第3章

CHAPTER 3

组　件　库

5min

在近三十年的前端发展过程中，前端和设计之间似乎总是有着千丝万缕的联系。诚然，前端工程师作为一个单独的职业被提及还要追溯到农业时代，而作为“初代目”前端工程师的各位先哲们则多全知全能。实际上，形成首批前端工程师的群体大都来自网页制作人员，而这些人群中又可以划分为两个方向，即网页设计人员和网页开发人员。在农业时代之前，彼时最为流行的 Web 架构方案是以 JSP(Java Server Pages)、ASP(Active Server Pages)等为主的服务器端渲染技术，并且浏览器中的网页仅仅包括文字、图片等媒体的展示。在这样的时代背景下，通常会将 Web 页面的设计交由网页设计人员来完成，而将业务数据拼接到页面中进行展示的工作交给网页开发人员来完成，因而，在前后端分离大势的催化下，多元文化融合下诞生的前端程序员通常兼有美学意识和开发功底。

随着业务开发需求的增加，有着多元理念下的前端工程师也在不断追寻开发过程中的效率，考虑实用与美学兼具。于是，在大量工程实践的提炼下，前端业界抽象出了一套功能复用、界面美观、交互流畅的功能体系，即组件库体系。组件库体系是凝结了最初网页设计人员审美和网页开发人员技术的智慧结晶，其在颗粒度与复用度方面都对前端工程效能进行了提升。事实上，前端技术都是基于 HTML、CSS、JavaScript 三大基石进行拓展的，而其各自虽然都可以形成独立的架构与设计方案，但融合三大基础能力的组件库体系似乎才是高效工程的最优解，因而，所谓“组件库”，其本质是高度抽象的以 HTML、CSS、JavaScript 为基底能够复用的最细粒度单元的体系集成方案。

通常来讲，为了解决业务场景中的复用问题，前端工程师都会分别封装对应的组件，然而，这些组件的封装一般涉及两个问题，即颗粒度问题与开闭性问题。基于这两个维度来考量组件的封装性是非常好的评价指标，同时也是多个组件组合是否能形成内聚体系最关键的核心所在，因此，要想实现体系化的组件库就必须对业务抽象层次很好地进行划分，即范围决定层次。

前端工程业界通常可将组件库区分为通用型组件库和业务型组件库，本章则主要介绍通用型组件库，因而，本章以工业时代下前端业界中两个常见的通用型组件库体系进行阐述，分别是 Element UI 和 Ant Design。不得不说，尽管前端业界中的组件库方案大都在框

架体系的大背景下进行讨论,但实际上农业时代也有很多诸如以 jQuery+Bootstrap 为基础封装的十分优秀的组件方案,因而,组件库方案并不能只着眼于基于框架限制下的工程构建。对于前端工程而言,有效地对工程方案进行合理划分及核心单元抽离才是关键。最后,本章也会简单总结设计与开发组件库所考量维度的建设思考,以期能够给读者提供更多的设计方面的参照。“学而不思则罔”,Why 有时要比 How 更重要,单纯地照搬照抄很难撼动现有组件库市场的固有格局,差异性才更能体现出价值。

16min

3.1 Element UI

在中国互联网历史上曾发生过无数次经典战役,时至今日仍被人津津乐道的“千团大战”绝对称得上是这些经典战役中尤为浓墨重彩的一笔。时间拨回 2016 年,从“千团大战”中杀出一条血路来的饿了么,随着其商户端 PC 业务量激增而产生了大量的同公司内部高度类似的管理后台类项目,并且此时野蛮生长下的饿了么也急需有足够辨识度的品牌标识出现在公众视野之中。于是,适逢 Vue 的兴起且饿了么又以 Vue 作为其前端技术选型等多方综合因素考量下,由饿了么 UED 部门操刀设计、大前端部门落地实现的面向中后台的组件库诞生了。以今日之视角观昨日之事态,彼时 Vue 生态下也刚好缺少适合进行二次开发的中后台组件库。到目前为止,尽管后续 Vue 的组件库万物丛生,但在 Vue 领域下能与之争锋者亦如凤毛麟角。正是在这样的天时地利下,饿了么团队出品的 Element UI 成为 Vue 生态下 PC 端组件库的标配首选。

除了配合 Vue 框架外,Element UI 也提供了满足 React 和 Angular 框架下的对应实现。相较于 Vue 体系下的统治地位,Element UI 在其他框架下的表现却并不尽如人意,先声夺人往往才能掌控优势。除此之外,配合 Element UI 组件库的二次开源最佳实践也层出不穷,以后台管理系统为例,就有十分多的 GitHub 开源项目,例如 Vue Element Admin 及 Vue Pure Admin 等。

本节将通过指南、组件、主题、国际化、文档及资源等几部分来分别介绍 Element UI 的整个组件库体系,以期能够让读者从多个维度了解 Element UI 的构成与价值。

3.1.1 指南

所谓“知止而后有定”,组件库的体系建构离不开设计理念的提炼与表达,谋定而动,知止方才有得,因此,本节将通过设计哲学、设计风格及设计样式 3 个方面来阐述 Element UI 的设计理念,以思考理念如何指导组件库体系的落地与实践。

1. 设计哲学

“思想决定意识,意识决定行为”,在设计体系中也不例外,每个设计体系语言都有其自

身所需要遵循的原则。在 Element UI 中，可将这些原则总结为 4 个关键词，即一致、反馈、效率及可控，其中，一致性体现在界面与生活中的一致，反馈在于行为与界面的响应，而效率与可控则重点讲究的是快速与正确。设计哲学体现的是设计师及设计团队整体的思想与调性，其对整个组件库体系而言具有决定性作用，如图 3-1 所示。

图 3-1　Element UI 设计哲学

2. 设计风格

在用户界面(User Interface)设计风格中，通常可以将其划分为拟物风格、扁平风格及轻拟物风格等。对 Element UI 而言，其遵循的是扁平化的设计理念。相较于拟物化风格对实际物体的模仿重现，扁平化风格则更强调抽象、极简和符号化，其通过去除冗余、厚重和繁杂的装饰效果来凸显信息本身的核心要义，如图 3-2 所示。

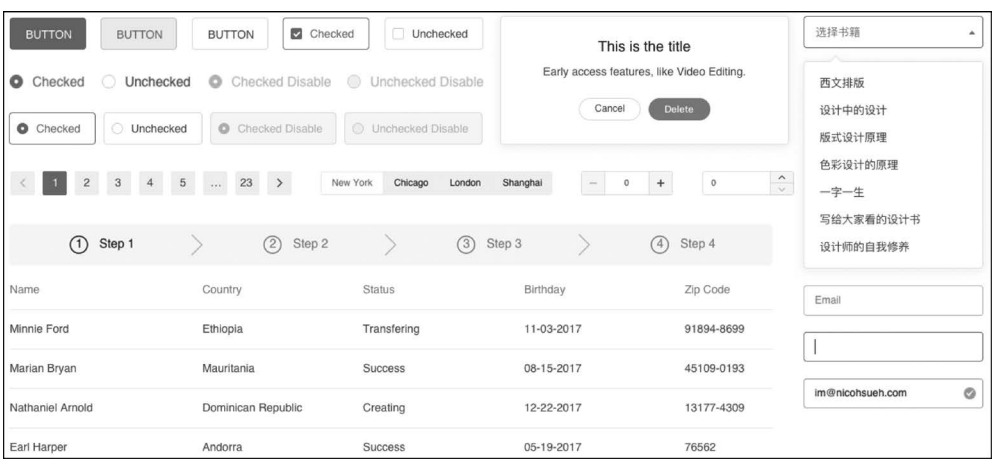


图 3-2　Element UI 设计风格

3. 设计样式

三大构成是现代艺术设计基础的重要组成部分，其主要包括平面构成、色彩构成及

立体构成。所谓“构成”,其含义是将不同形态的单元重新组成一个新的单元。同样地,UI设计中通常也会包括上述三大构成所需的要素,其具体表现为色彩、布局及字体的设计与呈现。

在Element UI中,其色彩体系主要通过“主色+功能色”的方式对色彩广度进行展现,而在深度方面则通过灰度对色彩层级进行递进,如图3-3所示。

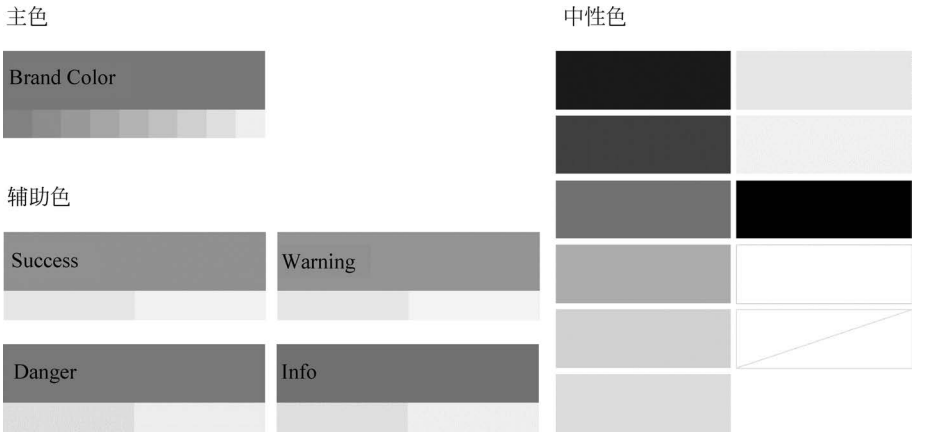


图3-3 Element UI色彩样式

对于布局体系而言,Element UI同样采用弹性(Flex)栅格布局进行自适应排布,其采用的是24分栏而非12栅格体系,如图3-4所示。

图3-4 Element UI布局样式

对于字体而言,Element UI采用统一的字体规范,提供不同的字体家族、字重等设计方案,力求在各个操作系统下都有最佳的展示效果,如图3-5所示。

图 3-5　Element UI 字体样式

3.1.2　组件

随着时代的发展，以 IT 技术为基础利用网络平台提供服务的互联网企业逐步登上了历史舞台。一般而言，互联网企业提供服务的形式通常以终端层作为其业务流量的入口，其中，通过需求分析来对功能进行梳理，从而便可进行流程分解与设计，以形成不同形式的页面来串联起整个商务活动。因此，为了更好、快速、高效地提供业务开发能力，以功能层、页面层、组件层对业务形态进行析构分解便可将组件库体系按一定规则归纳收敛到提供所需不同组件类型的划分之中。在 Element UI 中，其可规划为六大类组件，分别为通用类组件(Basic)、表单类组件(Form)、数据类组件(Data)、通知类组件(Notice)、导航类组件(Navigation)及其他类组件(Others)，如图 3-6 所示。

