Element UI 学习笔记

第8组

班级：07111505

学号：1120151831

姓名：李继领

日期：2017-10-10

目录

[简介 3](#_Toc495410486)

[设计指南 3](#_Toc495410487)

[设计原则 3](#_Toc495410488)

[导航 4](#_Toc495410489)

[**选择合适的导航** 4](#_Toc495410490)

[**侧栏导航** 4](#_Toc495410491)

[**顶部导航** 7](#_Toc495410492)

[安装 8](#_Toc495410493)

[npm 安装 8](#_Toc495410494)

[CDN 9](#_Toc495410495)

[快速上手 12](#_Toc495410496)

[内置过渡动画 59](#_Toc495410497)

[fade 淡入淡出 59](#_Toc495410498)

[zoom 缩放 63](#_Toc495410499)

[collapse 展开折叠 67](#_Toc495410500)

[按需引入 70](#_Toc495410501)

[基础组件 71](#_Toc495410502)

[BASIC 71](#_Toc495410503)

[Layout 布局 71](#_Toc495410504)

[Color 色彩 96](#_Toc495410505)

[Typography 字体 97](#_Toc495410506)

[Icon 图标 98](#_Toc495410507)

[FORM 101](#_Toc495410508)

[Radio 单选框 101](#_Toc495410509)

[按钮样式 105](#_Toc495410510)

# 简介

Element UI 是一套采用 Vue.js 2.0 作为基础框架实现的桌面端UI框架，提供了配套设计资源，它面向企业级的后台应用，能够帮助快速地搭建网站，极大地减少研发的人力与时间成本。Element UI由饿了么前端团队推出，手机端有对应框架Mint UI 。

# 设计指南

## 设计原则

* 一致性 Consistency

与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言和概念；

在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文本、元素的位置等。

* 反馈 Feedback

控制反馈：通过界面样式和交互动效让用户可以清晰的感知自己的操作；

页面反馈：操作后，通过页面元素的变化清晰地展现当前状态。

* 效率 Efficiency

简化流程：设计简洁直观的操作流程；

清晰明确：语言表达清晰且表意明确，让用户快速理解进而作出决策；

帮助用户识别：界面简单直白，让用户快速识别而非回忆，减少用户记忆负担。

* 可控 Controllability

用户决策：根据场景可给予用户操作建议或安全提示，但不能代替用户进行决策；

结果可控：用户可以自由的进行操作，包括撤销、回退和终止当前操作等。

## 导航

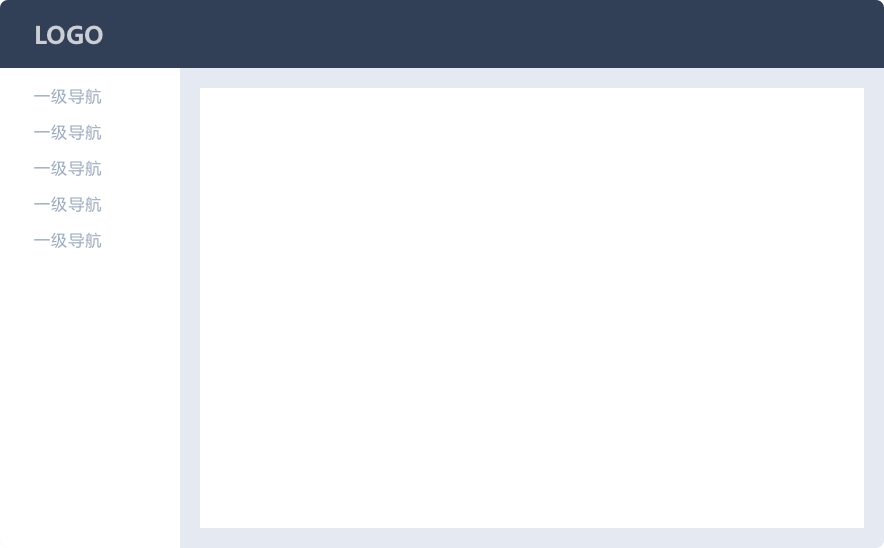
导航可以解决用户在访问页面时：在哪里，去哪里，怎样去的问题。一般导航会有「侧栏导航」和「顶部导航」2 种类型。

