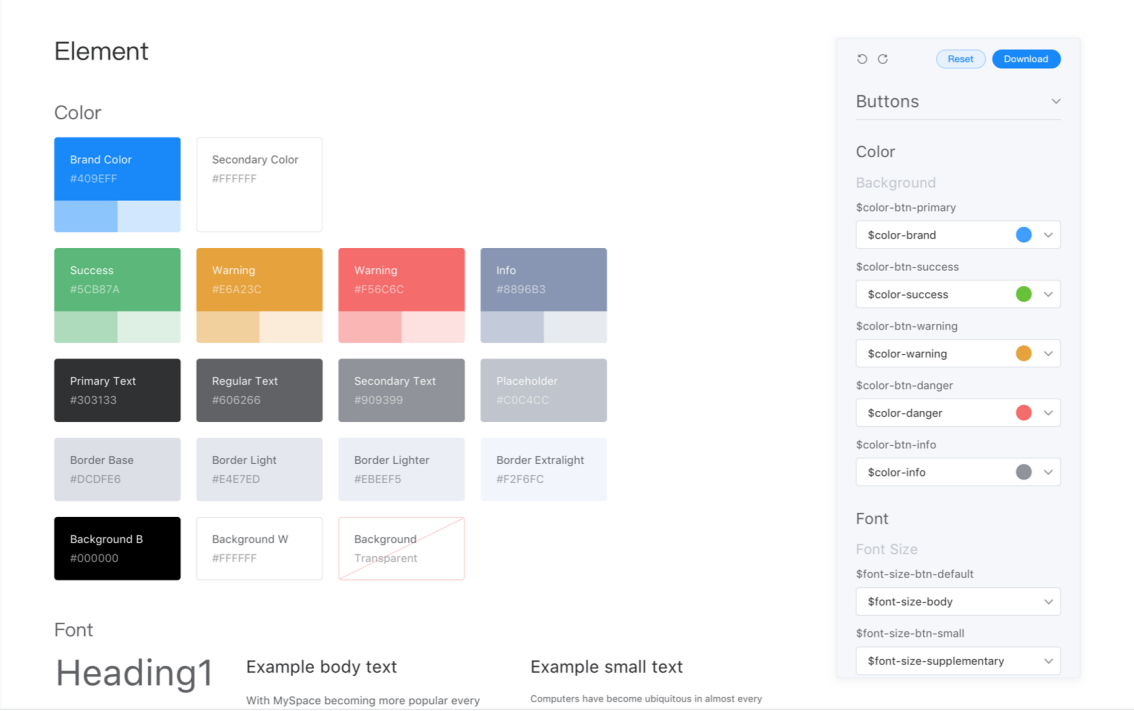
自定义主题

Element 默认提供一套主题，CSS 命名采用 BEM 的风格，方便使用者覆盖样式。我们提供了四种方法，可以进行不同程度的样式自定义。

主题编辑器

使用在线主题编辑器，可以修改定制 Element 所有全局和组件的 Design Tokens，并可以方便地实时预览样式改变后的视觉。同时它还可以基于新的定制样式生成完整的样式文件包，供直接下载使用（关于如何使用下载的主题包，请参考本节「引入自定义主题」部分）。

也可以使用主题编辑器 Chrome 插件，在任何使用 Element 开发的网站上配置并实时预览主题。



仅替换主题色

如果仅希望更换 Element 的主题色，推荐使用在线主题生成工具。Element 默认的主题色是鲜艳、友好的蓝色。通过替换主题色，能够让 Element 的视觉更加符合具体项目的定位。

使用上述工具，可以很方便地实时预览主题色改变之后的视觉，同时它还可以基于新的主题色生成完整的样式文件包，供直接下载使用（关于如何使用下载的主题包，请参考本节「引入自定义主题」和「搭配插件按需引入组件主题」部分）。

在项目中改变 SCSS 变量

Element 的 theme-chalk 使用 SCSS 编写，如果你的项目也使用了 SCSS，那么可以直接在项目中改变 Element 的样式变量。新建一个样式文件，例如 `element-variables.scss`，写入以下内容：

```
/* 改变主题色变量 */
$--color-primary: teal;

/* 改变 icon 字体路径变量，必需 */
```

```
$--font-path: '~element-ui/lib/theme-chalk/fonts';

@import "~element-ui/packages/theme-chalk/src/index";
```