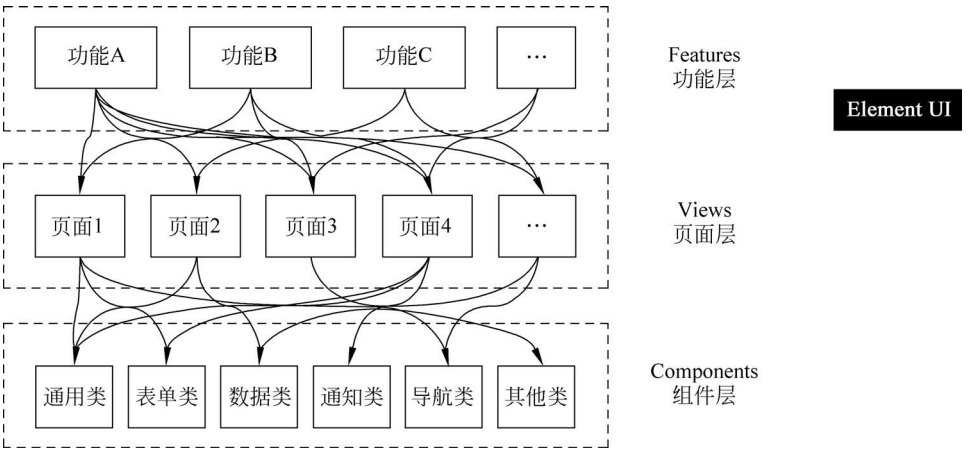


图 3-6　Element UI 组件分类

毋庸置疑，对于组件库的构建也同样离不开工程模板。同时，基于不同框架的组件库构建需要符合其对应框架的语法规则，因而，本节以 Element UI 2.x 版本代码为例相应地进

行实现及分析,其所依赖的框架为 Vue 2.x。

在 Vue 2.x 中,Vue 提供了 component()的组件注册方法,同时也提供了 use()方法进行插件接入,代码如下:

```
//封装 install 方法,用于注册组件及指令
const install = function(Vue, opts = {}) {};
if (typeof window !== 'undefined' && window.Vue) {
  install(window.Vue);
}
```

相应地,对于每个组件而言,需要提供一个 install()方法,模板代码如下:

```
//从 src 目录下引入组件,组件名称按照对应分类进行命名
import ComponentA from './src';
//提供 install 方法,用于注册
ComponentA.install = function(Vue) {
    Vue.component(ComponentA.name, ComponentA)
};
//导出组件
export default ComponentA;
```

注意:上述代码为抽象出来的组件收敛入口模板代码,对于具体的每个组件的实现思路可以查看源码进行学习,具体可见表 3-1～3-6 的备注中对应的源码网址。

1. 通用类组件

在通用类组件中,Element UI 提供了布局组件(Layout)、布局容器组件(Container)、图标组件(Icon)、按钮组件(Button)及文字链接组件(Link)等,见表 3-1。

表 3-1 Element UI 通用类组件

组件名称	组 件 包	备 注
Layout	el-row	ElementUI 仓库/packages/row
	el-col	ElementUI 仓库/packages/col
Container	el-container	ElementUI 仓库/packages/container
	el-header	ElementUI 仓库/packages/header
	el-main	ElementUI 仓库/packages/main
	el-footer	ElementUI 仓库/packages/footer
Icon	el-icon	ElementUI 仓库/packages/icon
Button	el-button	ElementUI 仓库/packages/button
	el-button-group	ElementUI 仓库/packages/button-group
Link	el-link	ElementUI 仓库/packages/link

2. 表单类组件

在表单类组件中,Element UI 提供了单选框组件(Radio)、复选框组件(Checkbox)、输入框组件(Input)、计数器组件(InputNumber)、选择器组件(Select)、级联选择器组件

(Cascader)、开关组件(Switch)、滑块组件(Slider)、时间选择器组件(TimePicker)、日期选择器组件(DatePicker)、上传组件(Upload)、评分组件(Rate)、颜色选择器组件(ColorPicker)、穿梭框组件(Transfer)及表单组件(Form)等,见表 3-2。

表 3-2 Element UI 表单类组件

组件名称	组 件 包	备 注
Radio	el-radio	ElementUI 仓库/packages/radio
	el-radio-group	ElementUI 仓库/packages/radio-group
	el-radio-button	ElementUI 仓库/packages/radio-button
Checkbox	el-checkbox	ElementUI 仓库/packages/checkbox
	el-checkbox-group	ElementUI 仓库/packages/checkbox-group
	el-checkbox-button	ElementUI 仓库/packages/checkbox-button
Input	el-input	ElementUI 仓库/packages/input
	el-autocomplete	ElementUI 仓库/packages/autocomplete
InputNumber	el-input-number	ElementUI 仓库/packages/input-number
Select	el-select	ElementUI 仓库/packages/select
	el-option	ElementUI 仓库/packages/option
	el-option-group	ElementUI 仓库/packages/option-group
Cascader	el-cascader	ElementUI 仓库/packages/cascader
	el-cascader-panel	ElementUI 仓库/packages/cascader-panel
Switch	el-switch	ElementUI 仓库/packages/switch
Slider	el-slider	ElementUI 仓库/packages/slider
TimePicker	el-time-select	ElementUI 仓库/packages/time-select
	el-time-picker	ElementUI 仓库/packages/time-picker
DatePicker	el-date-picker	ElementUI 仓库/packages/date-picker
Upload	el-upload	ElementUI 仓库/packages/upload
Rate	el-rate	ElementUI 仓库/packages/rate
ColorPicker	el-color-picker	ElementUI 仓库/packages/color-picker
Transfer	el-transfer	ElementUI 仓库/packages/transfer
Form	el-form	ElementUI 仓库/packages/form
	el-form-item	ElementUI 仓库/packages/form-item

3. 数据类组件

在数据类组件中,Element UI 提供了表格组件(Table)、标签组件(Tag)、进度条组件(Progress)、树形组件(Tree)、分页组件(Pagination)、标记组件(Badge)、头像组件(Avatar)、骨架屏组件(Skeleton)、空状态组件(Empty)、描述列表组件(Descriptions)、结果组件(Result)及统计数值组件(Statistic)等,见表 3-3。

表 3-3 Element UI 数据类组件

组件名称	组 件 包	备 注
Table	el-table	ElementUI 仓库/packages/table
	el-table-column	ElementUI 仓库/packages/table-column
Tag	el-tag	ElementUI 仓库/packages/tag
Progress	el-progress	ElementUI 仓库/packages/progress
Tree	el-tree	ElementUI 仓库/packages/tree
Pagination	el-pagination	ElementUI 仓库/packages/pagination
Badge	el-badge	ElementUI 仓库/packages/badge
Avatar	el-avatar	ElementUI 仓库/packages/avatar
Skeleton	el-skeleton	ElementUI 仓库/packages/skeleton
	el-skeleton-item	ElementUI 仓库/packages/skeleton-item
Empty	el-empty	ElementUI 仓库/packages/empty
Descriptions	el-descriptions	ElementUI 仓库/packages/descriptions
	el-descriptions-item	ElementUI 仓库/packages/descriptions-item
Result	el-result	ElementUI 仓库/packages/result
Statistic	el-statistic	ElementUI 仓库/packages/statistic

4. 通知类组件

在通知类组件中,Element UI 提供了警告组件(Alert)、加载组件(Loading)、消息提示组件(Message)及通知组件(Notification)等,见表 3-4。

表 3-4 Element UI 通知类组件

组件名称	组 件 包	备 注
Alert	el-alert	ElementUI 仓库/packages/alert
Loading	v-loading	ElementUI 仓库/packages/loading
Message	message	ElementUI 仓库/packages/message
Notification	notify	ElementUI 仓库/packages/notification

5. 导航类组件

在导航类组件中,Element UI 提供了导航菜单组件(NavMenu)、标签页组件(Tabs)、面包屑组件(Breadcrumb)、页头组件(PageHeader)、下拉菜单组件(Dropdown)及步骤条组件(Steps)等,见表 3-5。

表 3-5 Element UI 导航类组件

组件名称	组 件 包	备 注
NavMenu	el-menu	ElementUI 仓库/packages/menu
	el-menu-item	ElementUI 仓库/packages/menu-item
	el-menu-group	ElementUI 仓库/packages/menu-item-group
	el-submenu	ElementUI 仓库/packages/submenu

