大前端技术栈系列

-- 前端工程化之路

蜂泰科技 | 徐 健

徐 健

- 大前端工程师
- 研发管理中心架构组成员
- 超过8年客户端/前端研发经验

怎么看前端?

> 前端开发是创建WEB页面或APP等前端界面呈现给用户的过程，通过HTML，CSS及JavaScript以及衍生出来的各种技术、框架、解决方案，来实现互联网产品的用户界面交互。 —— 百度百科

前端技术演进历史:

上古时代
1990 - 2004

- 静态页面
- js诞生
- 简单交互

web2.0时代
2004 - 2008

- 交互需求增多
- Ajax流行、异步请求
- 典型:jQuery

发展大爆炸时代
2008 - 2015

- V8引擎发布
- Node.js爆发
- MVVM/MVC/SPA等类型应用

今天的前端
2015 - 2020

- 小程序、跨容器、跨平台、跨OS等跨端
- 包管理、构建、框架、混合等标准成熟

未来的前端
2020 ~ ?

- 可视化
- 智能化
- 工具链
- serverless
- lowcode/nocode
- WebAssembly

工程化的意义?

> The only constant in the world is change.
> 世界上唯一不变的是变化。——《谁动了我的奶酪》作者 斯宾塞·约翰逊

工程化之前遇到的问题:

- 最早的前端开发就是实现页面，顶多再写写JS让页面可以有交互的特效。
- 但是随着需求的增加，我们不仅要做Web应用，还要做App、小程序以及各种端。在这种需求日增的情况下，必须得考虑一种新的方式，优化前端的开发工作，
- 例如，解决代码冗余，项目可维护性，提升版本迭代速度等等一系列的问题。前端工程化的概念也就是在这中情况下被提出了。

什么是前端工程化?

> 解放生产力、提高生产效率。通过制定相应的规范，借助一系列工具和手段解决整个工作流程中的痛点。

特点：

模块化

组件化

规范化

自动化

以工程的角度去理解前端。工程是工程，而不是某项技术。

前端工程化-模块化

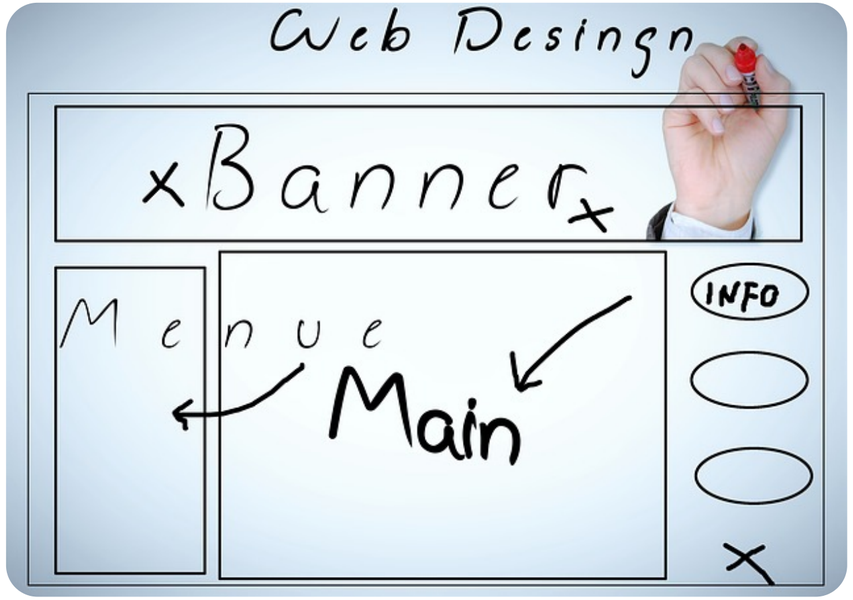
> 模块化是指将一个文件拆分成多个相互依赖的文件，最后进行统一的打包和加载，这样能够很好的保证高效的多人协作。

