客户端扩展模块 API

本文的将对扩展模块中的 API 进行介绍，如果需要了解扩展机制及扩展开发相关请参考 客户端扩展机制文档。

API 概览

客户端扩展模块 API 由客户端提供，扩展中的 JavaScript 模块代码中可以访问这些 API 来实现丰富的功能。实际运行时可用的 API 由扩展 API 声明和服务器端授权决定。

目前扩展模块中可供使用的 API 共有 288 个。根据扩展实际拥有的权限，在扩展模块中仅能够调用拥有权限的 API。

按模块划分

根据扩展所属的模块不同，可以分为 `user`、`members`、`im`、`ext`、`notification`、`contextmenu`、`commander`、`components`、`views`、`utils`、`window`、`nodeModules`、`node`、`lang`、`env`、`platform` 几类，下面分别列出不同模块下可用的 API：

- **user** 用户信息访问相关
  - `user.getCurrentUser` (Level 1)：获取当前登录的用户信息
- **members** 通讯录成员信息访问相关
  - `members.getDept` (Level 3)：获取部门信息
  - `members.getDeptsTree` (Level 3)：获取部门结构树信息
  - `members.getMember` (Level 3)：根据账号或用户 ID 获取用户对象信息
  - `members.getMembers` (Level 3)：获取系统中所有用户账号或者指定过滤条件，以列表形式返回
  - `members.getRoleName` (Level 3)：获取角色名称
- **im** 聊天相关
  - `im.createGroupChat` (Level 4)：创建一个讨论组会话
  - `im.dismissChat` (Level 4)：解散讨论组会话
  - `im.exitChat` (Level 4)：退出讨论组会话
  - `im.fetchChatsHistory` (Level 1)：请求获取会话聊天记录
  - `im.fetchPublicChats` (Level 3)：请求获取公开讨论组
  - `im.getChat` (Level 3)：根据 gid 或 ID 获取会话
  - `im.getChatMessages` (Level 3)：根据 gid 或 ID 获取会话消息列表（仅限在界面上显示的消息）
  - `im.getChats` (Level 3)：获取系统中所有会话或者指定过滤条件，以列表形式返回
  - `im.inviteMembersToChat` (Level 4)：邀请其他成员加入会话
  - `im.joinChat` (Level 4)：加入会话
  - `im.renameChat` (Level 4)：重命名会话
  - `im.sendCodeMessage` (Level 4)：请求发送代码类消息
  - `im.sendContentToChatSendbox` (Level 2)：向会话输入框添加内容
  - `im.sendEmojiMessage` (Level 4)：向会话发送 Emoji 表情消息
  - `im.sendFileMessage` (Level 4)：向会话发送文件消息
  - `im.sendImageMessage` (Level 4)：向会话发送图片消息
  - `im.sendTextMessage` (Level 4)：向会话发送文本消息
  - `im.shareContentToChat` (Level 4)：请求分享内容到会话
  - `im.toggleFreezeChat` (Level 4)：切换会话是否为从最近列表移除
  - `im.toggleMuteChat` (Level 4)：切换会话是否为开启免打扰
  - `im.toggleStarChat` (Level 4)：切换会话是否为收藏
- **ext** 扩展相关
  - `ext.MainView` (Level 6)：扩展应用主界面视图组件（当前扩展类型为 app，且应用类型为 insideView 时有效）
  - `ext.getExtension` (Level 0)：获取当前扩展对象
  - `ext.onAttach` (Level 0)：允许绑定扩展生命周期函数，当扩展被加载后调用，此时可以对扩展进行初始化
  - `ext.onDetach` (Level 0)：允许绑定扩展生命周期函数，当扩展被卸载时调用，此时应该将扩展使用的资源进行释放，例如销毁定时器等
  - `ext.onReady` (Level 0)：允许绑定扩展生命周期函数，当界面加载完毕时调用，此时扩展可以处理与界面相关操作
  - `ext.onReceiveChatMessages` (Level 4)：允许绑定扩展生命周期函数，当用户接收到聊天消息时调用
  - `ext.onRenderChatMessageContent` (Level 4)：允许绑定扩展生命周期函数，当在界面上需要转化 markdown 格式的消息文本为 html 时会调用此回调方法
  - `ext.onRequestOpenApp`：允许绑定扩展生命周期函数，当扩展类型为应用且应用类型为 `'custom'` 时，使用此函数来执行用户点击应用图标时的操作
  - `ext.onSendChatMessage` (Level 4)：允许绑定扩展生命周期函数，当用户发送聊天消息时调用
  - `ext.onUserLogin` (Level 1)：允许绑定扩展生命周期函数，当用户登录时调用
  - `ext.onUserLogout` (Level 1)：允许绑定扩展生命周期函数，当用户登录出调用
  - `ext.onUserStatusChange` (Level 1)：允许绑定扩展生命周期函数，当用户状态变更时调用
  - `ext.urlInspectors` (Level 5)：允许提供网址解释器
- **notification** 通知相关

  - `notification.requestAttention` (Level 5)：请求在用户操作系统桌面提示窗口界面有新内容
  - `notification.sendLocalNotification` (Level 5)：向通知中心发送一条本地通知
  - `notification.setAppBadge` (Level 5)：设置应用图标上的小红点提示
  - `notification.showDesktopNotification` (Level 5)：显示一条桌面弹窗通知
- **contextmenu** 上下文菜单相关
  - `contextmenu.addContextMenuCreator` (Level 5)：添加上下文菜单
  - `contextmenu.addContextMenuCreator.chat.actions` (Level 5)：添加会话操作菜单
  - `contextmenu.addContextMenuCreator.chat.member` (Level 5)：添加会话成员相关操作菜单
  - `contextmenu.addContextMenuCreator.chat.menu` (Level 5)：添加会话在会话列表上的菜单
  - `contextmenu.addContextMenuCreator.chat.sendbox.sendButton` (Level 5)：添加会话发送框发送按钮右键菜单
  - `contextmenu.addContextMenuCreator.chat.sendbox.toolbar` (Level 5)：添加会话发送框工具栏菜单
  - `contextmenu.addContextMenuCreator.chat.sidebar.file` (Level 5)：添加会话侧边栏文件列表操作菜单
  - `contextmenu.addContextMenuCreator.chat.sidebar.member` (Level 5)：添加会话侧边栏成员操作菜单
  - `contextmenu.addContextMenuCreator.chat.toolbar` (Level 5)：添加会话工具栏操作菜单
  - `contextmenu.addContextMenuCreator.chat.toolbar.more` (Level 5)：添加会话工具栏更多按钮操作菜单
  - `contextmenu.addContextMenuCreator.chats.menu` (Level 5)：添加会话列表类型切换菜单
  - `contextmenu.addContextMenuCreator.chats.menu.group` (Level 5)：添加会话列表分组操作菜单
  - `contextmenu.addContextMenuCreator.ext.app` (Level 6)：添加扩展应用操作菜单
  - `contextmenu.addContextMenuCreator.ext.apps.navbar` (Level 6)：添加应用在导航上的操作菜单
  - `contextmenu.addContextMenuCreator.ext.extension` (Level 6)：添加扩展操作菜单
  - `contextmenu.addContextMenuCreator.ext.extensionsMenu` (Level 6)：添加扩展列表上的操作菜单
  - `contextmenu.addContextMenuCreator.ext.openedApp` (Level 6)：添加已打开的扩展应用操作菜单
  - `contextmenu.addContextMenuCreator.image` (Level 4)：添加图片操作菜单
  - `contextmenu.addContextMenuCreator.member` (Level 5)：添加成员操作菜单
  - `contextmenu.addContextMenuCreator.member.profile` (Level 4)：添加成员资料菜单
  - `contextmenu.addContextMenuCreator.message` (Level 5)：添加聊天消息菜单
  - `contextmenu.addContextMenuCreator.message.basic` (Level 5)：添加聊天消息基本操作菜单
  - `contextmenu.addContextMenuCreator.message.code` (Level 5)：添加代码类聊天消息基本操作菜单
  - `contextmenu.addContextMenuCreator.message.image` (Level 5)：添加图片类聊天消息基本操作菜单
  - `contextmenu.addContextMenuCreator.message.text` (Level 5)：添加文本类聊天消息基本操作菜单
  - `contextmenu.addContextMenuCreator.message.url` (Level 5)：添加链接类聊天消息基本操作菜单
  - `contextmenu.addContextMenuCreator.profile.menu` (Level 4)：添加个人资料面板菜单
  - `contextmenu.removeContextMenuCreator` (Level 0)：移除添加的上下文菜单
  - `contextmenu.showContextMenu` (Level 4)：显示上下文菜单
- **commander** 命令相关
  - `commander.executeCommand` (Level 6)：执行命令
  - `commander.executeCommand.closeDisplay` (Level 2)：执行关闭弹出层命令
  - `commander.executeCommand.closeModal` (Level 2)：执行关闭对话框命令
  - `commander.executeCommand.confirmJoinPublicChat` (Level 4)：执行请求加入公开会话命令
  - `commander.executeCommand.mentionMemberInSendbox` (Level 2)：执行在输入框“@成员”命令
  - `commander.executeCommand.openInApp` (Level 5)：执行在扩展应用中打开链接命令
  - `commander.executeCommand.openUrlInBrowser` (Level 5)：执行在浏览器中打开链接命令
  - `commander.executeCommand.openUrlInDialog` (Level 5)：执行在对话框中打开链接命令
  - `commander.executeCommand.openWebviewDialog` (Level 5)：执行在 Webview 对话框中打开链接命令
  - `commander.executeCommand.sendContentToChat` (Level 2)：执行向输入框发送内容命令
  - `commander.executeCommand.setRoute` (Level 6)：执行设置界面路由命令
  - `commander.executeCommand.shortcut.captureScreenHotkey` (Level 2)：执行启动截图命令
  - `commander.executeCommand.shortcut.focusWindowHotkey` (Level 2)：执行激活窗口命令
  - `commander.executeCommand.showChatAddCategoryDialog` (Level 4)：执行显示将会话添加到分组对话框命令
  - `commander.executeCommand.showChatCommittersSettingDialog` (Level 4)：执行显示会话白名单对话框命令
  - `commander.executeCommand.showChatInviteDialog` (Level 4)：执行显示邀请成员到会话对话框命令
  - `commander.executeCommand.showChatSendCodeDialog` (Level 4)：执行显示发送代码对话框命令
  - `commander.executeCommand.showChatSendDialog` (Level 4)：执行显示发送内容到会话对话框命令
  - `commander.executeCommand.showChatShareDialog` (Level 4)：执行转发内容到会话对话框命令
  - `commander.executeCommand.showChatsHistoryDialog` (Level 2)：执行显示会话历史记录对话框命令
  - `commander.executeCommand.showConfirmSendFilesDialog` (Level 4)：执行显示确认发送文件对话框命令
  - `commander.executeCommand.showCreateChatDialog` (Level 4)：执行显示创建会话对话框命令
  - `commander.executeCommand.showEmojiPopover` (Level 4)：执行显示 Emoji 选择面板命令
  - `commander.executeCommand.showExtensionDialog` (Level 6)：执行显示扩展详情对话框命令
  - `commander.executeCommand.showExtensionInstallDialog` (Level 6)：执行显示安装扩展对话框命令
  - `commander.executeCommand.showHotkeySettingDialog` (Level 6)：执行显示快捷键设置对话框命令
  - `commander.executeCommand.showLanguageSwitchDialog` (Level 4)：执行显示界面语言切换对话框命令
  - `commander.executeCommand.showMemberProfile` (Level 4)：执行显示用户资料面板命令
  - `commander.executeCommand.showMessager` (Level 5)：执行显示提示消息命令
  - `commander.executeCommand.showTodoEditDialog` (Level 6)：执行显示待办编辑对话框命令
  - `commander.executeCommand.updateViewStyle` (Level 6)：执行请求更新视图样式命令
  - `commander.executeCommand.viewImage` (Level 2)：执行显示查看图片命令

    - `commander.executeCommandLine` (Level 6)：执行命令串
    - `commander.executeCommandWithContext` (Level 6)：执行包含上下文参数的命令
    - `commander.registerCommand` (Level 5)：注册命令
    - `commander.unregisterCommand` (Level 0)：取消注册的命令
- **components** 组件相关
    - `components.AppAvatar` (Level 6)：应用图标组件
    - `components.AreaSelector` (Level 6)：区域选择组件
    - `components.Avatar` (Level 6)：头像组件
    - `components.Button` (Level 6)：按钮组件
    - `components.Checkbox` (Level 6)：复选框组件
    - `components.ClickOutsideWrapper` (Level 6)：监听点击外部的辅助组件
    - `components.ContextMenu` (Level 6)：上下文菜单组件
    - `components.Display` (Level 6)：弹出层管理对象
    - `components.DisplayContainer` (Level 6)：弹出层容器组件
    - `components.DisplayLayer` (Level 6)：弹出层组件
    - `components.Emoji` (Level 6)：Emoji 组件
    - `components.HotkeyInputControl` (Level 6)：快捷键设置组件
    - `components.Icon` (Level 6)：图标组件
    - `components.ImageCutter` (Level 6)：图片剪切组件
    - `components.ImageViewer` (Level 6)：图片查看组件
    - `components.InputControl` (Level 6)：输入框组件
    - `components.Messager` (Level 6)：提示消息组件
    - `components.Modal` (Level 6)：对话框组件
    - `components.Pager` (Level 6)：分页组件
    - `components.Popover` (Level 6)：弹出面板组件
    - `components.SearchControl` (Level 6)：搜索组件
    - `components.SelectBox` (Level 6)：选择框组件
    - `components.Spinner` (Level 6)：显示加载中动画组件
    - `components.TabPane` (Level 6)：标签页面板组件
    - `components.Tabs` (Level 6)：标签页组件
- **views** 主界面视图相关
    - `views.chats` (Level 6)：主界面上聊天相关组件
    - `views.chats.ChatAvatar` (Level 6)：会话头像组件
    - `views.chats.ChatChangeFontPopover` (Level 6)：会话消息字体大小设置面板功能库
    - `views.chats.ChatCommittersSetting` (Level 6)：会话白名单设置界面组件
    - `views.chats.ChatCommittersSettingDialog` (Level 6)：会话白名单设置对话框功能库
    - `views.chats.ChatCreateDialog` (Level 6)：新建会话对话框功能库
    - `views.chats.ChatCreateGroups` (Level 6)：创建讨论组会话组件
    - `views.chats.ChatHeader` (Level 6)：会话界面头部组件
    - `views.chats.ChatHistory` (Level 6)：会话历史记录查看组件
    - `views.chats.ChatInviteDialog` (Level 6)：邀请加入会话对话框功能库
    - `views.chats.ChatJoinPublic` (Level 6)：加入公开讨论组界面组件
    - `views.chats.ChatListItem` (Level 6)：会话列表项组件
    - `views.chats.ChatMessages` (Level 6)：会话消息列表组件
    - `views.chats.ChatSearchResult` (Level 6)：会话搜索结果组件
    - `views.chats.ChatSendbox` (Level 6)：会话消息输入发送框
    - `views.chats.ChatShareDialog` (Level 6)：分享到会话对话框功能库
    - `views.chats.ChatSidebar` (Level 6)：会话侧边栏组件
    - `views.chats.ChatSidebarFiles` (Level 6)：会话文件侧边栏组件
    - `views.chats.ChatSidebarPeoples` (Level 6)：会话成员侧边栏组件
    - `views.chats.ChatSidebarProfile` (Level 6)：会话侧边栏个人资料界面组件
    - `views.chats.ChatTipPopover` (Level 6)：会话发送框小提示功能库
    - `views.chats.ChatTitle` (Level 6)：会话标题组件
    - `views.chats.ChatView` (Level 6)：会话组件
    - `views.chats.ChatsCache` (Level 6)：会话界面缓存组件
    - `views.chats.ChatsDndContainer` (Level 6)：会话拖放事件处理组件
    - `views.chats.ChatsHistory` (Level 6)：会话历史记录管理界面组件
    - `views.chats.ChatsHistoryDialog` (Level 6)：会话历史记录管理对话框功能库
    - `views.chats.Index` (Level 6)：会话界面首页组件
    - `views.chats.Menu` (Level 6)：会话列表界面组件
    - `views.chats.MenuList` (Level 6)：会话列表组件
    - `views.chats.MenuSearchList` (Level 6)：会话搜索结果列表组件
    - `views.chats.MessageBroadcast` (Level 6)：广播类会话消息组件
    - `views.chats.MessageContentFile` (Level 6)：文件类消息内容组件
    - `views.chats.MessageContentImage` (Level 6)：图片类消息内容组件
    - `views.chats.MessageContentText` (Level 6)：文本类消息内容组件
    - `views.chats.MessageDivider` (Level 6)：会话消息分割线组件

    - `views.chats.MessageList` (Level 6)：消息列表组件
    - `views.chats.MessageListItem` (Level 6)：会话消息列表项组件
    - `views.chats.MessagesPreviewDialog` (Level 6)：消息预览对话框功能库
    - `views.common` (Level 6)：主界面上的通用组件
    - `views.common.About` (Level 6)：关于界面组件
    - `views.common.AboutDialog` (Level 6)：关于对话框功能库
    - `views.common.BuildInfo` (Level 6)：客户端版本信息组件
    - `views.common.DraftEditor` (Level 6)：消息输入框组件
    - `views.common.EmojiPopover` (Level 6)：Emoji 选择面板功能库
    - `views.common.FileListItem` (Level 6)：文件列表项组件
    - `views.common.FileListView` (Level 6)：文件列表项组件
    - `views.common.HotkeySettingDialog` (Level 6)：快捷键设置对话框功能库
    - `views.common.MemberList` (Level 6)：成员列表组件
    - `views.common.MemberListItem` (Level 6)：成员列表项组件
    - `views.common.MemberProfile` (Level 6)：个人资料界面组件
    - `views.common.MemberProfileDialog` (Level 6)：个人资料对话框功能库
    - `views.common.Routes` (Level 6)：路由功能库
    - `views.common.SelectDialog` (Level 6)：选择对话框功能库
    - `views.common.SelectPanel` (Level 6)：选择面板组件
    - `views.common.SelectPanelWithActions` (Level 6)：带操作的选择面板组件
    - `views.common.StatusDot` (Level 6)：用户状态指示标签组件
    - `views.common.UserAvatar` (Level 6)：用户头像组件
    - `views.common.UserChangePasswordDialog` (Level 6)：修改密码对话框功能库
    - `views.common.UserListItem` (Level 6)：用户列表项组件
    - `views.common.UserProfileDialog` (Level 6)：用户个人资料对话框功能库
    - `views.common.UserSetting` (Level 6)：用户设置界面组件
    - `views.common.UserSettingDialog` (Level 6)：用户设置对话框功能库
    - `views.contacts` (Level 6)：通讯录界面组件
    - `views.contacts.ContactsHome` (Level 6)：通讯录界面组件
    - `views.contacts.ContactsIndex` (Level 6)：通讯录首页组件
    - `views.contacts.ContactsView` (Level 6)：通讯录列表界面组件
    - `views.contacts.DeptsTree` (Level 6)：部门树结构界面组件
    - `views.contacts.GroupsMenu` (Level 6)：讨论组菜单组件
    - `views.contacts.GroupsView` (Level 6)：讨论组列表组件
    - `views.exts` (Level 6)：主界面上的扩展相关组件
    - `views.exts.AppExtensions` (Level 6)：应用扩展管理界面组件
    - `views.exts.AppFiles` (Level 6)：文件应用界面组件
    - `views.exts.AppHome` (Level 6)：应用管理首页界面组件
    - `views.exts.ExtensionDetail` (Level 6)：扩展详情界面组件
    - `views.exts.ExtensionListItem` (Level 6)：扩展列表项组件
    - `views.exts.Index` (Level 6)：扩展首页界面组件
    - `views.exts.WebApp` (Level 6)：Web 应用界面组件
    - `views.index` (Level 6)：主界面上的首页相关组件
    - `views.index.AppView` (Level 6)：主窗口首页界面组件
    - `views.index.ImageCutterApp` (Level 6)：截屏窗口界面组件
    - `views.index.Index` (Level 6)：主窗口界面组件
    - `views.login` (Level 6)：主界面上的登录相关组件
    - `views.login.Form` (Level 6)：登录表单界面组件
    - `views.login.Index` (Level 6)：登录界面首页组件
    - `views.login.SwapUser` (Level 6)：切换已登录的账号界面组件
    - `views.login.SwapUserDialog` (Level 6)：切换已登录的账号对话框功能库
    - `views.main` (Level 6)：主界面上的界面框架组件
    - `views.main.AutoReconnectBar` (Level 6)：自动重连提示界面组件
    - `views.main.CacheContainer` (Level 6)：应用界面缓存组件
    - `views.main.Index` (Level 6)：主界面组件
    - `views.main.Navbar` (Level 6)：主导航组件
- **utils** 通用工具模块相关
    - `utils.Color` (Level 5)：颜色辅助类
    - `utils.DateHelper` (Level 5)：日期时间辅助方法
    - `utils.HtmlHelper` (Level 5)：HTML 操作辅助方法
    - `utils.Image` (Level 5)：图片操作辅助方法
    - `utils.LimitTimePromise` (Level 5)：限时异步实现辅助方法
    - `utils.Markdown` (Level 5)：Markdown 辅助方法
    - `utils.Pinyin` (Level 5)：拼音辅助方法
    - `utils.Plain` (Level 5)：对象扁平化辅助方法
    - `utils.Skin` (Level 5)：CSS 样式计算辅助方法
    - `utils.Store` (Level 5)：本地存储辅助方法

    - `utils.StringHelper` (Level 5)：字符串辅助方法
    - `utils.version` (Level 5)：语义化版本号辅助方法
- **window** 浏览器内置对象相关
    - `window.document` (Level 6)：访问浏览器 document 对象
    - `window.window` (Level 6)：访问浏览器 window 对象
- **nodeModules** 第三方 node 模块
    - `nodeModules.compareVersions` (Level 7)：compare-versions 模块，提供对版本号进行比较的方法
    - `nodeModules.draft-js` (Level 6)：draft-js 模块，提供 DraftJS 编辑器组件
    - `nodeModules.emoji-toolkit` (Level 7)：emoji-toolkit 模块，提供 Emoji 辅助方法
    - `nodeModules.emojione-picker` (Level 6)：emojione-picker 模块，提供 Emoji 表情选择面板组件
    - `nodeModules.extract-zip` (Level 7)：extract-zip 模块，提供对 zip 文件进行解压缩方法
    - `nodeModules.highlight-js` (Level 7)：hightlight.js 模块，提供对代码片段进行高亮化方法
    - `nodeModules.hotkeys-js` (Level 7)：hotkeys-js 模块，提供快捷键辅助方法
    - `nodeModules.htmlparser` (Level 7)：htmlparser 模块，提供 HTML 片段解析方法
    - `nodeModules.jquery` (Level 6)：jquery 模块，提供 jQuery 功能对象
    - `nodeModules.marked` (Level 7)：marked 模块，提供 Markdown 辅助方法
    - `nodeModules.md5` (Level 7)：md5 模块，提供 md5 算法方法
    - `nodeModules.prop-types` (Level 6)：prop-types，提供组件属性定义和检查辅助方法
    - `nodeModules.react` (Level 6)：react 模块
    - `nodeModules.react-dom` (Level 6)：react-dom 模块
    - `nodeModules.react-split-pane` (Level 6)：react-split-pane 模块，提供可拖放调整布局的容器组件
    - `nodeModules.tiny-pinyin` (Level 7)：tiny-pinyin 模块，提供汉语拼音操作方法
    - `nodeModules.uuid` (Level 6)：uuid 模块，提供全局唯一字符串生成辅助方法
- **node** NodeJS 内置模块
    - `node.child_process` (Level 7)：NodeJS 内置模块，子进程
    - `node.crypto` (Level 7)：NodeJS 内置模块，加密和解密
    - `node.dgram` (Level 7)：NodeJS 内置模块，数据报
    - `node.dns` (Level 7)：NodeJS 内置模块，域名服务器
    - `node.events` (Level 7)：NodeJS 内置模块，事件
    - `node.fs` (Level 7)：NodeJS 内置模块，文件读写
    - `node.global.process` (Level 7)：NodeJS global.process 对象
    - `node.http` (Level 7)：NodeJS 内置模块，HTTP
    - `node.http2` (Level 7)：NodeJS 内置模块，HTTP/2
    - `node.https` (Level 7)：NodeJS 内置模块，HTTPS
    - `node.net` (Level 7)：NodeJS 内置模块，网络
    - `node.os` (Level 7)：NodeJS 内置模块，操作系统
    - `node.path` (Level 7)：NodeJS 内置模块，路径
    - `node.string_decoder` (Level 7)：NodeJS 内置模块，字符串解码器
    - `node.url` (Level 7)：NodeJS 内置模块，URL 解析
    - `node.util` (Level 7)：NodeJS 实用工具库
    - `node.zlib` (Level 7)：NodeJS 内置模块，压缩
- **lang** 界面语言文本管理对象
    - `lang` (Level 7)：界面语言文本管理对象
- **env** 访问运行环境相关信息
    - `env.arch` (Level 5)：获取操作系统架构类型
    - `env.chrome` (Level 7)：获取所使用的 Chrome 版本
    - `env.displayName` (Level 5)：获取客户端显示名称
    - `env.displayVersion` (Level 5)：获取客户端显示的版本
    - `env.electron` (Level 7)：获取用户所使用的 Electron 版本
    - `env.lang` (Level 5)：获取用户所使用的界面语言类型
    - `env.modules` (Level 7)：获取所使用的 node modules 版本
    - `env.node` (Level 7)：获取所使用的 node 版本
    - `env.os` (Level 5)：获取操作系统类型
    - `env.productName` (Level 5)：获取客户端产品名称
    - `env.v8` (Level 7)：获取所使用的 v8 版本
    - `env.version` (Level 5)：获取客户端版本
- **platform** 访问平台提供的实用工具
    - `platform` (Level 7)：访问平台提供的实用工具

按等级划分

扩展 API 根据权限的不同划分为 7 个等级，等级越高的 API 拥有的权限更高，能够实现更复杂的功能。在 “基础版” 授权中客户端扩展可以使用**等级 5** 及小于等级 5 下的所有 API，在 “增强版” 和 “OEM” 版本中客户端扩展可以使用所有等级的扩展 API。