续表

组件名称	组 件 包	备 注
Tabs	el-tabs	ElementUI 仓库/packages/tabs
	el-tab-pane	ElementUI 仓库/packages/tab-pane
Breadcrumb	el-breadcrumb	ElementUI 仓库/packages/breadcrumb
	el-breadcrumb-item	ElementUI 仓库/packages/breadcrumb-item
PageHeader	el-page-header	ElementUI 仓库/packages/page-header
Dropdown	el-dropdown	ElementUI 仓库/packages/dropdown
	el-dropdown-menu	ElementUI 仓库/packages/dropdown-menu
	el-dropdown-item	ElementUI 仓库/packages/dropdown-item
Steps	el-steps	ElementUI 仓库/packages/steps
	el-step	ElementUI 仓库/packages/step

6. 其他类组件

对于上述未分类但开发中复用程度又很高的组件，Element UI 将其划分到了其他类组件中，包括对话框组件(Dialog)、文字提示组件(Tooltip)、弹出窗组件(Popover)、气泡确认框组件(Popconfirm)、卡片组件(Card)、走马灯组件(Carousel)、折叠面板组件(Collapse)、时间线组件(Timeline)、分割线组件(Divider)、日历组件(Calendar)、图片组件(Image)、回到顶部组件(Backtop)、无限滚动组件(InfiniteScroll)及抽屉组件(Drawer)，见表 3-6。

表 3-6 Element UI 其他类组件

组件名称	组 件 包	备 注
Dialog	el-dialog	ElementUI 仓库/packages/dialog
Tooltip	el-tooltip	ElementUI 仓库/packages/tooltip
Popover	el-popover	ElementUI 仓库/packages/popover
Popconfirm	el-popconfirm	ElementUI 仓库/packages/popconfirm
Card	el-card	ElementUI 仓库/packages/card
Carousel	el-carousel	ElementUI 仓库/packages/carousel
	el-carousel-item	ElementUI 仓库/packages/carousel-item
Collapse	el-collapse	ElementUI 仓库/packages/collapse
	el-collapse-item	ElementUI 仓库/packages/collapse-item
Timeline	el-timeline	ElementUI 仓库/packages/timeline
	el-timeline-item	ElementUI 仓库/packages/timeline-item
Divider	el-divider	ElementUI 仓库/packages/divider
Calendar	el-calendar	ElementUI 仓库/packages/calendar
Image	el-image	ElementUI 仓库/packages/image
Backtop	el-backtop	ElementUI 仓库/packages/backtop
InfiniteScroll	v-infinite-scroll	ElementUI 仓库/packages/infinite-scroll
Drawer	el-drawer	ElementUI 仓库/packages/drawer

3.1.3 主题

对于通用型组件库而言,一套成熟的组件库体系并不应该仅服务于某一种或几种特定的场景,而是应该定位于可扩展且支持各种定制化复用的生态平台,因此,对于多业务领域场景的主题配置方案便是组件库体系中不可或缺的部分。在 Element UI 中,其是通过修改变量映射对应色值的方式来完成个性化主题的定制功能,如图 3-7 所示。

图 3-7 Element UI 主题配置

在 Element UI 2.x 中,其实现主题变量的代码如下:

```
//第 3 章/var.scss
$ -- color - primary: # 409EFF !default;
$ -- color - white: # FFFFFF !default;
$ -- color - black: # 000000 !default;
$ -- color - success: # 67C23A !default;
$ -- color - warning: # E6A23C !default;
$ -- color - danger: # F56C6C !default;
$ -- color - info: # 909399 !default;
$ -- color - text - primary: # 303133 !default;
$ -- color - text - regular: # 606266 !default;
$ -- color - text - secondary: # 909399 !default;
$ -- color - text - placeholder: # C0C4CC !default;
```

3.1.4 国际化

正如前述的业务复杂化及拓展需求的不断深耕,商业化出海也成为各大国内互联网厂商竞相追逐获取更多新兴市场下的战略方向,因此,国际化方案不仅是前端工程领域的一项重要课题,就组件库体系而言也同样是一项不可忽视的重要内容。在 Element UI 中,其国际化方案主要提供了 t()、use()及 i18n()3 个功能函数,如图 3-8 所示。

其中,对于 i18n()及 use()的实现方式,代码如下:

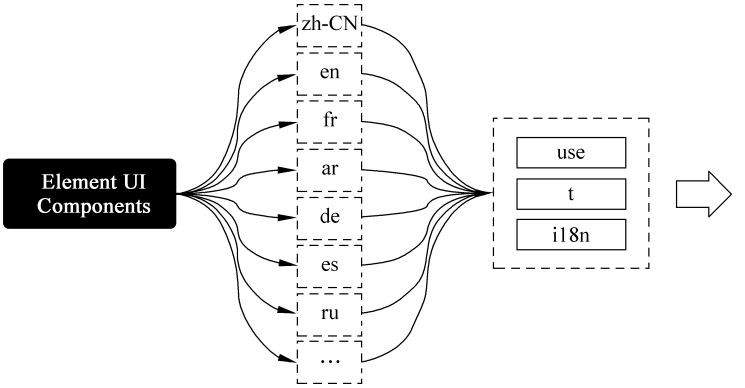


图 3-8　Element UI 国际化方案

```
//第 3 章/locale.js
const format = Format(Vue);
let i18nHandler = function() {
  const vuei18n = Object.getPrototypeOf(this || Vue).$t;
  if (typeof vuei18n === 'function' && !!Vue.locale) {
    if (!merged) {
      merged = true;
      Vue.locale(
        Vue.config.lang,
        deepmerge(lang, Vue.locale(Vue.config.lang) || {}, { clone: true })
      );
    }
    return vuei18n.apply(this, arguments);
  }
};
export const t = function(path, options) {
  let value = i18nHandler.apply(this, arguments);
  if (value !== null && value !== undefined) return value;
  const array = path.split('.');
  let current = lang;
  for (let i = 0, j = array.length; i < j; i++) {
    const property = array[i];
    value = current[property];
    if (i === j - 1) return format(value, options);
    if (!value) return '';
    current = value;
  }
  return '';
};
export const use = function(l) {
  lang = l || lang;
};
export const i18n = function(fn) {
  i18nHandler = fn || i18nHandler;
}
```

对于 t(),则需要用到 format()的字符串模板转化,其代码如下:

```
//第 3 章/format.js
const RE_NARGS = /(%|)\{([0-9a-zA-Z_]+)\}/g;
export default function(Vue) {
  function template(string, ...args) {
    if (args.length === 1 && typeof args[0] === 'object') {
      args = args[0];
    }
    if (!args || !args.hasOwnProperty) {
      args = {};
    }
    return string.replace(RE_NARGS, (match, prefix, i, index) => {
      let result;
      if (string[index - 1] === '{' &&
        string[index + match.length] === '}') {
        return i;
      } else {
        result = hasOwn(args, i) ? args[i] : null;
        if (result === null || result === undefined) {
          return '';
        }
        return result;
      }
    });
  }
  return template;
}
```

3.1.5 文档

在成熟的组件库体系中,为了能让开发及设计等人员更好地使用和了解组件库功能及价值,通常来讲也会对应提供官方文档以展现组件样例及设计理念。在 Element UI 2.x 中,由于其本身基于 Vue 2.x 的框架建设,故而其文档构建也采用了基于 Vue 全家桶的工程化方案,然而,不同于以 HTML 为主的网页应用建设,开发人员通常会选用以 MarkDown 为主的轻量级标记语言对文档进行编写,因此,对于组件库文档而言,就需要能够对 MarkDown 语法与 HTML 语法进行相应转化,然而,由于 Element UI 2.x 是基于 Vue 全家桶文档进行构建的,故而只需对.md 文件配合.vue 文件进行相应处理,如图 3-9 所示。

在 Element UI 2.x 中,其使用的是基于 Webpack 的打包构建,因而对于.md 文件需要通过 md-loader 和 vue-loader 进行处理,代码如下:

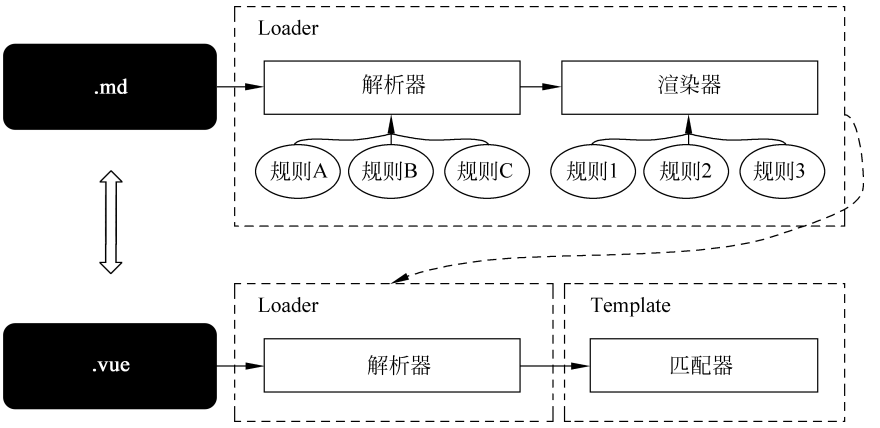


图 3-9 Element UI 文档建设

```
//第 3 章/webpack.demo.js
const webpackConfig = {
  module: {
    rules: [
      {
        test: /\.md$/,
        use: [
          //vue-loader
          {
            loader: 'vue-loader',
            options: {
              compilerOptions: {
                preserveWhitespace: false
              }
            }
          },
          //md-loader
          {
            loader: path.resolve(__dirname, './md-loader/index.js')
          }
        ]
      }
    ]
  }
}
```