**选择合适的导航**

选择合适的导航可以让用户在产品的使用过程中非常流畅，相反若是不合适就会引起用户操作不适（方向不明确），以下是「侧栏导航」和 「顶部导航」的区别。

**侧栏导航**

可将导航栏固定在左侧，提高导航可见性，方便页面之间切换；顶部可放置常用工具，如搜索条、帮助按钮、通知按钮等。适用于中后台的管理型、工具型网站。



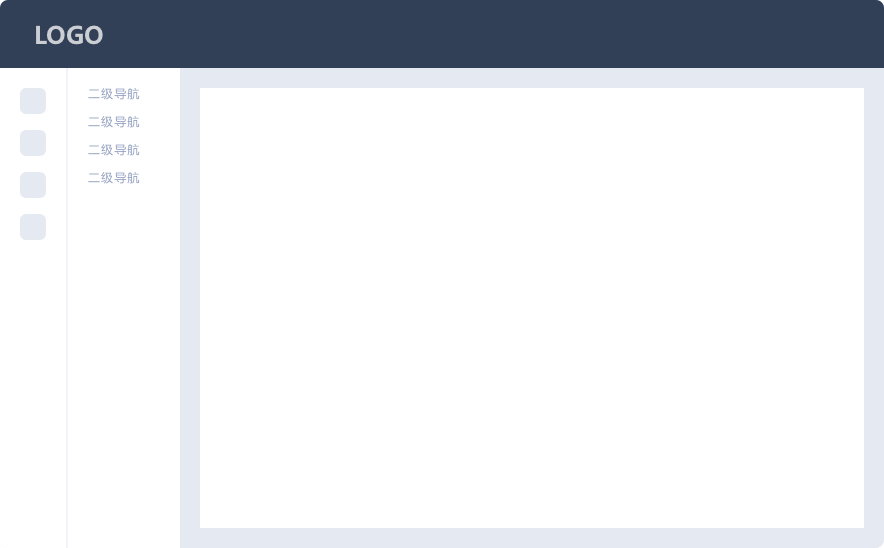
**一级类目**

适用于结构简单的网站：只有一级页面时，不需要使用面包屑。



**二级类目**

侧栏中最多可显示两级导航；当使用二级导航时，我们建议搭配使用面包屑，方便用户定位自己的位置和快速返回。



**三级类目**

适用于较复杂的工具型后台，左侧栏为一级导航，中间栏可显示其对应的二级导航，也可放置其他的工具型选项。

**顶部导航**

顺应了从上至下的正常浏览顺序，方便浏览信息；顶部宽度限制了导航的数量和文本长度。



适用于导航较少，页面篇幅较长的网站。

# 安装

## npm 安装

推荐使用 npm 的方式安装，它能更好地和 webpack 打包工具配合使用。

npm i element-ui -S

## CDN

目前可以通过 unpkg.com/element-ui 获取到最新版本的资源，在页面上引入 js 和 css 文件即可开始使用。

<!-- 引入样式 -->

<link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-default/index.css">

<!-- 引入组件库 -->

<script src="https://unpkg.com/element-ui/lib/index.js"></script>

CDN简单应用——Hello world

通过 CDN 的方式我们可以很容易地使用 Element 写出一个 Hello world 页面。

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<!-- 引入样式 -->

<link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-default/index.css">

</head>

<body>

<div id="app">

<el-button @click="visible = true">按钮</el-button>

<el-dialog v-model="visible" title="Hello world">

<p>欢迎使用 Element</p>

</el-dialog>

</div>

</body>

<!-- 先引入 Vue -->

<script src="https://unpkg.com/vue/dist/vue.js"></script>

<!-- 引入组件库 -->

<script src="https://unpkg.com/element-ui/lib/index.js"></script>

<script>

new Vue({

el: '#app',

data: function() {

return { visible: false }

}

})

</script>

</html>

# 快速上手

本节将介绍如何在项目中使用 Element。

使用 Starter Kit

配置文件

新建项目，项目结构为

|- src/ --------------------- 项目源代码

|- App.vue

|- main.js -------------- 入口文件

|- .babelrc ----------------- babel 配置文件

|- index.html --------------- HTML 模板

|- package.json ------------- npm 配置文件

|- README.md ---------------- 项目帮助文档

|- webpack.config.js -------- webpack 配置文件

几个配置文件的典型配置如下：

.babelrc

{

"presets": ["vue-app"]

}

package.json

{

"name": "element-starter",

"scripts": {

"dev": "cross-env NODE\_ENV=development webpack-dev-server --inline --hot --port 8086",

"build": "cross-env NODE\_ENV=production webpack --progress --hide-modules"

},

"dependencies": {

"element-ui": "^1.0.0",

"vue": "^2.1.6"

},

"devDependencies": {

"babel-core": "^6.0.0",

"babel-loader": "^6.0.0",

"babel-preset-vue-app": "^1.2.0",

"cross-env": "^1.0.6",

"css-loader": "^0.23.1",

"file-loader": "^0.8.5",

"style-loader": "^0.13.1",

"vue-loader": "^9.8.0",

"webpack": "beta",

"webpack-dev-server": "beta"

}

}

webpack.config.js

var path = require('path')

var webpack = require('webpack')

module.exports = {

entry: './src/main.js',

output: {

path: path.resolve(\_\_dirname, './dist'),

publicPath: '/dist/',

filename: 'build.js'

},

module: {

loaders: [

{

test: /\.vue$/,

loader: 'vue-loader'

},

{

test: /\.js$/,

loader: 'babel-loader',

exclude: /node\_modules/

},

{

test: /\.css$/,

loader: 'style-loader!css-loader'

},

{

test: /\.(eot|svg|ttf|woff|woff2)(\?\S\*)?$/,

loader: 'file-loader'

},

{

test: /\.(png|jpe?g|gif|svg)(\?\S\*)?$/,

loader: 'file-loader',

query: {

name: '[name].[ext]?[hash]'

}

}

]

},

devServer: {

historyApiFallback: true,

noInfo: true

},

devtool: '#eval-source-map'

}

if (process.env.NODE\_ENV === 'production') {

module.exports.devtool = '#source-map'

// http://vue-loader.vuejs.org/en/workflow/production.html

module.exports.plugins = (module.exports.plugins || []).concat([

new webpack.DefinePlugin({

'process.env': {

NODE\_ENV: '"production"'

}

}),

new webpack.optimize.UglifyJsPlugin({

compress: {

warnings: false

}

})

])

}

¶

引入 Element

你可以引入整个 Element，或是根据需要仅引入部分组件。我们先介绍如何引入完整的 Element。

¶

完整引入

在 main.js 中写入以下内容：

import Vue from 'vue'

import ElementUI from 'element-ui'

import 'element-ui/lib/theme-default/index.css'

import App from './App.vue'

Vue.use(ElementUI)

new Vue({

el: '#app',

render: h => h(App)

})