下面为每个等级下的 API 清单：

- **等级 0**：扩展模块默认拥有等级为 0 的 API

    - `commander.unregisterCommand`：取消注册的命令
    - `contextmenu.removeContextMenuCreator`：移除添加的上下文菜单
    - `ext.getExtension`：获取当前扩展对象
    - `ext.onAttach`：允许绑定扩展生命周期函数，当扩展被加载后调用，此时可以对扩展进行初始化
    - `ext.onDetach`：允许绑定扩展生命周期函数，当扩展被卸载时调用，此时应该将扩展使用的资源进行释放，例如销毁定时器等
    - `ext.onReady`：允许绑定扩展生命周期函数，当界面加载完毕时调用，此时扩展可以处理与界面相关操作
- **等级 1**：能够读取当前用户自身相关信息
    - `ext.onUserLogin`：允许绑定扩展生命周期函数，当用户登录时调用
    - `ext.onUserLogout`：允许绑定扩展生命周期函数，当用户登录出调用
    - `ext.onUserStatusChange`：允许绑定扩展生命周期函数，当用户状态变更时调用
    - `im.fetchChatsHistory`：请求获取会话聊天记录
    - `user.getCurrentUser`：获取当前登录的用户信息
- **等级 2**：能够执行与当前用户自身相关其他操作并能够使用界面上的一些通用组件和对话框
    - `commander.executeCommand.closeDisplay`：执行关闭弹出层命令
    - `commander.executeCommand.closeModal`：执行关闭对话框命令
    - `commander.executeCommand.mentionMemberInSendbox`：执行在输入框“@成员”命令
    - `commander.executeCommand.sendContentToChat`：执行向输入框发送内容命令
    - `commander.executeCommand.shortcut.captureScreenHotkey`：执行启动截图命令
    - `commander.executeCommand.shortcut.focusWindowHotkey`：执行激活窗口命令
    - `commander.executeCommand.showChatsHistoryDialog`：执行显示会话历史记录对话框命令
    - `commander.executeCommand.viewImage`：执行显示查看图片命令
    - `im.sendContentToChatSendbox`：向会话输入框添加内容
- **等级 3**：能够获取会话列表、消息记录、部门和组织成员信息
    - `im.fetchPublicChats`：请求获取公开讨论组
    - `im.getChat`：根据 gid 或 ID 获取会话
    - `im.getChatMessages`：根据 gid 或 ID 获取会话消息列表（仅限在界面上显示的消息）
    - `im.getChats`：获取系统中所有会话或者指定过滤条件，以列表形式返回
    - `members.getDept`：获取部门信息
    - `members.getDeptsTree`：获取部门结构树信息
    - `members.getMember`：根据账号或用户 ID 获取用户对象信息
    - `members.getMembers`：获取系统中所有用户账号或者指定过滤条件，以列表形式返回
    - `members.getRoleName`：获取角色名称
- **等级 4**：能够通过 API 发送消息，对会话进行操作以及通过命令使用功能对话框
    - `commander.executeCommand.confirmJoinPublicChat`：执行请求加入公开会话命令
    - `commander.executeCommand.showChatAddCategoryDialog`：执行显示将会话添加到分组对话框命令
    - `commander.executeCommand.showChatCommittersSettingDialog`：执行显示会话白名单对话框命令
    - `commander.executeCommand.showChatInviteDialog`：执行显示邀请成员到会话对话框命令
    - `commander.executeCommand.showChatSendCodeDialog`：执行显示发送代码对话框命令
    - `commander.executeCommand.showChatSendDialog`：执行显示发送内容到会话对话框命令
    - `commander.executeCommand.showChatShareDialog`：执行转发内容到会话对话框命令
    - `commander.executeCommand.showConfirmSendFilesDialog`：执行显示确认发送文件对话框命令
    - `commander.executeCommand.showCreateChatDialog`：执行显示创建会话对话框命令
    - `commander.executeCommand.showEmojiPopover`：执行显示 Emoji 选择面板命令
    - `commander.executeCommand.showLanguageSwitchDialog`：执行显示界面语言切换对话框命令
    - `commander.executeCommand.showMemberProfile`：执行显示用户资料面板命令
    - `contextmenu.addContextMenuCreator.image`：添加图片操作菜单
    - `contextmenu.addContextMenuCreator.member.profile`：添加成员资料菜单
    - `contextmenu.addContextMenuCreator.profile.menu`：添加个人资料面板菜单
    - `contextmenu.showContextMenu`：显示上下文菜单
    - `ext.onReceiveChatMessages`：允许绑定扩展生命周期函数，当用户接收到聊天消息时调用
    - `ext.onRenderChatMessageContent`：允许绑定扩展生命周期函数，当在界面上需要转化 markdown 格式的消息文本为 html 时会调用此回调方法
    - `ext.onSendChatMessage`：允许绑定扩展生命周期函数，当用户发送聊天消息时调用
    - `im.createGroupChat`：创建一个讨论组会话
    - `im.dismissChat`：解散讨论组会话
    - `im.exitChat`：退出讨论组会话
    - `im.inviteMembersToChat`：邀请其他成员加入会话
    - `im.joinChat`：加入会话
    - `im.renameChat`：重命名会话
    - `im.sendCodeMessage`：请求发送代码类消息
    - `im.sendEmojiMessage`：向会话发送 Emoji 表情消息
    - `im.sendFileMessage`：向会话发送文件消息
    - `im.sendImageMessage`：向会话发送图片消息
    - `im.sendTextMessage`：向会话发送文本消息
    - `im.shareContentToChat`：请求分享内容到会话
    - `im.toggleFreezeChat`：切换会话是否为从最近列表移除
    - `im.toggleMuteChat`：切换会话是否为开启免打扰

    - `im.toggleStarChat`：切换会话是否为收藏
- **等级 5**：能够自定义消息卡片、右键菜单，发送本地通知并能够使用更多的内置辅助功能 API
    - `commander.executeCommand.openInApp`：执行在扩展应用中打开链接命令
    - `commander.executeCommand.openUrlInBrowser`：执行在浏览器中打开链接命令
    - `commander.executeCommand.openUrlInDialog`：执行在对话框中打开链接命令
    - `commander.executeCommand.openWebviewDialog`：执行在 Webview 对话框中打开链接命令
    - `commander.executeCommand.showMessager`：执行显示提示消息命令
    - `commander.registerCommand`：注册命令
    - `contextmenu.addContextMenuCreator`：添加上下文菜单
    - `contextmenu.addContextMenuCreator.chat.actions`：添加会话操作菜单
    - `contextmenu.addContextMenuCreator.chat.member`：添加会话成员相关操作菜单
    - `contextmenu.addContextMenuCreator.chat.menu`：添加会话在会话列表上的菜单
    - `contextmenu.addContextMenuCreator.chat.sendbox.sendButton`：添加会话发送框发送按钮右键菜单
    - `contextmenu.addContextMenuCreator.chat.sendbox.toolbar`：添加会话发送框工具栏菜单
    - `contextmenu.addContextMenuCreator.chat.sidebar.file`：添加会话侧边栏文件列表操作菜单
    - `contextmenu.addContextMenuCreator.chat.sidebar.member`：添加会话侧边栏成员操作菜单
    - `contextmenu.addContextMenuCreator.chat.toolbar`：添加会话工具栏操作菜单
    - `contextmenu.addContextMenuCreator.chat.toolbar.more`：添加会话工具栏更多按钮操作菜单
    - `contextmenu.addContextMenuCreator.chats.menu`：添加会话列表类型切换菜单
    - `contextmenu.addContextMenuCreator.chats.menu.group`：添加会话列表分组操作菜单
    - `contextmenu.addContextMenuCreator.member`：添加成员操作菜单
    - `contextmenu.addContextMenuCreator.message`：添加聊天消息菜单
    - `contextmenu.addContextMenuCreator.message.basic`：添加聊天消息基本操作菜单
    - `contextmenu.addContextMenuCreator.message.code`：添加代码类聊天消息基本操作菜单
    - `contextmenu.addContextMenuCreator.message.image`：添加图片类聊天消息基本操作菜单
    - `contextmenu.addContextMenuCreator.message.text`：添加文本类聊天消息基本操作菜单
    - `contextmenu.addContextMenuCreator.message.url`：添加链接类聊天消息基本操作菜单
    - `env.arch`：获取操作系统架构类型
    - `env.displayName`：获取客户端显示名称
    - `env.displayVersion`：获取客户端显示的版本
    - `env.lang`：获取用户所使用的界面语言类型
    - `env.os`：获取操作系统类型
    - `env.productName`：获取客户端产品名称
    - `env.version`：获取客户端版本
    - `ext.urlInspectors`：允许提供网址解释器
    - `notification.requestAttention`：请求在用户操作系统桌面提示窗口界面有新内容
    - `notification.sendLocalNotification`：向通知中心发送一条本地通知
    - `notification.setAppBadge`：设置应用图标上的小红点提示
    - `notification.showDesktopNotification`：显示一条桌面弹窗通知
    - `utils.Color`：颜色辅助类
    - `utils.DateHelper`：日期时间辅助方法
    - `utils.HtmlHelper`：HTML 操作辅助方法
    - `utils.Image`：图片操作辅助方法
    - `utils.LimitTimePromise`：限时异步实现辅助方法
    - `utils.Markdown`：Markdown 辅助方法
    - `utils.Pinyin`：拼音辅助方法
    - `utils.Plain`：对象扁平化辅助方法
    - `utils.Skin`：CSS 样式计算辅助方法
    - `utils.Store`：本地存储辅助方法
    - `utils.StringHelper`：字符串辅助方法
    - `utils.version`：语义化版本号辅助方法
- **等级 6**：能够使用 React 来开发应用，直接使用内置的通用组件和界面上的功能组件，主动执行功能命令
    - `commander.executeCommand`：执行命令
    - `commander.executeCommand.setRoute`：执行设置界面路由命令
    - `commander.executeCommand.showExtensionDialog`：执行显示扩展详情对话框命令
    - `commander.executeCommand.showExtensionInstallDialog`：执行显示安装扩展对话框命令
    - `commander.executeCommand.showHotkeySettingDialog`：执行显示快捷键设置对话框命令
    - `commander.executeCommand.showTodoEditDialog`：执行显示待办编辑对话框命令
    - `commander.executeCommand.updateViewStyle`：执行请求更新视图样式命令
    - `commander.executeCommandLine`：执行命令串
    - `commander.executeCommandWithContext`：执行包含上下文参数的命令
    - `components.AppAvatar`：应用图标组件
    - `components.AreaSelector`：区域选择组件
    - `components.Avatar`：头像组件
    - `components.Button`：按钮组件
    - `components.Checkbox`：复选框组件
    - `components.ClickOutsideWrapper`：监听点击外部的辅助组件

- `components.ContextMenu`：上下文菜单组件
- `components.Display`：弹出层管理对象
- `components.DisplayContainer`：弹出层容器组件
- `components.DisplayLayer`：弹出层组件
- `components.Emoji`：Emoji 组件
- `components.HotkeyInputControl`：快捷键设置组件
- `components.Icon`：图标组件
- `components.ImageCutter`：图片剪切组件
- `components.ImageViewer`：图片查看组件
- `components.InputControl`：输入框组件
- `components.Messager`：提示消息组件
- `components.Modal`：对话框组件
- `components.Pager`：分页组件
- `components.Popover`：弹出面板组件
- `components.SearchControl`：搜索组件
- `components.SelectBox`：选择框组件
- `components.Spinner`：显示加载中动画组件
- `components.TabPane`：标签页面板组件
- `components.Tabs`：标签页组件
- `contextmenu.addContextMenuCreator.ext.app`：添加扩展应用操作菜单
- `contextmenu.addContextMenuCreator.ext.apps.navbar`：添加应用在导航上的操作菜单
- `contextmenu.addContextMenuCreator.ext.extension`：添加扩展操作菜单
- `contextmenu.addContextMenuCreator.ext.extensionsMenu`：添加扩展列表上的操作菜单
- `contextmenu.addContextMenuCreator.ext.openedApp`：添加已打开的扩展应用操作菜单
- `ext.MainView`：扩展应用主界面视图组件（当前扩展类型为 app，且应用类型为 insideView 时有效）
- `nodeModules.draft-js`：draft-js 模块，提供 DraftJS 编辑器组件
- `nodeModules.emojione-picker`：emojione-picker 模块，提供 Emoji 表情选择面板组件
- `nodeModules.jquery`：jquery 模块，提供 jQuery 功能对象
- `nodeModules.prop-types`：prop-types，提供组件属性定义和检查辅助方法
- `nodeModules.react`：react 模块
- `nodeModules.react-dom`：react-dom 模块
- `nodeModules.react-split-pane`：react-split-pane 模块，提供可拖放调整布局的容器组件
- `nodeModules.uuid`：uuid 模块，提供全局唯一字符串生成辅助方法
- `views.chats`：主界面上聊天相关组件
- `views.chats.ChatAvatar`：会话头像组件
- `views.chats.ChatChangeFontPopover`：会话消息字体大小设置面板功能库
- `views.chats.ChatCommittersSetting`：会话白名单设置界面组件
- `views.chats.ChatCommittersSettingDialog`：会话白名单设置对话框功能库
- `views.chats.ChatCreateDialog`：新建会话对话框功能库
- `views.chats.ChatCreateGroups`：创建讨论组会话组件
- `views.chats.ChatHeader`：会话界面头部组件
- `views.chats.ChatHistory`：会话历史记录查看组件
- `views.chats.ChatInviteDialog`：邀请加入会话对话框功能库
- `views.chats.ChatJoinPublic`：加入公开讨论组界面组件
- `views.chats.ChatListItem`：会话列表项组件
- `views.chats.ChatMessages`：会话消息列表组件
- `views.chats.ChatSearchResult`：会话搜索结果组件
- `views.chats.ChatSendbox`：会话消息输入发送框
- `views.chats.ChatShareDialog`：分享到会话对话框功能库
- `views.chats.ChatSidebar`：会话侧边栏组件
- `views.chats.ChatSidebarFiles`：会话文件侧边栏组件
- `views.chats.ChatSidebarPeoples`：会话成员侧边栏组件
- `views.chats.ChatSidebarProfile`：会话侧边栏个人资料界面组件
- `views.chats.ChatTipPopover`：会话发送框小提示功能库
- `views.chats.ChatTitle`：会话标题组件
- `views.chats.ChatView`：会话组件
- `views.chats.ChatsCache`：会话界面缓存组件
- `views.chats.ChatsDndContainer`：会话拖放事件处理组件
- `views.chats.ChatsHistory`：会话历史记录管理界面组件
- `views.chats.ChatsHistoryDialog`：会话历史记录管理对话框功能库
- `views.chats.Index`：会话界面首页组件
- `views.chats.Menu`：会话列表界面组件
- `views.chats.MenuList`：会话列表组件
- `views.chats.MenuSearchList`：会话搜索结果列表组件
- `views.chats.MessageBroadcast`：广播类会话消息组件
- `views.chats.MessageContentFile`：文件类消息内容组件
- `views.chats.MessageContentImage`：图片类消息内容组件
- `views.chats.MessageContentText`：文本类消息内容组件

    - `views.chats.MessageDivider`：会话消息分割线组件
    - `views.chats.MessageList`：消息列表组件
    - `views.chats.MessageListItem`：会话消息列表项组件
    - `views.chats.MessagesPreviewDialog`：消息预览对话框功能库
    - `views.common`：主界面上的通用组件
    - `views.common.About`：关于界面组件
    - `views.common.AboutDialog`：关于对话框功能库
    - `views.common.BuildInfo`：客户端版本信息组件
    - `views.common.DraftEditor`：消息输入框组件
    - `views.common.EmojiPopover`：Emoji 选择面板功能库
    - `views.common.FileListItem`：文件列表项组件
    - `views.common.FileListView`：文件列表项组件
    - `views.common.HotkeySettingDialog`：快捷键设置对话框功能库
    - `views.common.MemberList`：成员列表组件
    - `views.common.MemberListItem`：成员列表项组件
    - `views.common.MemberProfile`：个人资料界面组件
    - `views.common.MemberProfileDialog`：个人资料对话框功能库
    - `views.common.Routes`：路由功能库
    - `views.common.SelectDialog`：选择对话框功能库
    - `views.common.SelectPanel`：选择面板组件
    - `views.common.SelectPanelWithActions`：带操作的选择面板组件
    - `views.common.StatusDot`：用户状态指示标签组件
    - `views.common.UserAvatar`：用户头像组件
    - `views.common.UserChangePasswordDialog`：修改密码对话框功能库
    - `views.common.UserListItem`：用户列表项组件
    - `views.common.UserProfileDialog`：用户个人资料对话框功能库
    - `views.common.UserSetting`：用户设置界面组件
    - `views.common.UserSettingDialog`：用户设置对话框功能库
    - `views.contacts`：通讯录界面组件
    - `views.contacts.ContactsHome`：通讯录界面组件
    - `views.contacts.ContactsIndex`：通讯录首页组件
    - `views.contacts.ContactsView`：通讯录列表界面组件
    - `views.contacts.DeptsTree`：部门树结构界面组件
    - `views.contacts.GroupsMenu`：讨论组菜单组件
    - `views.contacts.GroupsView`：讨论组列表组件
    - `views.exts`：主界面上的扩展相关组件
    - `views.exts.AppExtensions`：应用扩展管理界面组件
    - `views.exts.AppFiles`：文件应用界面组件
    - `views.exts.AppHome`：应用管理首页界面组件
    - `views.exts.ExtensionDetail`：扩展详情界面组件
    - `views.exts.ExtensionListItem`：扩展列表项组件
    - `views.exts.Index`：扩展首页界面组件
    - `views.exts.WebApp`：Web 应用界面组件
    - `views.index`：主界面上的首页相关组件
    - `views.index.AppView`：主窗口首页界面组件
    - `views.index.ImageCutterApp`：截屏窗口界面组件
    - `views.index.Index`：主窗口界面组件
    - `views.login`：主界面上的登录相关组件
    - `views.login.Form`：登录表单界面组件
    - `views.login.Index`：登录界面首页组件
    - `views.login.SwapUser`：切换已登录的账号界面组件
    - `views.login.SwapUserDialog`：切换已登录的账号对话框功能库
    - `views.main`：主界面上的界面框架组件
    - `views.main.AutoReconnectBar`：自动重连提示界面组件
    - `views.main.CacheContainer`：应用界面缓存组件
    - `views.main.Index`：主界面组件
    - `views.main.Navbar`：主导航组件
    - `window.document`：访问浏览器 document 对象
    - `window.window`：访问浏览器 window 对象
- **等级 7**：能够使用 NodeJS 内置模块，访问平台（例如 electron）提供的额外实用工具库
    - `env.chrome`：获取所使用的 Chrome 版本
    - `env.electron`：获取用户所使用的 Electron 版本
    - `env.modules`：获取所使用的 node modules 版本
    - `env.node`：获取所使用的 node 版本
    - `env.v8`：获取所使用的 v8 版本
    - `lang`：界面语言文本管理对象
    - `node.child_process`：NodeJS 内置模块，子进程

- `node.crypto`：NodeJS 内置模块，加密和解密
- `node.dgram`：NodeJS 内置模块，数据报
- `node.dns`：NodeJS 内置模块，域名服务器
- `node.events`：NodeJS 内置模块，事件
- `node.fs`：NodeJS 内置模块，文件读写
- `node.global.process`：NodeJS global.process 对象
- `node.http`：NodeJS 内置模块，HTTP
- `node.http2`：NodeJS 内置模块，HTTP/2
- `node.https`：NodeJS 内置模块，HTTPS
- `node.net`：NodeJS 内置模块，网络
- `node.os`：NodeJS 内置模块，操作系统
- `node.path`：NodeJS 内置模块，路径
- `node.string_decoder`：NodeJS 内置模块，字符串解码器
- `node.url`：NodeJS 内置模块，URL 解析
- `node.util`：NodeJS 实用工具库
- `node.zlib`：NodeJS 内置模块，压缩
- `nodeModules.compareVersions`：compare-versions 模块，提供对版本号进行比较的方法
- `nodeModules.emoji-toolkit`：emoji-toolkit 模块，提供 Emoji 辅助方法
- `nodeModules.extract-zip`：extract-zip 模块，提供对 zip 文件进行解压缩方法
- `nodeModules.highlight-js`：hightlight.js 模块，提供对代码片段进行高亮化方法
- `nodeModules.hotkeys-js`：hotkeys-js 模块，提供快捷键辅助方法
- `nodeModules.htmlparser`：htmlparser 模块，提供 HTML 片段解析方法
- `nodeModules.marked`：marked 模块，提供 Markdown 辅助方法
- `nodeModules.md5`：md5 模块，提供 md5 算法方法
- `nodeModules.tiny-pinyin`：tiny-pinyin 模块，提供汉语拼音操作方法
- `platform`：访问平台提供的实用工具

扩展 API 权限

扩展 API 调用鉴权机制

在扩展模块调用 API 时会实时进行权限验证，如果当前用户所使用的扩展没有对应的 API 权限则调用 API 会失败。目前扩展在运行时最终可用的权限由如下环境共同决定：

- 客户端自身允许提供的 API 权限，一些特别定制的客户端可能只拥有有限的扩展 API 权限；
- 当前服务器授权中允许的扩展 API 权限，不同的授权版本可能具备的权限不一样，通常更高级版的授权拥有更多权限；
- 扩展自身在扩展描述文件中申请的权限。

权限定义格式

扩展 API 权限由一个列表定义（在代码中使用字符串数组），列表中每一项表示一个或一组权限，也可以用于排除部分权限，下面是可能的扩展权限定义：

- `权限完整名称`，例如 `user.getCurrentUser` 表示拥有获取当前用户信息权限;
- `权限名称前缀`，例如 `members` 表示拥有权限名称前缀为 `members.` 的所有权限，通常使用前缀来表示拥有对应模块下的所有权限;
- `@L权限等级`，例如 `@L2` 表示拥有等级为 2 或者等级小于 2 的所有权限；
- `@L权限等级!`，例如 `@L2!` 表示拥有仅等级为 2 所有权限，但没有等级小于 2 的权限；
- `-权限完整名称`，例如 `-members.getMember` 从权限清单中排除名称为 `members.getMember`（获取成员信息）的权限；
- `-权限名称前缀`，例如 `-members` 从权限清单中排除名称前缀为 `members.` 的所有权限；

在描述文件中申请权限

开发扩展时，扩展需要在描述文件中申明扩展模块中所需要的权限，如果没有申明权限，则扩展模块运行时可能出错。在扩展描述文件中申明权限的格式为：

```
{
    "name": "xext-my-extension",

    // 使用 usesPermissions 来在一个数组中申明权限
    "usesPermissions": [
        "user",
        "members",
        "@L3",
        "nodeModules"
    ],

    // ...此处省略扩展描述文件中的其他属性
}
```

API 定义

user： 用户信息访问相关

§ user.getCurrentUser

获取当前登录的用户信息。

基本信息

等级	1
类型	函数

函数返回值

- `User`： 当前登录的用户对象。

用法

```
// 通过 `xext.*` 调用:
const user = xext.user.getCurrentUser();

// 或通过 `require()` 调用:
// const {getCurrentUser} = require("xext/user");
// const user = getCurrentUser();
```

members： 通讯录成员信息访问相关

§ members.getDept

获取部门信息。

基本信息

等级	3
类型	函数

函数定义

- *function* **getDept**(deptID)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
deptID	`string`	必须		部门 ID。例如：`"18"`。

函数返回值

- `object`： 部门信息对象。

用法

```
const deptID = "18";
// 通过 `xext.*` 调用:
const dept = xext.members.getDept(deptID);

// 或通过 `require()` 调用:
// const {getDept} = require("xext/members");
// const dept = getDept(deptID);
```

§ members.getDeptsTree

获取部门结构树信息。

基本信息

等级	3
类型	函数

函数返回值

- `object[]`： 部门树结构数据。

用法

```
// 通过 `xext.*` 调用:
const deptsTree = xext.members.getDeptsTree();

// 或通过 `require()` 调用:
// const {getDeptsTree} = require("xext/members");
// const deptsTree = getDeptsTree();
```

§ members.getMember

根据账号或用户 ID 获取用户对象信息。

基本信息

等级	3
类型	函数

函数定义

- *function* **getMember**(memberID)
- *function* **getMember**(memberAccount)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
memberID	`number`	必须		
memberAccount	`string`	必须		

函数返回值

- `Member`：用户成员对象。

用法

```
const memberID = 42;
// 通过 `xext.*` 调用:
const member = xext.members.getMember(memberID);

// 或通过 `require()` 调用:
// const {getMember} = require("xext/members");
// const member = getMember(memberID);
```

§ members.getMembers

获取系统中所有用户账号或者指定过滤条件，以列表形式返回。

基本信息

等级	3
类型	函数

函数定义

- *function* **getMembers**(memberFilter)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
memberFilter	`function`	可选		用户成员筛选回调函数。例如：`member => (member.gender === "f");`。

函数返回值

- `Member[]`：成员列表。

用法

```
const memberFilter = member => (member.gender === "f");;
// 通过 `xext.*` 调用:
const members = xext.members.getMembers(memberFilter);

// 或通过 `require()` 调用:
// const {getMembers} = require("xext/members");
// const members = getMembers(memberFilter);
```

§ members.getRoleName

获取角色名称。

基本信息

等级	3
类型	函数

函数定义

- *function* **getRoleName**(roleID)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
roleID	`string`	必须		成员角色 ID。例如：`"projectManager"`。

函数返回值

- **`string`**：成员角色名称。例如：`"产品经理"`。

用法

```
const roleID = "projectManager";
// 通过 `xext.*` 调用:
const roleName = xext.members.getRoleName(roleID);

// 或通过 `require()` 调用:
// const {getRoleName} = require("xext/members");
// const roleName = getRoleName(roleID);
```

im： 聊天相关

§ im.createGroupChat

创建一个讨论组会话。

基本信息

等级	4
类型	函数

函数定义

- *function* **createGroupChat**(members, name)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
members	`number[]|Set<number>`	必须		成员列表。
name	`string`	可选		

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.createGroupChat(members, name);

// 或通过 `require()` 调用:
// const {createGroupChat} = require("xext/im");
// const promise = createGroupChat(members, name);
```

§ im.dismissChat

解散讨论组会话。

基本信息

等级	4
类型	函数

函数返回值

- `Promise`：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.dismissChat();

// 或通过 `require()` 调用:
// const {dismissChat} = require("xext/im");
// const promise = dismissChat();
```

§ im.exitChat

退出讨论组会话。

基本信息

等级	4
类型	函数

函数定义

- *function* **exitChat**(chatGid)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。

用法

```
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
// 通过 `xext.*` 调用:
xext.im.exitChat(chatGid);

// 或通过 `require()` 调用:
// const {exitChat} = require("xext/im");
// exitChat(chatGid);
```

§ im.fetchChatsHistory

请求获取会话聊天记录。

基本信息

等级	1
类型	函数

函数定义

- *function* **fetchChatsHistory**(pager, continued, startDate)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
pager	`object`	可选		
continued	`boolean`	可选		
startDate	`number`	可选		

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.fetchChatsHistory(pager, continued, startDate);

// 或通过 `require()` 调用:
// const {fetchChatsHistory} = require("xext/im");
// const promise = fetchChatsHistory(pager, continued, startDate);
```

§ im.fetchPublicChats

请求获取公开讨论组。

基本信息

等级	3
类型	函数

函数返回值

- **`Promise<Chat[]>`**：会话列表。该方法通过 Promise 异步返回结果。

用法

```
// 通过 `xext.*` 调用:
const chats = xext.im.fetchPublicChats();

// 或通过 `require()` 调用:
// const {fetchPublicChats} = require("xext/im");
// const chats = fetchPublicChats();
```

§ im.getChat

根据 gid 或 ID 获取会话。

基本信息

等级	3
类型	函数

函数定义

- *function* **getChat**(chatGid)
- *function* **getChat**(chatID)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	必须		
chatID	`number`	必须		

函数返回值

- **`Chat`**：会话对象。

用法

```
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
// 通过 `xext.*` 调用:
const chat = xext.im.getChat(chatGid);

// 或通过 `require()` 调用:
// const {getChat} = require("xext/im");
// const chat = getChat(chatGid);
```

§ im.getChatMessages

根据 gid 或 ID 获取会话消息列表（仅限在界面上显示的消息）。

基本信息

等级	3
类型	函数

函数定义

- *function* **getChatMessages**(chatGid)
- *function* **getChatMessages**(chatID)
- *function* **getChatMessages**(chatGid, chatMessageFilter)
- *function* **getChatMessages**(chatID, chatMessageFilter)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	必须		
chatID	`number`	必须		
chatMessageFilter	`function`	可选		

函数返回值

- `Promise<ChatMessage[]>` 该方法通过 Promise 异步返回结果。

用法