在 Element UI 2.x 中，其通过 markdown-it 及其相关插件实现了 md-loader，代码如下：

```
//第 3 章/md-loader.js
module.exports = function(source) {
  //输出逻辑
  return `
```

```
      <template>
        <section class="content element-doc">
          ${output.join('')}
        </section>
      </template>
      ${pageScript}
    `;
  }
```

注意：loader 是 Webpack 中处理模块资源的一种转换器，其本质是一个函数且支持自定义，对于 Webpack 的具体分析会在后续章节中进行阐述。

其中，经过.md 文件到.vue 文件的转换后，配合自定义的 demo-block 显示组件便可实现组件及源码的同步展现，代码如下：

```
<!-- 第3章/demo-block.vue -->
<template>
  <div
    class="demo-block">
    <div class="source">
      <!-- 源码 -->
      <slot name="source"></slot>
    </div>
    <div class="meta" ref="meta">
      <div class="description" v-if="$slots.default">
        <slot></slot>
      </div>
      <div class="highlight">
        <slot name="highlight"></slot>
      </div>
    </div>
  </div>
</template>
```

3.1.6 资源

为了更好地提供资产沉淀，组件库体系同样也需要提供相应的资源文件来支持业务方的自定义扩展。通常来讲，资源文件默认会提供设计规范说明书、设计组件包及设计工具插件等内容。在 Element UI 中，其提供了 Axure 及 Sketch 相关设计软件的设计组件包，如图 3-10 所示。

图 3-10　Element UI 设计资源

3.2　Ant Design

22min

如果说“千团大战”是中国互联网巨头在团购市场的一次“烧钱大赛”，则促使现象产生背后而颠覆用户商业行为的底层原始推动力完全源自支付方式的彻底变革。毋庸置疑，诞生于 2004 年的支付宝起初的作用仅仅是为了解决阿里巴巴旗下的淘宝平台交易当中的信任问题，然而，随着移动互联网的发展，由于其为用户提供了极大便利，支付宝逐渐渗入衣食住行的各个环节中，同时也极大地改变了人们的行为和生活方式。在 2013 年“All In 无线”之后，随着支付宝前线业务的开疆拓土，中后台需求与日俱增，而缺少与之相应的成熟体系方案却成为用户增长的瓶颈。再者，加之肯德基接入支付宝的不顺，终于成为压死受中后台羸弱弊病困扰下支付宝的最后一根稻草。面对如此量级的中后台产品，建立统一的 UI 设计/前端框架便成为支付宝时下的当务之急。于是，在 2015 年 4 月，为了未来能支撑起庞大的中后台业务，也为各自的专业发展，与支付宝母公司蚂蚁金服同名且同时寓意着“艺术与科技”(Art and Technology)的组件库体系——Ant Design 终于呱呱坠地。

Ant Design 从诞生以来，大致经历了 6 个时间发展节点。在 2015 年，Ant Design 以 0.6 版本首次向世人掀开自己的神秘面纱；仅隔一年之后，Ant Design 1.0 版本正式发布。随着体系的不断沉淀，2.0 版本的 Ant Design 同时提供了面向移动端的 Ant Design Mobile 及交互动效规范的 Ant Motion 及 Ant Landing 等。到了 3.0 时代，Ant Design 则推出了提升设计工作效率的 Sketch 工具集——Kitchen。更进一步，Ant Design 4.0 又对插画资产提出了自己的最佳解决方案，即海兔(HiTu)插画组件库。到目前为止，以 5.0 规范下的领域子模型的提出为例，Ant Design 组件库体系一直都致力于通过科技与艺术来不断地提升用户体验，如图 3-11 所示。

正如 Ant Design 版本升级之路一样，相较于 Element UI，Ant Design 庞大的生态体系则包罗万象，其成熟的生态产品包括 Ant Design Pro、HiTu、Tech UI、Ant Design Vue、Ant Landing、Ant Motion、Kitchen、Ant Design Mobile 等。

同样地，本节仍将通过指南、组件、主题、国际化、文档及资源等几部分来分别展开介绍

图 3-11 Ant Design 历史背景

Ant Design 的整个组件库体系，以期能够让读者更加全面地透析成熟组件库体系所必须涵盖的范围与层次。

3.2.1 指南

所谓“内正其心，外正其容”，组件库体系的价值判断决定了组件库的价值选择，也为设计者提供评价设计好坏的内在标准，并且对组件库本身的使命与意义提供最本质的衡量基准，启示并激发了组件库体系中趋向发展的判定，进而为具体设计问题提供向导和一般解决方案，因此，本节仍将通过设计哲学、设计风格及设计样式 3 个方面来阐述 Ant Design 的设计理念，以对比不同价值体系下的组件库方案的具体呈现。

1. 设计哲学

“知性始于感性，理性行于知性”，在人机交互设计中，无意识与有意识常常造成了人与系统的力量融合和互相影响。不同于其他设计体系的理念坚持，Ant Design 有 4 点与众不同，可将这些不同归纳为自然、确定、生长及意义，这也可看作其设计原则。其中，自然主要体现在感知自然和行为自然，其为解决数字世界的复杂化与注意力资源稀缺化的矛盾提供了追寻的方向；确定主要表现为设计确定及用户确定，其决定了人机交互的高确定性与低合作熵的状态；生长则重点包括价值连接和人机共生，其表达了设计者对产品功能价值的可发现性及创造性；意义更多是承载用户的需求与产品的使命，其可分为结果的意义和过程的意义。设计哲学高度凝练地归纳总结了设计体系的价值体现与表达诉求，其能大幅度提升研发团队的确定性且保持系统一致性，如图 3-12 所示。

2. 设计风格

同样地，对 Ant Design 而言，其遵循的也是扁平化的设计理念。不同于 Element UI，Ant Design 结合时下设计趋势及企业发展变化，在不同版本中对扁平化风格相应地进行了细节调整。以 Ant Design 5. x 为例，其设计风格变得更加轻量与简洁，如图 3-13 所示。

3. 设计样式

用户界面作为系统和用户之间进行交互和信息交换的媒介，其不仅包括界面美观的设

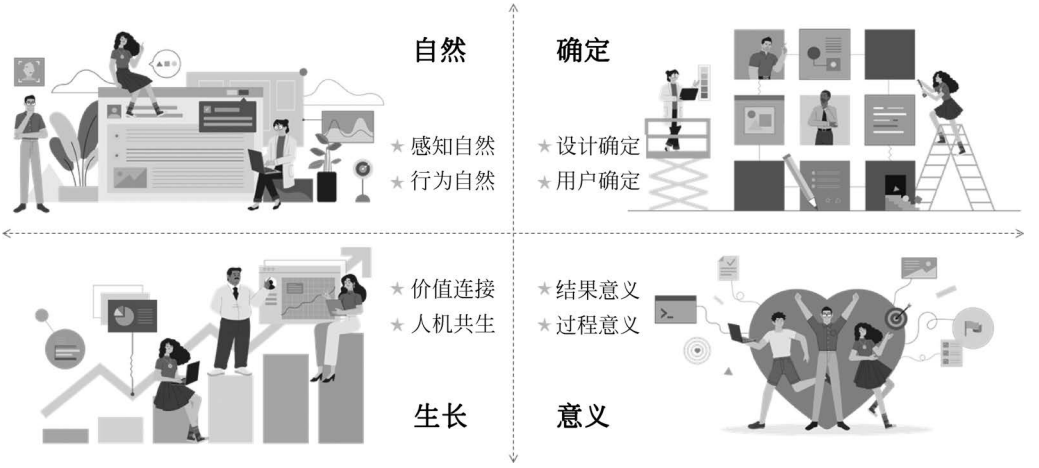


图 3-12　Ant Design 设计哲学

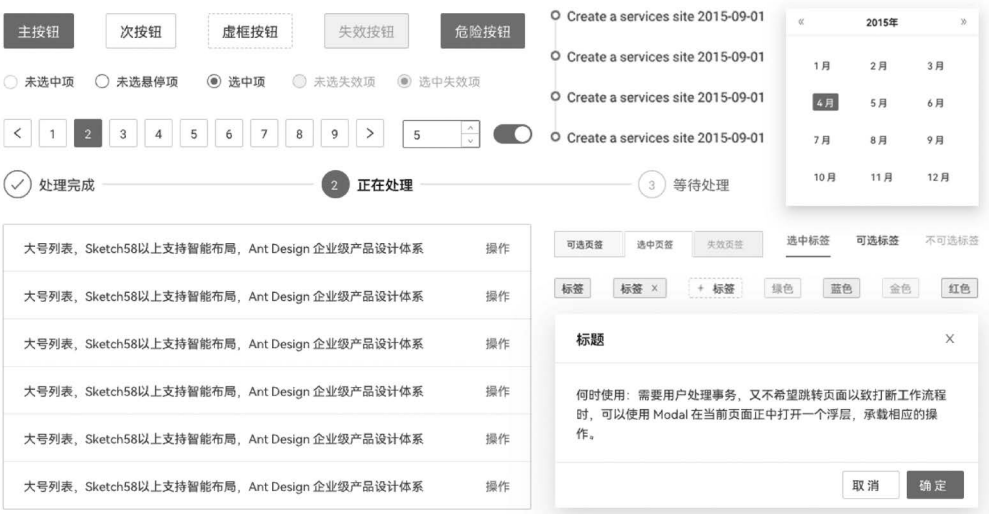


图 3-13　Ant Design 设计风格

计，也同时囊括人机交互和操作逻辑的设计，因此，对于色彩、布局及字体样式而言，也同样要追寻设计中的韵律感与动态感。

在 Ant Design 中，其色彩体系可解读成系统级色彩体系和产品级色彩体系两个层面。系统级色彩体系主要以色板的功能方式进行提供，将色彩通过一定的模型算法而延展出自然规律下的变化，平衡了可读性、美感及可得性，如图 3-14 所示。