之后，在项目的入口文件中，直接引入以上样式文件即可（无需引入 Element 编译好的 CSS 文件）：

```
import Vue from 'vue'
import Element from 'element-ui'
import './element-variables.scss'

Vue.use(Element)
```

:::tip 需要注意的是，覆盖字体路径变量是必需的，将其赋值为 Element 中 icon 图标所在的相对路径即可。 :::

命令行主题工具

如果你的项目没有使用 SCSS，那么可以使用命令行主题工具进行深层次的主题定制：

安装工具

首先安装「主题生成工具」，可以全局安装或者安装在当前项目下，推荐安装在项目里，方便别人 clone 项目时能直接安装依赖并启动，这里以全局安装做演示。

```
npm i element-theme -g
```

安装白垩主题，可以从 npm 安装或者从 GitHub 拉取最新代码。

```
# 从 npm
npm i element-theme-chalk -D

# 从 GitHub
npm i https://github.com/ElementUI/theme-chalk -D
```

初始化变量文件

主题生成工具安装成功后，如果全局安装可以在命令行里通过 `et` 调用工具，如果安装在当前目录下，需要通过 `node_modules/.bin/et` 访问到命令。执行 `-i` 初始化变量文件。默认输出到 `element-variables.scss`，当然你可以传参数指定文件输出目录。

```
et -i [可以自定义变量文件]

> ✓ Generator variables file
```

如果使用默认配置，执行后当前目录会有一个 `element-variables.scss` 文件。内部包含了主题所用到的所有变量，它们使用 SCSS 的格式定义。大致结构如下：

```
$--color-primary: #409EFF !default;
$--color-primary-light-1: mix($--color-white, $--color-primary, 10%) !default; /*
53a8ff */
$--color-primary-light-2: mix($--color-white, $--color-primary, 20%) !default; /*
```

```
66b1ff */
$--color-primary-light-3: mix($--color-white, $--color-primary, 30%) !default; /*
79bbff */
$--color-primary-light-4: mix($--color-white, $--color-primary, 40%) !default; /*
8cc5ff */
$--color-primary-light-5: mix($--color-white, $--color-primary, 50%) !default; /*
a0cfff */
$--color-primary-light-6: mix($--color-white, $--color-primary, 60%) !default; /*
b3d8ff */
$--color-primary-light-7: mix($--color-white, $--color-primary, 70%) !default; /*
c6e2ff */
$--color-primary-light-8: mix($--color-white, $--color-primary, 80%) !default; /*
d9ecff */
$--color-primary-light-9: mix($--color-white, $--color-primary, 90%) !default; /*
ecf5ff */

$--color-success: #67c23a !default;
$--color-warning: #e6a23c !default;
$--color-danger: #f56c6c !default;
$--color-info: #909399 !default;

...
```

修改变量

直接编辑 `element-variables.scss` 文件，例如修改主题色为红色。

```
$--color-primary: red;
```

编译主题

保存文件后，到命令行里执行 `et` 编译主题，如果你想启用 `watch` 模式，实时编译主题，增加 `-w` 参数；如果你在初始化时指定了自定义变量文件，则需要增加 `-c` 参数，并带上你的变量文件名。默认情况下编译的主题目录是放在 `./theme` 下，你可以通过 `-o` 参数指定打包目录。

```
et

> ✓ build theme font
> ✓ build element theme
```

使用自定义主题

引入自定义主题

和引入默认主题一样，在代码里直接引用「在线主题编辑器」或「命令行工具」生成的主题的 `theme/index.css` 文件即可。

```
import '../theme/index.css'
import ElementUI from 'element-ui'
import Vue from 'vue'
```

```
Vue.use(ElementUI)
```

搭配插件按需引入组件主题

如果是搭配 `babel-plugin-component` 一起使用，只需要修改 `.babelrc` 的配置，指定 `styleLibraryName` 路径为自定义主题相对于 `.babelrc` 的路径，注意要加 `~`。

```
{
  "plugins": [
    [
      "component",
      {
        "libraryName": "element-ui",
        "styleLibraryName": "~theme"
      }
    ]
  ]
}
```

如果不清楚 `babel-plugin-component` 是什么，请阅读 快速上手 一节。更多 `element-theme` 用法请参考项目仓库。