```
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
// 通过 `xext.*` 调用:
const chatMessages = xext.im.getChatMessages(chatGid);

// 或通过 `require()` 调用:
// const {getChatMessages} = require("xext/im");
// const chatMessages = getChatMessages(chatGid);
```

§ im.getChats

获取系统中所有会话或者指定过滤条件，以列表形式返回。

基本信息

等级	3
类型	函数

函数定义

- *function* **getChats**(chatFilter)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatFilter	`function`	可选		会话筛选回调函数。例如：`chat => (chat.type === "one2one");`。

函数返回值

- `Chat[]`：会话列表。

用法

```
const chatFilter = chat => (chat.type === "one2one");;
// 通过 `xext.*` 调用:
const chats = xext.im.getChats(chatFilter);

// 或通过 `require()` 调用:
// const {getChats} = require("xext/im");
// const chats = getChats(chatFilter);
```

§ im.inviteMembersToChat

邀请其他成员加入会话。

基本信息

等级	4
类型	函数

函数定义

- *function* **inviteMembersToChat**(chatGid, membersIDList)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
membersIDList	`number[]`	可选		

函数返回值

- `Promise`：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.inviteMembersToChat(chatGid, membersIDList);

// 或通过 `require()` 调用:
// const {inviteMembersToChat} = require("xext/im");
// const promise = inviteMembersToChat(chatGid, membersIDList);
```

§ im.joinChat

加入会话。

基本信息

等级	4
类型	函数

函数定义

- *function* **joinChat**(chatGid)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。

函数返回值

- `Promise`：异步返回操作结果。

用法

```
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
// 通过 `xext.*` 调用:
const promise = xext.im.joinChat(chatGid);

// 或通过 `require()` 调用:
// const {joinChat} = require("xext/im");
// const promise = joinChat(chatGid);
```

§ im.renameChat

重命名会话。

基本信息

等级	4
类型	函数

函数定义

- *function* **renameChat**(chatGid, newName)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
newName	`string`	可选		

函数返回值

- `Promise`：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用：
const promise = xext.im.renameChat(chatGid, newName);

// 或通过 `require()` 调用：
// const {renameChat} = require("xext/im");
// const promise = renameChat(chatGid, newName);
```

§ im.sendCodeMessage

请求发送代码类消息。

基本信息

等级	4
类型	函数

函数定义

- *function* **sendCodeMessage**(messageCodeContent, chatGid)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageCodeContent	`object`	必须		文件消息内容。例如：`{"code":"console.log(\"Hello, world!\");","name":"Hello World","lang":"js"}`。
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。

函数返回值

- `Promise`：异步返回操作结果。

用法

```
const messageCodeContent = {"code":"console.log(\"Hello, world!\");","name":"Hello World","lang":"js"};
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
// 通过 `xext.*` 调用：
const promise = xext.im.sendCodeMessage(messageCodeContent, chatGid);

// 或通过 `require()` 调用：
// const {sendCodeMessage} = require("xext/im");
// const promise = sendCodeMessage(messageCodeContent, chatGid);
```

§ im.sendContentToChatSendbox

向会话输入框添加内容。

基本信息

等级	2
类型	函数

函数定义

- *function* **sendContentToChatSendbox**(messageTextContent, messageContentType, chatGid, clearOldContent)
- *function* **sendContentToChatSendbox**(messageFileContent, messageContentType, chatGid, clearOldContent)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageTextContent	`string`	可选		
messageFileContent	`object`	可选		
chatGid	`string`	可选		
messageContentType	`string`	可选		
clearOldContent	`string`	可选		

用法

```
const messageTextContent = "@all 今天中午吃饺子！";
const messageContentType = "image";
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
const clearOldContent = false;
// 通过 `xext.*` 调用:
xext.im.sendContentToChatSendbox(messageTextContent, messageContentType, chatGid, clearOldContent);

// 或通过 `require()` 调用:
// const {sendContentToChatSendbox} = require("xext/im");
// sendContentToChatSendbox(messageTextContent, messageContentType, chatGid, clearOldContent);
```

§ im.sendEmojiMessage

向会话发送 Emoji 表情消息。

基本信息

等级	4
类型	函数

函数定义

- *function* **sendEmojiMessage**(messageEmojiContent, chatGid)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageEmojiContent	`string`	可选		
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.sendEmojiMessage(messageEmojiContent, chatGid);

// 或通过 `require()` 调用:
// const {sendEmojiMessage} = require("xext/im");
// const promise = sendEmojiMessage(messageEmojiContent, chatGid);
```

§ im.sendFileMessage

向会话发送文件消息。

基本信息

等级	4
类型	函数

函数定义

- *function* **sendFileMessage**(messageFileContent, chatGid, progressChangeCallback)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageFileContent	`string`	必须		文件消息内容。例如：`{"path":"/Users/xuan/image.png","name":"image.png"}`。
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
progressChangeCallback	`function`	可选		进度变更回调函数。例如：`(progress) => {console.log(进度变更 ${progress});}`。

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
const messageFileContent = {"path":"/Users/xuan/image.png","name":"image.png"};
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
const progressChangeCallback = (progress) => {console.log(`进度变更 ${progress}`);};
// 通过 `xext.*` 调用:
const promise = xext.im.sendFileMessage(messageFileContent, chatGid, progressChangeCallback);

// 或通过 `require()` 调用:
// const {sendFileMessage} = require("xext/im");
// const promise = sendFileMessage(messageFileContent, chatGid, progressChangeCallback);
```

§ im.sendImageMessage

向会话发送图片消息。

基本信息

等级	4
类型	函数

函数定义

- *function* **sendImageMessage**(messageImageContent, chatGid, progressChangeCallback)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageImageContent	`string`	可选		
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
progressChangeCallback	`function`	可选		进度变更回调函数。例如：`(progress) => {console.log(进度变更 ${progress});}`。

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.sendImageMessage(messageImageContent, chatGid, progressChangeCallback);

// 或通过 `require()` 调用:
// const {sendImageMessage} = require("xext/im");
// const promise = sendImageMessage(messageImageContent, chatGid, progressChangeCallback);
```

§ im.sendTextMessage

向会话发送文本消息。

基本信息

等级	4
类型	函数

函数定义

- *function* **sendTextMessage**(messageTextContent, chatGid, isMarkdown)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageTextContent	`string`	必须		文本消息内容。例如：`"@all 今天中午吃饺子！"`。
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
isMarkdown	`boolean`	可选		

函数返回值

- `Promise`：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用：
const promise = xext.im.sendTextMessage(messageTextContent, chatGid, isMarkdown);

// 或通过 `require()` 调用：
// const {sendTextMessage} = require("xext/im");
// const promise = sendTextMessage(messageTextContent, chatGid, isMarkdown);
```

§ im.shareContentToChat

请求分享内容到会话。

基本信息

等级	4
类型	函数

函数定义

- *function* **shareContentToChat**(messageTextContent, chatGid)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
messageTextContent	`string`	必须		文本消息内容。例如：`"@all 今天中午吃饺子！"`。
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。

函数返回值

- `Promise`：异步返回操作结果。

用法

```
const messageTextContent = "@all 今天中午吃饺子！";
const chatGid = "58113bb5-6a66-e94f-eb12-fb94271f6f7c";
// 通过 `xext.*` 调用：
const promise = xext.im.shareContentToChat(messageTextContent, chatGid);

// 或通过 `require()` 调用：
// const {shareContentToChat} = require("xext/im");
// const promise = shareContentToChat(messageTextContent, chatGid);
```

§ im.toggleFreezeChat

切换会话是否为从最近列表移除。

基本信息

等级	4
类型	函数

函数定义

- *function* **toggleFreezeChat**(chatGid, toggle)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
toggle	`boolean`	可选		切换标识。

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.toggleFreezeChat(chatGid, toggle);

// 或通过 `require()` 调用:
// const {toggleFreezeChat} = require("xext/im");
// const promise = toggleFreezeChat(chatGid, toggle);
```

§ im.toggleMuteChat

切换会话是否为开启免打扰。

基本信息

等级	4
类型	函数

函数定义

- *function* **toggleMuteChat**(chatGid, toggle)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
toggle	`boolean`	可选		切换标识。

函数返回值

- **`Promise`**：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用:
const promise = xext.im.toggleMuteChat(chatGid, toggle);

// 或通过 `require()` 调用:
// const {toggleMuteChat} = require("xext/im");
// const promise = toggleMuteChat(chatGid, toggle);
```

§ im.toggleStarChat

切换会话是否为收藏。

基本信息

等级	4
类型	函数

函数定义

- *function* **toggleStarChat**(chatGid, toggle)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
chatGid	`string`	可选		会话 GID。例如：`"58113bb5-6a66-e94f-eb12-fb94271f6f7c"`。
toggle	`boolean`	可选		切换标识。

函数返回值

- `Promise`：异步返回操作结果。

用法

```
// 通过 `xext.*` 调用：
const promise = xext.im.toggleStarChat(chatGid, toggle);

// 或通过 `require()` 调用：
// const {toggleStarChat} = require("xext/im");
// const promise = toggleStarChat(chatGid, toggle);
```

ext： 扩展相关

§ ext.MainView

扩展应用主界面视图组件（当前扩展类型为 app，且应用类型为 insideView 时有效）。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.getExtension

获取当前扩展对象。

基本信息

等级	0
类型	函数

函数返回值

- `any`

用法

```
// 通过 `xext.*` 调用：
const any = xext.ext.getExtension();

// 或通过 `require()` 调用：
// const {getExtension} = require("xext/ext");
// const any = getExtension();
```

§ ext.onAttach

允许绑定扩展生命周期函数，当扩展被加载后调用，此时可以对扩展进行初始化。

基本信息

等级	0
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onDetach

允许绑定扩展生命周期函数，当扩展被卸载时调用，此时应该将扩展使用的资源进行释放，例如销毁定时器等。

基本信息

等级	0
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onReady

允许绑定扩展生命周期函数，当界面加载完毕时调用，此时扩展可以处理与界面相关操作。

基本信息

等级	0
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onReceiveChatMessages

允许绑定扩展生命周期函数，当用户接收到聊天消息时调用。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onRenderChatMessageContent

允许绑定扩展生命周期函数，当在界面上需要转化 markdown 格式的消息文本为 html 时会调用此回调方法。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onRequestOpenApp

允许绑定扩展生命周期函数，当扩展类型为应用且应用类型为 `'custom'` 时，使用此函数来执行用户点击应用图标时的操作。

基本信息

等级	未定义
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onSendChatMessage

允许绑定扩展生命周期函数，当用户发送聊天消息时调用。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onUserLogin

允许绑定扩展生命周期函数，当用户登录时调用。

基本信息

等级	1
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onUserLogout

允许绑定扩展生命周期函数，当用户登录出调用。

基本信息

等级	1
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.onUserStatusChange

允许绑定扩展生命周期函数，当用户状态变更时调用。

基本信息

等级	1
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ ext.urlInspectors

允许提供网址解释器。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

notification： 通知相关

§ notification.requestAttention

请求在用户操作系统桌面提示窗口界面有新内容。

基本信息

等级	5
类型	函数

用法

```
// 通过 `xext.*` 调用:
xext.notification.requestAttention();

// 或通过 `require()` 调用:
// const {requestAttention} = require("xext/notification");
// requestAttention();
```

§ notification.sendLocalNotification

向通知中心发送一条本地通知。

基本信息

等级	5
类型	函数

用法

```
// 通过 `xext.*` 调用:
xext.notification.sendLocalNotification();

// 或通过 `require()` 调用:
// const {sendLocalNotification} = require("xext/notification");
// sendLocalNotification();
```

§ notification.setAppBadge

设置应用图标上的小红点提示。

基本信息

等级	5
类型	函数

用法

```
// 通过 `xext.*` 调用:
xext.notification.setAppBadge();

// 或通过 `require()` 调用:
// const {setAppBadge} = require("xext/notification");
// setAppBadge();
```

§ notification.showDesktopNotification

显示一条桌面弹窗通知。

基本信息

等级	5
类型	函数

用法

```
// 通过 `xext.*` 调用:
xext.notification.showDesktopNotification();

// 或通过 `require()` 调用:
// const {showDesktopNotification} = require("xext/notification");
// showDesktopNotification();
```

contextmenu：上下文菜单相关

§ contextmenu.addContextMenuCreator

添加上下文菜单。

基本信息

等级	5
类型	函数

函数定义

- *function* **addContextMenuCreator**(contextmenuName, contextmenuCreatorFunc)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
contextmenuName	`string`	必须		上下文菜单类型名称。例如：`"message.text"`。
contextmenuCreatorFunc	`function`	必须		上下文菜单生成器函数。例如：`(context) => [{label: "测试菜单项", url: "http://xuanim.com"}]`。

函数返回值

- **`string`**：上下文菜单生成器 ID。例如：`"0fxm381p"`。

用法

```
const contextmenuName = "message.text";
const contextmenuCreatorFunc = (context) => [{label: "测试菜单项", url: "http://xuanim.com"}];
// 通过 `xext.*` 调用:
const contextmenuCreatorID = xext.contextmenu.addContextMenuCreator(contextmenuName,
contextmenuCreatorFunc);

// 或通过 `require()` 调用:
// const {addContextMenuCreator} = require("xext/contextmenu");
// const contextmenuCreatorID = addContextMenuCreator(contextmenuName, contextmenuCreatorFunc);
```

§ contextmenu.addContextMenuCreator.chat.actions

添加会话操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.member

添加会话成员相关操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.menu

添加会话在会话列表上的菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.sendbox.sendButton

添加会话发送框发送按钮右键菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.sendbox.toolbar

添加会话发送框工具栏菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.sidebar.file

添加会话侧边栏文件列表操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.sidebar.member

添加会话侧边栏成员操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.toolbar

添加会话工具栏操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chat.toolbar.more

添加会话工具栏更多按钮操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chats.menu

添加会话列表类型切换菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.chats.menu.group

添加会话列表分组操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.ext.app

添加扩展应用操作菜单。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.ext.apps.navbar

添加应用在导航上的操作菜单。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.ext.extension

添加扩展操作菜单。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.ext.extensionsMenu

添加扩展列表上的操作菜单。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.ext.openedApp

添加已打开的扩展应用操作菜单。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.image

添加图片操作菜单。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.member

添加成员操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.member.profile

添加成员资料菜单。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.message

添加聊天消息菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.message.basic

添加聊天消息基本操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.message.code

添加代码类聊天消息基本操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.message.image

添加图片类聊天消息基本操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.message.text

添加文本类聊天消息基本操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.message.url

添加链接类聊天消息基本操作菜单。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.addContextMenuCreator.profile.menu

添加个人资料面板菜单。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ contextmenu.removeContextMenuCreator

移除添加的上下文菜单。

基本信息

等级	0
类型	函数

函数定义

- *function* **removeContextMenuCreator**(contextmenuCreatorID)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
contextmenuCreatorID	`string`	必须		上下文菜单生成器 ID。例如：`"0fxm381p"`。

函数返回值

- **`any`**：操作结果，如果为 `true` 则操作成功，否则为操作失败。例如：`true`。

用法

```
const contextmenuCreatorID = "0fxm381p";
// 通过 `xext.*` 调用:
const operationResult = xext.contextmenu.removeContextMenuCreator(contextmenuCreatorID);

// 或通过 `require()` 调用:
// const {removeContextMenuCreator} = require("xext/contextmenu");
// const operationResult = removeContextMenuCreator(contextmenuCreatorID);
```

§ contextmenu.showContextMenu

显示上下文菜单。

基本信息

等级	4
类型	函数

函数定义

- *function* **showContextMenu**(contextmenuName, contextmenuContext)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
contextmenuName	`string`	必须		上下文菜单类型名称。例如：`"message.text"`。
contextmenuContext	`object`	可选	`null`	上下文菜单创建时的上下文参数对象。例如：`{"myContextProp":"Hello"}`。

函数返回值

- `any`：操作结果，如果为 `true` 则操作成功，否则为操作失败。例如：`true`。

用法

```
const contextmenuName = "message.text";
const contextmenuContext = {"myContextProp":"Hello"};
// 通过 `xext.*` 调用:
const operationResult = xext.contextmenu.showContextMenu(contextmenuName, contextmenuContext);

// 或通过 `require()` 调用:
// const {showContextMenu} = require("xext/contextmenu");
// const operationResult = showContextMenu(contextmenuName, contextmenuContext);
```

commander： 命令相关

§ commander.executeCommand

执行命令。

基本信息

等级	6
类型	函数

函数定义

- *function* **executeCommand**(commandName, ...commandParams)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
commandName	`string`	必须		命令名称。例如：`"showMessageDetail"`。
commandParams	`...any`	可选		执行命令传递给命令回调函数的参数。例如：`["Hello World!"]`。

函数返回值

- `any`：命令执行完毕后的返回值。

用法

```
const commandName = "showMessageDetail";
const commandParams1 = "Hello World!";
// 通过 `xext.*` 调用:
const commandResult = xext.commander.executeCommand(commandName, commandParams1);

// 或通过 `require()` 调用:
// const {executeCommand} = require("xext/commander");
// const commandResult = executeCommand(commandName, commandParams1);
```

§ commander.executeCommand.closeDisplay

执行关闭弹出层命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.closeModal

执行关闭对话框命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.confirmJoinPublicChat

执行请求加入公开会话命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.mentionMemberInSendbox

执行在输入框"@成员"命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.openInApp

执行在扩展应用中打开链接命令。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.openUrlInBrowser

执行在浏览器中打开链接命令。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.openUrlInDialog

执行在对话框中打开链接命令。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.openWebviewDialog

执行在 Webview 对话框中打开链接命令。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.sendContentToChat

执行向输入框发送内容命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.setRoute

执行设置界面路由命令。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.shortcut.captureScreenHotkey

执行启动截图命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.shortcut.focusWindowHotkey

执行激活窗口命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatAddCategoryDialog

执行显示将会话添加到分组对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatCommittersSettingDialog

执行显示会话白名单对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatInviteDialog

执行显示邀请成员到会话对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatSendCodeDialog

执行显示发送代码对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatSendDialog

执行显示发送内容到会话对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatShareDialog

执行转发内容到会话对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showChatsHistoryDialog

执行显示会话历史记录对话框命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showConfirmSendFilesDialog

执行显示确认发送文件对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showCreateChatDialog

执行显示创建会话对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showEmojiPopover

执行显示 Emoji 选择面板命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showExtensionDialog

执行显示扩展详情对话框命令。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showExtensionInstallDialog

执行显示安装扩展对话框命令。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showHotkeySettingDialog

执行显示快捷键设置对话框命令。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showLanguageSwitchDialog

执行显示界面语言切换对话框命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showMemberProfile

执行显示用户资料面板命令。

基本信息

等级	4
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showMessager

执行显示提示消息命令。

基本信息

等级	5
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.showTodoEditDialog

执行显示待办编辑对话框命令。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.updateViewStyle

执行请求更新视图样式命令。

基本信息

等级	6
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommand.viewImage

执行显示查看图片命令。

基本信息

等级	2
类型	托管的接口

提示：此 API 由系统托管，系统托管的 API 无法在扩展模块中直接调用，由系统在合适的实际调用；如果扩展没有此 API 权限，系统将无法调用此 API。

§ commander.executeCommandLine

执行命令串。

基本信息

等级	6
类型	函数

函数定义

- *function* **executeCommandLine**(commandLine, commandContext)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
commandLine	`string`	可选		
commandContext	`object`	可选	`null`	命令执行时的上下文参数对象。例如：`{"myContextProp":"Hello"}`。

函数返回值

- **`any`**：命令执行完毕后的返回值。

用法

```
// 通过 `xext.*` 调用：
const commandResult = xext.commander.executeCommandLine(commandLine, commandContext);

// 或通过 `require()` 调用：
// const {executeCommandLine} = require("xext/commander");
// const commandResult = executeCommandLine(commandLine, commandContext);
```

§ commander.executeCommandWithContext

执行包含上下文参数的命令。

基本信息

等级	6
类型	函数

函数定义

- *function* **executeCommandWithContext**(commandName, commandContext, ...commandParams)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
commandName	`string`	必须		命令名称。例如：`"showMessageDetail"`。
commandContext	`object`	可选	`null`	命令执行时的上下文参数对象。例如：`{"myContextProp":"Hello"}`。
commandParams	`...any`	可选		执行命令传递给命令回调函数的参数。例如：`["Hello World!"]`。

函数返回值

- **`boolean`**：操作结果，如果为 `true` 则操作成功，否则为操作失败。例如：`true`。

用法

```
const commandName = "showMessageDetail";
const commandContext = {"myContextProp":"Hello"};
const commandParams1 = "Hello World!";
// 通过 `xext.*` 调用：
const operationResult = xext.commander.executeCommandWithContext(commandName, commandContext,
commandParams1);

// 或通过 `require()` 调用：
// const {executeCommandWithContext} = require("xext/commander");
// const operationResult = executeCommandWithContext(commandName, commandContext, commandParams1);
```

§ commander.registerCommand

注册命令。

基本信息

等级	5
类型	函数

函数定义

- *function* **registerCommand**(commandName, commandCallback, commandContext)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
commandName	`string`	必须		命令名称。例如：`"showMessageDetail"`。
commandCallback	`function`	必须		执行命令时调用的回调函数。例如：`(message) => alert(message);`。
commandContext	`object`	可选	`null`	命令执行时的上下文参数对象。例如：`{"myContextProp":"Hello"}`。

函数返回值

- **`boolean`**：操作结果，如果为 `true` 则操作成功，否则为操作失败。例如：`true`。

用法

```
const commandName = "showMessageDetail";
const commandCallback = (message) => alert(message);;
const commandContext = {"myContextProp":"Hello"};
// 通过 `xext.*` 调用：
const operationResult = xext.commander.registerCommand(commandName, commandCallback, commandContext);

// 或通过 `require()` 调用：
// const {registerCommand} = require("xext/commander");
// const operationResult = registerCommand(commandName, commandCallback, commandContext);
```

§ commander.unregisterCommand

取消注册的命令。

基本信息

等级	0
类型	函数

函数定义

- *function* **unregisterCommand**(commandName)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
commandName	`string`	必须		命令名称。例如：`"showMessageDetail"`。

函数返回值

- **`boolean`**：操作结果，如果为 `true` 则操作成功，否则为操作失败。例如：`true`。

用法

```
const commandName = "showMessageDetail";
// 通过 `xext.*` 调用：
const operationResult = xext.commander.unregisterCommand(commandName);

// 或通过 `require()` 调用：
// const {unregisterCommand} = require("xext/commander");
// const operationResult = unregisterCommand(commandName);
```

components： 组件相关

§ components.AppAvatar

应用图标组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
avatar	`object`	可选	`null`	
label	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	
badge	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const AppAvatar = require('components/AppAvatar');

// 或通过 `xext.*` 调用:
// const AppAvatar = xext.components.AppAvatar;

// 使用组件:
const render = () => {
    return (
        <AppAvatar
            avatar={avatar}
            label={label}
            className={className}
            badge={badge}
        >
            {children}
        </AppAvatar>
    );
}
```

§ components.AreaSelector

区域选择组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
onSelectArea	`object`	可选	`null`	
toolbarStyle	`object`	可选	`null`	
toolbarHeight	`number`	可选	`40`	
style	`object`	可选	`null`	
toolbar	`object`	可选	`null`	
img	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
```

```
const AreaSelector = require('components/AreaSelector');

// 或通过 `xext.*` 调用:
// const AreaSelector = xext.components.AreaSelector;

// 使用组件:
const render = () => {
    return (
        <AreaSelector
            onSelectArea={onSelectArea}
            toolbarStyle={toolbarStyle}
            toolbarHeight={40}
            style={style}
            toolbar={toolbar}
            img={img}
        />
    );
}
```

§ components.Avatar

头像组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
auto	`object`	可选	`null`	
skin	`object`	可选	`null`	
image	`object`	可选	`null`	
icon	`object`	可选	`null`	
label	`object`	可选	`null`	
size	`object`	可选	`null`	
iconSize	`object`	可选	`null`	
className	`object`	可选	`null`	
foreColor	`object`	可选	`null`	
imageClassName	`object`	可选	`null`	
iconClassName	`object`	可选	`null`	
style	`object`	可选	`null`	
children	`object`	可选	`null`	
badge	`object`	可选	`null`	
leftstyle	`object`	可选	`null`	
rightstyle	`object`	可选	`null`	
imageErrorView	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Avatar = require('components/Avatar');

// 或通过 `xext.*` 调用:
// const Avatar = xext.components.Avatar;
```

```
// 使用组件:
const render = () => {
    return (
        <Avatar
            auto={auto}
            skin={skin}
            image={image}
            icon={icon}
            label={label}
            size={size}
            iconSize={iconSize}
            className={className}
            foreColor={foreColor}
            imageClassName={imageClassName}
            iconClassName={iconClassName}
            style={style}
            badge={badge}
            leftstyle={leftstyle}
            rightstyle={rightstyle}
            imageErrorView={imageErrorView}
        >
            {children}
        </Avatar>
    );
}
```

§ components.Button

按钮组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
skin	`object`	可选	`null`	
icon	`object`	可选	`null`	
label	`object`	可选	`null`	
className	`object`	可选	`null`	
style	`object`	可选	`null`	
children	`object`	可选	`null`	
btnClass	`string`	可选	`"btn"`	
type	`string`	可选	`"button"`	
disabled	`boolean`	可选	`false`	

用法

```
// 通过 `require()` 调用:
const Button = require('components/Button');

// 或通过 `xext.*` 调用:
// const Button = xext.components.Button;

// 使用组件:
const render = () => {
    return (
        <Button
            skin={skin}
            icon={icon}
```

```
            label={label}
            className={className}
            style={style}
            btnClass="btn"
            type="button"
            disabled={disabled}
        >
            {children}
        </Button>
    );
}
```

§ components.Checkbox

复选框组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
checked	`boolean`	可选	`false`	
label	`object`	可选	`null`	
className	`object`	可选	`null`	
inputProps	`object`	可选	`null`	
onChange	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Checkbox = require('components/Checkbox');

// 或通过 `xext.*` 调用:
// const Checkbox = xext.components.Checkbox;

// 使用组件:
const render = () => {
    return (
        <Checkbox
            checked={checked}
            label={label}
            className={className}
            inputProps={inputProps}
            onChange={onChange}
        >
            {children}
        </Checkbox>
    );
}
```

§ components.ClickOutsideWrapper

监听点击外部的辅助组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
onClickOutside	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ClickOutsideWrapper = require('components/ClickOutsideWrapper');

// 或通过 `xext.*` 调用:
// const ClickOutsideWrapper = xext.components.ClickOutsideWrapper;

// 使用组件:
const render = () => {
    return (
        <ClickOutsideWrapper
            onClickOutside={onClickOutside}
        >
            {children}
        </ClickOutsideWrapper>
    );
}
```

§ components.ContextMenu

上下文菜单组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const ContextMenu = require('components/ContextMenu');

// 或通过 `xext.*` 调用:
// const ContextMenu = xext.components.ContextMenu;

// 使用组件:
const render = () => {
    return <ContextMenu />;
}
```

§ components.Display

弹出层管理对象。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const Display = require('components/Display');

// 或通过 `xext.*` 调用:
// const Display = xext.components.Display;

// 使用组件:
const render = () => {
    return <Display />;
}
```

§ components.DisplayContainer

弹出层容器组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const DisplayContainer = require('components/DisplayContainer');