类似地，与 Element UI 的色彩样式的分类方式相仿，产品级色彩体系则重点以“主色＋功能色＋中性色”的方式提供企业级的视觉传达，如图 3-15 所示。

空间布局是体系化视觉设计的起点，相较于传统平面设计，UI 设计中的空间布局则更

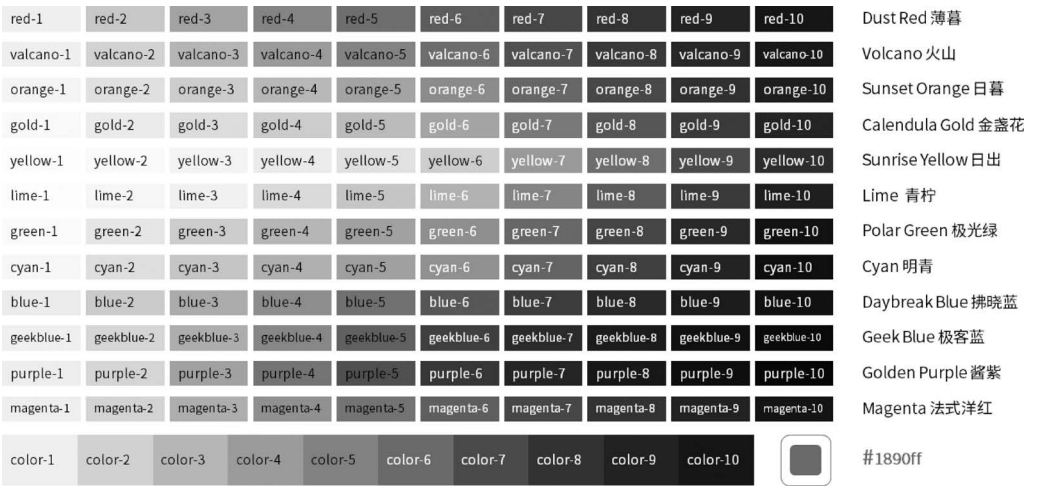
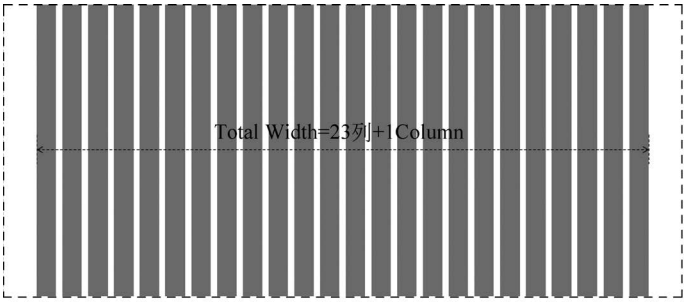


图 3-14　Ant Design 系统级色彩样式

图 3-15　Ant Design 产品级色彩样式

加动态且富有变化。受现代主义建筑大师勒·柯布西耶(Le Corbusier)的模度思想启发，Ant Design 采用网格基数为 8 的偶数分隔粒度提供具备动态感和韵律感的布局决策。同样地，Ant Design 也采用了 24 分栏的栅格体系，如图 3-16 所示。

字体是体系化界面设计中的重要构成之一，Ant Design 基于动态秩序的有效原则，分别对字体家族、字体基准、字体颜色等相应地进行了约束，力求在视觉展现上满足路德维希·密斯·凡德罗(Ludwig Mies van der Rohe)所提出的“少即是多”(less is more)的现代主义风格设计理念，如图 3-17 所示。



4	16	24	40	64	104	168	272	440	720	1152	1920
8	12	20	32	48	80	128	208	336	552	896	1440

图 3-16　Ant Design 布局样式

Font Size	12	14	16	20	24	30	38	46	56	…
Line Height	20	22	24	28	32	38	46	54	64	…

图 3-17　Ant Design 字体样式

3.2.2　组件

在面向对象编程中，常常会用到设计模式的解决方案。同样地，在 UI 设计中，设计模式也被广泛地应用到组件库体系设计的方方面面，其可以显著地增加研发团队的确定性及系统一致性，节省不必要的开销，提高设计与开发的协作效率。作为企业级业务的最佳践行者，Ant Design 中落地设计模式也同样遵循其设计哲学，并为企业产品中反复出现的设计问题提供通用的解决方案。设计者既可以直接使用设计模式来完成界面设计，也可以从设计模式出发，衍生更加贴合业务的解决方案，以满足个性化的设计需求，因此，为了更好快速高效地提供业务开发能力，Ant Design 以功能层、页面层、模块层、组件层来对业务形态进行拆解，其可规划为六大类组件，分别为通用类组件（General）、布局类组件（Layout）、导航类组件（Navigation）、数据类组件（Data）、反馈类组件（Feedback）及其他类组件（Other），如图 3-18 所示。

注意：在 Ant Design 中，数据类组件包括数据录入类组件和数据展示类组件。

回顾 Ant Design 1.0 到 5.0 的进化过程，开源社区的参与度也在逐步增加，广大社区开发者提供了许多功能组件，例如卡片组件（Card）、评分组件（Rate）、自动完成组件

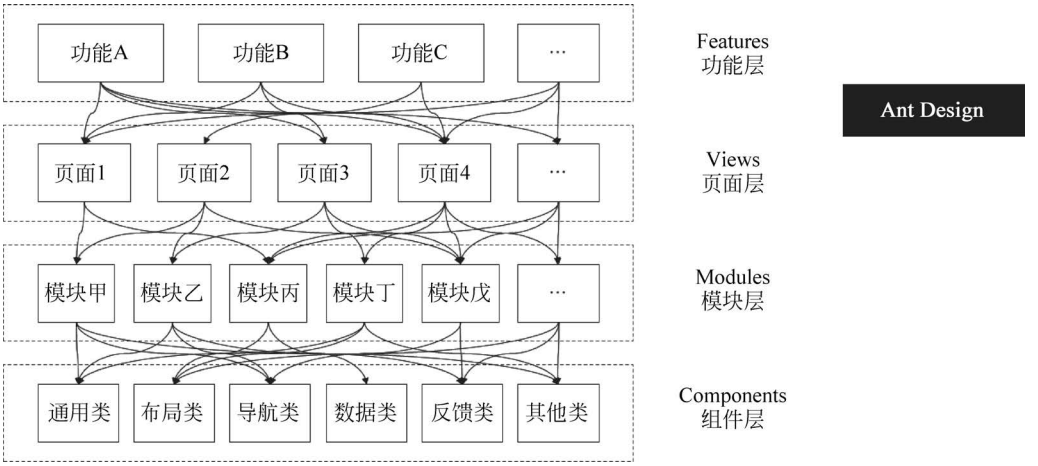


图 3-18　Ant Design 组件分类

(AutoComplete)、分割线组件(Divider)、分段控制器组件(Segmented)、包裹组件(App)、二维码组件(QRCode)等,因而,本节以 Ant Design 5.5.x 版本代码为例相应地进行实现分析,其所依赖的框架为 React 18.x,模板代码如下:

```
//从当前组件目录下引入组件,组件名称按照对应分类进行命名
import ComponentA from './componentA';
import ComponentB from './componentB';
function Component() {}
Component.ComponentA = ComponentA
Component.ComponentB = ComponentB
//导出组件
export default Component;
```

注意:上述代码为抽象出来的组件模板代码,具体对应组件源码网址可见对应分类组件中的表格备注项。

1. 通用类组件

在通用类组件中,Ant Design 提供了按钮组件(Button)、图标组件(Icon)及排版组件(Typography)等,见表 3-7。

表 3-7　Ant Design 通用类组件

组件名称	组 件 包	备　　注
Button	Button	Ant Design 仓库/components/button/button.tsx
	ButtonGroup	Ant Design 仓库/components/button/button-group.tsx
Icon	Icon	Ant Design 仓库/components/icon/index.ts

续表

组件名称	组 件 包	备　　注
Typography	Typography	Ant Design 仓库/components/typography/Typography. tsx
	Text	Ant Design 仓库/components/typography/Text. tsx
	Title	Ant Design 仓库/components/typography/Title. tsx
	Paragraph	Ant Design 仓库/components/typography/Paragraph. tsx
	Link	Ant Design 仓库/components/typography/Link. tsx

2. 布局类组件

在布局类组件中，Ant Design 提供了分割线组件(Divider)、栅格组件(Grid)、布局组件(Layout)及间距组件(Space)等，见表 3-8。

表 3-8　Ant Design 布局类组件

组件名称	组 件 包	备　　注
Divider	Divider	Ant Design 仓库/components/divider/index. tsx
Grid	Row	Ant Design 仓库/components/grid/row. tsx
	Col	Ant Design 仓库/components/grid/col. tsx
Layout	Layout	Ant Design 仓库/components/layout/layout. tsx Ant Design 仓库/components/layout/Sider. tsx
	Header	
	Content	
	Footer	
	Sider	
Space	Space	Ant Design 仓库/components/space/index. tsx
	Compact	Ant Design 仓库/components/space/Compact. tsx

3. 导航类组件

在导航类组件中，Ant Design 提供了锚点组件(Anchor)、面包屑组件(Breadcrumb)、下拉菜单组件(Dropdown)、导航菜单组件(Menu)、分页组件(Pagination)及步骤条组件(Steps)等，见表 3-9。