以上代码便完成了 Element 的引入。需要注意的是，样式文件需要单独引入。

¶

按需引入

借助 babel-plugin-component，我们可以只引入需要的组件，以达到减小项目体积的目的。

首先，安装 babel-plugin-component：

npm install babel-plugin-component -D

然后，将 .babelrc 修改为：

{

"presets": [

["es2015", { "modules": false }]

],

"plugins": [["component", [

{

"libraryName": "element-ui",

"styleLibraryName": "theme-default"

}

]]]

}

接下来，如果你只希望引入部分组件，比如 Button 和 Select，那么需要在 main.js 中写入以下内容：

import Vue from 'vue'

import { Button, Select } from 'element-ui'

import App from './App.vue'

Vue.component(Button.name, Button)

Vue.component(Select.name, Select)

/\* 或写为

\* Vue.use(Button)

\* Vue.use(Select)

\*/

new Vue({

el: '#app',

render: h => h(App)

})

完整组件列表和引入方式（完整组件列表以 components.json 为准）

import Vue from 'vue'

import {

Pagination,

Dialog,

Autocomplete,

Dropdown,

DropdownMenu,

DropdownItem,

Menu,

Submenu,

MenuItem,

MenuItemGroup,

Input,

InputNumber,

Radio,

RadioGroup,

RadioButton,

Checkbox,

CheckboxGroup,

Switch,

Select,

Option,

OptionGroup,

Button,

ButtonGroup,

Table,

TableColumn,

DatePicker,

TimeSelect,

TimePicker,

Popover,

Tooltip,

Breadcrumb,

BreadcrumbItem,

Form,

FormItem,

Tabs,

TabPane,

Tag,

Tree,

Alert,

Slider,

Icon,

Row,

Col,

Upload,

Progress,

Spinner,

Badge,

Card,

Rate,

Steps,

Step,

Carousel,

Scrollbar,

CarouselItem,

Collapse,

CollapseItem,

Cascader,

ColorPicker,

Loading,

MessageBox,

Message,

Notification

} from 'element-ui'

Vue.use(Pagination)

Vue.use(Dialog)

Vue.use(Autocomplete)

Vue.use(Dropdown)

Vue.use(DropdownMenu)

Vue.use(DropdownItem)

Vue.use(Menu)

Vue.use(Submenu)

Vue.use(MenuItem)

Vue.use(MenuItemGroup)

Vue.use(Input)

Vue.use(InputNumber)

Vue.use(Radio)

Vue.use(RadioGroup)

Vue.use(RadioButton)

Vue.use(Checkbox)

Vue.use(CheckboxGroup)

Vue.use(Switch)

Vue.use(Select)

Vue.use(Option)

Vue.use(OptionGroup)

Vue.use(Button)

Vue.use(ButtonGroup)

Vue.use(Table)

Vue.use(TableColumn)

Vue.use(DatePicker)

Vue.use(TimeSelect)

Vue.use(TimePicker)

Vue.use(Popover)

Vue.use(Tooltip)

Vue.use(Breadcrumb)

Vue.use(BreadcrumbItem)

Vue.use(Form)

Vue.use(FormItem)

Vue.use(Tabs)

Vue.use(TabPane)

Vue.use(Tag)

Vue.use(Tree)

Vue.use(Alert)

Vue.use(Slider)

Vue.use(Icon)

Vue.use(Row)

Vue.use(Col)

Vue.use(Upload)

Vue.use(Progress)

Vue.use(Spinner)

Vue.use(Badge)

Vue.use(Card)

Vue.use(Rate)

Vue.use(Steps)

Vue.use(Step)

Vue.use(Carousel)

Vue.use(Scrollbar)

Vue.use(CarouselItem)

Vue.use(Collapse)

Vue.use(CollapseItem)

Vue.use(Cascader)

Vue.use(ColorPicker)

Vue.use(Loading.directive)

Vue.prototype.$loading = Loading.service

Vue.prototype.$msgbox = MessageBox

Vue.prototype.$alert = MessageBox.alert

Vue.prototype.$confirm = MessageBox.confirm

Vue.prototype.$prompt = MessageBox.prompt

Vue.prototype.$notify = Notification

Vue.prototype.$message = Message

¶

开始使用

至此，一个基于 Vue 和 Element 的开发环境已经搭建完毕，现在就可以编写代码了。启动开发模式：

# 执行如下命令后访问 localhost:8086

npm run dev

编译：

npm run build

国际化

Element 组件内部默认使用中文，若希望使用其他语言，则需要进行多语言设置。以英文为例，在 main.js 中：

// 完整引入 Element

import Vue from 'vue'

import ElementUI from 'element-ui'

import locale from 'element-ui/lib/locale/lang/en'

Vue.use(ElementUI, { locale })

或

// 按需引入 Element

import Vue from 'vue'

import { Button, Select } from 'element-ui'

import lang from 'element-ui/lib/locale/lang/en'

import locale from 'element-ui/lib/locale'

// 设置语言

locale.use(lang)

// 引入组件

Vue.component(Button.name, Button)

Vue.component(Select.name, Select)

如果使用其它语言，默认情况下中文语言包依旧是被引入的，可以使用 webpack 的 NormalModuleReplacementPlugin 替换默认语言包。

webpack.config.js

{

plugins: [

new webpack.NormalModuleReplacementPlugin(/element-ui[\/\\]lib[\/\\]locale[\/\\]lang[\/\\]zh-CN/, 'element-ui/lib/locale/lang/en')

]