// 或通过 `xext.*` 调用:
// const DisplayContainer = xext.components.DisplayContainer;

// 使用组件:
const render = () => {
    return <DisplayContainer />;
}
```

§ components.DisplayLayer

弹出层组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
content	`string`	可选	`""`	
contentLoadFail	`object`	可选	`null`	
id	`object`	可选	`null`	
animation	`string`	可选	`"scale-from-top"`	
onShown	`object`	可选	`null`	
onHidden	`object`	可选	`null`	
onLoad	`object`	可选	`null`	
listenUpdateStyle	`boolean`	可选	`false`	
show	`boolean`	可选	`true`	
hotkey	`boolean`	可选	`true`	
cache	`boolean`	可选	`false`	
loadingContent	`boolean`	可选	`true`	
rootClassName	`string`	可选	`""`	
className	`string`	可选	`"layer"`	
backdrop	`boolean`	可选	`true`	
backdropClassName	`string`	可选	`""`	
contentClassName	`string`	可选	`""`	
footer	`object`	可选	`null`	
header	`object`	可选	`null`	
plugName	`object`	可选	`null`	
modal	`boolean`	可选	`false`	
children	`object`	可选	`null`	
style	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const DisplayLayer = require('components/DisplayLayer');

// 或通过 `xext.*` 调用:
// const DisplayLayer = xext.components.DisplayLayer;

// 使用组件:
const render = () => {
    return (
        <DisplayLayer
            content={content}
            contentLoadFail={contentLoadFail}
            id={id}
            animation="scale-from-top"
            onShown={onShown}
            onHidden={onHidden}
            onLoad={onLoad}
            listenUpdateStyle={listenUpdateStyle}
            show={show}
            hotkey={hotkey}
            cache={cache}
            loadingContent={loadingContent}
            rootClassName={rootClassName}
            className="layer"
            backdrop={backdrop}
            backdropClassName={backdropClassName}
            contentClassName={contentClassName}
            footer={footer}
            header={header}
```

```
            plugName={plugName}
            modal={modal}
            style={style}
        >
            {children}
        </DisplayLayer>
    );
}
```

§ components.Emoji

Emoji 组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const Emoji = require('components/Emoji');

// 或通过 `xext.*` 调用:
// const Emoji = xext.components.Emoji;

// 使用组件:
const render = () => {
    return <Emoji />;
}
```

§ components.HotkeyInputControl

快捷键设置组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
defaultValue	`string`	可选	`""`	
className	`object`	可选	`null`	
onChange	`object`	可选	`null`	
inputProps	`object`	可选	`null`	
onlyMotifyKeysText	`string`	可选	`""`	

用法

```
// 通过 `require()` 调用:
const HotkeyInputControl = require('components/HotkeyInputControl');

// 或通过 `xext.*` 调用:
// const HotkeyInputControl = xext.components.HotkeyInputControl;

// 使用组件:
const render = () => {
    return (
        <HotkeyInputControl
```

```
            defaultValue={defaultValue}
            className={className}
            onChange={onChange}
            inputProps={inputProps}
            onlyMotifyKeysText={onlyMotifyKeysText}
        />
    );
}
```

§ components.Icon

图标组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
size	`number`	可选	`0`	
style	`object`	可选	`null`	
square	`boolean`	可选	`true`	
className	`string`	可选	`""`	
color	`string`	可选	`""`	
name	`string`	可选	`""`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Icon = require('components/Icon');

// 或通过 `xext.*` 调用:
// const Icon = xext.components.Icon;

// 使用组件:
const render = () => {
    return (
        <Icon
            size={size}
            style={style}
            square={square}
            className={className}
            color={color}
            name={name}
        >
            {children}
        </Icon>
    );
}
```

§ components.ImageCutter

图片剪切组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
sourceImage	`object`	可选	`null`	
style	`object`	可选	`null`	
onFinish	`object`	可选	`null`	
onCancel	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ImageCutter = require('components/ImageCutter');

// 或通过 `xext.*` 调用:
// const ImageCutter = xext.components.ImageCutter;

// 使用组件:
const render = () => {
    return (
        <ImageCutter
            sourceImage={sourceImage}
            style={style}
            onFinish={onFinish}
            onCancel={onCancel}
        />
    );
}
```

§ components.ImageViewer

图片查看组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const ImageViewer = require('components/ImageViewer');

// 或通过 `xext.*` 调用:
// const ImageViewer = xext.components.ImageViewer;

// 使用组件:
const render = () => {
    return <ImageViewer />;
}
```

§ components.InputControl

输入框组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
value	`string`	可选	`""`	
defaultValue	`undefined`	可选		
label	`string`	可选	`" "`	
className	`string`	可选	`""`	
placeholder	`string`	可选	`""`	
autoFocus	`boolean`	可选	`false`	
style	`object`	可选	`null`	
labelStyle	`object`	可选	`null`	
inputType	`string`	可选	`"text"`	
inputStyle	`object`	可选	`null`	
inputProps	`object`	可选	`null`	
helpText	`object`	可选	`null`	
onChange	`object`	可选	`null`	
disabled	`boolean`	可选	`false`	
inputClassName	`string`	可选	`"rounded"`	
children	`object`	可选	`null`	
name	`string`	可选	`""`	
hotkeyScope	`object`	可选	`null`	
hotKeys	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const InputControl = require('components/InputControl');

// 或通过 `xext.*` 调用:
// const InputControl = xext.components.InputControl;

// 使用组件:
const render = () => {
    return (
        <InputControl
            value={value}
            defaultValue={defaultValue}
            label=" "
            className={className}
            placeholder={placeholder}
            autoFocus={autoFocus}
            style={style}
            labelStyle={labelStyle}
            inputType="text"
            inputStyle={inputStyle}
            inputProps={inputProps}
            helpText={helpText}
            onChange={onChange}
            disabled={disabled}
            inputClassName="rounded"
            name={name}
            hotkeyScope={hotkeyScope}
            hotKeys={hotKeys}
        >
            {children}
        </InputControl>
    );
}
```

§ components.Messager

提示消息组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const Messager = require('components/Messager');

// 或通过 `xext.*` 调用:
// const Messager = xext.components.Messager;

// 使用组件:
const render = () => {
    return <Messager />;
}
```

§ components.Modal

对话框组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const Modal = require('components/Modal');

// 或通过 `xext.*` 调用:
// const Modal = xext.components.Modal;

// 使用组件:
const render = () => {
    return <Modal />;
}
```

§ components.Pager

分页组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
page	`number`	可选	`1`	
recTotal	`number`	可选	`0`	
recPerPage	`number`	可选	`20`	
pageRecCount	`number`	可选	`0`	
className	`object`	可选	`null`	
onPageChange	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Pager = require('components/Pager');

// 或通过 `xext.*` 调用:
// const Pager = xext.components.Pager;

// 使用组件:
const render = () => {
    return (
        <Pager
            page={1}
            recTotal={recTotal}
            recPerPage={20}
            pageRecCount={pageRecCount}
            className={className}
            onPageChange={onPageChange}
        />
    );
}
```

§ components.Popover

弹出面板组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const Popover = require('components/Popover');

// 或通过 `xext.*` 调用:
// const Popover = xext.components.Popover;

// 使用组件:
const render = () => {
    return <Popover />;
}
```

§ components.SearchControl

搜索组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
placeholder	`object`	可选	`null`	
changeDelay	`number`	可选	`100`	
onSearchChange	`object`	可选	`null`	
onBlur	`object`	可选	`null`	
onFocus	`object`	可选	`null`	
onFocusChange	`object`	可选	`null`	
defaultValue	`string`	可选	`""`	
children	`object`	可选	`null`	
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const SearchControl = require('components/SearchControl');

// 或通过 `xext.*` 调用:
// const SearchControl = xext.components.SearchControl;

// 使用组件:
const render = () => {
    return (
        <SearchControl
            placeholder={placeholder}
            changeDelay={100}
            onSearchChange={onSearchChange}
            onBlur={onBlur}
            onFocus={onFocus}
            onFocusChange={onFocusChange}
            defaultValue={defaultValue}
            className={className}
        >
            {children}
        </SearchControl>
    );
}
```

§ components.SelectBox

选择框组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
value	`string`	可选	`""`	
onChange	`object`	可选	`null`	
onFocus	`object`	可选	`null`	
onBlur	`object`	可选	`null`	
children	`object`	可选	`null`	
selectProps	`object`	可选	`null`	
className	`object`	可选	`null`	
selectClassName	`object`	可选	`null`	
options	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const SelectBox = require('components/SelectBox');

// 或通过 `xext.*` 调用:
// const SelectBox = xext.components.SelectBox;

// 使用组件:
const render = () => {
    return (
        <SelectBox
            value={value}
            onChange={onChange}
            onFocus={onFocus}
            onBlur={onBlur}
            selectProps={selectProps}
            className={className}
            selectClassName={selectClassName}
            options={options}
        >
            {children}
        </SelectBox>
    );
}
```

§ components.Spinner

显示加载中动画组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
iconSize	`number`	可选	`24`	
iconClassName	`string`	可选	`"spin text-gray inline-block"`	
iconName	`string`	可选	`"loading"`	
label	`string`	可选	`""`	
className	`string`	可选	`""`	
children	`object`	可选	`null`	
labelClassName	`string`	可选	`""`	

用法

```
// 通过 `require()` 调用:
const Spinner = require('components/Spinner');

// 或通过 `xext.*` 调用:
// const Spinner = xext.components.Spinner;

// 使用组件:
const render = () => {
    return (
        <Spinner
            iconSize={24}
            iconClassName="spin text-gray inline-block"
            iconName="loading"
            label={label}
            className={className}
            labelClassName={labelClassName}
        >
            {children}
        </Spinner>
    );
}
```

§ components.TabPane

标签页面板组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
label	`string`	可选	`"tab"`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const TabPane = require('components/TabPane');

// 或通过 `xext.*` 调用:
// const TabPane = xext.components.TabPane;

// 使用组件:
const render = () => {
    return (
        <TabPane
            label="tab"
        >
            {children}
        </TabPane>
    );
}
```

§ components.Tabs

标签页组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
navClassName	`string`	可选	`""`	
activeClassName	`string`	可选	`"active"`	
tabPaneClass	`string`	可选	`""`	
contentClassName	`string`	可选	`"active"`	
className	`string`	可选	`""`	
children	`object`	可选	`null`	
cache	`boolean`	可选	`false`	
defaultActivePaneKey	`object`	可选	`null`	
onPaneChange	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Tabs = require('components/Tabs');

// 或通过 `xext.*` 调用:
// const Tabs = xext.components.Tabs;

// 使用组件:
const render = () => {
    return (
        <Tabs
            navClassName={navClassName}
            activeClassName="active"
            tabPaneClass={tabPaneClass}
            contentClassName="active"
            className={className}
            cache={cache}
            defaultActivePaneKey={defaultActivePaneKey}
            onPaneChange={onPaneChange}
        >
            {children}
        </Tabs>
    );
}
```

views： 主界面视图相关

§ views.chats

主界面上聊天相关组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
ChatAvatar	`class`	
ChatTipPopover	`object`	例如：`{"default":{}}`。
ChatChangeFontPopover	`object`	例如：`{"default":{}}`。
ChatTitle	`class`	
ChatCommittersSettingDialog	`object`	例如：`{"default":{}}`。
ChatView	`class`	
ChatCommittersSetting	`class`	
ChatsCache	`class`	
ChatCreateDialog	`object`	例如：`{"default":{}}`。
ChatsDndContainer	`class`	
ChatCreateGroups	`class`	
ChatsHistoryDialog	`object`	例如：`{"default":{}}`。
ChatsHistory	`class`	
ChatHeader	`class`	
ChatHistory	`class`	
Index	`class`	
ChatInviteDialog	`object`	例如：`{"default":{}}`。
MenuSearchList	`class`	
MenuList	`class`	
ChatJoinPublic	`class`	
Menu	`class`	
ChatListItem	`class`	
MessageBroadcast	`class`	
ChatMessages	`class`	
MessageContentFile	`class`	
ChatSearchResult	`class`	
MessageContentImage	`class`	
ChatSendbox	`class`	
MessageContentText	`class`	
ChatSidebarFiles	`class`	
MessageDivider	`class`	
ChatSidebarPeoples	`class`	
MessageListItem	`class`	
ChatSidebarProfile	`class`	
MessageList	`class`	
ChatSidebar	`class`	
MessagesPreviewDialog	`object`	例如：`{"default":{}}`。
ChatShareDialog	`object`	例如：`{"default":{}}`。

用法

```
// 通过 `require()` 调用:
const {ChatAvatar, ChatTipPopover, ChatChangeFontPopover, ChatTitle, ChatCommittersSettingDialog, ChatView,
ChatCommittersSetting, ChatsCache, ChatCreateDialog, ChatsDndContainer, ChatCreateGroups,
ChatsHistoryDialog, ChatsHistory, ChatHeader, ChatHistory, Index, ChatInviteDialog, MenuSearchList,
MenuList, ChatJoinPublic, Menu, ChatListItem, MessageBroadcast, ChatMessages, MessageContentFile,
ChatSearchResult, MessageContentImage, ChatSendbox, MessageContentText, ChatSidebarFiles, MessageDivider,
ChatSidebarPeoples, MessageListItem, ChatSidebarProfile, MessageList, ChatSidebar, MessagesPreviewDialog,
ChatShareDialog} = require('views/chats');

// 或者通过 `xext.*` 调用:
// const {ChatAvatar, ChatTipPopover, ChatChangeFontPopover, ChatTitle, ChatCommittersSettingDialog,
ChatView, ChatCommittersSetting, ChatsCache, ChatCreateDialog, ChatsDndContainer, ChatCreateGroups,
ChatsHistoryDialog, ChatsHistory, ChatHeader, ChatHistory, Index, ChatInviteDialog, MenuSearchList,
MenuList, ChatJoinPublic, Menu, ChatListItem, MessageBroadcast, ChatMessages, MessageContentFile,
ChatSearchResult, MessageContentImage, ChatSendbox, MessageContentText, ChatSidebarFiles, MessageDivider,
ChatSidebarPeoples, MessageListItem, ChatSidebarProfile, MessageList, ChatSidebar, MessagesPreviewDialog,
ChatShareDialog} = xext.views.chats;
```

§ views.chats.ChatAvatar

会话头像组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
chat	`object`	可选	`null`	
grayOffline	`boolean`	可选	`false`	
className	`object`	可选	`null`	
avatarSize	`object`	可选	`null`	
iconSize	`object`	可选	`null`	
avatarClassName	`object`	可选	`null`	
iconClassName	`object`	可选	`null`	
showNoticeBadge	`boolean`	可选	`true`	
showStar	`boolean`	可选	`false`	

用法

```
// 通过 `require()` 调用:
const ChatAvatar = require('views/chats/ChatAvatar');

// 或通过 `xext.*` 调用:
// const ChatAvatar = xext.views.chats.ChatAvatar;

// 使用组件:
const render = () => {
    return (
        <ChatAvatar
            chat={chat}
            grayOffline={grayOffline}
            className={className}
            avatarSize={avatarSize}
            iconSize={iconSize}
            avatarClassName={avatarClassName}
            iconClassName={iconClassName}
            showNoticeBadge={showNoticeBadge}
            showStar={showStar}
        />
```

```
    );
}
```

§ views.chats.ChatChangeFontPopover

会话消息字体大小设置面板功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
ChangeFontView	`class`	
showChangeFontPopover	`function`	

用法

```
// 通过 `require()` 调用:
const {ChangeFontView, showChangeFontPopover} = require('views/chats/ChatChangeFontPopover');

// 或者通过 `xext.*` 调用:
// const {ChangeFontView, showChangeFontPopover} = xext.views.chats.ChatChangeFontPopover;
```

§ views.chats.ChatCommittersSetting

会话白名单设置界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
chat	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatCommittersSetting = require('views/chats/ChatCommittersSetting');

// 或通过 `xext.*` 调用:
// const ChatCommittersSetting = xext.views.chats.ChatCommittersSetting;

// 使用组件:
const render = () => {
    return (
        <ChatCommittersSetting
            chat={chat}
            className={className}
        >
            {children}
        </ChatCommittersSetting>
    );
}
```

§ views.chats.ChatCommittersSettingDialog

会话白名单设置对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showChatCommittersSettingDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showChatCommittersSettingDialog} = require('views/chats/ChatCommittersSettingDialog');

// 或者通过 `xext.*` 调用:
// const {showChatCommittersSettingDialog} = xext.views.chats.ChatCommittersSettingDialog;
```

§ views.chats.ChatCreateDialog

新建会话对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showCreateChatDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showCreateChatDialog} = require('views/chats/ChatCreateDialog');

// 或者通过 `xext.*` 调用:
// const {showCreateChatDialog} = xext.views.chats.ChatCreateDialog;
```

§ views.chats.ChatCreateGroups

创建讨论组会话组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
onRequestClose	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatCreateGroups = require('views/chats/ChatCreateGroups');
```

```
// 或通过 `xext.*` 调用:
// const ChatCreateGroups = xext.views.chats.ChatCreateGroups;

// 使用组件:
const render = () => {
    return (
        <ChatCreateGroups
            onRequestClose={onRequestClose}
            className={className}
        >
            {children}
        </ChatCreateGroups>
    );
}
```

§ views.chats.ChatHeader

会话界面头部组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
chat	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	
showSidebarIcon	`string`	可选	`"auto"`	

用法

```
// 通过 `require()` 调用:
const ChatHeader = require('views/chats/ChatHeader');

// 或通过 `xext.*` 调用:
// const ChatHeader = xext.views.chats.ChatHeader;

// 使用组件:
const render = () => {
    return (
        <ChatHeader
            chat={chat}
            className={className}
            showSidebarIcon="auto"
        >
            {children}
        </ChatHeader>
    );
}
```

§ views.chats.ChatHistory

会话历史记录查看组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
chat	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	
gotoMessage	`object`	可选	`null`	
searchKeys	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatHistory = require('views/chats/ChatHistory');

// 或通过 `xext.*` 调用:
// const ChatHistory = xext.views.chats.ChatHistory;

// 使用组件:
const render = () => {
    return (
        <ChatHistory
            chat={chat}
            className={className}
            gotoMessage={gotoMessage}
            searchKeys={searchKeys}
        >
            {children}
        </ChatHistory>
    );
}
```

§ views.chats.ChatInviteDialog

邀请加入会话对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showChatInviteDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showChatInviteDialog} = require('views/chats/ChatInviteDialog');

// 或者通过 `xext.*` 调用:
// const {showChatInviteDialog} = xext.views.chats.ChatInviteDialog;
```

§ views.chats.ChatJoinPublic

加入公开讨论组界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
children	`object`	可选	`null`	
onRequestClose	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatJoinPublic = require('views/chats/ChatJoinPublic');

// 或通过 `xext.*` 调用:
// const ChatJoinPublic = xext.views.chats.ChatJoinPublic;

// 使用组件:
const render = () => {
    return (
        <ChatJoinPublic
            className={className}
            onRequestClose={onRequestClose}
        >
            {children}
        </ChatJoinPublic>
    );
}
```

§ views.chats.ChatListItem

会话列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
children	`object`	可选	`null`	
chat	`object`	可选	`null`	
badge	`object`	可选	`null`	
subname	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatListItem = require('views/chats/ChatListItem');

// 或通过 `xext.*` 调用:
// const ChatListItem = xext.views.chats.ChatListItem;

// 使用组件:
const render = () => {
    return (
        <ChatListItem
            className={className}
            chat={chat}
            badge={badge}
            subname={subname}
        >
            {children}
        </ChatListItem>
    );
```

```
}
```

§ views.chats.ChatMessages

会话消息列表组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatMessages = require('views/chats/ChatMessages');

// 或通过 `xext.*` 调用:
// const ChatMessages = xext.views.chats.ChatMessages;

// 使用组件:
const render = () => {
    return (
        <ChatMessages
            className={className}
            chat={chat}
        />
    );
}
```

§ views.chats.ChatSearchResult

会话搜索结果组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
children	`object`	可选	`null`	
chat	`object`	可选	`null`	
searchKeys	`object`	可选	`null`	
searchCount	`number`	可选	`0`	
searchFilterTime	`number`	可选	`0`	
requestGoto	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
```

```
const ChatSearchResult = require('views/chats/ChatSearchResult');

// 或通过 `xext.*` 调用:
// const ChatSearchResult = xext.views.chats.ChatSearchResult;

// 使用组件:
const render = () => {
    return (
        <ChatSearchResult
            className={className}
            chat={chat}
            searchKeys={searchKeys}
            searchCount={searchCount}
            searchFilterTime={searchFilterTime}
            requestGoto={requestGoto}
        >
            {children}
        </ChatSearchResult>
    );
}
```

§ views.chats.ChatSendbox

会话消息输入发送框。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatSendbox = require('views/chats/ChatSendbox');

// 或通过 `xext.*` 调用:
// const ChatSendbox = xext.views.chats.ChatSendbox;

// 使用组件:
const render = () => {
    return (
        <ChatSendbox
            className={className}
            chat={chat}
        />
    );
}
```

§ views.chats.ChatShareDialog

分享到会话对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showChatShareDialog	`function`	
showChatSendDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showChatShareDialog, showChatSendDialog} = require('views/chats/ChatShareDialog');

// 或者通过 `xext.*` 调用:
// const {showChatShareDialog, showChatSendDialog} = xext.views.chats.ChatShareDialog;
```

§ views.chats.ChatSidebar

会话侧边栏组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	
children	`object`	可选	`null`	
closeButton	`boolean`	可选	`true`	

用法

```
// 通过 `require()` 调用:
const ChatSidebar = require('views/chats/ChatSidebar');

// 或通过 `xext.*` 调用:
// const ChatSidebar = xext.views.chats.ChatSidebar;

// 使用组件:
const render = () => {
    return (
        <ChatSidebar
            className={className}
            chat={chat}
            closeButton={closeButton}
        >
            {children}
        </ChatSidebar>
    );
}
```

§ views.chats.ChatSidebarFiles

会话文件侧边栏组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatSidebarFiles = require('views/chats/ChatSidebarFiles');

// 或通过 `xext.*` 调用:
// const ChatSidebarFiles = xext.views.chats.ChatSidebarFiles;

// 使用组件:
const render = () => {
    return (
        <ChatSidebarFiles
            className={className}
            chat={chat}
        >
            {children}
        </ChatSidebarFiles>
    );
}
```

§ views.chats.ChatSidebarPeoples

会话成员侧边栏组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatSidebarPeoples = require('views/chats/ChatSidebarPeoples');

// 或通过 `xext.*` 调用:
// const ChatSidebarPeoples = xext.views.chats.ChatSidebarPeoples;

// 使用组件:
const render = () => {
    return (
        <ChatSidebarPeoples
            className={className}
            chat={chat}
        >
            {children}
        </ChatSidebarPeoples>
    );
}
```

§ views.chats.ChatSidebarProfile

会话侧边栏个人资料界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatSidebarProfile = require('views/chats/ChatSidebarProfile');

// 或通过 `xext.*` 调用:
// const ChatSidebarProfile = xext.views.chats.ChatSidebarProfile;

// 使用组件:
const render = () => {
    return (
        <ChatSidebarProfile
            className={className}
            chat={chat}
        >
            {children}
        </ChatSidebarProfile>
    );
}
```

§ views.chats.ChatTipPopover

会话发送框小提示功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showChatTipPopover	`function`	

用法

```
// 通过 `require()` 调用:
const {showChatTipPopover} = require('views/chats/ChatTipPopover');

// 或者通过 `xext.*` 调用:
// const {showChatTipPopover} = xext.views.chats.ChatTipPopover;
```

§ views.chats.ChatTitle

会话标题组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatTitle = require('views/chats/ChatTitle');

// 或通过 `xext.*` 调用:
// const ChatTitle = xext.views.chats.ChatTitle;

// 使用组件:
const render = () => {
    return (
        <ChatTitle
            className={className}
            chat={chat}
        >
            {children}
        </ChatTitle>
    );
}
```

§ views.chats.ChatView

会话组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chatGid	`object`	可选	`null`	
children	`object`	可选	`null`	
hidden	`boolean`	可选	`false`	

用法

```
// 通过 `require()` 调用:
const ChatView = require('views/chats/ChatView');

// 或通过 `xext.*` 调用:
// const ChatView = xext.views.chats.ChatView;

// 使用组件:
const render = () => {
    return (
        <ChatView
            className={className}
            chatGid={chatGid}
            hidden={hidden}
        >
            {children}
        </ChatView>
```

```
    );
}
```

§ views.chats.ChatsCache

会话界面缓存组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
children	`object`	可选	`null`	
filterType	`boolean`	可选	`false`	
activeChatId	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatsCache = require('views/chats/ChatsCache');

// 或通过 `xext.*` 调用:
// const ChatsCache = xext.views.chats.ChatsCache;

// 使用组件:
const render = () => {
    return (
        <ChatsCache
            className={className}
            filterType={filterType}
            activeChatId={activeChatId}
        >
            {children}
        </ChatsCache>
    );
}
```

§ views.chats.ChatsDndContainer

会话拖放事件处理组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const ChatsDndContainer = require('views/chats/ChatsDndContainer');

// 或通过 `xext.*` 调用:
```

```
// const ChatsDndContainer = xext.views.chats.ChatsDndContainer;

// 使用组件:
const render = () => {
    return (
        <ChatsDndContainer
            className={className}
        />
    );
}
```

§ views.chats.ChatsHistory

会话历史记录管理界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
chat	`object`	可选	`null`	
children	`object`	可选	`null`	
startPageSize	`number`	可选	`20`	
morePageSize	`number`	可选	`20`	

用法

```
// 通过 `require()` 调用:
const ChatsHistory = require('views/chats/ChatsHistory');

// 或通过 `xext.*` 调用:
// const ChatsHistory = xext.views.chats.ChatsHistory;

// 使用组件:
const render = () => {
    return (
        <ChatsHistory
            className={className}
            chat={chat}
            startPageSize={20}
            morePageSize={20}
        >
            {children}
        </ChatsHistory>
    );
}
```

§ views.chats.ChatsHistoryDialog

会话历史记录管理对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showChatsHistoryDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showChatsHistoryDialog} = require('views/chats/ChatsHistoryDialog');

// 或者通过 `xext.*` 调用:
// const {showChatsHistoryDialog} = xext.views.chats.ChatsHistoryDialog;
```

§ views.chats.Index

会话界面首页组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
match		可选		
hidden	`boolean`	可选	`false`	
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Index = require('views/chats/Index');

// 或通过 `xext.*` 调用:
// const Index = xext.views.chats.Index;

// 使用组件:
const render = () => {
    return (
        <Index
            match={match}
            hidden={hidden}
            className={className}
        />
    );
}
```

§ views.chats.Menu

会话列表界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
filterType	`object`	可选	`null`	
children	`object`	可选	`null`	
activeChatId	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Menu = require('views/chats/Menu');

// 或通过 `xext.*` 调用:
// const Menu = xext.views.chats.Menu;

// 使用组件:
const render = () => {
    return (
        <Menu
            className={className}
            filterType={filterType}
            activeChatId={activeChatId}
        >
            {children}
        </Menu>
    );
}
```

§ views.chats.MenuList

会话列表组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
search	`object`	可选	`null`	
filter	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	
activeChatId	`object`	可选	`null`	
onClearSearch	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MenuList = require('views/chats/MenuList');

// 或通过 `xext.*` 调用:
// const MenuList = xext.views.chats.MenuList;