表 3-9　Ant Design 导航类组件

组件名称	组 件 包	备　　注
Anchor	Anchor	Ant Design 仓库/components/anchor/Anchor. tsx
	AnchorLink	Ant Design 仓库/components/anchor/AnchorLink. tsx
Breadcrumb	Breadcrumb	Ant Design 仓库/components/breadcrumb/Breadcrumb. tsx
	BreadcrumbItem	Ant Design 仓库/components/breadcrumb/BreadcrumbItem. tsx
	BreadcrumbSeparator	Ant Design 仓库/components/breadcrumb/BreadcrumbSeparator. tsx
Dropdown	Dropdown	Ant Design 仓库/components/dropdown/dropdown. tsx
	DropdownButton	Ant Design 仓库/components/dropdown/dropdown-button. tsx

续表

组件名称	组 件 包	备　　注
Menu	Menu	Ant Design 仓库/components/menu/index. tsx
	Item	Ant Design 仓库/components/menu/MenuItem. tsx
	SubMenu	Ant Design 仓库/components/menu/SubMenu. tsx
	MenuDivider	Ant Design 仓库/components/menu/MenuDivider. tsx
	ItemGroup	React Component 仓库/src/MenuItemGroup. tsx
Pagination	Pagination	Ant Design 仓库/components/pagination/Pagination. tsx
Steps	Steps	Ant Design 仓库/components/steps/Steps. tsx
	Step	React Component 仓库/src/Step. tsx

4. 数据类组件

在 Ant Design 中,数据类组件可分为两种,分别是数据录入类组件及数据展示类组件。

在数据录入类组件中,Ant Design 提供了自动完成组件(AutoComplete)、级联选择组件(Cascader)、复选框组件(Checkbox)、颜色选择器组件(ColorPicker)、日期选择器框组件(DatePicker)、表单组件(Form)、输入框组件(Input)、数字输入框组件(InputNumber)、提及组件(Mentions)、单选框组件(Radio)、评分组件(Rate)、选择器组件(Select)、滑动输入条组件(Slider)、开关组件(Switch)、时间选择框组件(TimePicker)、穿梭框组件(Transfer)、树选择组件(TreeSelect)及上传组件(Upload)等,见表 3-10。

表 3-10　Ant Design 数据录入类组件

组件名称	组 件 包	备　　注
AutoComplete	AutoComplete	Ant Design 仓库/components/auto-complete/index. tsx
Cascader	Cascader	Ant Design 仓库/components/cascader/index. tsx
Checkbox	Checkbox	Ant Design 仓库/components/checkbox/Checkbox. tsx
	CheckboxGroup	Ant Design 仓库/components/checkbox/Group. tsx
ColorPicker	ColorPicker	Ant Design 仓库/components/color-picker/ColorPicker. tsx
DatePicker	DatePicker	Ant Design 仓库/components/date-picker/index. ts
	RangePicker	Ant Design 仓库/components/date-picker/index. ts
Form	Form	Ant Design 仓库/components/form/Form. tsx
	FormItem	Ant Design 仓库/components/form/FormItem/index. tsx
	FormList	Ant Design 仓库/components/form/FormList. tsx
	ErrorList	Ant Design 仓库/components/form/ErrorList. tsx
	FormProvider	Ant Design 仓库/components/form/context. tsx
Input	Input	Ant Design 仓库/components/input/Input. tsx
	Group	Ant Design 仓库/components/input/Group. tsx
	Search	Ant Design 仓库/components/input/Search. tsx
	TextArea	Ant Design 仓库/components/input/TextArea. tsx
	Password	Ant Design 仓库/components/input/Password. tsx
InputNumber	InputNumber	Ant Design 仓库/components/input-number/index. tsx

续表

组件名称	组 件 包	备 注
Mentions	Mentions	Ant Design 仓库/components/mentions/index. tsx
	Option	React Component 仓库/src/Option. tsx
Radio	Radio	Ant Design 仓库/components/radio/radio. tsx
	RadioGroup	Ant Design 仓库/components/radio/group. tsx
	RadioButton	Ant Design 仓库/components/radio/radioButton. tsx
Rate	Rate	Ant Design 仓库/components/rate/index. tsx
Select	Select	Ant Design 仓库/components/select/index. tsx
	Option	React Component 仓库/src/Option. tsx
	OptionGroup	React Component 仓库/src/OptGroup. tsx
Slider	Slider	Ant Design 仓库/components/slider/index. tsx
Switch	Switch	Ant Design 仓库/components/switch/index. tsx
TimePicker	TimePicker	Ant Design 仓库/components/time-picker/index. tsx
	RangePicker	
Transfer	Transfer	Ant Design 仓库/components/transfer/index. tsx
	TransferList	Ant Design 仓库/components/transfer/list. tsx
	Search	Ant Design 仓库/components/transfer/search. tsx
	Operation	Ant Design 仓库/components/transfer/operation. tsx
TreeSelect	TreeSelect	Ant Design 仓库/components/tree-select/index. tsx
	TreeNode	React Component 仓库/src/TreeNode. tsx
Upload	Upload	Ant Design 仓库/components/upload/Upload. tsx
	Dragger	Ant Design 仓库/components/upload/Dragger. tsx

在数据展示类组件中，Ant Design 提供了头像组件（Avatar）、徽标组件（Badge）、日历组件（Calendar）、卡片组件（Card）、走马灯组件（Carousel）、折叠面板组件（Collapse）、描述列表组件（Descriptions）、空状态组件（Empty）、图片组件（Image）、列表组件（List）、气泡卡片组件（Popover）、二维码组件（QRCode）、分段控制器组件（Segmented）、统计数值组件（Statistic）、表格组件（Table）、标签页组件（Tabs）、标签组件（Tag）、时间轴组件（Timeline）、文字提示组件（Tooltip）、漫游式引导组件（Tour）及树形组件（Tree）等，见表 3-11。

表 3-11 Ant Design 数据展示类组件

组件名称	组 件 包	备 注
Avatar	Avatar	Ant Design 仓库/components/avatar/avatar. tsx
	Group	Ant Design 仓库/components/avatar/group. tsx
Badge	Badge	Ant Design 仓库/components/badge/index. tsx
	Ribbon	Ant Design 仓库/components/badge/Ribbon. tsx
Calendar	Calendar	Ant Design 仓库/components/calendar/index. tsx

续表

组件名称	组 件 包	备　　注
Card	Card	Ant Design 仓库/components/card/Card. tsx
	Grid	Ant Design 仓库/components/card/Grid. tsx
	Meta	Ant Design 仓库/components/card/Meta. tsx
Carousel	Carousel	Ant Design 仓库/components/carousel/index. tsx
Collapse	Collapse	Ant Design 仓库/components/collapse/index. ts
Descriptions	Descriptions	Ant Design 仓库/components/descriptions/index. tsx
	DescriptionsItem	Ant Design 仓库/components/descriptions/Item. tsx
Empty	Empty	Ant Design 仓库/components/empty/index. tsx
Image	Image	Ant Design 仓库/components/image/index. tsx
	PreviewGroup	Ant Design 仓库/components/image/PreviewGroup. tsx
List	List	Ant Design 仓库/components/list/index. tsx
	Item	Ant Design 仓库/components/list/Item. tsx
	Meta	
Popover	Popover	Ant Design 仓库/components/popover/index. tsx
QRCode	QRCode	Ant Design 仓库/components/qrcode/index. tsx
Segmented	Segmented	Ant Design 仓库/components/segmented/index. tsx
Statistic	Statistic	Ant Design 仓库/components/statistic/Statistic. tsx
	Countdown	Ant Design 仓库/components/statistic/Countdown. tsx
Table	Table	Ant Design 仓库/components/table/Table. tsx
	Column	Ant Design 仓库/components/table/Column. ts
	ColumnGroup	Ant Design 仓库/components/table/ColumnGroup. ts
	Summary	React Component 仓库/src/Footer/Summary. tsx
Tabs	Tabs	Ant Design 仓库/components/tabs/index. tsx
	TabPane	Ant Design 仓库/components/tabs/TabPane. ts
Tag	Tag	Ant Design 仓库/components/tag/index. tsx
Timeline	Timeline	Ant Design 仓库/components/timeline/Timeline. tsx
	TimelineItem	Ant Design 仓库/components/timeline/TimelineItem. tsx
Tooltip	Tooltip	Ant Design 仓库/components/tooltip/index. tsx
Tour	Tour	Ant Design 仓库/components/tour/index. tsx
Tree	Tree	Ant Design 仓库/components/tree/index. ts
	TreeNode	React Component 仓库/src/TreeNode. tsx
	DirectoryTree	Ant Design 仓库/components/tree/DirectoryTree. tsx

5. 反馈类组件

在反馈类组件中，Ant Design 提供了警告提示组件(Alert)、抽屉组件(Drawer)、全局提示组件(Message)、对话框组件(Modal)、通知提醒组件(Notification)、气泡确认框组件(Popconfirm)、进度条组件(Progress)、结果组件(Result)、骨架屏组件(Skeleton)及加载中组件(Spin)等，见表 3-12。