}

¶

兼容 vue-i18n@5.x

Element 兼容 vue-i18n@5.x，搭配使用能更方便地实现多语言切换。

import Vue from 'vue'

import VueI18n from 'vue-i18n'

import Element from 'element-ui'

import enLocale from 'element-ui/lib/locale/lang/en'

import zhLocale from 'element-ui/lib/locale/lang/zh-CN'

Vue.use(VueI18n)

Vue.use(Element)

Vue.config.lang = 'zh-cn'

Vue.locale('zh-cn', zhLocale)

Vue.locale('en', enLocale)

¶

兼容其他 i18n 插件

如果不使用 vue-i18n@5.x，而是用其他的 i18n 插件，Element 将无法兼容，但是可以自定义 Element 的 i18n 的处理方法。

import Vue from 'vue'

import Element from 'element-ui'

import enLocale from 'element-ui/lib/locale/lang/en'

import zhLocale from 'element-ui/lib/locale/lang/zh-CN'

Vue.use(Element, {

i18n: function (path, options) {

// ...

}

})

¶

兼容 vue-i18n@6.x

默认不支持 6.x 的 vue-i18n，你需要手动处理。

import Vue from 'vue'

import Element from 'element-ui'

import VueI18n from 'vue-i18n'

import enLocale from 'element-ui/lib/locale/lang/en'

import zhLocale from 'element-ui/lib/locale/lang/zh-CN'

Vue.use(VueI18n)

const messages = {

en: {

message: 'hello',

...enLocale // 或者用 Object.assign({ message: 'hello' }, enLocale)

},

zh: {

message: '你好',

...zhLocale // 或者用 Object.assign({ message: '你好' }, zhLocale)

}

}

// Create VueI18n instance with options

const i18n = new VueI18n({

locale: 'en', // set locale

messages, // set locale messages

})

Vue.use(Element, {

i18n: (key, value) => i18n.vm.\_t(key, value)

})

new Vue({ i18n }).$mount('#app')

¶

按需加载里定制 i18n

import Vue from 'vue'

import DatePicker from 'element/lib/date-picker'

import VueI18n from 'vue-i18n'

import enLocale from 'element-ui/lib/locale/lang/en'

import zhLocale from 'element-ui/lib/locale/lang/zh-CN'

import ElementLocale from 'element-ui/lib/locale'

Vue.use(VueI18n)

Vue.use(DatePicker)

const messages = {

en: {

message: 'hello',

...enLocale

},

zh: {

message: '你好',

...zhLocale

}

}

// Create VueI18n instance with options

const i18n = new VueI18n({

locale: 'en', // set locale

messages, // set locale messages

})

ElementLocale.i18n((key, value) => i18n.t(key, value))

¶

通过 CDN 的方式加载语言文件

<script src="//unpkg.com/vue"></script>

<script src="//unpkg.com/element-ui"></script>

<script src="//unpkg.com/element-ui/lib/umd/locale/en.js"></script>

<script>

ELEMENT.locale(ELEMENT.lang.en)

</script>

搭配 vue-i18n 使用

<script src="//unpkg.com/vue"></script>

<script src="//unpkg.com/vue-i18n/dist/vue-i18n.js"></script>

<script src="//unpkg.com/element-ui"></script>

<script src="//unpkg.com/element-ui/lib/umd/locale/zh-CN.js"></script>

<script src="//unpkg.com/element-ui/lib/umd/locale/en.js"></script>

<script>

Vue.locale('en', ELEMENT.lang.en)

Vue.locale('zh-cn', ELEMENT.lang.zhCN)

</script>

目前 Element 内置了以下语言：

简体中文（zh-CN）

英语（en）

德语（de）

葡萄牙语（pt）

西班牙语（es）

丹麦语（da）

法语（fr）

挪威语（nb-NO）

繁体中文（zh-TW）

意大利语（it）

韩语（ko）

日语（ja）

荷兰语（nl）

越南语（vi）

俄语（ru-RU）

土耳其语（tr-TR）

巴西葡萄牙语（pt-br）

波斯语（fa）

泰语（th）

印尼语（id）

保加利亚语（bg）

波兰语（pl）

芬兰语（fi）

瑞典语（sv-SE）

希腊语（el）

斯洛伐克语（sk）

加泰罗尼亚语（ca）

捷克语（cz）

乌克兰语（ua）

土库曼语（tk）

泰米尔语（ta）

拉脱维亚语（lv）

南非荷兰语（af-ZA）

爱沙尼亚语（ee）

斯洛文尼亚语（sl）

阿拉伯语（ar）

希伯来语（he）

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。如果你想完全替换主题色或者部分样式，可以使用下面的方法。

¶

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）。

如果希望进行除主题色之外的样式定制，请继续阅读。

¶

深层次的定制

如果仅仅改变主题色不能满足你的需求，请按以下步骤进行更深层次的主题定制：

¶

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

安装默认主题，可以从 npm 安装或者从 GitHub 拉取最新代码。

# 从 npm

npm i element-theme-default -D

# 从 GitHub

npm i https://github.com/ElementUI/theme-default -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.css，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.css 文件。内部包含了主题所用到的所有变量，它们使用 CSS4 的风格定义。大致结构如下：

:root {

/\* Colors

-------------------------- \*/

--color-primary: #20a0ff;

--color-success: #13ce66;

--color-warning: #f7ba2a;

--color-danger: #ff4949;

--color-info: #50BFFF;

--color-blue: #2e90fe;

--color-blue-light: #5da9ff;

--color-blue-lighter: rgba(var(--color-blue), 0.12);

--color-white: #fff;

--color-black: #000;

--color-grey: #C0CCDA;