// 使用组件:
const render = () => {
    return (
        <MenuList
            search={search}
            filter={filter}
            className={className}
            activeChatId={activeChatId}
            onClearSearch={onClearSearch}
        >
            {children}
        </MenuList>
    );
}
```

§ views.chats.MenuSearchList

会话搜索结果列表组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
search	`object`	可选	`null`	
filter	`object`	可选	`null`	
children	`object`	可选	`null`	
onClickSearchItem	`object`	可选	`null`	
startPageSize	`number`	可选	`20`	
morePageSize	`number`	可选	`20`	
defaultPage	`number`	可选	`1`	
activeChatId	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MenuSearchList = require('views/chats/MenuSearchList');

// 或通过 `xext.*` 调用:
// const MenuSearchList = xext.views.chats.MenuSearchList;

// 使用组件:
const render = () => {
    return (
        <MenuSearchList
            className={className}
            search={search}
            filter={filter}
            onClickSearchItem={onClickSearchItem}
            startPageSize={20}
            morePageSize={20}
            defaultPage={1}
            activeChatId={activeChatId}
        >
            {children}
        </MenuSearchList>
    );
}
```

§ views.chats.MessageBroadcast

广播类会话消息组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
prefix	`object`	可选	`null`	
children	`object`	可选	`null`	
contentConverter	`object`	可选	`null`	
message		可选		

用法

```
// 通过 `require()` 调用:
const MessageBroadcast = require('views/chats/MessageBroadcast');

// 或通过 `xext.*` 调用:
// const MessageBroadcast = xext.views.chats.MessageBroadcast;

// 使用组件:
const render = () => {
    return (
        <MessageBroadcast
            className={className}
            prefix={prefix}
            contentConverter={contentConverter}
            message={message}
        >
            {children}
        </MessageBroadcast>
    );
}
```

§ views.chats.MessageContentFile

文件类消息内容组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
message		可选		

用法

```
// 通过 `require()` 调用:
const MessageContentFile = require('views/chats/MessageContentFile');

// 或通过 `xext.*` 调用:
// const MessageContentFile = xext.views.chats.MessageContentFile;

// 使用组件:
const render = () => {
    return (
        <MessageContentFile
            className={className}
            message={message}
        />
    );
}
```

§ views.chats.MessageContentImage

图片类消息内容组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
message		可选		

用法

```
// 通过 `require()` 调用:
const MessageContentImage = require('views/chats/MessageContentImage');

// 或通过 `xext.*` 调用:
// const MessageContentImage = xext.views.chats.MessageContentImage;

// 使用组件:
const render = () => {
    return (
        <MessageContentImage
            className={className}
            message={message}
        />
    );
}
```

§ views.chats.MessageContentText

文本类消息内容组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
message		可选		
contentConverter	`object`	可选	`null`	
fontSize	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MessageContentText = require('views/chats/MessageContentText');

// 或通过 `xext.*` 调用:
// const MessageContentText = xext.views.chats.MessageContentText;

// 使用组件:
const render = () => {
    return (
        <MessageContentText
```

```
            className={className}
            message={message}
            contentConverter={contentConverter}
            fontSize={fontSize}
        />
    );
}
```

§ views.chats.MessageDivider

会话消息分割线组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
children	`object`	可选	`null`	
date	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MessageDivider = require('views/chats/MessageDivider');

// 或通过 `xext.*` 调用:
// const MessageDivider = xext.views.chats.MessageDivider;

// 使用组件:
const render = () => {
    return (
        <MessageDivider
            className={className}
            date={date}
        >
            {children}
        </MessageDivider>
    );
}
```

§ views.chats.MessageList

消息列表组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
messages		可选		
stayBottom	`boolean`	可选	`true`	
staticUI	`boolean`	可选	`false`	
showDateDivider	`number`	可选	`0`	
className	`object`	可选	`null`	
font	`object`	可选	`null`	
listItemProps	`object`	可选	`null`	
children	`object`	可选	`null`	
listItemCreator	`object`	可选	`null`	
header	`object`	可选	`null`	
onScroll	`object`	可选	`null`	
sleepUrlCard	`object`	可选	`null`	
inverse	`boolean`	可选	`false`	

用法

```
// 通过 `require()` 调用:
const MessageList = require('views/chats/MessageList');

// 或通过 `xext.*` 调用:
// const MessageList = xext.views.chats.MessageList;

// 使用组件:
const render = () => {
    return (
        <MessageList
            messages={messages}
            stayBottom={stayBottom}
            staticUI={staticUI}
            showDateDivider={showDateDivider}
            className={className}
            font={font}
            listItemProps={listItemProps}
            listItemCreator={listItemCreator}
            header={header}
            onScroll={onScroll}
            sleepUrlCard={sleepUrlCard}
            inverse={inverse}
        >
            {children}
        </MessageList>
    );
}
```

§ views.chats.MessageListItem

会话消息列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
message		可选		
lastMessage	`object`	可选	`null`	
font	`object`	可选	`null`	
ignoreStatus	`boolean`	可选	`false`	
showDateDivider	`number`	可选	`0`	
hideHeader	`number`	可选	`0`	
staticUI	`boolean`	可选	`false`	
avatarSize	`object`	可选	`null`	
dateFormater	`string`	可选	`"hh:mm"`	
textContentConverter	`object`	可选	`null`	
className	`object`	可选	`null`	
children	`object`	可选	`null`	
sleepUrlCard	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MessageListItem = require('views/chats/MessageListItem');

// 或通过 `xext.*` 调用:
// const MessageListItem = xext.views.chats.MessageListItem;

// 使用组件:
const render = () => {
    return (
        <MessageListItem
            message={message}
            lastMessage={lastMessage}
            font={font}
            ignoreStatus={ignoreStatus}
            showDateDivider={showDateDivider}
            hideHeader={hideHeader}
            staticUI={staticUI}
            avatarSize={avatarSize}
            dateFormater="hh:mm"
            textContentConverter={textContentConverter}
            className={className}
            sleepUrlCard={sleepUrlCard}
        >
            {children}
        </MessageListItem>
    );
}
```

§ views.chats.MessagesPreviewDialog

消息预览对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showMessagesPreviewDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showMessagesPreviewDialog} = require('views/chats/MessagesPreviewDialog');

// 或者通过 `xext.*` 调用:
// const {showMessagesPreviewDialog} = xext.views.chats.MessagesPreviewDialog;
```

§ views.common

主界面上的通用组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
AboutDialog	`object`	例如：`{"default":{}}`。
MemberProfileDialog	`object`	例如：`{"default":{}}`。
About	`class`	
MemberProfile	`class`	
BuildInfo	`class`	
Routes	`object`	
DraftEditor	`class`	
StatusDot	`class`	
EmojiPopover	`object`	例如：`{"default":{}}`。
UserAvatar	`class`	
UserChangePasswordDialog	`object`	例如：`{"default":{}}`。
FileListItem	`class`	
UserListItem	`class`	
FileListView	`class`	
UserProfileDialog	`object`	例如：`{"default":{}}`。
HotkeySettingDialog	`object`	例如：`{"default":{}}`。
UserSettingDialog	`object`	例如：`{"default":{}}`。
MemberListItem	`class`	
UserSetting	`class`	
MemberList	`class`	
SelectPanel	`class`	
SelectPanelWithActions	`class`	
SelectDialog	`object`	例如：`{"default":{}}`。

用法

```
// 通过 `require()` 调用:
const {AboutDialog, MemberProfileDialog, About, MemberProfile, BuildInfo, Routes, DraftEditor, StatusDot,
EmojiPopover, UserAvatar, UserChangePasswordDialog, FileListItem, UserListItem, FileListView,
UserProfileDialog, HotkeySettingDialog, UserSettingDialog, MemberListItem, UserSetting, MemberList,
SelectPanel, SelectPanelWithActions, SelectDialog} = require('views/common');

// 或者通过 `xext.*` 调用:
// const {AboutDialog, MemberProfileDialog, About, MemberProfile, BuildInfo, Routes, DraftEditor,
StatusDot, EmojiPopover, UserAvatar, UserChangePasswordDialog, FileListItem, UserListItem, FileListView,
UserProfileDialog, HotkeySettingDialog, UserSettingDialog, MemberListItem, UserSetting, MemberList,
SelectPanel, SelectPanelWithActions, SelectDialog} = xext.views.common;
```

§ views.common.About

关于界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const About = require('views/common/About');

// 或通过 `xext.*` 调用:
// const About = xext.views.common.About;

// 使用组件:
const render = () => {
    return (
        <About
            className={className}
        />
    );
}
```

§ views.common.AboutDialog

关于对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showAboutDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showAboutDialog} = require('views/common/AboutDialog');

// 或者通过 `xext.*` 调用:
// const {showAboutDialog} = xext.views.common.AboutDialog;
```

§ views.common.BuildInfo

客户端版本信息组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const BuildInfo = require('views/common/BuildInfo');

// 或通过 `xext.*` 调用:
// const BuildInfo = xext.views.common.BuildInfo;

// 使用组件:
const render = () => {
    return <BuildInfo />;
}
```

§ views.common.DraftEditor

消息输入框组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
placeholder	`object`	可选	`null`	
onChange	`object`	可选	`null`	
handleKey	`boolean`	可选	`false`	
onReturnKeyDown	`object`	可选	`null`	
onPastedText	`object`	可选	`null`	
onPastedFiles	`object`	可选	`null`	
defaultState	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const DraftEditor = require('views/common/DraftEditor');

// 或通过 `xext.*` 调用:
// const DraftEditor = xext.views.common.DraftEditor;

// 使用组件:
const render = () => {
    return (
        <DraftEditor
            placeholder={placeholder}
            onChange={onChange}
            handleKey={handleKey}
            onReturnKeyDown={onReturnKeyDown}
            onPastedText={onPastedText}
            onPastedFiles={onPastedFiles}
            defaultState={defaultState}
        />
```

```
    );
}
```

§ views.common.EmojiPopover

Emoji 选择面板功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showEmojiPopover	`function`	

用法

```
// 通过 `require()` 调用:
const {showEmojiPopover} = require('views/common/EmojiPopover');

// 或者通过 `xext.*` 调用:
// const {showEmojiPopover} = xext.views.common.EmojiPopover;
```

§ views.common.FileListItem

文件列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
file		可选		
hideIcon	`boolean`	可选	`false`	
className	`string`	可选	`"flex-middle"`	
showDate	`boolean`	可选	`false`	
onFileSaved	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const FileListItem = require('views/common/FileListItem');

// 或通过 `xext.*` 调用:
// const FileListItem = xext.views.common.FileListItem;

// 使用组件:
const render = () => {
    return (
        <FileListItem
            file={file}
            hideIcon={hideIcon}
            className="flex-middle"
            showDate={showDate}
            onFileSaved={onFileSaved}
        />
    );
}
```

§ views.common.FileListView

文件列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
files		可选		
listItemProps	`object`	可选	`null`	
className	`object`	可选	`null`	
startPageSize	`number`	可选	`20`	
morePageSize	`number`	可选	`20`	
defaultPage	`number`	可选	`1`	

用法

```
// 通过 `require()` 调用:
const FileListView = require('views/common/FileListView');

// 或通过 `xext.*` 调用:
// const FileListView = xext.views.common.FileListView;

// 使用组件:
const render = () => {
    return (
        <FileListView
            files={files}
            listItemProps={listItemProps}
            className={className}
            startPageSize={20}
            morePageSize={20}
            defaultPage={1}
        />
    );
}
```

§ views.common.HotkeySettingDialog

快捷键设置对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showHotkeySettingDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showHotkeySettingDialog} = require('views/common/HotkeySettingDialog');

// 或者通过 `xext.*` 调用:
// const {showHotkeySettingDialog} = xext.views.common.HotkeySettingDialog;
```

§ views.common.MemberList

成员列表组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
chosen	`object`	可选	`null`	
members		可选		
listItemProps	`object`	可选	`null`	
onItemClick	`object`	可选	`null`	
onItemContextMenu	`object`	可选	`null`	
itemRender	`object`	可选	`null`	
contentRender	`object`	可选	`null`	
className	`object`	可选	`null`	
avatarClassName	`object`	可选	`null`	
heading	`object`	可选	`null`	
startPageSize	`number`	可选	`20`	
morePageSize	`number`	可选	`20`	
defaultPage	`number`	可选	`1`	
eventBindObject	`object`	可选	`null`	
selection	`boolean`	可选	`false`	

用法

```
// 通过 `require()` 调用:
const MemberList = require('views/common/MemberList');

// 或通过 `xext.*` 调用:
// const MemberList = xext.views.common.MemberList;

// 使用组件:
const render = () => {
    return (
        <MemberList
            chosen={chosen}
            members={members}
            listItemProps={listItemProps}
            onItemClick={onItemClick}
            onItemContextMenu={onItemContextMenu}
            itemRender={itemRender}
            contentRender={contentRender}
            className={className}
            avatarClassName={avatarClassName}
            heading={heading}
            startPageSize={20}
            morePageSize={20}
            defaultPage={1}
            eventBindObject={eventBindObject}
            selection={selection}
        />
    );
}
```

§ views.common.MemberListItem

成员列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
title	`object`	可选	`null`	
member		可选		
children	`object`	可选	`null`	
className	`string`	可选	`"flex-middle"`	
avatarSize	`number`	可选	`24`	
showStatusDot	`boolean`	可选	`true`	
avatarClassName	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MemberListItem = require('views/common/MemberListItem');

// 或通过 `xext.*` 调用:
// const MemberListItem = xext.views.common.MemberListItem;

// 使用组件:
const render = () => {
    return (
        <MemberListItem
            title={title}
            member={member}
            className="flex-middle"
            avatarSize={24}
            showStatusDot={showStatusDot}
            avatarClassName={avatarClassName}
        >
            {children}
        </MemberListItem>
    );
}
```

§ views.common.MemberProfile

个人资料界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
memberId		可选		
className	`object`	可选	`null`	
compact	`boolean`	可选	`false`	
hideChatBtn	`boolean`	可选	`false`	
onRequestClose	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const MemberProfile = require('views/common/MemberProfile');

// 或通过 `xext.*` 调用:
// const MemberProfile = xext.views.common.MemberProfile;

// 使用组件:
const render = () => {
    return (
        <MemberProfile
            memberId={memberId}
            className={className}
            compact={compact}
            hideChatBtn={hideChatBtn}
            onRequestClose={onRequestClose}
        />
    );
}
```

§ views.common.MemberProfileDialog

个人资料对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showMemberProfileDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showMemberProfileDialog} = require('views/common/MemberProfileDialog');

// 或者通过 `xext.*` 调用:
// const {showMemberProfileDialog} = xext.views.common.MemberProfileDialog;
```

§ views.common.Routes

路由功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
chats	`object`	
contacts	`object`	
exts	`object`	例如：`{"_":"/exts","app":{"__":"/exts/app/:id"}}`。
apps	`object`	例如：`{"__":"/:app/:filterType?/:id?/:params?"}`。

用法

```
// 通过 `require()` 调用：
const {chats, contacts, exts, apps} = require('views/common/Routes');

// 或者通过 `xext.*` 调用：
// const {chats, contacts, exts, apps} = xext.views.common.Routes;
```

§ views.common.SelectDialog

选择对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showSelectDialog	`function`	
showSelectDialogWithPreview	`function`	

用法

```
// 通过 `require()` 调用：
const {showSelectDialog, showSelectDialogWithPreview} = require('views/common/SelectDialog');

// 或者通过 `xext.*` 调用：
// const {showSelectDialog, showSelectDialogWithPreview} = xext.views.common.SelectDialog;
```

§ views.common.SelectPanel

选择面板组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
listBuilder	`object`	可选	`null`	
onSelectionsChange	`object`	可选	`null`	
className	`object`	可选	`null`	
searchPlaceHolderText	`object`	可选	`null`	
children	`object`	可选	`null`	
selections	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用：
const SelectPanel = require('views/common/SelectPanel');
```

```
// 或通过 `xext.*` 调用:
// const SelectPanel = xext.views.common.SelectPanel;

// 使用组件:
const render = () => {
    return (
        <SelectPanel
            listBuilder={listBuilder}
            onSelectionsChange={onSelectionsChange}
            className={className}
            searchPlaceHolderText={searchPlaceHolderText}
            selections={selections}
        >
            {children}
        </SelectPanel>
    );
}
```

§ views.common.SelectPanelWithActions

带操作的选择面板组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
onFinish	`object`	可选	`null`	
primaryBtnText	`string`	可选	`""`	
cancelBtnText	`string`	可选	`""`	
message	`object`	可选	`null`	
header	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const SelectPanelWithActions = require('views/common/SelectPanelWithActions');

// 或通过 `xext.*` 调用:
// const SelectPanelWithActions = xext.views.common.SelectPanelWithActions;

// 使用组件:
const render = () => {
    return (
        <SelectPanelWithActions
            onFinish={onFinish}
            primaryBtnText={primaryBtnText}
            cancelBtnText={cancelBtnText}
            message={message}
            header={header}
        />
    );
}
```

§ views.common.StatusDot

用户状态指示标签组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
size	`number`	可选	`14`	
className	`string`	可选	`"circle"`	
label	`object`	可选	`null`	
style	`object`	可选	`null`	
status	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const StatusDot = require('views/common/StatusDot');

// 或通过 `xext.*` 调用:
// const StatusDot = xext.views.common.StatusDot;

// 使用组件:
const render = () => {
    return (
        <StatusDot
            size={14}
            className="circle"
            label={label}
            style={style}
            status={status}
        />
    );
}
```

§ views.common.UserAvatar

用户头像组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
user	`object`	可选	`null`	
className	`object`	可选	`null`	
showStatusDot	`object`	可选	`null`	
shape	`string`	可选	`"circle"`	
avatarSize	`number`	可选	`24`	

用法

```
// 通过 `require()` 调用:
const UserAvatar = require('views/common/UserAvatar');
```

```
// 或通过 `xext.*` 调用:
// const UserAvatar = xext.views.common.UserAvatar;

// 使用组件:
const render = () => {
    return (
        <UserAvatar
            user={user}
            className={className}
            showStatusDot={showStatusDot}
            shape="circle"
            avatarSize={24}
        />
    );
}
```

§ views.common.UserChangePasswordDialog

修改密码对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
UserChangePassword	`class`	
showUserChangePasswordDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {UserChangePassword, showUserChangePasswordDialog} =
require('views/common/UserChangePasswordDialog');

// 或者通过 `xext.*` 调用:
// const {UserChangePassword, showUserChangePasswordDialog} = xext.views.common.UserChangePasswordDialog;
```

§ views.common.UserListItem

用户列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
user		可选		
className	`string`	可选	`"flex-middle"`	
avatarSize	`number`	可选	`30`	
avatarClassName	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const UserListItem = require('views/common/UserListItem');
```

```
// 或通过 `xext.*` 调用:
// const UserListItem = xext.views.common.UserListItem;

// 使用组件:
const render = () => {
    return (
        <UserListItem
            user={user}
            className="flex-middle"
            avatarSize={30}
            avatarClassName={avatarClassName}
        >
            {children}
        </UserListItem>
    );
}
```

§ views.common.UserProfileDialog

用户个人资料对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showUserProfileDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showUserProfileDialog} = require('views/common/UserProfileDialog');

// 或者通过 `xext.*` 调用:
// const {showUserProfileDialog} = xext.views.common.UserProfileDialog;
```

§ views.common.UserSetting

用户设置界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
settings		可选		
localsettings	`object`	可选	`null`	
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const UserSetting = require('views/common/UserSetting');

// 或通过 `xext.*` 调用:
// const UserSetting = xext.views.common.UserSetting;

// 使用组件:
const render = () => {
```

```
    return (
        <UserSetting
            settings={settings}
            localsettings={localsettings}
            className={className}
        />
    );
}
```

§ views.common.UserSettingDialog

用户设置对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showUserSettingDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showUserSettingDialog} = require('views/common/UserSettingDialog');

// 或者通过 `xext.*` 调用:
// const {showUserSettingDialog} = xext.views.common.UserSettingDialog;
```

§ views.contacts

通讯录界面组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
ContactsHome	`class`	
ContactsView	`class`	
DeptsTree	`class`	
ContactsIndex	`class`	
GroupsMenu	`class`	
GroupsView	`class`	

用法

```
// 通过 `require()` 调用:
const {ContactsHome, ContactsView, DeptsTree, ContactsIndex, GroupsMenu, GroupsView} =
require('views/contacts');

// 或者通过 `xext.*` 调用:
// const {ContactsHome, ContactsView, DeptsTree, ContactsIndex, GroupsMenu, GroupsView} =
xext.views.contacts;
```

§ views.contacts.ContactsHome

通讯录界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
hidden	`boolean`	可选	`false`	
className	`object`	可选	`null`	
objectType		可选		
filterType		可选		

用法

```
// 通过 `require()` 调用:
const ContactsHome = require('views/contacts/ContactsHome');

// 或通过 `xext.*` 调用:
// const ContactsHome = xext.views.contacts.ContactsHome;

// 使用组件:
const render = () => {
    return (
        <ContactsHome
            hidden={hidden}
            className={className}
            objectType={objectType}
            filterType={filterType}
        />
    );
}
```

§ views.contacts.ContactsIndex

通讯录首页组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const ContactsIndex = require('views/contacts/ContactsIndex');

// 或通过 `xext.*` 调用:
// const ContactsIndex = xext.views.contacts.ContactsIndex;

// 使用组件:
const render = () => {
    return <ContactsIndex />;
}
```

§ views.contacts.ContactsView

通讯录列表界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
deptID		可选		

用法

```
// 通过 `require()` 调用:
const ContactsView = require('views/contacts/ContactsView');

// 或通过 `xext.*` 调用:
// const ContactsView = xext.views.contacts.ContactsView;

// 使用组件:
const render = () => {
    return (
        <ContactsView
            className={className}
            deptID={deptID}
        />
    );
}
```

§ views.contacts.DeptsTree

部门树结构界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
activeGroupID	`object`	可选	`null`	
showCompany	`boolean`	可选	`true`	

用法

```
// 通过 `require()` 调用:
const DeptsTree = require('views/contacts/DeptsTree');

// 或通过 `xext.*` 调用:
// const DeptsTree = xext.views.contacts.DeptsTree;

// 使用组件:
const render = () => {
    return (
        <DeptsTree
            className={className}
            activeGroupID={activeGroupID}
            showCompany={showCompany}
        />
    );
```

```
}
```

§ views.contacts.GroupsMenu

讨论组菜单组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
filterType	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const GroupsMenu = require('views/contacts/GroupsMenu');

// 或通过 `xext.*` 调用:
// const GroupsMenu = xext.views.contacts.GroupsMenu;

// 使用组件:
const render = () => {
    return (
        <GroupsMenu
            className={className}
            filterType={filterType}
        />
    );
}
```

§ views.contacts.GroupsView

讨论组列表组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
filterType		可选		

用法

```
// 通过 `require()` 调用:
const GroupsView = require('views/contacts/GroupsView');

// 或通过 `xext.*` 调用:
// const GroupsView = xext.views.contacts.GroupsView;

// 使用组件:
const render = () => {
    return (
        <GroupsView
```

```
            className={className}
            filterType={filterType}
        />
    );
}
```

§ views.exts

主界面上的扩展相关组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
AppExtensions	`class`	
ExtensionListItem	`class`	
AppFiles	`class`	
AppHome	`class`	
Index	`class`	
WebApp	`class`	
ExtensionDetail	`class`	

用法

```
// 通过 `require()` 调用:
const {AppExtensions, ExtensionListItem, AppFiles, AppHome, Index, WebApp, ExtensionDetail} =
require('views/exts');

// 或者通过 `xext.*` 调用:
// const {AppExtensions, ExtensionListItem, AppFiles, AppHome, Index, WebApp, ExtensionDetail} =
xext.views.exts;
```

§ views.exts.AppExtensions

应用扩展管理界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
app		可选		

用法

```
// 通过 `require()` 调用:
const AppExtensions = require('views/exts/AppExtensions');

// 或通过 `xext.*` 调用:
// const AppExtensions = xext.views.exts.AppExtensions;
```

```
// 使用组件:
const render = () => {
    return (
        <AppExtensions
            className={className}
            app={app}
        />
    );
}
```

§ views.exts.AppFiles

文件应用界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
app		可选		

用法

```
// 通过 `require()` 调用:
const AppFiles = require('views/exts/AppFiles');

// 或通过 `xext.*` 调用:
// const AppFiles = xext.views.exts.AppFiles;

// 使用组件:
const render = () => {
    return (
        <AppFiles
            className={className}
            app={app}
        />
    );
}
```

§ views.exts.AppHome

应用管理首页界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const AppHome = require('views/exts/AppHome');
```

```
// 或通过 `xext.*` 调用:
// const AppHome = xext.views.exts.AppHome;

// 使用组件:
const render = () => {
    return (
        <AppHome
            className={className}
        />
    );
}
```

§ views.exts.ExtensionDetail

扩展详情界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
onRequestClose	`object`	可选	`null`	
extension		可选		

用法

```
// 通过 `require()` 调用:
const ExtensionDetail = require('views/exts/ExtensionDetail');

// 或通过 `xext.*` 调用:
// const ExtensionDetail = xext.views.exts.ExtensionDetail;

// 使用组件:
const render = () => {
    return (
        <ExtensionDetail
            className={className}
            onRequestClose={onRequestClose}
            extension={extension}
        />
    );
}
```

§ views.exts.ExtensionListItem

扩展列表项组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
extension		可选		
onSettingBtnClick	`object`	可选	`null`	
showType	`boolean`	可选	`true`	

用法

```
// 通过 `require()` 调用:
const ExtensionListItem = require('views/exts/ExtensionListItem');

// 或通过 `xext.*` 调用:
// const ExtensionListItem = xext.views.exts.ExtensionListItem;

// 使用组件:
const render = () => {
    return (
        <ExtensionListItem
            className={className}
            extension={extension}
            onSettingBtnClick={onSettingBtnClick}
            showType={showType}
        />
    );
}
```

§ views.exts.Index

扩展首页界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
match		可选		
hidden	`boolean`	可选	`false`	
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Index = require('views/exts/Index');

// 或通过 `xext.*` 调用:
// const Index = xext.views.exts.Index;

// 使用组件:
const render = () => {
    return (
        <Index
            match={match}
            hidden={hidden}
            className={className}
        />
    );
}
```

§ views.exts.WebApp

Web 应用界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
app		可选		
className	`object`	可选	`null`	
onLoadingChange	`object`	可选	`null`	
onPageTitleUpdated	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const WebApp = require('views/exts/WebApp');

// 或通过 `xext.*` 调用:
// const WebApp = xext.views.exts.WebApp;

// 使用组件:
const render = () => {
    return (
        <WebApp
            app={app}
            className={className}
            onLoadingChange={onLoadingChange}
            onPageTitleUpdated={onPageTitleUpdated}
        />
    );
}
```

§ views.index

主界面上的首页相关组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
ImageCutterApp	`class`	
AppView	`class`	
Index	`class`	

用法

```
// 通过 `require()` 调用:
const {ImageCutterApp, AppView, Index} = require('views/index');

// 或者通过 `xext.*` 调用:
// const {ImageCutterApp, AppView, Index} = xext.views.index;
```

§ views.index.AppView

主窗口首页界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const AppView = require('views/index/AppView');

// 或通过 `xext.*` 调用:
// const AppView = xext.views.index.AppView;

// 使用组件:
const render = () => {
    return <AppView />;
}
```

§ views.index.ImageCutterApp

截屏窗口界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const ImageCutterApp = require('views/index/ImageCutterApp');

// 或通过 `xext.*` 调用:
// const ImageCutterApp = xext.views.index.ImageCutterApp;

// 使用组件:
const render = () => {
    return <ImageCutterApp />;
}
```