表 3-12 Ant Design 反馈类组件

组件名称	组 件 包	备 注
Alert	Alert	Ant Design 仓库/components/alert/index. tsx
	ErrorBoundary	Ant Design 仓库/components/alert/ErrorBoundary. tsx
Drawer	Drawer	Ant Design 仓库/components/drawer/index. tsx
Message	message	Ant Design 仓库/components/message/index. tsx
Modal	Modal	Ant Design 仓库/components/modal/Modal. tsx
Notification	notification	Ant Design 仓库/components/notification/index. tsx
Popconfirm	Popconfirm	Ant Design 仓库/components/popconfirm/index. tsx
Progress	Progress	Ant Design 仓库/components/progress/index. tsx
Result	Result	Ant Design 仓库/components/result/index. tsx
Skeleton	Skeleton	Ant Design 仓库/components/skeleton/Skeleton. tsx
	SkeletonButton	Ant Design 仓库/components/skeleton/Button. tsx
	SkeletonAvatar	Ant Design 仓库/components/skeleton/Avatar. tsx
	SkeletonInput	Ant Design 仓库/components/skeleton/Input. tsx
	SkeletonImage	Ant Design 仓库/components/skeleton/Image. tsx
	SkeletonNode	Ant Design 仓库/components/skeleton/Node. tsx
Spin	Spin	Ant Design 仓库/components/spin/index. tsx

6. 其他类组件

对于没有具体类别的组件，Ant Design 将其划分到其他类组件中，包括固钉组件(Affix)、包裹组件(App)、全局化配置组件(ConfigProvider)、悬浮按钮组件(FloatButton)及水印组件(Watermark)等，见表 3-13。

表 3-13 Ant Design 其他类组件

组件名称	组 件 包	备 注
Affix	Affix	Ant Design 仓库/components/affix/index. tsx
App	App	Ant Design 仓库/components/app/index. tsx
ConfigProvider	ConfigProvider	Ant Design 仓库/components/config-provider/index. tsx
	ConfigContext	Ant Design 仓库/components/config-provider/context. tsx
	SizeContext	Ant Design 仓库/components/config-provider/SizeContext. tsx
FloatButton	FloatButton	Ant Design 仓库/components/float-button/FloatButton. tsx
	FloatButtonGroup	Ant Design 仓库/components/float-button/FloatButtonGroup. tsx
	BackTop	Ant Design 仓库/components/float-button/BackTop. tsx
Watermark	Watermark	Ant Design 仓库/components/watermark/index. tsx

3.2.3 主题

为满足业务和品牌多样化的视觉需求，组件库体系都会在设计规范和技术上支持灵活的样式定制，包括但不限于全局样式和指定组件的视觉呈现，例如主色、圆角、边框、阴影、层

级及动效等。相较于前几个版本的主题定制方案，Ant Design 5.0 提供了一套基于设计令牌(Design Token)思想而抽提出符合自身设计理念的全新体系化定制主题方案，如图 3-19 所示。

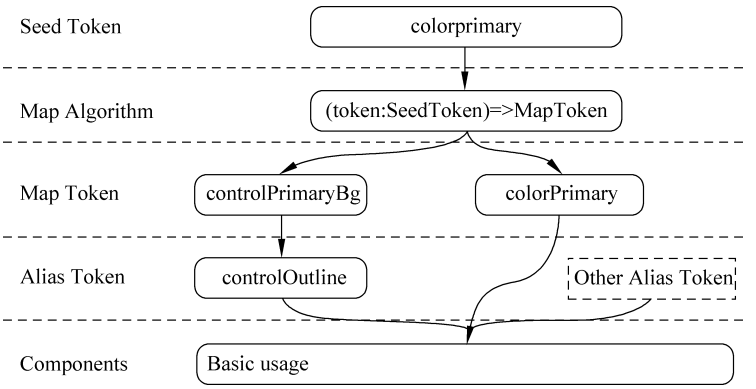


图 3-19　Ant Design 主题配置

注意：Design Token 是设计系统中的视觉原子，可翻译为设计指令或者设计令牌，是帮助设计师和开发工程师建立起表达决策的通用语言。

在 Ant Design 5.0 中，其提供了一套通用的 Token 生成算法，代码如下：

```
//第 3 章/genCommonMapToken.ts
export default function genCommonMapToken(token: SeedToken): CommonMapToken {
  const { motionUnit, motionBase, borderRadius, lineWidth } = token;
  return {
    //motion
    motionDurationFast: `${(motionBase + motionUnit).toFixed(1)}s`,
    motionDurationMid: `${(motionBase + motionUnit * 2).toFixed(1)}s`,
    motionDurationSlow: `${(motionBase + motionUnit * 3).toFixed(1)}s`,
    //line
    lineWidthBold: lineWidth + 1,
    //radius
    ...genRadius(borderRadius),
  };
}
```

以色彩生成方案为例，Ant Design 将自定义颜色的色板生成算法抽离到一个通用的@ant-design/colors 库中。值得一提的是，在 Ant Design 不同版本中对于色板生成的方案有着十分巨大的差异。

在 Ant Design 1.x 色板算法中，其主要使用的是基于 RGB 模型线性组合的黑白混合算法，然而，RGB 色彩模型线性换算过程中间隔变化较大，对于人眼视觉效果及感性认知不友好。

在 2.x 色板算法中，Ant Design 使用了基于 HSL 模型进行贝塞尔曲线（Bézier Curve）拟合后再通过对灰度的判断分别进行加深及减弱的分化粒度算法。相较于 RGB 色彩模型，HSL 色彩模型则可以利用色相（Hue）的旋转对应地进行加深和减弱，更加符合人的视觉感知。

注意：贝塞尔曲线是应用于二维图形应用程序的数学曲线，常用来进行数据拟合。

对于 HSL 色彩模型而言，HSV 模型的变化梯度则更加柔和且易于控制，因而，在 Ant Design 3.x 中，Ant Design 对于灰度判断的理论依据不再基于贝塞尔曲线拟合的 HSL 色彩模型而重新改为线性调整的 HSV 的色彩模型，简化算法的混合方案。

注意：色彩模型指的是某个三维颜色空间中的一个可见光子集，包含某个色彩域的所有色彩，常见的色彩模型有 RGB、CMYK、HSL、HSV 等。

在 Ant Design 5.0 中，其仍沿袭了自 3.0 版本之后的基于 HSV 色彩模型的自定义色板算法，对应代码如下：

```
//第 3 章/generate.ts
export default function generate(color: string, opts: Opts = {}): string[]
  {
  const patterns: string[] = [];
  const pColor = inputToRGB(color);
  //light 主题
  for (let i = lightColorCount; i > 0; i -= 1) {}
  //dark 主题
  for (let i = 1; i <= darkColorCount; i += 1) {}
  //暗黑主题
  if (opts.theme === 'dark') {
    return darkColorMap.map(({ index, opacity }) => {
      const darkColorString: string = toHex(
        mix(
          inputToRGB(opts.backgroundColor || '#141414'),
          inputToRGB(patterns[index]),
          opacity * 100,
        ),
      );
      return darkColorString;
    });
  }
  return patterns;
}
```

可以看出，对于 Ant Design 5.0 的分割粒度方案而言，色相（Hue）的判断区位为 60～240，饱和度（Saturation）和明度（Value）的加深及减弱梯度则有明显区别，并且减弱的幅度更大。

注意：学术界对色彩的研究十分复杂，包含色彩理论、色彩模型、色彩空间、色彩计算等，感兴趣的读者可查阅相关资料进行深入学习。

3.2.4 国际化

相应地，作为面向服务全球金融市场的引领者，蚂蚁金服前端的国际化之路同样早已深耕入局。在 Ant Design 中，其国际化方案主要提供了 ConfigProvider 的组件，用于全局地进行国际化配置，如图 3-20 所示。

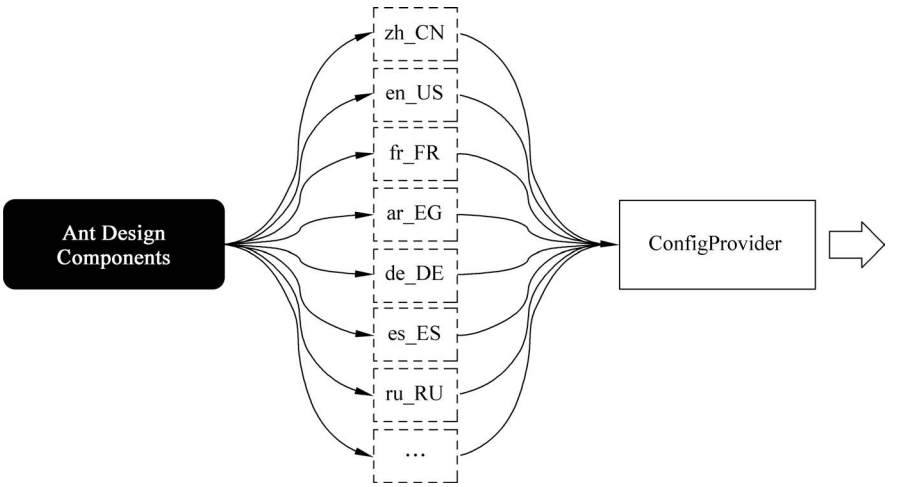


图 3-20 Ant Design 国际化方案

其中，ConfigProvider 配置提供组件的实现，代码如下：

```
//第 3 章/config - provider.tsx
const ConfigProvider: React.FC < ConfigProviderProps > & {
  ConfigContext: typeof ConfigContext;
  SizeContext: typeof SizeContext;
  config: typeof setGlobalConfig;
  useConfig: typeof useConfig;
} = (props) => {
  const context = React.useContext < ConfigConsumerProps >(ConfigContext);
  const antLocale = React.useContext < LocaleContextProps | undefined >(LocaleContext);
  return < ProviderChildren parentContext = {context} legacyLocale = {antLocale!} {...props}
/>;
}
```