¶

修改变量

直接编辑 element-variables.css 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

et

> ✔ build theme font

> ✔ build element theme

¶

引入自定义主题

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。像引入默认主题一样，在代码里直接引用 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"

}

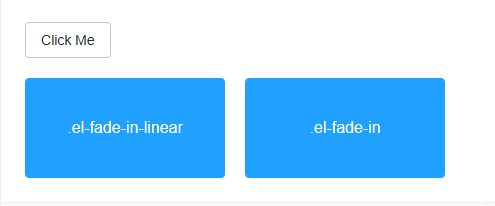
]]]

}

# 内置过渡动画

Element 内应用在部分组件的过渡动画

## fade 淡入淡出





提供 el-fade-in-linear 和 el-fade-in 两种效果。

<template>

<div>

<el-button @click="show = !show">Click Me</el-button>

<div style="display: flex; margin-top: 20px; height: 100px;">

<transition name="el-fade-in-linear">

<div v-show="show" class="transition-box">.el-fade-in-linear</div>

</transition>

<transition name="el-fade-in">

<div v-show="show" class="transition-box">.el-fade-in</div>

</transition>

</div>

</div>

</template>

<script>

export default {

data: () => ({

show: true

})

}

</script>

<style>

.transition-box {

margin-bottom: 10px;

width: 200px;

height: 100px;

border-radius: 4px;

background-color: #20A0FF;

text-align: center;

color: #fff;

padding: 40px 20px;

box-sizing: border-box;

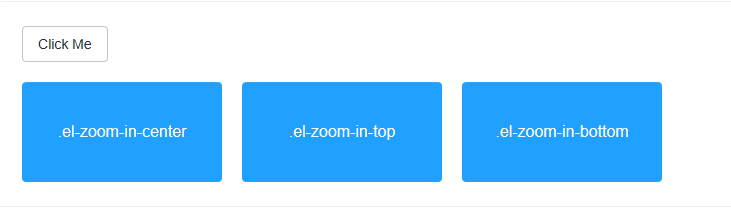
margin-right: 20px;

}

</style>

显示代码

## zoom 缩放



提供 el-zoom-in-center，el-zoom-in-top 和 el-zoom-in-bottom 三种效果。

<template>

<div>

<el-button @click="show2 = !show2">Click Me</el-button>

<div style="display: flex; margin-top: 20px; height: 100px;">

<transition name="el-zoom-in-center">

<div v-show="show2" class="transition-box">.el-zoom-in-center</div>

</transition>

<transition name="el-zoom-in-top">

<div v-show="show2" class="transition-box">.el-zoom-in-top</div>

</transition>

<transition name="el-zoom-in-bottom">

<div v-show="show2" class="transition-box">.el-zoom-in-bottom</div>

</transition>

</div>

</div>

</template>

<script>

export default {

data: () => ({

show2: true

})

}

</script>

<style>

.transition-box {

margin-bottom: 10px;

width: 200px;

height: 100px;

border-radius: 4px;

background-color: #20A0FF;

text-align: center;

color: #fff;

padding: 40px 20px;

box-sizing: border-box;

margin-right: 20px;

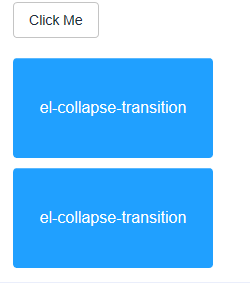
}

</style>

显示代码

## collapse 展开折叠

使用 el-collapse-transition 组件实现折叠展开效果。



<template>

<div>

<el-button @click="show3 = !show3">Click Me</el-button>

<div style="margin-top: 20px; height: 200px;">

<el-collapse-transition>

<div v-show="show3">

<div class="transition-box">el-collapse-transition</div>

<div class="transition-box">el-collapse-transition</div>

</div>

</el-collapse-transition>

</div>

</div>

</template>

<script>

export default {

data: () => ({

show3: true

})

}

</script>

<style>

.transition-box {

margin-bottom: 10px;

width: 200px;

height: 100px;

border-radius: 4px;

background-color: #20A0FF;

text-align: center;

color: #fff;

padding: 40px 20px;

box-sizing: border-box;

margin-right: 20px;

}

</style>

## 显示代码按需引入

// fade/zoom 等

import 'element-ui/lib/theme-defaut/base.css';

// collapse 展开折叠

import CollapseTransition from 'element-ui/lib/transitions/collapse-transition';

import Vue from 'vue'

Vue.component(CollapseTransition.name, CollapseTransition)

# 基础组件

## BASIC

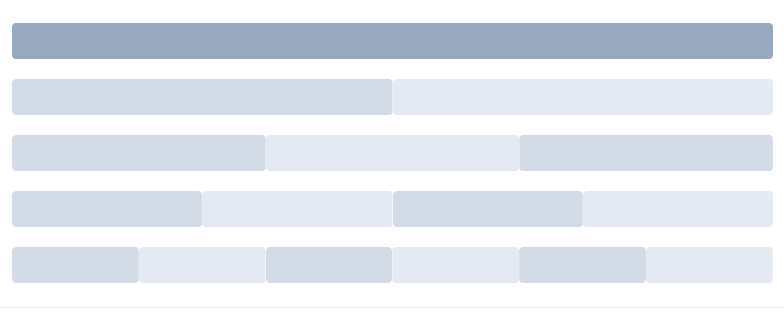
### Layout 布局

通过基础的 24 分栏，迅速简便地创建布局。

¶

#### 基础布局

使用单一分栏创建基础的栅格布局。



<el-row>

<el-col :span="24"><div class="grid-content bg-purple-dark"></div></el-col>

</el-row>

<el-row>

<el-col :span="12"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="12"><div class="grid-content bg-purple-light"></div></el-col>