§ views.index.Index

主窗口界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

用法

```
// 通过 `require()` 调用:
const Index = require('views/index/Index');

// 或通过 `xext.*` 调用:
// const Index = xext.views.index.Index;

// 使用组件:
const render = () => {
    return <Index />;
}
```

§ views.login

主界面上的登录相关组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
Form	`class`	
SwapUserDialog	`object`	例如: `{"default":{}}`。
Index	`class`	
SwapUser	`class`	

用法

```
// 通过 `require()` 调用:
const {Form, SwapUserDialog, Index, SwapUser} = require('views/login');

// 或者通过 `xext.*` 调用:
// const {Form, SwapUserDialog, Index, SwapUser} = xext.views.login;
```

§ views.login.Form

登录表单界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Form = require('views/login/Form');

// 或通过 `xext.*` 调用:
// const Form = xext.views.login.Form;

// 使用组件:
const render = () => {
```

```
    return (
        <Form
            className={className}
        />
    );
}
```

§ views.login.Index

登录界面首页组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
userStatus	`object`	可选	`null`	
children	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Index = require('views/login/Index');

// 或通过 `xext.*` 调用:
// const Index = xext.views.login.Index;

// 使用组件:
const render = () => {
    return (
        <Index
            className={className}
            userStatus={userStatus}
        >
            {children}
        </Index>
    );
}
```

§ views.login.SwapUser

切换已登录的账号界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
identify	`object`	可选	`null`	
server	`object`	可选	`null`	
onSelectUser	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const SwapUser = require('views/login/SwapUser');

// 或通过 `xext.*` 调用:
// const SwapUser = xext.views.login.SwapUser;

// 使用组件:
const render = () => {
    return (
        <SwapUser
            className={className}
            identify={identify}
            server={server}
            onSelectUser={onSelectUser}
        />
    );
}
```

§ views.login.SwapUserDialog

切换已登录的账号对话框功能库。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
showSwapUserDialog	`function`	

用法

```
// 通过 `require()` 调用:
const {showSwapUserDialog} = require('views/login/SwapUserDialog');

// 或者通过 `xext.*` 调用:
// const {showSwapUserDialog} = xext.views.login.SwapUserDialog;
```

§ views.main

主界面上的界面框架组件。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
Navbar	`class`	
CacheContainer	`class`	
Index	`class`	
AutoReconnectBar	`class`	

用法

```
// 通过 `require()` 调用:
const {Navbar, CacheContainer, Index, AutoReconnectBar} = require('views/main');

// 或者通过 `xext.*` 调用:
// const {Navbar, CacheContainer, Index, AutoReconnectBar} = xext.views.main;
```

§ views.main.AutoReconnectBar

自动重连提示界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const AutoReconnectBar = require('views/main/AutoReconnectBar');

// 或通过 `xext.*` 调用:
// const AutoReconnectBar = xext.views.main.AutoReconnectBar;

// 使用组件:
const render = () => {
    return (
        <AutoReconnectBar
            className={className}
        />
    );
}
```

§ views.main.CacheContainer

应用界面缓存组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
match	`object`	可选	`null`	
location	`object`	可选	`null`	
history	`function`	可选	`function () { [native code] }`	
staticContext	`function`	可选	`function () { [native code] }`	

用法

```
// 通过 `require()` 调用:
const CacheContainer = require('views/main/CacheContainer');

// 或通过 `xext.*` 调用:
```

```
// const CacheContainer = xext.views.main.CacheContainer;

// 使用组件:
const render = () => {
    return (
        <CacheContainer
            match={match}
            location={location}
            history={history}
            staticContext={staticContext}
        />
    );
}
```

§ views.main.Index

主界面组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
userStatus	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Index = require('views/main/Index');

// 或通过 `xext.*` 调用:
// const Index = xext.views.main.Index;

// 使用组件:
const render = () => {
    return (
        <Index
            className={className}
            userStatus={userStatus}
        />
    );
}
```

§ views.main.Navbar

主导航组件。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
className	`object`	可选	`null`	
userStatus	`object`	可选	`null`	

用法

```
// 通过 `require()` 调用:
const Navbar = require('views/main/Navbar');

// 或通过 `xext.*` 调用:
// const Navbar = xext.views.main.Navbar;

// 使用组件:
const render = () => {
    return (
        <Navbar
            className={className}
            userStatus={userStatus}
        />
    );
}
```

utils： 通用工具模块相关

§ utils.Color

颜色辅助类。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
hexToRgb	`function`	
isColor	`function`	
hslToRgb	`function`	
Color	`class`	

用法

```
// 通过 `require()` 调用:
const {hexToRgb, isColor, hslToRgb, Color} = require('utils/Color');

// 或者通过 `xext.*` 调用:
// const {hexToRgb, isColor, hslToRgb, Color} = xext.utils.Color;
```

§ utils.DateHelper

日期时间辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
TIME_DAY	`number`	例如：`86400000`。
createDate	`function`	
createPhpTimestamp	`function`	
isSameDay	`function`	
isSameYear	`function`	
isSameMonth	`function`	
isToday	`function`	
isYesterday	`function`	
isDBY	`function`	
formatDate	`function`	
formatDateSpan	`function`	
getTimeBeforeDesc	`function`	

用法

```
// 通过 `require()` 调用:
const {TIME_DAY, createDate, createPhpTimestamp, isSameDay, isSameYear, isSameMonth, isToday, isYesterday,
isDBY, formatDate, formatDateSpan, getTimeBeforeDesc} = require('utils/DateHelper');

// 或者通过 `xext.*` 调用:
// const {TIME_DAY, createDate, createPhpTimestamp, isSameDay, isSameYear, isSameMonth, isToday,
isYesterday, isDBY, formatDate, formatDateSpan, getTimeBeforeDesc} = xext.utils.DateHelper;
```

§ utils.HtmlHelper

HTML 操作辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
classes	`function`	
rem	`function`	
getSearchParam	`function`	
strip	`function`	
escape	`function`	
isWebUrl	`function`	
linkify	`function`	

用法

```
// 通过 `require()` 调用:
const {classes, rem, getSearchParam, strip, escape, isWebUrl, linkify} = require('utils/HtmlHelper');

// 或者通过 `xext.*` 调用:
// const {classes, rem, getSearchParam, strip, escape, isWebUrl, linkify} = xext.utils.HtmlHelper;
```

§ utils.Image

图片操作辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
getImageInfo	`function`	
cutImage	`function`	

用法

```
// 通过 `require()` 调用:
const {getImageInfo, cutImage} = require('utils/Image');

// 或者通过 `xext.*` 调用:
// const {getImageInfo, cutImage} = xext.utils.Image;
```

§ utils.LimitTimePromise

限时异步实现辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
limitTimePromise	`function`	

用法

```
// 通过 `require()` 调用:
const {limitTimePromise} = require('utils/LimitTimePromise');

// 或者通过 `xext.*` 调用:
// const {limitTimePromise} = xext.utils.LimitTimePromise;
```

§ utils.Markdown

Markdown 辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
renderer	`object`	
marked	`function`	

用法

```
// 通过 `require()` 调用:
const {renderer, marked} = require('utils/Markdown');

// 或者通过 `xext.*` 调用:
// const {renderer, marked} = xext.utils.Markdown;
```

§ utils.Pinyin

拼音辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
getPinyin	`function`	

用法

```
// 通过 `require()` 调用:
const {getPinyin} = require('utils/Pinyin');

// 或者通过 `xext.*` 调用:
// const {getPinyin} = xext.utils.Pinyin;
```

§ utils.Plain

对象扁平化辅助方法。

基本信息

等级	5
类型	模块或库

用法

```
// 通过 `xext.*` 调用:
const Plain = xext.utils.Plain;

// 或者通过 `require()` 调用:
// const Plain = require('utils/Plain');
```

§ utils.Skin

CSS 样式计算辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
getCodeFromString	`function`	
longShadow	`function`	
skinStyle	`function`	

用法

```
// 通过 `require()` 调用:
const {getCodeFromString, longShadow, skinStyle} = require('utils/Skin');

// 或者通过 `xext.*` 调用:
// const {getCodeFromString, longShadow, skinStyle} = xext.utils.Skin;
```

§ utils.Store

本地存储辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
setStoreItem	`function`	
getStoreItem	`function`	
removeStoreItem	`function`	
clearStore	`function`	
getStoreLength	`function`	
storeForEach	`function`	
storeGetAll	`function`	

用法

```
// 通过 `require()` 调用:
const {setStoreItem, getStoreItem, removeStoreItem, clearStore, getStoreLength, storeForEach, storeGetAll}
= require('utils/Store');

// 或者通过 `xext.*` 调用:
// const {setStoreItem, getStoreItem, removeStoreItem, clearStore, getStoreLength, storeForEach,
storeGetAll} = xext.utils.Store;
```

§ utils.StringHelper

字符串辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
formatString	`function`	
BYTE_UNITS	`object`	
formatBytes	`function`	
isEmptyString	`function`	
isNotEmptyString	`function`	
ifEmptyStringThen	`function`	
limitStringLength	`function`	

用法

```
// 通过 `require()` 调用:
const {formatString, BYTE_UNITS, formatBytes, isEmptyString, isNotEmptyString, ifEmptyStringThen,
limitStringLength} = require('utils/StringHelper');

// 或者通过 `xext.*` 调用:
// const {formatString, BYTE_UNITS, formatBytes, isEmptyString, isNotEmptyString, ifEmptyStringThen,
limitStringLength} = xext.utils.StringHelper;
```

§ utils.version

语义化版本号辅助方法。

基本信息

等级	5
类型	模块或库

模块成员

成员	类型	描述和示例
formatVersion	`function`	
simplifyVersion	`function`	
compareVersions	`function`	

用法

```
// 通过 `require()` 调用:
const {formatVersion, simplifyVersion, compareVersions} = require('utils/version');

// 或者通过 `xext.*` 调用:
// const {formatVersion, simplifyVersion, compareVersions} = xext.utils.version;
```

window： 浏览器内置对象相关

§ window.document

访问浏览器 document 对象。

基本信息

等级	6
类型	数据
依赖	`window.window`

数据类型

- **`Document`**： 浏览器 Document 对象。。

用法

```
// 通过 `xext.*` 调用:
const document = xext.window.document;

// 或通过 `require()` 调用:
// const document = require('xext/window');
```

§ window.window

访问浏览器 window 对象。

基本信息

等级	6
类型	数据

数据类型

- **`Window`**： 浏览器 Window 对象。。

用法

```
// 通过 `require()` 调用:
const window = require('window');

// 或通过 `xext.*` 调用:
// const window = xext.window.window;
```

nodeModules： 第三方 node 模块

§ nodeModules.compareVersions

compare-versions 模块，提供对版本号进行比较的方法。详情请参考 https://npmjs.com/package/compareVersions 。

基本信息

等级	7
类型	函数

函数定义

- *function* **compareVersions**(firstVersion, secondVersion)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
firstVersion	`string`	必须		第一个版本字符串。例如：`"2.6.0"`。
secondVersion	`string`	必须		第二个版本字符串。例如：`"3.1.0"`。

函数返回值

- **`number`**：比较结果，如果等于 0，说明两个参数的值相等；如果大于 0，说明第一个值大于第二个值；如果小于 0，说明第一个值小于第二个值。

用法

```
const firstVersion = "2.6.0";
const secondVersion = "3.1.0";
// 通过 `require()` 调用:
const {compareVersions} = require("compareVersions");
const compareResult = compareVersions(firstVersion, secondVersion);

// 或通过 `xext.*` 调用:
// const compareResult = xext.nodeModules.compareVersions(firstVersion, secondVersion);
```

§ nodeModules.draft-js

draft-js 模块，提供 DraftJS 编辑器组件。详情请参考 https://npmjs.com/package/draft-js 。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
Editor	`class`	
EditorBlock	`class`	
EditorState	`class`	
CompositeDecorator	`class`	
Entity	`object`	例如：`{}`。
EntityInstance	`class`	
BlockMapBuilder	`object`	例如：`{}`。
CharacterMetadata	`class`	
ContentBlock	`class`	
ContentState	`class`	
RawDraftContentState	`object`	例如：`{}`。
SelectionState	`class`	
AtomicBlockUtils	`object`	例如：`{}`。
KeyBindingUtil	`object`	例如：`{}`。
Modifier	`object`	例如：`{}`。
RichUtils	`object`	例如：`{}`。
DefaultDraftBlockRenderMap	`object`	
DefaultDraftInlineStyle	`object`	
convertFromHTML	`function`	
convertFromRaw	`function`	
convertToRaw	`function`	
genKey	`function`	
getDefaultKeyBinding	`function`	
getVisibleSelectionRect	`function`	

用法

```
// 通过 `require()` 调用:
const {Editor, EditorBlock, EditorState, CompositeDecorator, Entity, EntityInstance, BlockMapBuilder,
CharacterMetadata, ContentBlock, ContentState, RawDraftContentState, SelectionState, AtomicBlockUtils,
KeyBindingUtil, Modifier, RichUtils, DefaultDraftBlockRenderMap, DefaultDraftInlineStyle, convertFromHTML,
convertFromRaw, convertToRaw, genKey, getDefaultKeyBinding, getVisibleSelectionRect} = require('draft-js');

// 或者通过 `xext.*` 调用:
// const {Editor, EditorBlock, EditorState, CompositeDecorator, Entity, EntityInstance, BlockMapBuilder,
CharacterMetadata, ContentBlock, ContentState, RawDraftContentState, SelectionState, AtomicBlockUtils,
KeyBindingUtil, Modifier, RichUtils, DefaultDraftBlockRenderMap, DefaultDraftInlineStyle, convertFromHTML,
convertFromRaw, convertToRaw, genKey, getDefaultKeyBinding, getVisibleSelectionRect} =
xext.nodeModules.DraftJs;
```

§ nodeModules.emoji-toolkit

emoji-toolkit 模块，提供 Emoji 辅助方法。详情请参考 https://npmjs.com/package/emoji-toolkit 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
emojiList	`object`	
asciiList	`object`	
asciiRegexp	`string`	
emojiVersion	`string`	例如：`"5.0"`。
emojiSize	`string`	例如：`"32"`。
blacklistChars	`string`	例如：`""`。
imagePathPNG	`string`	例如：`"media/twemoji/png/"`。
defaultPathPNG	`string`	
fileExtension	`string`	例如：`".png"`。
imageTitleTag	`boolean`	例如：`true`。
sprites	`boolean`	例如：`false`。
unicodeAlt	`boolean`	例如：`true`。
ascii	`boolean`	例如：`false`。
riskyMatchAscii	`boolean`	例如：`false`。
regAscii	`object`	例如：`{}`。
regAsciiRisky	`object`	例如：`{}`。
regUnicode	`object`	例如：`{}`。
convert	`function`	
shortnameLookup	`object`	例如：`[]`。
altShortNames	`object`	例如：`[]`。
shortnames	`string`	
regShortNames	`object`	例如：`{}`。
toImage	`function`	
unicodeToImage	`function`	
unifyUnicode	`function`	
shortnameToAscii	`function`	
shortnameToUnicode	`function`	
shortnameToImage	`function`	
toShort	`function`	
escapeHTML	`function`	
unescapeHTML	`function`	
shortnameConversionMap	`function`	
unicodeCharRegex	`function`	
mapEmojiList	`function`	
mapUnicodeToShort	`function`	
memorizeReplacement	`function`	
mapUnicodeCharactersToShort	`function`	
objectFlip	`function`	
escapeRegExp	`function`	
replaceAll	`function`	
imageType	`string`	例如：`"png"`。

用法

```
// 通过 `require()` 调用:
const {emojiList, asciiList, asciiRegexp, emojiVersion, emojiSize, blacklistChars, imagePathPNG,
defaultPathPNG, fileExtension, imageTitleTag, sprites, unicodeAlt, ascii, riskyMatchAscii, regAscii,
regAsciiRisky, regUnicode, convert, shortnameLookup, altShortNames, shortnames, regShortNames, toImage,
unicodeToImage, unifyUnicode, shortnameToAscii, shortnameToUnicode, shortnameToImage, toShort, escapeHTML,
unescapeHTML, shortnameConversionMap, unicodeCharRegex, mapEmojiList, mapUnicodeToShort,
memorizeReplacement, mapUnicodeCharactersToShort, objectFlip, escapeRegExp, replaceAll, imageType} =
require('emoji-toolkit');

// 或者通过 `xext.*` 调用:
// const {emojiList, asciiList, asciiRegexp, emojiVersion, emojiSize, blacklistChars, imagePathPNG,
defaultPathPNG, fileExtension, imageTitleTag, sprites, unicodeAlt, ascii, riskyMatchAscii, regAscii,
regAsciiRisky, regUnicode, convert, shortnameLookup, altShortNames, shortnames, regShortNames, toImage,
unicodeToImage, unifyUnicode, shortnameToAscii, shortnameToUnicode, shortnameToImage, toShort, escapeHTML,
unescapeHTML, shortnameConversionMap, unicodeCharRegex, mapEmojiList, mapUnicodeToShort,
memorizeReplacement, mapUnicodeCharactersToShort, objectFlip, escapeRegExp, replaceAll, imageType} =
xext.nodeModules.EmojiToolkit;
```

§ nodeModules.emojione-picker

emojione-picker 模块，提供 Emoji 表情选择面板组件。详情请参考 https://npmjs.com/package/emoji-picker 。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `nodeModules.emoji-toolkit`, `window.document`

组件属性

属性	类型	是否必须	默认值	描述和示例
emojione		可选		
search	`string`	可选	`""`	
searchPlaceholder	`string`	可选	`"Search…"`	
className		可选		
onChange		可选		
categories	`object`	可选	`{"people":{"title":"People","emoji":"smile"},"nature":{"title":"Nature","emoji":"hamster"},"food":{"title":"Food & Drink","emoji":"pizza"},"activity":{"title":"Activity","emoji":"soccer"},"travel":{"title":"Travel & Places","emoji":"earth_americas"},"objects":{"title":"Objects","emoji":"bulb"},"symbols":{"title":"Symbols","emoji":"clock9"},"flags":{"title":"Flags","emoji":"flag_gb"}}`	

用法

```
// 通过 `require()` 调用:
const EmojiPicker = require('emoji-picker');

// 或通过 `xext.*` 调用:
// const EmojiPicker = xext.nodeModules.EmojiPicker;

// 使用组件:
const render = () => {
    return (
        <EmojiPicker
            emojione={emojione}
            search={search}
            searchPlaceholder="Search…"
            className={className}
            onChange={onChange}
            categories={{"people":{"title":"People","emoji":"smile"},"nature":
{"title":"Nature","emoji":"hamster"},"food":{"title":"Food & Drink","emoji":"pizza"},"activity":
{"title":"Activity","emoji":"soccer"},"travel":{"title":"Travel &
Places","emoji":"earth_americas"},"objects":{"title":"Objects","emoji":"bulb"},"symbols":
{"title":"Symbols","emoji":"clock9"},"flags":{"title":"Flags","emoji":"flag_gb"}}}
```

```
        />
    );
}
```

§ nodeModules.extract-zip

extract-zip 模块，提供对 zip 文件进行解压缩方法。详情请参考 https://npmjs.com/package/extract-zip 。

基本信息

等级	7
类型	函数

函数定义

- *function* **extractZip**(sourceFile, extractZipOptions, actionCallback)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
sourceFile	`string`	必须		
extractZipOptions	`object`	可选	`{dir: process.cwd(), defaultDirMode: 493, defaultFileMode: 432, onEntry: null}`	解压缩参数对象。
actionCallback	`function`	可选		操作完成回调函数，如果操作失败，第一个参数 `error` 会包含错误信息对象。例如：`(error) => {if (error) console.log('操作失败');}`。

用法

```
// 通过 `require()` 调用:
const {extractZip} = require("extract-zip");
extractZip(sourceFile, extractZipOptions, actionCallback);

// 或通过 `xext.*` 调用:
// xext.nodeModules.extractZip(sourceFile, extractZipOptions, actionCallback);
```

§ nodeModules.highlight-js

hightlight.js 模块，提供对代码片段进行高亮化方法。详情请参考 https://npmjs.com/package/highlight-js 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
highlight	`function`	
highlightAuto	`function`	
fixMarkup	`function`	
highlightBlock	`function`	
configure	`function`	
initHighlighting	`function`	
initHighlightingOnLoad	`function`	
registerLanguage	`function`	
listLanguages	`function`	
getLanguage	`function`	
inherit	`function`	
IDENT_RE	`string`	例如：`"[a-zA-Z]\\w*"`。
UNDERSCORE_IDENT_RE	`string`	例如：`"[a-zA-Z_]\\w*"`。
NUMBER_RE	`string`	例如：`"\\b\\d+(\\.\\d+)?"`。
C_NUMBER_RE	`string`	
BINARY_NUMBER_RE	`string`	例如：`"\\b(0b[01]+)"`。
RE_STARTERS_RE	`string`	
BACKSLASH_ESCAPE	`object`	例如：`{"begin":"\\\\[\\s\\S]","relevance":0}`。
APOS_STRING_MODE	`object`	
QUOTE_STRING_MODE	`object`	
PHRASAL_WORDS_MODE	`object`	例如：`{"begin":{}}`。
COMMENT	`class`	
C_LINE_COMMENT_MODE	`object`	
C_BLOCK_COMMENT_MODE	`object`	
HASH_COMMENT_MODE	`object`	
NUMBER_MODE	`object`	
C_NUMBER_MODE	`object`	
BINARY_NUMBER_MODE	`object`	
CSS_NUMBER_MODE	`object`	
REGEXP_MODE	`object`	
TITLE_MODE	`object`	
UNDERSCORE_TITLE_MODE	`object`	
METHOD_GUARD	`object`	例如：`{"begin":"\\.\\s*[a-zA-Z_]\\w*","relevance":0}`。

用法

```
// 通过 `require()` 调用:
const {highlight, highlightAuto, fixMarkup, highlightBlock, configure, initHighlighting,
initHighlightingOnLoad, registerLanguage, listLanguages, getLanguage, inherit, IDENT_RE,
UNDERSCORE_IDENT_RE, NUMBER_RE, C_NUMBER_RE, BINARY_NUMBER_RE, RE_STARTERS_RE, BACKSLASH_ESCAPE,
APOS_STRING_MODE, QUOTE_STRING_MODE, PHRASAL_WORDS_MODE, COMMENT, C_LINE_COMMENT_MODE,
C_BLOCK_COMMENT_MODE, HASH_COMMENT_MODE, NUMBER_MODE, C_NUMBER_MODE, BINARY_NUMBER_MODE, CSS_NUMBER_MODE,
REGEXP_MODE, TITLE_MODE, UNDERSCORE_TITLE_MODE, METHOD_GUARD} = require('highlight-js');

// 或者通过 `xext.*` 调用:
// const {highlight, highlightAuto, fixMarkup, highlightBlock, configure, initHighlighting,
initHighlightingOnLoad, registerLanguage, listLanguages, getLanguage, inherit, IDENT_RE,
UNDERSCORE_IDENT_RE, NUMBER_RE, C_NUMBER_RE, BINARY_NUMBER_RE, RE_STARTERS_RE, BACKSLASH_ESCAPE,
APOS_STRING_MODE, QUOTE_STRING_MODE, PHRASAL_WORDS_MODE, COMMENT, C_LINE_COMMENT_MODE,
C_BLOCK_COMMENT_MODE, HASH_COMMENT_MODE, NUMBER_MODE, C_NUMBER_MODE, BINARY_NUMBER_MODE, CSS_NUMBER_MODE,
REGEXP_MODE, TITLE_MODE, UNDERSCORE_TITLE_MODE, METHOD_GUARD} = xext.nodeModules.HighlightJS;
```

§ nodeModules.hotkeys-js

hotkeys-js 模块，提供快捷键辅助方法。详情请参考 https://npmjs.com/package/hotkeys-js 。

基本信息

等级	7
类型	函数

函数定义

- *function* **HotkeysJS**(hotkeys, eventHandler)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
hotkeys	`string`	必须		快捷键。例如：`"ctrl+a,ctrl+b,r,f"`。
eventHandler	`function`	必须		事件处理函数。例如：`(event) => {}`。

用法

```
const hotkeys = "ctrl+a,ctrl+b,r,f";
const eventHandler = (event) => {};
// 通过 `require()` 调用:
const {HotkeysJS} = require("hotkeys-js");
HotkeysJS(hotkeys, eventHandler);

// 或通过 `xext.*` 调用:
// xext.nodeModules.HotkeysJS(hotkeys, eventHandler);
```

§ nodeModules.htmlparser

htmlparser 模块，提供 HTML 片段解析方法。详情请参考 https://npmjs.com/package/htmlparser 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
Parser	`class`	
DefaultHandler	`class`	
RssHandler	`class`	
ElementType	`object`	
DomUtils	`object`	例如：`{}`。

用法

```
// 通过 `require()` 调用:
const {Parser, DefaultHandler, RssHandler, ElementType, DomUtils} = require('htmlparser');

// 或者通过 `xext.*` 调用:
// const {Parser, DefaultHandler, RssHandler, ElementType, DomUtils} = xext.nodeModules.HTMLParser;
```

§ nodeModules.jquery

jquery 模块，提供 jQuery 功能对象。详情请参考 https://npmjs.com/package/jquery 。

基本信息

等级	6
类型	模块或库
依赖	`window.document`

模块成员

成员	类型	描述和示例
fn	`object`	例如：`{"jquery":"3.3.1","length":0}`。
extend	`function`	
expando	`string`	例如：`"jQuery331003256905408316868"`。

isReady	`boolean`	例如：`false`。
error	`function`	
noop	`function`	
isPlainObject	`function`	
isEmptyObject	`function`	
globalEval	`function`	
each	`function`	
trim	`function`	
makeArray	`function`	
inArray	`function`	
merge	`function`	
grep	`function`	
map	`function`	
guid	`number`	例如：`1`。
support	`object`	
find	`function`	
expr	`object`	
unique	`function`	
uniqueSort	`function`	
text	`function`	
isXMLDoc	`function`	
contains	`function`	
escapeSelector	`function`	
filter	`function`	
Callbacks	`class`	
Deferred	`class`	
when	`function`	
readyException	`function`	
readyWait	`number`	例如：`1`。
ready	`function`	
hasData	`function`	
data	`function`	
removeData	`function`	
queue	`function`	
dequeue	`function`	
event	`object`	
removeEvent	`function`	
Event	`class`	
htmlPrefilter	`function`	
clone	`function`	
cleanData	`function`	
cssHooks	`object`	
cssNumber	`object`	
cssProps	`object`	例如：`{}`。
style	`function`	
css	`function`	
Tween	`class`	
easing	`object`	例如：`{"_default":"swing"}`。
fx	`function`	
Animation	`class`	
speed	`function`	

timers	`object`	例如：`[]`。
attr	`function`	
attrHooks	`object`	例如：`{"type":{}}`。
removeAttr	`function`	
prop	`function`	
propHooks	`object`	例如：`{"tabIndex":{}}`。
propFix	`object`	
valHooks	`object`	
parseXML	`function`	
param	`function`	
active	`number`	例如：`0`。
lastModified	`object`	例如：`{}`。
etag	`object`	例如：`{}`。
ajaxSettings	`object`	
ajaxSetup	`function`	
ajaxPrefilter	`function`	
ajaxTransport	`function`	
ajax	`function`	
getJSON	`function`	
getScript	`function`	
get	`function`	
post	`function`	
parseHTML	`function`	
offset	`object`	例如：`{}`。
proxy	`function`	
holdReady	`function`	
isArray	`function`	
parseJSON	`function`	
nodeName	`function`	
isFunction	`function`	
isWindow	`function`	
camelCase	`function`	
type	`function`	
now	`function`	
isNumeric	`function`	
noConflict	`function`	

用法