除此之外，对于现代化的 React 开发体系，Ant Design 5.0 也提供了配置方案的 hooks 函数 useConfig()，其对应代码如下：

```
//第 3 章/useConfig.ts
function useConfig() {
  const componentDisabled = useContext(DisabledContext);
  const componentSize = useContext(SizeContext);
  return {
    componentDisabled,
    componentSize,
  };
}
```

3.2.5 文档

在软件工程领域，开发文档是软件开发使用和维护过程中的必备资料。诚然，文档在组件库体系中也同样重要，其不仅有指导、帮助、解惑的作用，更关键的是它也是树立组件库体系品牌、提升团队影响力的平台媒介。

在 Ant Design 文档化构建的方案中，除了 5.0 版本外，其他构建方案都是通过内部自研的 MarkDown 转化工具 bisheng 实现的，其是通过 mark-twain 的前端工具包进行的 MarkDown 语法解析，代码如下：

```
//第 3 章/markdown.js
module.exports = function (filename, fileContent) {
  const markdown = markTwain(fileContent);
  markdown.meta.filename = toUriPath(filename);
  return markdown;
}
```

从 Ant Design 5.0 开始，其官方文档采用基于蚂蚁金服自研乌米(umi)框架而衍生出来的嘟米(dumi)文档工具进行组件库官网构建及部署。与业界常见的文档构建相比，嘟米是一款为组件开发场景而生的静态站点框架，如图 3-21 所示。

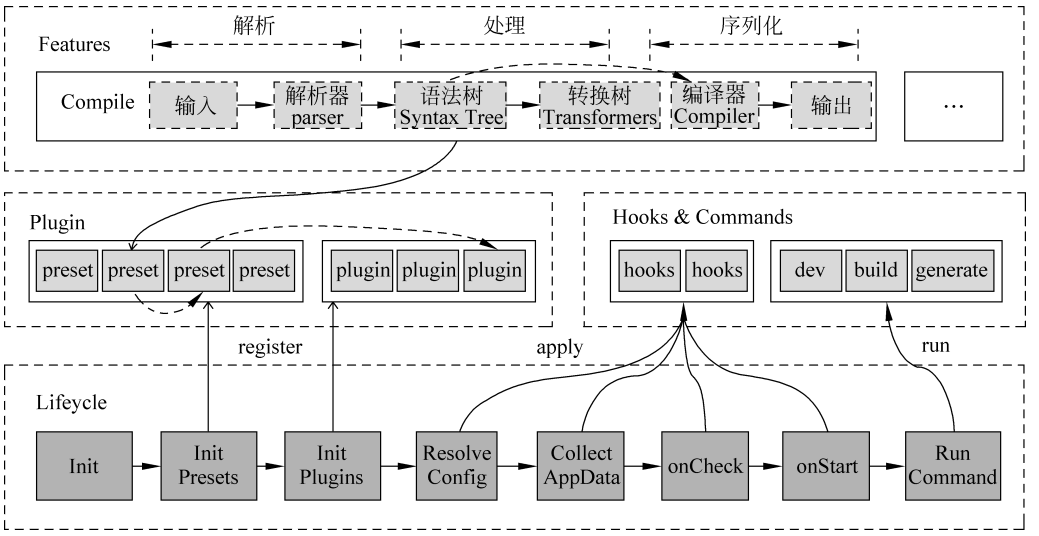


图 3-21 Ant Design 文档建设

在 Ant Design 5.0 官网构建中,Ant Design 团队采用的是 dumi 2.x 版本。对 dumi 2.x 而言,其整个构建的核心是对 demo、markdown、page 及 pre-raw 提供相应的 loader,以 markdown 为例,代码如下:

```
function emit() {
  //模板语法
  return Mustache.render()
}
//loader
export default function mdLoader() {}
```

其中,对 MarkDown 文档的转化则是通过 unified 内容转化工具集的自定义实现的,包括 rehype、remark 及 recma 等,代码如下:

```
//第 3 章/transformer.ts
export default async (raw: string, opts: IMdTransformerOptions) => {
  //加载额外的 remark 插件
  opts.extraRemarkPlugins?.forEach((plugin) =>
    applyUnifiedPlugin({
      plugin,
      processor,
      cwd: opts.cwd,
    }),
  );
  //加载额外的 rehype 插件
  opts.extraRehypePlugins?.forEach((plugin) =>
    applyUnifiedPlugin({
      plugin,
      processor,
      cwd: opts.cwd,
    }),
  );
  const result = await processor.use().process();
  return {
    content: String(result.value),
    meta: result.data,
  };
}
```

3.2.6 资源

庞大的组件库生态通常会包含多种多样的资源与模板。除此之外,组件库体系如果能够提供独有的资产套件,并且推广相应的软件应用,则能够提升业界的影响力并且也能发现更多的机会。对 Ant Design 而言,其提供了官方与社区两种资源物料,其中,官方资产包括设计资源和设计工具,例如 Sketch 组件包、Mobile Components、Ant Design Pro、Kitchen、Ant Design Landing 及 Chart 组件包等;相应地,社区贡献了对应平台或者通用设计软件的

套包，例如 xiaopiu 原型资源、Figma 组件包资源、墨刀原型资源、即时设计资源、MasterGo 组件包资源及 Raycast 拓展资源等，如图 3-22 所示。

图 3-22　Ant Design 设计资源

3.3　本章小结

2min

本章以工业时代下依托框架的两大组件库入手，分别介绍了 Element UI 和 Ant Design 组件库体系中的指南、组件、主题、国际化、文档及资源等内容，也简要地介绍了每个组件库所产生的背景与发展。为了更好地观察组件库体系从设计到落地的全貌，本节将通过对比分析不同的设计系统及前端发展历史阶段的典型组件库方案，以此来对本章做一个总结。

设计系统[1]（Design System），也称作设计体系，是指服务于客体的一系列具有关联性、有序性、标准性的设计集合整体。一般来讲，设计系统通常包含核心准则（Core Guidelines）、设计准则（Design Guidelines）、设计资产（Design Assets）、组件库（Component Libraries）、元系统（Meta Systems）、编码工具及其他资产（Code Tooling & other assets）等，其有着清晰的标准引导、机制化的组织流程及具象的指南和工具，用来帮助开发者、设计师及产品经理等高效地沟通协作，动态地确保用户体验的一致性。

注意：设计系统建设是一项十分庞大的工程，不仅包含前端工程师，同时也涵盖设计师、产品经理、运营人员等多个团队成员角色，有兴趣的读者可以在工作中不断地探索与实践。

由此看出，组件库体系不仅是设计系统的一部分，而且也拓展出了具有特定领域场景的新形态，因此，除了本章所介绍的设计体系之外，对已成熟商业化落地的设计系统例举如下，

感兴趣的读者可参考相关资料进行进一步学习,如图 3-23 所示。

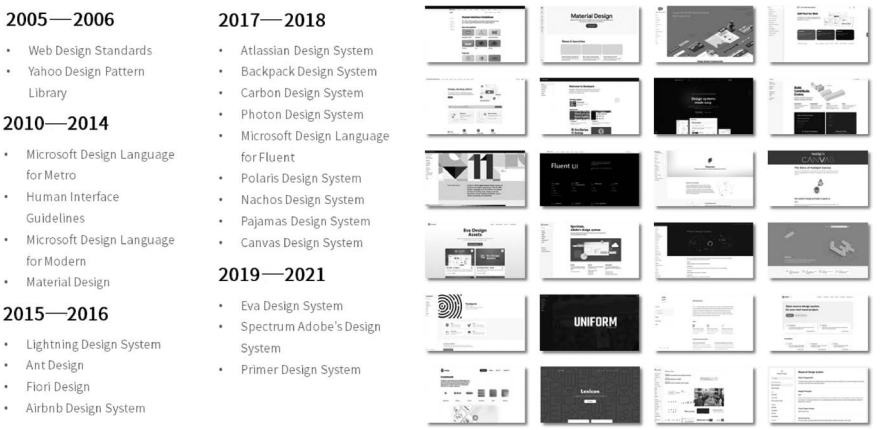


图 3-23 常见设计系统

以农业时代为起点,前端领域中出现了各种各样的组件库体系,对于业界常见的组件库体系进行如下总结,见表 3-14。

表 3-14 常见组件库体系对比

组件库名称	设计系统	团 队	诞生时期	工程
YUI	YUI	雅虎	农业时代	PC 端
Bootstrap	Bootstrap UI	Twitter	农业时代	PC 端 移动端
Angular Material	Material Design	谷歌	工业时代	PC 端 移动端
Element	Element UI	饿了么	工业时代	PC 端
Ant Design	Ant Design	蚂蚁金服	工业时代	PC 端 移动端
Iceworks	Ice Design	淘宝	信息时代	PC 端 移动端
Vant	Vant UI	有赞	信息时代	移动端 小程序
IView	View Design	视图更新科技	信息时代	PC 端
WeUI	We UI	微信	信息时代	小程序
Nut	Nut UI	京东	信息时代	PC 端
Chameleon	Chameleon UI	滴滴	信息时代	PC 端 移动端 小程序
DevUI	Dev UI	华为	信息时代	PC 端
Arco	Arco Design	今日头条	云边端时代	PC 端
Semi	Semi Design	抖音	云边端时代	PC 端

最后，前端组件库仅仅是组件库体系中的一个方向，对于多端体系下的组件库方案，希望各位读者能够通过组件库篇章的学习，从而构建出符合自己团队或公司特性的组件库体系。

从第 4 章开始，本书将会对前端工程中的包管理方案进行阐述。所有的工程方案都有各自的包管理系统，前端工程方案同样离不开系统级别的工具组合，希望各位读者能通过包管理的学习，制定出符合自己团队的包管理方案。