API - `$e.components`

目录

- 方法__
- Examples
- Guidelines

新的组件 API（自 2.7.0 版本起），提供了一种简单便捷的方式，将属于一个 UI 组件的所有路由、命令和键盘快捷键绑定到一个控制器中。

完整的组件列表，包括自定义和第三方路由，可通过 `$e.components.getAll();` 获取。每个组件都有其唯一的命名空间，其所有命令和路由都嵌套在该命名空间下。

当路由指向组件路由，且组件变为 “活动 ”状态时，它允许为每个组件绑定键盘快捷键。组件是可扩展的，因此第三方插件可以向现有组件添加一些路由、命令和快捷键。

- 描述： `$e.components` API 是整个 **api** 组件的管理器。
- 位置： `core/common/assets/js/api/core/components.js`
- 父级： `elementorModules.Module`

方法

方法	参数	返回值	描述
`$e.components.getAll()`		`{array.<string>}`	获取所有组件。
`$e.components.register()`	`{ComponentBase}` *component*	`{ComponentBase}` *component*	注册一个组件。
`$e.components.get()`	`{String}` *id*	`{ComponentBase}` *component*	通过 id 获取组件实例。
`$e.components.getActive()`		`{Object.<ComponentBase>}` *activeComponents*	获取活动组件。
`$e.components.activate()`	`{String}` *namespace*		激活组件。
`$e.components.inactivate()`	`{String}` *namespace*		停用组件。
`$e.components.isActive()`	`{String}` *namespace*	`{Boolean}` *isActive*	组件是否处于活动状态。

::: warning □□ □□□□□□□□□□ {$e.commands.getAll()} :::

Examples

```
// Example of creating and registering a new component, available to run in the console and
does not depend on anything else.
class CustomComponent extends $e.modules.ComponentBase {
□getNamespace() {
□□return 'custom-component';
□}
□
□defaultCommands() {
□□// Object of all the component commands.
□□return {
□□□// 'example' command.
□□□example: ( args ) => {
□□□□// Output command args to console.
□□□□console.log( 'ExampleCommand: ', args );
□□
□□□□// Return object as example.
□□□□return {
□□□□□example: 'result from ExampleCommand',
□□□□};
□□□},
□□};
□}
□}
□
□// Register the new component.
□$e.components.register( new CustomComponent() );
□
□// Runs 'example' command from 'custom-component'.
□result = $e.run( 'custom-component/example', {
□property: 'value',
□} );
□
□// Output command run result.
□console.log( 'e-components-eg-1-result: ', result );
```

Guidelines

- You can view your component as a namespace that holds your commands, hooks, routes, tabs, shortcuts & utils.
- Component class file should be named `component.js`
- Component folder name should be named as a component namespace or a sub-component namespace.
- Components and sub-components convention example, described in next scenario: Assuming you create a `Document` component which creates a sub-component `Elements`

```
├──document
│   ├──component.js
│   ├──index.js           ( has all sub-components exported )
│
└──┬──elements
│   │   ├──component.js
│   │   │   ...
```

`document/index.js` file at line *3*:

```
export { default as ElementsComponent } from './elements/component.js';
```

`document/component.js` file at line *2*:

```
import * as components from './';

export default class Component extends $e.modules.ComponentBase {
    getNamespace() {
        return 'document';
    }

    registerAPI() {
        // Register sub components.
        Object.values( components ).forEach( ( ComponentClass ) =>
            $e.components.register( new ComponentClass )
        );
```

```
        super.registerAPI();
    }
}

export default class Component;
```

`document/elements/component.js` file at line *6*:

```
export default class Component extends $e.modules.ComponentBase {
    getNamespace() {
        return 'elements';
    }
}

export default class Component;
```

□□□ #2
□ □□□ □□□ 2026-03-26 17:08:44 UTC
□ □□□ □□□ 2026-03-26 17:08:44 UTC