```
// 通过 `require()` 调用:
const {fn, extend, expando, isReady, error, noop, isPlainObject, isEmptyObject, globalEval, each, trim,
makeArray, inArray, merge, grep, map, guid, support, find, expr, unique, uniqueSort, text, isXMLDoc,
contains, escapeSelector, filter, Callbacks, Deferred, when, readyException, readyWait, ready, hasData,
data, removeData, queue, dequeue, event, removeEvent, Event, htmlPrefilter, clone, cleanData, cssHooks,
cssNumber, cssProps, style, css, Tween, easing, fx, Animation, speed, timers, attr, attrHooks, removeAttr,
prop, propHooks, propFix, valHooks, parseXML, param, active, lastModified, etag, ajaxSettings, ajaxSetup,
ajaxPrefilter, ajaxTransport, ajax, getJSON, getScript, get, post, parseHTML, offset, proxy, holdReady,
isArray, parseJSON, nodeName, isFunction, isWindow, camelCase, type, now, isNumeric, noConflict} =
require('jquery');

// 或者通过 `xext.*` 调用:
// const {fn, extend, expando, isReady, error, noop, isPlainObject, isEmptyObject, globalEval, each, trim,
makeArray, inArray, merge, grep, map, guid, support, find, expr, unique, uniqueSort, text, isXMLDoc,
contains, escapeSelector, filter, Callbacks, Deferred, when, readyException, readyWait, ready, hasData,
data, removeData, queue, dequeue, event, removeEvent, Event, htmlPrefilter, clone, cleanData, cssHooks,
cssNumber, cssProps, style, css, Tween, easing, fx, Animation, speed, timers, attr, attrHooks, removeAttr,
prop, propHooks, propFix, valHooks, parseXML, param, active, lastModified, etag, ajaxSettings, ajaxSetup,
ajaxPrefilter, ajaxTransport, ajax, getJSON, getScript, get, post, parseHTML, offset, proxy, holdReady,
isArray, parseJSON, nodeName, isFunction, isWindow, camelCase, type, now, isNumeric, noConflict} =
xext.nodeModules.JQuery;
```

§ nodeModules.marked

marked 模块，提供 Markdown 辅助方法。详情请参考 https://npmjs.com/package/marked 。

基本信息

等级	7
类型	函数

函数定义

- *function* **marked**(srcString, actionCallback, parseResult)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
srcString	`string`	必须		
actionCallback	`function`	可选		操作完成回调函数，如果操作失败，第一个参数 `error` 会包含错误信息对象。例如：`(error) => {if (error) console.log('操作失败');}` 。
parseResult	`string`	可选		

函数返回值

- `any`

用法

```
// 通过 `require()` 调用:
const {marked} = require("marked");
const string = marked(srcString, actionCallback, parseResult);

// 或通过 `xext.*` 调用:
// const string = xext.nodeModules.marked(srcString, actionCallback, parseResult);
```

§ nodeModules.md5

md5 模块，提供 md5 算法方法。详情请参考 https://npmjs.com/package/md5 。

基本信息

等级	7
类型	函数

函数定义

- *function* **md5**(str)

函数参数定义

参数	类型	是否必须	默认值	描述和示例
str	`string|Buffer|number[]`	可选		

函数返回值

- `string`

用法

```
// 通过 `require()` 调用:
const {md5} = require("md5");
const md5Str = md5(str);

// 或通过 `xext.*` 调用:
// const md5Str = xext.nodeModules.md5(str);
```

§ nodeModules.prop-types

prop-types，提供组件属性定义和检查辅助方法。详情请参考 https://npmjs.com/package/prop-types 。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
PropTypes.array	`function`	
PropTypes.bool	`function`	
PropTypes.func	`function`	
PropTypes.number	`function`	
PropTypes.object	`function`	
PropTypes.string	`function`	
PropTypes.symbol	`function`	
PropTypes.any	`function`	
PropTypes.arrayOf	`function`	
PropTypes.element	`function`	
PropTypes.instanceOf	`function`	
PropTypes.node	`function`	
PropTypes.objectOf	`function`	
PropTypes.oneOf	`function`	
PropTypes.oneOfType	`function`	
PropTypes.shape	`function`	
PropTypes.exact	`function`	
PropTypes.checkPropTypes	`function`	
PropTypes.PropTypes	`object`	

用法

```
// 通过 `xext.*` 调用:
const PropTypes = xext.nodeModules.PropTypes;

// 或者通过 `require()` 调用:
// const PropTypes = require('prop-types');
```

§ nodeModules.react

react 模块。详情请参考 https://npmjs.com/package/react 。

基本信息

等级	6
类型	模块或库
依赖	`window.document`

模块成员

成员	类型	描述和示例
Children	`object`	例如：`{}`。
createRef	`function`	
Component	`class`	
PureComponent	`class`	
createContext	`function`	
forwardRef	`function`	
lazy	`function`	
memo	`function`	
useCallback	`function`	
useContext	`function`	
useEffect	`function`	
useImperativeHandle	`function`	
useDebugValue	`function`	
useLayoutEffect	`function`	
useMemo	`function`	
useReducer	`function`	
useRef	`function`	
useState	`function`	
Fragment	`symbol`	例如：`undefined`。
Profiler	`symbol`	例如：`undefined`。
StrictMode	`symbol`	例如：`undefined`。
Suspense	`symbol`	例如：`undefined`。
createElement	`function`	
cloneElement	`function`	
createFactory	`function`	
isValidElement	`function`	
version	`string`	例如：`"16.11.0"`。

用法

```
// 通过 `require()` 调用:
const {Children, createRef, Component, PureComponent, createContext, forwardRef, lazy, memo, useCallback,
useContext, useEffect, useImperativeHandle, useDebugValue, useLayoutEffect, useMemo, useReducer, useRef,
useState, Fragment, Profiler, StrictMode, Suspense, createElement, cloneElement, createFactory,
isValidElement, version} = require('react');

// 或者通过 `xext.*` 调用:
// const {Children, createRef, Component, PureComponent, createContext, forwardRef, lazy, memo,
useCallback, useContext, useEffect, useImperativeHandle, useDebugValue, useLayoutEffect, useMemo,
useReducer, useRef, useState, Fragment, Profiler, StrictMode, Suspense, createElement, cloneElement,
createFactory, isValidElement, version} = xext.nodeModules.React;
```

§ nodeModules.react-dom

react-dom 模块。详情请参考 https://npmjs.com/package/react-dom 。

基本信息

等级	6
类型	模块或库
依赖	`nodeModules.react`, `window.document`

模块成员

成员	类型	描述和示例
createPortal	`function`	
findDOMNode	`function`	
hydrate	`function`	
render	`function`	
unstable_renderSubtreeIntoContainer	`function`	
unmountComponentAtNode	`function`	
unstable_createPortal	`function`	
unstable_batchedUpdates	`function`	
flushSync	`function`	

用法

```
// 通过 `require()` 调用:
const {createPortal, findDOMNode, hydrate, render, unstable_renderSubtreeIntoContainer,
unmountComponentAtNode, unstable_createPortal, unstable_batchedUpdates, flushSync} = require('react-dom');

// 或者通过 `xext.*` 调用:
// const {createPortal, findDOMNode, hydrate, render, unstable_renderSubtreeIntoContainer,
unmountComponentAtNode, unstable_createPortal, unstable_batchedUpdates, flushSync} =
xext.nodeModules.ReactDOM;
```

§ nodeModules.react-split-pane

react-split-pane 模块，提供可拖放调整布局的容器组件。详情请参考 https://npmjs.com/package/react-split-pane 。

基本信息

等级	6
类型	React 组件
依赖	`nodeModules.react`, `window.document`

组件属性

属性	类型	是否必须	默认值
allowResize	`boolean`	可选	`true`
children		可选	
className		可选	
primary	`string`	可选	`"first"`
minSize	`number`	可选	`50`
maxSize		可选	
defaultSize		可选	
size		可选	
split	`string`	可选	`"vertical"`
onDragStarted		可选	
onDragFinished		可选	
onChange		可选	
onResizerClick		可选	
onResizerDoubleClick		可选	
prefixer	`object`	可选	`{"_userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.128 Electron/4.2.12 Safari/537.36","_keepUnprefixed":false,"_browserInfo":{"name":"Chrome","chrome":true,"version":"69.0","blink":true,"mac":true,"osname":"macOS","osversion":"10.14.6","a":true,"jsPrefix":"Webkit","cssPrefix":"-webkit-","browserName":"chrome","browserVersion":69,"osVersion":10.14},"prefixedKeyframes":"keyframes","_requiresPrefix":{},"_hasPropsRequiringPrefix":false,"_metaData":{"browserVersion":69,"browserName":"chrome","cssPrefix":"-webkit-","jsPrefix":"Webkit","keepUnprefixed":false,"requiresPrefix":{}}}`
style		可选	
resizerStyle		可选	
paneClassName	`string`	可选	`""`
pane1ClassName	`string`	可选	`""`
pane2ClassName	`string`	可选	`""`
paneStyle		可选	
pane1Style		可选	
pane2Style		可选	
resizerClassName		可选	
step		可选	

用法

```
// 通过 `require()` 调用:
const ReactSplitPane = require('react-split-pane');

// 或通过 `xext.*` 调用:
// const ReactSplitPane = xext.nodeModules.ReactSplitPane;

// 使用组件:
const render = () => {
    return (
        <ReactSplitPane
            allowResize={allowResize}
            className={className}
            primary="first"
            minSize={50}
            maxSize={maxSize}
            defaultSize={defaultSize}
            size={size}
            split="vertical"
            onDragStarted={onDragStarted}
            onDragFinished={onDragFinished}
            onChange={onChange}
            onResizerClick={onResizerClick}
```

```
            onResizerDoubleClick={onResizerDoubleClick}
            prefixer={{"_userAgent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/69.0.3497.128 Electron/4.2.12
Safari/537.36","_keepUnprefixed":false,"_browserInfo":
{"name":"Chrome","chrome":true,"version":"69.0","blink":true,"mac":true,"osname":"macOS","osversion":"10.14
.6","a":true,"jsPrefix":"Webkit","cssPrefix":"-webkit-
","browserName":"chrome","browserVersion":69,"osVersion":10.14},"prefixedKeyframes":"keyframes","_requiresP
refix":{},"_hasPropsRequiringPrefix":false,"_metaData":
{"browserVersion":69,"browserName":"chrome","cssPrefix":"-webkit-
","jsPrefix":"Webkit","keepUnprefixed":false,"requiresPrefix":{}}}}
            style={style}
            resizerStyle={resizerStyle}
            paneClassName={paneClassName}
            pane1ClassName={pane1ClassName}
            pane2ClassName={pane2ClassName}
            paneStyle={paneStyle}
            pane1Style={pane1Style}
            pane2Style={pane2Style}
            resizerClassName={resizerClassName}
            step={step}
        >
            {children}
        </ReactSplitPane>
    );
}
```

§ nodeModules.tiny-pinyin

tiny-pinyin 模块，提供汉语拼音操作方法。详情请参考 https://npmjs.com/package/tiny-pinyin 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
isSupported	`function`	
parse	`function`	
patchDict	`function`	
genToken	`function`	
convertToPinyin	`function`	

用法

```
// 通过 `require()` 调用:
const {isSupported, parse, patchDict, genToken, convertToPinyin} = require('tiny-pinyin');

// 或者通过 `xext.*` 调用:
// const {isSupported, parse, patchDict, genToken, convertToPinyin} = xext.nodeModules.TinyPinyin;
```

§ nodeModules.uuid

uuid 模块，提供全局唯一字符串生成辅助方法。详情请参考 https://npmjs.com/package/uuid 。

基本信息

等级	6
类型	模块或库

模块成员

成员	类型	描述和示例
uuid.v1	`function`	
uuid.v4	`function`	

用法

```
// 通过 `xext.*` 调用:
const uuid = xext.nodeModules.uuid;

// 或者通过 `require()` 调用:
// const uuid = require('uuid');
```

node： NodeJS 内置模块

§ node.child_process

NodeJS 内置模块，子进程。详情请参考 https://nodejs.org/docs/latest-v10.x/api/child_process.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
ChildProcess	`class`	
fork	`function`	
exec	`function`	
execFile	`function`	
spawn	`function`	
spawnSync	`function`	
execFileSync	`function`	
execSync	`function`	

用法

```
// 通过 `require()` 调用:
const {ChildProcess, fork, exec, execFile, spawn, spawnSync, execFileSync, execSync} =
require('child_process');

// 或者通过 `xext.*` 调用:
// const {ChildProcess, fork, exec, execFile, spawn, spawnSync, execFileSync, execSync} =
xext.node.child_process;
```

§ node.crypto

NodeJS 内置模块，加密和解密。详情请参考 https://nodejs.org/docs/latest-v10.x/api/crypto.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
createCipher	`function`	
createCipheriv	`function`	
createDecipher	`function`	
createDecipheriv	`function`	
createDiffieHellman	`function`	
createDiffieHellmanGroup	`function`	
createECDH	`function`	

createHash	`function`	
createHmac	`function`	
createSign	`function`	
createVerify	`function`	
getCiphers	`function`	
getCurves	`function`	
getDiffieHellman	`function`	
getHashes	`function`	
pbkdf2	`function`	
pbkdf2Sync	`function`	
privateDecrypt	`function`	
privateEncrypt	`function`	
prng	`function`	
pseudoRandomBytes	`function`	
publicDecrypt	`function`	
publicEncrypt	`function`	
randomBytes	`function`	
randomFill	`function`	
randomFillSync	`function`	
rng	`function`	
scrypt	`function`	
scryptSync	`function`	
setEngine	`function`	
timingSafeEqual	`function`	
getFips	`function`	
setFips	`function`	
Certificate	`class`	
Cipher	`class`	
Cipheriv	`class`	
Decipher	`class`	
Decipheriv	`class`	
DiffieHellman	`class`	
DiffieHellmanGroup	`class`	
ECDH	`class`	
Hash	`class`	
Hmac	`class`	
Sign	`class`	
Verify	`class`	
DEFAULT_ENCODING	`string`	例如：`"buffer"`。
constants	`object`	
createCredentials	`function`	
Credentials	`class`	

用法

```
// 通过 `require()` 调用:
const {createCipher, createCipheriv, createDecipher, createDecipheriv, createDiffieHellman,
createDiffieHellmanGroup, createECDH, createHash, createHmac, createSign, createVerify, getCiphers,
getCurves, getDiffieHellman, getHashes, pbkdf2, pbkdf2Sync, privateDecrypt, privateEncrypt, prng,
pseudoRandomBytes, publicDecrypt, publicEncrypt, randomBytes, randomFill, randomFillSync, rng, scrypt,
scryptSync, setEngine, timingSafeEqual, getFips, setFips, Certificate, Cipher, Cipheriv, Decipher,
Deciphériv, DiffieHellman, DiffieHellmanGroup, ECDH, Hash, Hmac, Sign, Verify, DEFAULT_ENCODING, constants,
createCredentials, Credentials} = require('crypto');

// 或者通过 `xext.*` 调用:
// const {createCipher, createCipheriv, createDecipher, createDecipheriv, createDiffieHellman,
createDiffieHellmanGroup, createECDH, createHash, createHmac, createSign, createVerify, getCiphers,
getCurves, getDiffieHellman, getHashes, pbkdf2, pbkdf2Sync, privateDecrypt, privateEncrypt, prng,
pseudoRandomBytes, publicDecrypt, publicEncrypt, randomBytes, randomFill, randomFillSync, rng, scrypt,
scryptSync, setEngine, timingSafeEqual, getFips, setFips, Certificate, Cipher, Cipheriv, Decipher,
Deciphériv, DiffieHellman, DiffieHellmanGroup, ECDH, Hash, Hmac, Sign, Verify, DEFAULT_ENCODING, constants,
createCredentials, Credentials} = xext.node.crypto;
```

§ node.dgram

NodeJS 内置模块，数据报。详情请参考 https://nodejs.org/docs/latest-v10.x/api/dgram.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
createSocket	`function`	
Socket	`class`	

用法

```
// 通过 `require()` 调用:
const {createSocket, Socket} = require('dgram');

// 或者通过 `xext.*` 调用:
// const {createSocket, Socket} = xext.node.dgram;
```

§ node.dns

NodeJS 内置模块，域名服务器。详情请参考 https://nodejs.org/docs/latest-v10.x/api/dns.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
lookup	`function`	
lookupService	`function`	
Resolver	`class`	
setServers	`function`	
ADDRCONFIG	`number`	例如：`1024`。
V4MAPPED	`number`	例如：`2048`。
NODATA	`string`	例如：`"ENODATA"`。
FORMERR	`string`	例如：`"EFORMERR"`。
SERVFAIL	`string`	例如：`"ESERVFAIL"`。
NOTFOUND	`string`	例如：`"ENOTFOUND"`。
NOTIMP	`string`	例如：`"ENOTIMP"`。
REFUSED	`string`	例如：`"EREFUSED"`。
BADQUERY	`string`	例如：`"EBADQUERY"`。
BADNAME	`string`	例如：`"EBADNAME"`。
BADFAMILY	`string`	例如：`"EBADFAMILY"`。
BADRESP	`string`	例如：`"EBADRESP"`。
CONNREFUSED	`string`	例如：`"ECONNREFUSED"`。
TIMEOUT	`string`	例如：`"ETIMEOUT"`。
EOF	`string`	例如：`"EOF"`。
FILE	`string`	例如：`"EFILE"`。
NOMEM	`string`	例如：`"ENOMEM"`。
DESTRUCTION	`string`	例如：`"EDESTRUCTION"`。
BADSTR	`string`	例如：`"EBADSTR"`。
BADFLAGS	`string`	例如：`"EBADFLAGS"`。
NONAME	`string`	例如：`"ENONAME"`。
BADHINTS	`string`	例如：`"EBADHINTS"`。
NOTINITIALIZED	`string`	例如：`"ENOTINITIALIZED"`。
LOADIPHLPAPI	`string`	例如：`"ELOADIPHLPAPI"`。
ADDRGETNETWORKPARAMS	`string`	例如：`"EADDRGETNETWORKPARAMS"`。
CANCELLED	`string`	例如：`"ECANCELLED"`。
getServers	`function`	
resolve	`function`	
resolveAny	`function`	
resolve4	`function`	
resolve6	`function`	
resolveCname	`function`	
resolveMx	`function`	
resolveNs	`function`	
resolveTxt	`function`	
resolveSrv	`function`	
resolvePtr	`function`	
resolveNaptr	`function`	
resolveSoa	`function`	
reverse	`function`	

用法

```
// 通过 `require()` 调用:
const {lookup, lookupService, Resolver, setServers, ADDRCONFIG, V4MAPPED, NODATA, FORMERR, SERVFAIL,
NOTFOUND, NOTIMP, REFUSED, BADQUERY, BADNAME, BADFAMILY, BADRESP, CONNREFUSED, TIMEOUT, EOF, FILE, NOMEM,
DESTRUCTION, BADSTR, BADFLAGS, NONAME, BADHINTS, NOTINITIALIZED, LOADIPHLPAPI, ADDRGETNETWORKPARAMS,
CANCELLED, getServers, resolve, resolveAny, resolve4, resolve6, resolveCname, resolveMx, resolveNs,
resolveTxt, resolveSrv, resolvePtr, resolveNaptr, resolveSoa, reverse} = require('dns');

// 或者通过 `xext.*` 调用:
// const {lookup, lookupService, Resolver, setServers, ADDRCONFIG, V4MAPPED, NODATA, FORMERR, SERVFAIL,
NOTFOUND, NOTIMP, REFUSED, BADQUERY, BADNAME, BADFAMILY, BADRESP, CONNREFUSED, TIMEOUT, EOF, FILE, NOMEM,
DESTRUCTION, BADSTR, BADFLAGS, NONAME, BADHINTS, NOTINITIALIZED, LOADIPHLPAPI, ADDRGETNETWORKPARAMS,
CANCELLED, getServers, resolve, resolveAny, resolve4, resolve6, resolveCname, resolveMx, resolveNs,
resolveTxt, resolveSrv, resolvePtr, resolveNaptr, resolveSoa, reverse} = xext.node.dns;
```

§ node.events

NodeJS 内置模块，事件。详情请参考 https://nodejs.org/docs/latest-v10.x/api/events.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
EventEmitter	`class`	
usingDomains	`boolean`	例如：`false`。
defaultMaxListeners	`number`	例如：`10`。
init	`function`	
listenerCount	`function`	

用法

```
// 通过 `require()` 调用:
const {EventEmitter, usingDomains, defaultMaxListeners, init, listenerCount} = require('events');

// 或者通过 `xext.*` 调用:
// const {EventEmitter, usingDomains, defaultMaxListeners, init, listenerCount} = xext.node.events;
```

§ node.fs

NodeJS 内置模块，文件读写。详情请参考 https://nodejs.org/docs/latest-v10.x/api/fs.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
appendFile	`function`	
appendFileSync	`function`	
access	`function`	
accessSync	`function`	
chown	`function`	
chownSync	`function`	
chmod	`function`	
chmodSync	`function`	
close	`function`	

closeSync	`function`	
copyFile	`function`	
copyFileSync	`function`	
createReadStream	`function`	
createWriteStream	`function`	
exists	`function`	
existsSync	`function`	
fchown	`function`	
fchownSync	`function`	
fchmod	`function`	
fchmodSync	`function`	
fdatasync	`function`	
fdatasyncSync	`function`	
fstat	`function`	
fstatSync	`function`	
fsync	`function`	
fsyncSync	`function`	
ftruncate	`function`	
ftruncateSync	`function`	
futimes	`function`	
futimesSync	`function`	
lchown	`function`	
lchownSync	`function`	
lchmod	`function`	
lchmodSync	`function`	
link	`function`	
linkSync	`function`	
lstat	`function`	
lstatSync	`function`	
mkdir	`function`	
mkdirSync	`function`	
mkdtemp	`function`	
mkdtempSync	`function`	
open	`function`	
openSync	`function`	
readdir	`function`	
readdirSync	`function`	
read	`function`	
readSync	`function`	
readFile	`function`	
readFileSync	`function`	
readlink	`function`	

readlinkSync	`function`	
realpath	`function`	
realpathSync	`function`	
rename	`function`	
renameSync	`function`	
rmdir	`function`	
rmdirSync	`function`	
stat	`function`	
statSync	`function`	
symlink	`function`	
symlinkSync	`function`	
truncate	`function`	
truncateSync	`function`	
unwatchFile	`function`	
unlink	`function`	
unlinkSync	`function`	
utimes	`function`	
utimesSync	`function`	
watch	`function`	
watchFile	`function`	
writeFile	`function`	
writeFileSync	`function`	
write	`function`	
writeSync	`function`	
Dirent	`class`	
Stats	`class`	
ReadStream	`class`	
WriteStream	`class`	
FileReadStream	`class`	
FileWriteStream	`class`	
F_OK	`number`	例如：`0`。
R_OK	`number`	例如：`4`。
W_OK	`number`	例如：`2`。
X_OK	`number`	例如：`1`。
constants	`object`	

用法

```
// 通过 `require()` 调用:
const {appendFile, appendFileSync, access, accessSync, chown, chownSync, chmod, chmodSync, close,
closeSync, copyFile, copyFileSync, createReadStream, createWriteStream, exists, existsSync, fchown,
fchownSync, fchmod, fchmodSync, fdatasync, fdatasyncSync, fstat, fstatSync, fsync, fsyncSync, ftruncate,
ftruncateSync, futimes, futimesSync, lchown, lchownSync, lchmod, lchmodSync, link, linkSync, lstat,
lstatSync, mkdir, mkdirSync, mkdtemp, mkdtempSync, open, openSync, readdir, readdirSync, read, readSync,
readFile, readFileSync, readlink, readlinkSync, realpath, realpathSync, rename, renameSync, rmdir,
rmdirSync, stat, statSync, symlink, symlinkSync, truncate, truncateSync, unwatchFile, unlink, unlinkSync,
utimes, utimesSync, watch, watchFile, writeFile, writeFileSync, write, writeSync, Dirent, Stats,
ReadStream, WriteStream, FileReadStream, FileWriteStream, F_OK, R_OK, W_OK, X_OK, constants} =
require('fs');

// 或者通过 `xext.*` 调用:
// const {appendFile, appendFileSync, access, accessSync, chown, chownSync, chmod, chmodSync, close,
closeSync, copyFile, copyFileSync, createReadStream, createWriteStream, exists, existsSync, fchown,
fchownSync, fchmod, fchmodSync, fdatasync, fdatasyncSync, fstat, fstatSync, fsync, fsyncSync, ftruncate,
ftruncateSync, futimes, futimesSync, lchown, lchownSync, lchmod, lchmodSync, link, linkSync, lstat,
lstatSync, mkdir, mkdirSync, mkdtemp, mkdtempSync, open, openSync, readdir, readdirSync, read, readSync,
readFile, readFileSync, readlink, readlinkSync, realpath, realpathSync, rename, renameSync, rmdir,
rmdirSync, stat, statSync, symlink, symlinkSync, truncate, truncateSync, unwatchFile, unlink, unlinkSync,
utimes, utimesSync, watch, watchFile, writeFile, writeFileSync, write, writeSync, Dirent, Stats,
ReadStream, WriteStream, FileReadStream, FileWriteStream, F_OK, R_OK, W_OK, X_OK, constants} =
xext.node.fs;
```

§ node.global.process

NodeJS global.process 对象。详情请参考 https://nodejs.org/docs/latest-v10.x/api/undefined.html 。

基本信息

等级	7
类型	数据

数据类型

- `object`：NodeJS global.process 对象。

用法

```
// 通过 `xext.*` 调用:
const process = xext.node.global.process;

// 或通过 `require()` 调用:
// const process = require('xext/node/global');
```

§ node.http

NodeJS 内置模块，HTTP。详情请参考 https://nodejs.org/docs/latest-v10.x/api/http.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
METHODS	`object`	
STATUS_CODES	`object`	
Agent	`class`	
ClientRequest	`class`	
globalAgent	`object`	
IncomingMessage	`class`	
OutgoingMessage	`class`	
Server	`class`	
ServerResponse	`class`	
createServer	`function`	
get	`function`	
request	`function`	

用法

```
// 通过 `require()` 调用:
const {METHODS, STATUS_CODES, Agent, ClientRequest, globalAgent, IncomingMessage, OutgoingMessage, Server,
ServerResponse, createServer, get, request} = require('http');

// 或者通过 `xext.*` 调用:
// const {METHODS, STATUS_CODES, Agent, ClientRequest, globalAgent, IncomingMessage, OutgoingMessage,
Server, ServerResponse, createServer, get, request} = xext.node.http;
```

§ node.http2

NodeJS 内置模块，HTTP/2。详情请参考 https://nodejs.org/docs/latest-v10.x/api/http2.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
constants	`object`	
getDefaultSettings	`function`	
getPackedSettings	`function`	
getUnpackedSettings	`function`	
createServer	`function`	
createSecureServer	`function`	
connect	`function`	
Http2ServerResponse	`class`	
Http2ServerRequest	`class`	

用法

```
// 通过 `require()` 调用:
const {constants, getDefaultSettings, getPackedSettings, getUnpackedSettings, createServer,
createSecureServer, connect, Http2ServerResponse, Http2ServerRequest} = require('http2');