</el-row>

<el-row>

<el-col :span="8"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="8"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="8"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

</el-row>

<el-row>

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple-light"></div></el-col>

</el-row>

<style>

.el-row {

margin-bottom: 20px;

&:last-child {

margin-bottom: 0;

}

}

.el-col {

border-radius: 4px;

}

.bg-purple-dark {

background: #99a9bf;

}

.bg-purple {

background: #d3dce6;

}

.bg-purple-light {

background: #e5e9f2;

}

.grid-content {

border-radius: 4px;

min-height: 36px;

}

.row-bg {

padding: 10px 0;

background-color: #f9fafc;

}

</style>

分栏间隔

分栏之间存在间隔。



<el-row :gutter="20">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<style>

.el-row {

margin-bottom: 20px;

&:last-child {

margin-bottom: 0;

}

}

.el-col {

border-radius: 4px;

}

.bg-purple-dark {

background: #99a9bf;

}

.bg-purple {

background: #d3dce6;

}

.bg-purple-light {

background: #e5e9f2;

}

.grid-content {

border-radius: 4px;

min-height: 36px;

}

.row-bg {

padding: 10px 0;

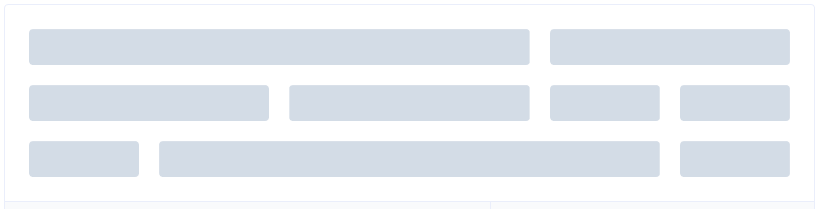
background-color: #f9fafc;

}

</style>

#### 混合布局

通过基础的 1/24 分栏任意扩展组合形成较为复杂的混合布局。



<el-row :gutter="20">

<el-col :span="16"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="8"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="8"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="8"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="16"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="4"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<style>

.el-row {

margin-bottom: 20px;

&:last-child {

margin-bottom: 0;

}

}

.el-col {

border-radius: 4px;

}

.bg-purple-dark {

background: #99a9bf;

}

.bg-purple {

background: #d3dce6;

}

.bg-purple-light {

background: #e5e9f2;

}

.grid-content {

border-radius: 4px;

min-height: 36px;

}

.row-bg {

padding: 10px 0;

background-color: #f9fafc;

}

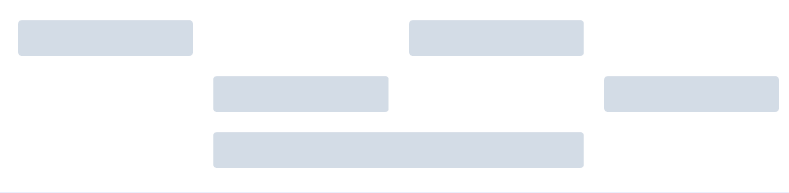
</style>

隐藏代码

#### 分栏偏移

支持偏移指定的栏数。

通过制定 col 组件的 offset 属性可以指定分栏偏移的栏数。



<el-row :gutter="20">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6" :offset="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="6" :offset="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6" :offset="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row :gutter="20">

<el-col :span="12" :offset="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<style>

.el-row {

margin-bottom: 20px;

&:last-child {

margin-bottom: 0;

}

}

.el-col {

border-radius: 4px;

}

.bg-purple-dark {

background: #99a9bf;

}

.bg-purple {

background: #d3dce6;

}

.bg-purple-light {

background: #e5e9f2;

}

.grid-content {

border-radius: 4px;

min-height: 36px;

}

.row-bg {

padding: 10px 0;

background-color: #f9fafc;

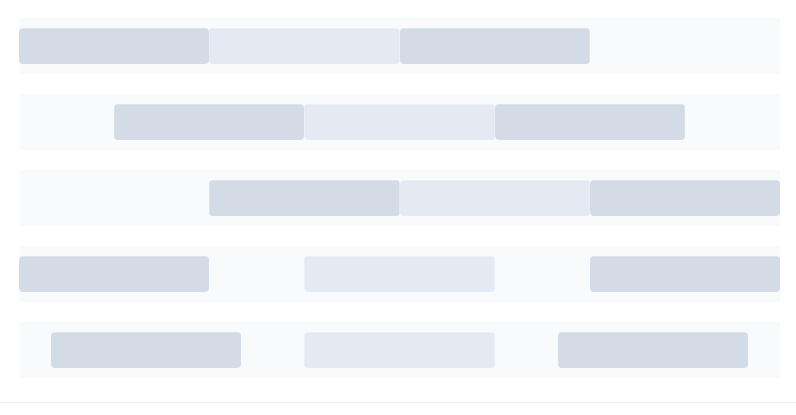
}

</style>

#### 显示代码对齐方式

通过 flex 布局来对分栏进行灵活的对齐。

将 type 属性赋值为 'flex'，可以启用 flex 布局，并可通过 justify 属性来指定 start, center, end, space-between, space-around 其中的值来定义子元素的排版方式。



<el-row type="flex" class="row-bg">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row type="flex" class="row-bg" justify="center">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row type="flex" class="row-bg" justify="end">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row type="flex" class="row-bg" justify="space-between">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<el-row type="flex" class="row-bg" justify="space-around">

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :span="6"><div class="grid-content bg-purple"></div></el-col>