// 或者通过 `xext.*` 调用:
// const {constants, getDefaultSettings, getPackedSettings, getUnpackedSettings, createServer,
createSecureServer, connect, Http2ServerResponse, Http2ServerRequest} = xext.node.http2;
```

§ node.https

NodeJS 内置模块，HTTPS。详情请参考 https://nodejs.org/docs/latest-v10.x/api/https.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
Agent	`class`	
globalAgent	`object`	
Server	`class`	
createServer	`function`	
get	`function`	
request	`function`	

用法

```
// 通过 `require()` 调用:
const {Agent, globalAgent, Server, createServer, get, request} = require('https');

// 或者通过 `xext.*` 调用:
// const {Agent, globalAgent, Server, createServer, get, request} = xext.node.https;
```

§ node.net

NodeJS 内置模块，网络。详情请参考 https://nodejs.org/docs/latest-v10.x/api/net.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
connect	`function`	
createConnection	`function`	
createServer	`function`	
isIP	`function`	
isIPv4	`function`	
isIPv6	`function`	
Server	`class`	
Socket	`class`	
Stream	`class`	

用法

```
// 通过 `require()` 调用:
const {connect, createConnection, createServer, isIP, isIPv4, isIPv6, Server, Socket, Stream} =
require('net');

// 或者通过 `xext.*` 调用:
// const {connect, createConnection, createServer, isIP, isIPv4, isIPv6, Server, Socket, Stream} =
xext.node.net;
```

§ node.os

NodeJS 内置模块，操作系统。详情请参考 https://nodejs.org/docs/latest-v10.x/api/os.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
arch	`function`	
cpus	`function`	
endianness	`function`	
freemem	`function`	
getPriority	`function`	
homedir	`function`	
hostname	`function`	
loadavg	`function`	
networkInterfaces	`function`	
platform	`function`	
release	`function`	
setPriority	`function`	
tmpdir	`function`	
totalmem	`function`	
type	`function`	
userInfo	`function`	
uptime	`function`	
getNetworkInterfaces	`function`	
tmpDir	`function`	
constants	`object`	
EOL	`string`	例如：`"\n"`。

用法

```
// 通过 `require()` 调用：
const {arch, cpus, endianness, freemem, getPriority, homedir, hostname, loadavg, networkInterfaces,
platform, release, setPriority, tmpdir, totalmem, type, userInfo, uptime, getNetworkInterfaces, tmpDir,
constants, EOL} = require('os');

// 或者通过 `xext.*` 调用：
// const {arch, cpus, endianness, freemem, getPriority, homedir, hostname, loadavg, networkInterfaces,
platform, release, setPriority, tmpdir, totalmem, type, userInfo, uptime, getNetworkInterfaces, tmpDir,
constants, EOL} = xext.node.os;
```

§ node.path

NodeJS 内置模块，路径。详情请参考 https://nodejs.org/docs/latest-v10.x/api/path.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
resolve	`function`	
normalize	`function`	
isAbsolute	`function`	
join	`function`	
relative	`function`	
toNamespacedPath	`function`	
dirname	`function`	
basename	`function`	
extname	`function`	
format	`function`	
parse	`function`	
sep	`string`	例如：`"/"`。
delimiter	`string`	例如：`":"`。
win32	`object`	
posix	`object`	

用法

```
// 通过 `require()` 调用:
const {resolve, normalize, isAbsolute, join, relative, toNamespacedPath, dirname, basename, extname,
format, parse, sep, delimiter, win32, posix} = require('path');

// 或者通过 `xext.*` 调用:
// const {resolve, normalize, isAbsolute, join, relative, toNamespacedPath, dirname, basename, extname,
format, parse, sep, delimiter, win32, posix} = xext.node.path;
```

§ node.string_decoder

NodeJS 内置模块，字符串解码器。详情请参考 https://nodejs.org/docs/latest-v10.x/api/string_decoder.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
StringDecoder	`class`	

用法

```
// 通过 `require()` 调用:
const {StringDecoder} = require('string_decoder');

// 或者通过 `xext.*` 调用:
// const {StringDecoder} = xext.node.string_decoder;
```

§ node.url

NodeJS 内置模块，URL 解析。详情请参考 https://nodejs.org/docs/latest-v10.x/api/url.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
Url	`class`	
parse	`function`	
resolve	`function`	
resolveObject	`function`	
format	`function`	
URL	`class`	
URLSearchParams	`class`	
domainToASCII	`function`	
domainToUnicode	`function`	

用法

```
// 通过 `require()` 调用:
const {Url, parse, resolve, resolveObject, format, URL, URLSearchParams, domainToASCII, domainToUnicode} =
require('url');

// 或者通过 `xext.*` 调用:
// const {Url, parse, resolve, resolveObject, format, URL, URLSearchParams, domainToASCII, domainToUnicode}
= xext.node.url;
```

§ node.util

NodeJS 实用工具库。详情请参考 https://nodejs.org/docs/latest-v10.x/api/util.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
callbackify	`function`	
debuglog	`function`	
deprecate	`function`	
format	`function`	
formatWithOptions	`function`	
getSystemErrorName	`function`	
inherits	`function`	
inspect	`function`	
isArray	`function`	
isBoolean	`function`	
isBuffer	`function`	
isDeepStrictEqual	`function`	
isNull	`function`	
isNullOrUndefined	`function`	
isNumber	`function`	
isString	`function`	
isSymbol	`function`	
isUndefined	`function`	
isRegExp	`function`	
isObject	`function`	
isDate	`function`	
isError	`function`	
isFunction	`function`	
isPrimitive	`function`	
log	`function`	
promisify	`function`	
TextDecoder	`class`	
TextEncoder	`class`	
types	`object`	例如：`{}`。
debug	`function`	
error	`function`	
print	`function`	
puts	`function`	

用法

```
// 通过 `require()` 调用:
const {callbackify, debuglog, deprecate, format, formatWithOptions, getSystemErrorName, inherits, inspect,
isArray, isBoolean, isBuffer, isDeepStrictEqual, isNull, isNullOrUndefined, isNumber, isString, isSymbol,
isUndefined, isRegExp, isObject, isDate, isError, isFunction, isPrimitive, log, promisify, TextDecoder,
TextEncoder, types, debug, error, print, puts} = require('util');

// 或者通过 `xext.*` 调用:
// const {callbackify, debuglog, deprecate, format, formatWithOptions, getSystemErrorName, inherits,
inspect, isArray, isBoolean, isBuffer, isDeepStrictEqual, isNull, isNullOrUndefined, isNumber, isString,
isSymbol, isUndefined, isRegExp, isObject, isDate, isError, isFunction, isPrimitive, log, promisify,
TextDecoder, TextEncoder, types, debug, error, print, puts} = xext.node.util;
```

§ node.zlib

NodeJS 内置模块，压缩。详情请参考 https://nodejs.org/docs/latest-v10.x/api/zlib.html 。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
Deflate	`class`	
Inflate	`class`	
Gzip	`class`	
Gunzip	`class`	
DeflateRaw	`class`	
InflateRaw	`class`	
Unzip	`class`	
deflate	`function`	
deflateSync	`function`	
gzip	`function`	
gzipSync	`function`	
deflateRaw	`function`	
deflateRawSync	`function`	
unzip	`function`	
unzipSync	`function`	
inflate	`function`	
inflateSync	`function`	
gunzip	`function`	
gunzipSync	`function`	
inflateRaw	`function`	
inflateRawSync	`function`	
createDeflate	`function`	
createInflate	`function`	
createDeflateRaw	`function`	
createInflateRaw	`function`	
createGzip	`function`	
createGunzip	`function`	

createUnzip	`function`	
constants	`object`	
codes	`object`	
Z_NO_FLUSH	`number`	例如：`0`。
Z_PARTIAL_FLUSH	`number`	例如：`1`。
Z_SYNC_FLUSH	`number`	例如：`2`。
Z_FULL_FLUSH	`number`	例如：`3`。
Z_FINISH	`number`	例如：`4`。
Z_BLOCK	`number`	例如：`5`。
Z_OK	`number`	例如：`0`。
Z_STREAM_END	`number`	例如：`1`。
Z_NEED_DICT	`number`	例如：`2`。
Z_ERRNO	`number`	例如：`-1`。
Z_STREAM_ERROR	`number`	例如：`-2`。
Z_DATA_ERROR	`number`	例如：`-3`。
Z_MEM_ERROR	`number`	例如：`-4`。
Z_BUF_ERROR	`number`	例如：`-5`。
Z_VERSION_ERROR	`number`	例如：`-6`。
Z_NO_COMPRESSION	`number`	例如：`0`。
Z_BEST_SPEED	`number`	例如：`1`。
Z_BEST_COMPRESSION	`number`	例如：`9`。
Z_DEFAULT_COMPRESSION	`number`	例如：`-1`。
Z_FILTERED	`number`	例如：`1`。
Z_HUFFMAN_ONLY	`number`	例如：`2`。
Z_RLE	`number`	例如：`3`。
Z_FIXED	`number`	例如：`4`。
Z_DEFAULT_STRATEGY	`number`	例如：`0`。
ZLIB_VERNUM	`number`	例如：`4784`。
DEFLATE	`number`	例如：`1`。
INFLATE	`number`	例如：`2`。
GZIP	`number`	例如：`3`。
GUNZIP	`number`	例如：`4`。
DEFLATERAW	`number`	例如：`5`。
INFLATERAW	`number`	例如：`6`。
UNZIP	`number`	例如：`7`。
Z_MIN_WINDOWBITS	`number`	例如：`8`。
Z_MAX_WINDOWBITS	`number`	例如：`15`。
Z_DEFAULT_WINDOWBITS	`number`	例如：`15`。
Z_MIN_CHUNK	`number`	例如：`64`。
Z_MAX_CHUNK	`number`	例如：`null`。
Z_DEFAULT_CHUNK	`number`	例如：`16384`。
Z_MIN_MEMLEVEL	`number`	例如：`1`。

Z_MAX_MEMLEVEL	`number`	例如：`9`。
Z_DEFAULT_MEMLEVEL	`number`	例如：`8`。
Z_MIN_LEVEL	`number`	例如：`-1`。
Z_MAX_LEVEL	`number`	例如：`9`。
Z_DEFAULT_LEVEL	`number`	例如：`-1`。

用法

```
// 通过 `require()` 调用:
const {Deflate, Inflate, Gzip, Gunzip, DeflateRaw, InflateRaw, Unzip, deflate, deflateSync, gzip, gzipSync,
deflateRaw, deflateRawSync, unzip, unzipSync, inflate, inflateSync, gunzip, gunzipSync, inflateRaw,
inflateRawSync, createDeflate, createInflate, createDeflateRaw, createInflateRaw, createGzip, createGunzip,
createUnzip, constants, codes, Z_NO_FLUSH, Z_PARTIAL_FLUSH, Z_SYNC_FLUSH, Z_FULL_FLUSH, Z_FINISH, Z_BLOCK,
Z_OK, Z_STREAM_END, Z_NEED_DICT, Z_ERRNO, Z_STREAM_ERROR, Z_DATA_ERROR, Z_MEM_ERROR, Z_BUF_ERROR,
Z_VERSION_ERROR, Z_NO_COMPRESSION, Z_BEST_SPEED, Z_BEST_COMPRESSION, Z_DEFAULT_COMPRESSION, Z_FILTERED,
Z_HUFFMAN_ONLY, Z_RLE, Z_FIXED, Z_DEFAULT_STRATEGY, ZLIB_VERNUM, DEFLATE, INFLATE, GZIP, GUNZIP,
DEFLATERAW, INFLATERAW, UNZIP, Z_MIN_WINDOWBITS, Z_MAX_WINDOWBITS, Z_DEFAULT_WINDOWBITS, Z_MIN_CHUNK,
Z_MAX_CHUNK, Z_DEFAULT_CHUNK, Z_MIN_MEMLEVEL, Z_MAX_MEMLEVEL, Z_DEFAULT_MEMLEVEL, Z_MIN_LEVEL, Z_MAX_LEVEL,
Z_DEFAULT_LEVEL} = require('zlib');

// 或者通过 `xext.*` 调用:
// const {Deflate, Inflate, Gzip, Gunzip, DeflateRaw, InflateRaw, Unzip, deflate, deflateSync, gzip,
gzipSync, deflateRaw, deflateRawSync, unzip, unzipSync, inflate, inflateSync, gunzip, gunzipSync,
inflateRaw, inflateRawSync, createDeflate, createInflate, createDeflateRaw, createInflateRaw, createGzip,
createGunzip, createUnzip, constants, codes, Z_NO_FLUSH, Z_PARTIAL_FLUSH, Z_SYNC_FLUSH, Z_FULL_FLUSH,
Z_FINISH, Z_BLOCK, Z_OK, Z_STREAM_END, Z_NEED_DICT, Z_ERRNO, Z_STREAM_ERROR, Z_DATA_ERROR, Z_MEM_ERROR,
Z_BUF_ERROR, Z_VERSION_ERROR, Z_NO_COMPRESSION, Z_BEST_SPEED, Z_BEST_COMPRESSION, Z_DEFAULT_COMPRESSION,
Z_FILTERED, Z_HUFFMAN_ONLY, Z_RLE, Z_FIXED, Z_DEFAULT_STRATEGY, ZLIB_VERNUM, DEFLATE, INFLATE, GZIP,
GUNZIP, DEFLATERAW, INFLATERAW, UNZIP, Z_MIN_WINDOWBITS, Z_MAX_WINDOWBITS, Z_DEFAULT_WINDOWBITS,
Z_MIN_CHUNK, Z_MAX_CHUNK, Z_DEFAULT_CHUNK, Z_MIN_MEMLEVEL, Z_MAX_MEMLEVEL, Z_DEFAULT_MEMLEVEL, Z_MIN_LEVEL,
Z_MAX_LEVEL, Z_DEFAULT_LEVEL} = xext.node.zlib;
```

lang： 界面语言文本管理对象

§ lang

界面语言文本管理对象。

基本信息

等级	7
类型	模块或库

模块成员

成员	类型	描述和示例
lang.onLangChange	`function`	
lang.isJustLangSwitched	`function`	
lang.getAllLangList	`function`	
lang.getLangDisplayName	`function`	
lang.describeDateFromNow	`function`	
lang.isSupportMultiLanguages	`function`	
lang.getMediaPath	`function`	
lang.getStringFromObject	`function`	

用法

```
// 通过 `xext.*` 调用:
const lang = xext.lang;

// 或者通过 `require()` 调用:
// const lang = require('lang');
```

env： 访问运行环境相关信息

§ env.arch

获取操作系统架构类型。

基本信息

等级	5
类型	数据

数据类型

- `string`：操作系统架构类型。例如：`"x64"`。

用法

```
// 通过 `xext.*` 调用：
const arch = xext.env.arch;

// 或通过 `require()` 调用：
// const arch = require('xext/env');
```

§ env.chrome

获取所使用的 Chrome 版本。

基本信息

等级	7
类型	数据

数据类型

- `string`：使用的 Chrome 版本。例如：`"69.0.3497.128"`。

用法

```
// 通过 `xext.*` 调用：
const chromeVersion = xext.env.chrome;

// 或通过 `require()` 调用：
// const chromeVersion = require('xext/env');
```

§ env.displayName

获取客户端显示名称。

基本信息

等级	5
类型	数据

数据类型

- `string`：客户端显示名称。例如：`"喧喧"`。

用法

```
// 通过 `xext.*` 调用：
const displayName = xext.env.displayName;

// 或通过 `require()` 调用：
// const displayName = require('xext/env');
```

§ env.displayVersion

获取客户端显示的版本。

基本信息

等级	5
类型	数据

数据类型

- `string`：客户端版显示的本号。例如：`"3.1"`。

用法

```
// 通过 `xext.*` 调用：
const displayVersion = xext.env.displayVersion;

// 或通过 `require()` 调用：
// const displayVersion = require('xext/env');
```

§ env.electron

获取用户所使用的 Electron 版本。

基本信息

等级	7
类型	数据

数据类型

- `string`：使用的 Electron 版本。例如：`"4.2.12"`。

用法

```
// 通过 `xext.*` 调用：
const electronVersion = xext.env.electron;

// 或通过 `require()` 调用：
// const electronVersion = require('xext/env');
```

§ env.lang

获取用户所使用的界面语言类型。

基本信息

等级	5
类型	数据

数据类型

- `string`

用法

```
// 通过 `xext.*` 调用：
const langName = xext.env.lang;

// 或通过 `require()` 调用：
// const langName = require('xext/env');
```

§ env.modules

获取所使用的 node modules 版本。

基本信息

等级	7
类型	数据

数据类型

- `string`：使用的 Node modules 版本。例如：`"69"`。

用法

```
// 通过 `xext.*` 调用：
const modulesVersion = xext.env.modules;

// 或通过 `require()` 调用：
// const modulesVersion = require('xext/env');
```

§ env.node

获取所使用的 node 版本。

基本信息

等级	7
类型	数据

数据类型

- `string`：使用的 Node 版本。例如：`"10.11.0"`。

用法

```
// 通过 `xext.*` 调用：
const nodeVersion = xext.env.node;

// 或通过 `require()` 调用：
// const nodeVersion = require('xext/env');
```

§ env.os

获取操作系统类型。

基本信息

等级	5
类型	数据

数据类型

- `string`：操作系统类型。例如：`"mac"`。

用法

```
// 通过 `xext.*` 调用：
const os = xext.env.os;

// 或通过 `require()` 调用：
// const os = require('xext/env');
```

§ env.productName

获取客户端产品名称。

基本信息

等级	5
类型	数据

数据类型

- `string`

用法

```
// 通过 `xext.*` 调用：
const productName = xext.env.productName;

// 或通过 `require()` 调用：
// const productName = require('xext/env');
```

§ env.v8

获取所使用的 v8 版本。

基本信息

等级	7
类型	数据

数据类型

- `string`：使用的 v8 版本。例如：`"6.9.427.31-electron.0"`。

用法

```
// 通过 `xext.*` 调用：
const v8Version = xext.env.v8;

// 或通过 `require()` 调用：
// const v8Version = require('xext/env');
```

§ env.version

获取客户端版本。

基本信息

等级	5
类型	数据

数据类型

- `string`：客户端版本号。例如：`"3.1.0"`。

用法

```
// 通过 `xext.*` 调用：
const version = xext.env.version;

// 或通过 `require()` 调用：
// const version = require('xext/env');
```

platform：访问平台提供的实用工具

§ platform

访问平台提供的实用工具。

基本信息

等级	7
类型	数据

用法

```
// 通过 `require()` 调用：
const platform = require('platform');

// 或通过 `xext.*` 调用：
// const platform = xext.platform;
```

API 中的数据类型

在 API 中使用或获得的数据有些是特定的类型，包括 `User`、`Member`、`Chat`、`ChatMessage`、`Extension`、`AppExtension`。下面分别说明不同类型数据上可访问的属性。

`User`

当前用户。

属性	类型	描述和示例
id	`number`	只读属性，ID。
account	`string`	只读属性，成员账号。
email	`string`	只读属性，电子邮件。
phone	`string`	只读属性，电话。
mobile	`string`	只读属性，移动电话。
realname	`string`	只读属性，真实姓名。
site	`string`	只读属性，站点。
avatar	`string`	只读属性，头像。
role	`string`	只读属性，角色。
gender	`string`	只读属性，性别。
dept	`string`	只读属性，部门 ID。
admin	`string`	只读属性，是否为管理员。
deleted	`boolean`	只读属性，是否已删除。
displayName	`string`	只读属性，显示名称。
namePinyin	`string`	只读属性，拼音字符串。
lastLoginTime	`number`	只读属性，上次登录的时间戳。
server	`string`	只读属性，服务器地址。
serverVersion	`string`	只读属性，服务器版本。
uploadFileSize	`number`	只读属性，文件上传最大限制。
signed	`boolean`	只读属性，是否签到。
backendType	`string`	只读属性，后端服务器类型。
serverNowTime	`Date`	只读属性，当前服务器时间。

`Member`

组织成员。

属性	类型	描述和示例
id	`number`	只读属性，ID。
account	`string`	只读属性，成员账号。
email	`string`	只读属性，电子邮件。
phone	`string`	只读属性，电话。
mobile	`string`	只读属性，移动电话。
realname	`string`	只读属性，真实姓名。
site	`string`	只读属性，站点。
avatar	`string`	只读属性，头像。
role	`string`	只读属性，角色。
gender	`string`	只读属性，性别。
dept	`string`	只读属性，部门 ID。
admin	`string`	只读属性，是否为管理员。
deleted	`boolean`	只读属性，是否已删除。
displayName	`string`	只读属性，显示名称。
namePinyin	`string`	只读属性，拼音字符串。

`Chat`

会话信息。

属性	类型	描述和示例
gid	`string`	只读属性，GID 属性。
id	`number`	只读属性，ID。
type	`string`	只读属性，会话类型。
name	`string`	只读属性，讨论组名称。
createdDate	`number`	只读属性，创建时间。
createdBy	`string`	只读属性，创建者。
editedDate	`number`	只读属性，编辑时间。
lastActiveTime	`number`	只读属性，上次激活的时间。
dismissDate	`number`	只读属性，解散时间。
star	`boolean`	只读属性，是否标记为收藏。
mute	`boolean`	只读属性，是否开启免打扰。
public	`boolean`	只读属性，是否为公开讨论组。
admins	`string`	只读属性，管理员。
members	`string`	只读属性，成员 ID 列表。
committers	`string`	只读属性，白名单类型。
category	`string`	只读属性，分组信息。
freeze	`boolean`	只读属性，是否已从最近会话列表移除。
noticeCount	`number`	只读属性，未读消息数目。

`ChatMessage`

会话聊天消息。

属性	类型	描述和示例
gid	`string`	只读属性，GID。
id	`number`	只读属性，ID。
cgid	`string`	只读属性，会话 GID。
user	`number`	只读属性，发送者 ID。
date	`number`	只读属性，发送时间戳。
type	`string`	只读属性，消息类型。
contentType	`string`	只读属性，消息内容类型。
content	`string`	只读属性，消息内容原始信息。
unread	`boolean`	只读属性，是否未读。
deleted	`boolean`	只读属性，是否已删除。
data	`object`	只读属性，附加数据。
codeContent	`object`	只读属性，代码内容对象。
fileContent	`object`	只读属性，文件内容对象。
imageContent	`object`	只读属性，图片内容对象。
objectContentType	`string`	只读属性，对象内容类型。
objectContent	`string`	只读属性，对象内容对象。

`Extension`

扩展信息。

属性	类型	描述和示例
name	`string`	只读属性，内部名称。
displayName	`string`	只读属性，显示名称。
type	`string`	只读属性，类型。
isTheme	`boolean`	只读属性，是否为主题类型。
isApp	`boolean`	只读属性，是否为应用类型。
isPlugin	`boolean`	只读属性，是否为插件类型。
accentColor	`string`	只读属性，主题色。
icon	`string`	只读属性，图标。
description	`string`	只读属性，描述。
version	`string`	只读属性，版本号。
author	`object`	只读属性，作者信息。
publisher	`object`	只读属性，发布者信息。
category	`string`	只读属性，分组。
company	`string`	只读属性，开发公司。
homepage	`string`	只读属性，扩展网站主页。
keywords	`string`	只读属性，关键字。
repository	`object`	只读属性，源码版本库信息。
bugs	`string`	只读属性，Bugs 反馈。
installTime	`number`	只读属性，安装时间。
updateTime	`number`	只读属性，上次更新时间。
hot	`boolean`	只读属性，是否支持热加载。
serverData	`any`	只读属性，服务器提供的数据。
isRemote	`boolean`	只读属性，是否为远程扩展。
isDev	`boolean`	只读属性，是否为开发中的扩展。
getConfig	`function`	方法，获取扩展配置数据。用法：`const configValue = extension.getConfig('customData')`。
setConfig	`function`	方法，设置扩展配置数据。用法：`const configValue = extension.setConfig('customData', {myCustomData: 'test'})`。
getUserConfig	`function`	方法，获取当前用户数据。用法：`const userConfigValue = extension.getUserConfig('customData')`。
setUserConfig	`function`	方法，设置当前用户数据。用法：`const userConfigValue = extension.setUserConfig('customData', {myCustomData: 'test'})`。
getAbsolutePath	`function`	方法，获取基于当前扩展目录的完整路径。用法：`const myDataFilePath = extension.getAbsolutePath('./myData.json')`。

`AppExtension`

应用扩展信息。

属性	类型	描述和示例
name	`string`	只读属性，内部名称。
displayName	`string`	只读属性，显示名称。
type	`string`	只读属性，类型。
isTheme	`boolean`	只读属性，是否为主题类型。
isApp	`boolean`	只读属性，是否为应用类型。
isPlugin	`boolean`	只读属性，是否为插件类型。
accentColor	`string`	只读属性，主题色。
icon	`string`	只读属性，图标。
description	`string`	只读属性，描述。
version	`string`	只读属性，版本号。
author	`object`	只读属性，作者信息。
publisher	`object`	只读属性，发布者信息。
category	`string`	只读属性，分组。
company	`string`	只读属性，开发公司。
homepage	`string`	只读属性，扩展网站主页。
keywords	`string`	只读属性，关键字。
repository	`object`	只读属性，源码版本库信息。
bugs	`string`	只读属性，Bugs 反馈。
installTime	`number`	只读属性，安装时间。
updateTime	`number`	只读属性，上次更新时间。
hot	`boolean`	只读属性，是否支持热加载。
serverData	`any`	只读属性，服务器提供的数据。
isRemote	`boolean`	只读属性，是否为远程扩展。
isDev	`boolean`	只读属性，是否为开发中的扩展。
getConfig	`function`	方法，获取扩展配置数据。用法：`const configValue = extension.getConfig('customData')`。
setConfig	`function`	方法，设置扩展配置数据。用法：`const configValue = extension.setConfig('customData', {myCustomData: 'test'})`。
getUserConfig	`function`	方法，获取当前用户数据。用法：`const userConfigValue = extension.getUserConfig('customData')`。
setUserConfig	`function`	方法，设置当前用户数据。用法：`const userConfigValue = extension.setUserConfig('customData', {myCustomData: 'test'})`。
getAbsolutePath	`function`	方法，获取基于当前扩展目录的完整路径。用法：`const myDataFilePath = extension.getAbsolutePath('./myData.json')`。
isWebview	`boolean`	只读属性，是否为 Webview 类型。
isCustomApp	`boolean`	只读属性，是否为自定义应用。
webViewUrl	`string`	只读属性，Webview 应用地址。
injectCSS	`string`	只读属性，向 Webview 注入的 CSS 代码。
injectScript	`string`	只读属性，向 Webview 注入的 JavaScript 代码。
isLocalWebView	`boolean`	只读属性，是否为本地 Webview。
appAccentColor	`string`	只读属性，应用主题色。
appBackColor	`string`	只读属性，应用背景色。
appIcon	`string`	只读属性，应用图标。
menuIcon	`string`	只读属性，应用在导航菜单上的图标。
hidden	`boolean`	只读属性，此应用是否在应用列表隐藏。
canPinnedOnMenu	`boolean`	只读属性，此应用是否能固定在导航上。
pinnedOnMainMenu	`boolean`	只读属性，此应用是否固定在导航上。
noticeCount	`number`	只读属性，此应用未读标记数目。
hasNotice	`boolean`	只读属性，此应用是否有未读标记。
muteNoticeOnActive	`boolean`	只读属性，此应用是否在激活时自动清除未读标记。