</el-row>

<style>

.el-row {

margin-bottom: 20px;

&:last-child {

margin-bottom: 0;

}

}

.el-col {

border-radius: 4px;

}

.bg-purple-dark {

background: #99a9bf;

}

.bg-purple {

background: #d3dce6;

}

.bg-purple-light {

background: #e5e9f2;

}

.grid-content {

border-radius: 4px;

min-height: 36px;

}

.row-bg {

padding: 10px 0;

background-color: #f9fafc;

}

</style>

#### 显示代码响应式布局

参照了 Bootstrap 的 响应式设计，预设了四个响应尺寸：xs、sm、md和lg。



在线运行

<el-row :gutter="10">

<el-col :xs="8" :sm="6" :md="4" :lg="3"><div class="grid-content bg-purple"></div></el-col>

<el-col :xs="4" :sm="6" :md="8" :lg="9"><div class="grid-content bg-purple-light"></div></el-col>

<el-col :xs="4" :sm="6" :md="8" :lg="9"><div class="grid-content bg-purple"></div></el-col>

<el-col :xs="8" :sm="6" :md="4" :lg="3"><div class="grid-content bg-purple-light"></div></el-col>

</el-row>

<style>

.el-col {

border-radius: 4px;

}

.bg-purple-dark {

background: #99a9bf;

}

.bg-purple {

background: #d3dce6;

}

.bg-purple-light {

background: #e5e9f2;

}

.grid-content {

border-radius: 4px;

min-height: 36px;

}

</style>

#### 显示代码Row Attributes

| **参数** | **说明** | **类型** | **可选值** | **默认值** |
| --- | --- | --- | --- | --- |
| gutter | 栅格间隔 | number | — | 0 |
| type | 布局模式，可选 flex，现代浏览器下有效 | string | — | — |
| justify | flex 布局下的水平排列方式 | string | start/end/center/space-around/space-between | start |
| align | flex 布局下的垂直排列方式 | string | top/middle/bottom | top |
| tag | 自定义元素标签 | string | \* | div |

### Color 色彩

Element 为了避免视觉传达差异，使用一套特定的调色板来规定颜色，为你所搭建的产品提供一致的外观视觉感受。

#### 主色

Element 主要品牌颜色是鲜艳、友好的蓝色。



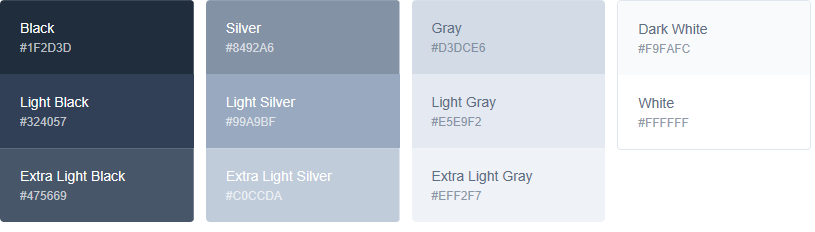
#### 辅助色

除了主色外的场景色，需要在不同的场景中使用（例如危险色表示危险的操作）。



中性色

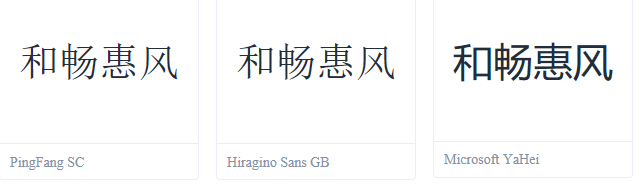
中性色用于文本、背景和边框颜色。通过运用不同的中性色，来表现层次结构。



### Typography 字体

我们对字体进行统一规范，力求在各个操作系统下都有最佳展示效果。

#### 中文字体



#### 英文／数字字体



#### Font-family 代码

font-family: "Helvetica Neue",Helvetica,"PingFang SC","Hiragino Sans GB","Microsoft YaHei","微软雅黑",Arial,sans-serif;

#### 字体使用规范

|  |  |  |
| --- | --- | --- |
| 主标题 | 用 Element 快速搭建页面 | 20px Extra large |
| 标题 | 用 Element 快速搭建页面 | 18px large |
| 小标题 | 用 Element 快速搭建页面 | 16px Medium |
| 正文 | 用 Element 快速搭建页面 | 14px Small |
| 正文（小） | 用 Element 快速搭建页面 | 13px Extra Small |
| 辅助文字 | 用 Element 快速搭建页面 | 12px Extra Extra Small |

### Icon 图标

提供了一套常用的图标集合。

#### 使用方法

直接通过设置类名为 el-icon-iconName 来使用即可。例如：



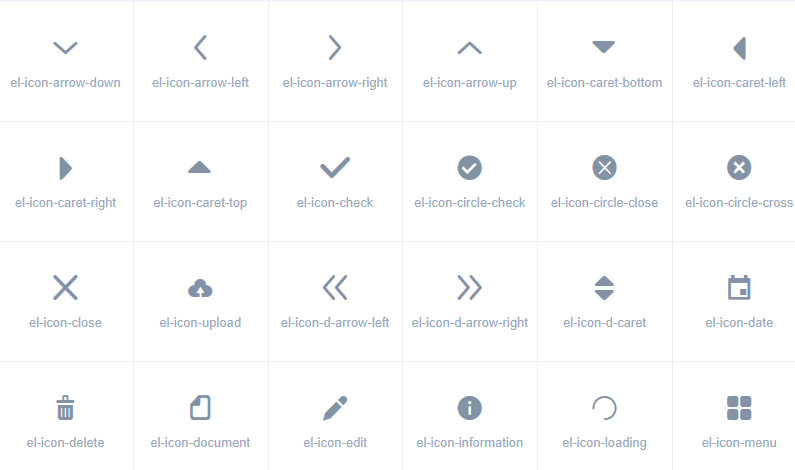
<i class="el-icon-edit"></i>

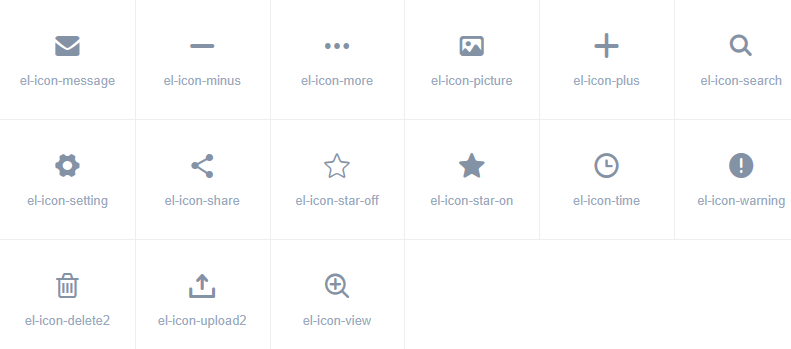
<i class="el-icon-share"></i>

<i class="el-icon-delete"></i>

<el-button type="primary" icon="search">搜索</el-button>

#### 图标集合





#### 第三方图标库

Element 的一些组件提供了 icon 属性（如 Input），如果希望传入第三方图标库中的图标，需要进行如下操作：

1. 修改第三方图标库的前缀（见下方说明）

2. 添加以下 CSS：

[class^="el-icon-my"], [class\*=" el-icon-my"] {

font-family:"your-font-family" !important;

/\* 以下内容参照第三方图标库本身的规则 \*/

font-size: inherit;

font-style:normal;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

}

之后就可以像使用 Element 内置图标一样使用第三方图标。比如在 el-input 中：

<el-input icon="my-xxx" />

## FORM

### Radio 单选框

在一组备选项中进行单选

#### 基础用法

由于选项默认可见，不宜过多，若选项过多，建议使用 Select 选择器。

备选项备选项



<template>

<el-radio class="radio" v-model="radio" label="1">备选项</el-radio>

<el-radio class="radio" v-model="radio" label="2">备选项</el-radio>

</template>

<script>

export default {

data () {

return {

radio: '1'

};

}

}

</script>

#### 禁用状态

单选框不可用的状态。



<template>

<el-radio disabled v-model="radio1" label="禁用">备选项</el-radio>

<el-radio disabled v-model="radio1" label="选中且禁用">备选项</el-radio>

</template>

<script>

export default {

data () {

return {

radio1: '选中且禁用'

};

}

}

</script>

#### 单选框组

适用于在多个互斥的选项中选择的场景



<template>

<el-radio-group v-model="radio2">

<el-radio :label="3">备选项</el-radio>

<el-radio :label="6">备选项</el-radio>

<el-radio :label="9">备选项</el-radio>

</el-radio-group>

</template>

<script>

export default {

data () {

return {

radio2: 3

};

}

}

</script>

### 按钮样式

按钮样式的单选组合。



<template>

<el-radio-group v-model="radio3">

<el-radio-button label="上海"></el-radio-button>

<el-radio-button label="北京"></el-radio-button>

<el-radio-button label="广州"></el-radio-button>

<el-radio-button label="深圳"></el-radio-button>

</el-radio-group>

<div style="margin: 15px 0;"></div>

<el-radio-group v-model="radio4">

<el-radio-button label="上海"></el-radio-button>

<el-radio-button label="北京" :disabled="true"></el-radio-button>

<el-radio-button label="广州"></el-radio-button>

<el-radio-button label="深圳"></el-radio-button>

</el-radio-group>

<div style="margin: 15px 0;"></div>

<el-radio-group v-model="radio5" :disabled="true">

<el-radio-button label="上海"></el-radio-button>

<el-radio-button label="北京"></el-radio-button>

<el-radio-button label="广州"></el-radio-button>

<el-radio-button label="深圳"></el-radio-button>

</el-radio-group>

</template>

<script>

export default {

data () {

return {

radio3: '上海',

radio4: '上海',

radio5: '上海'

};

}

}

</script>

#### 隐藏代码Radio Attributes

| **参数** | **说明** | **类型** | **可选值** | **默认值** |
| --- | --- | --- | --- | --- |
| label | Radio 的 value | string,number,boolean | — | — |
| disabled | 是否禁用 | boolean | — | false |
| name | 原生 name 属性 | string | — | — |

#### Radio-group Attributes

| **参数** | **说明** | **类型** | **可选值** | **默认值** |
| --- | --- | --- | --- | --- |
| size | Radio 按钮组尺寸 | string | large, small | — |
| fill | 按钮激活时的填充色和边框色 | string | — | #20a0ff |
| text-color | 按钮激活时的文本颜色 | string | — | #ffffff |

#### Radio-group Events

| **事件名称** | **说明** | **回调参数** |
| --- | --- | --- |
| change | 绑定值变化时触发的事件 | 选中的 Radio label 值 |

#### Radio-button Attributes

| **参数** | **说明** | **类型** | **可选值** | **默认值** |
| --- | --- | --- | --- | --- |
| label | Radio 的 value | string,number | — | — |
| disabled | 是否禁用 | boolean | — | false |

Element UI 提供了许多其他FROM组件以及其他类型的组件，详情见链接：

http://element.eleme.io/#/zh-CN/component