- JS 模块化：CommonJS、AMD、CMD 以及 ES6 Module。
- CSS 模块化：Sass、Less、Stylus、BEM、CSS Modules 等。其中预处理器和 BEM 都会有的一个问题就是样式覆盖。而 CSS Modules 则是通过 JS 来管理依赖，最大化的结合了 JS 模块化和 CSS 生态，比如 Vue 中的 style scoped。
- 资源模块化：任何资源都能以模块的形式进行加载，目前大部分项目中的文件、CSS、图片等都能直接通过 JS 做统一的依赖关系处理。

前端工程化-组件化

> 不同于模块化，模块化是对文件、对代码和资源拆分，而组件化则是对 UI 层面的拆分。

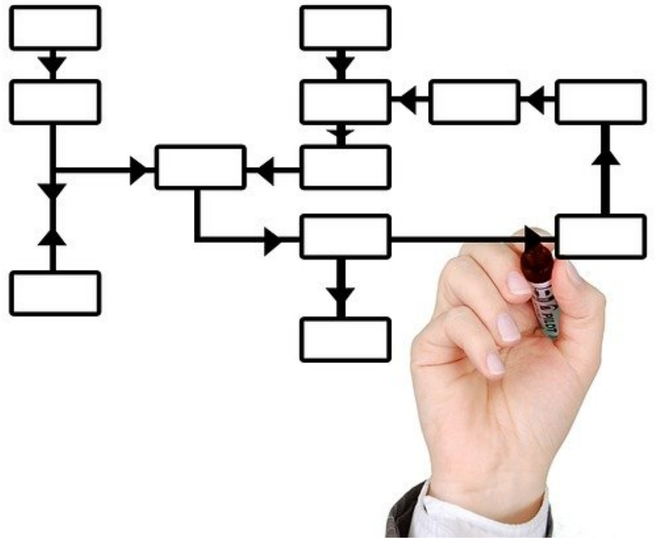
- 通常，我们会需要对页面进行拆分，将其拆分成一个一个的零件，然后分别去实现这一个个零件，最后再进行组装。
- 在我们的实际业务开发中，对于组件的拆分我们需要做不同程度的考量，其中主要包括细粒度和通用性这两块的考虑。
- 对于业务组件，你更多需要考量的是针对你负责业务线的一个适用度，即你设计的业务组件是否成为你当前业务的“通用”组件。



前端工程化-规范化

> 正所谓无规矩不成方圆，为了更好的协作和维护代码。在开发过程中会不断演进各种规范。

- 项目目录结构、命名规范;
- 编码规范：对于编码这块的约束，一般我们都会采用一些强制措施，比如 ESLint、StyleLint 等;
- 联调规范;
- 样式管理规范：目前流行的样式管理有 BEM、Sass、Less、Stylus、CSS Modules 等方式;
- git flow 工作流：其中包含分支命名规范、代码合并规范等;
- code review;



前端工程化-自动化

> 自动化合并、构建、打包能为我们节省很多工作。

- webpack/parcel/ESbuild;
- CI/CD;
- 容器;

本次宣讲的目标

- 了解前端工程化；
- 掌握从0到1「愉快地」开发一个现代化web应用的能力；

如何开始

> Well begun is half done.
>
> 好的开始，是成功的一半。 —— 贺拉斯

技术选型

开始工程

制造组件

生产构建

技术选型

> 如何进行技术选型（VUE / REACT / ANGULAR / ...）？

	Angular	React	Vue
发行时间	2010年	2013年	2014
开发方式	TypeScript	一切都是JavaScript的方式	JavaScript/HTML/TypeScript
模型	Incremental DOM	Virtual DOM	Virtual DOM
学习曲线	陡峭	中等	平滑
性能	一般	高	高
复杂性	高	中	低
灵活性	低	中	高
适合场景	大规模、功能丰富的应用	现代Web渲染SPA应用	中小型Web SPA应用

技术选型 - Incremental DOM

> 每个组件都被编译成一系列指令。这些指令创建 DOM 树并在数据更改时就地更新它们。

编写

```
@Component({
  selector: 'todos-cmp',
  template: `
    <div *ngFor="let t of todos|async">
        {{t.description}}
    </div>
  `
})
class TodosComponent {
  todos: Observable<Todo[]> = this.store.pipe(select('todo
  constructor(private store: Store<AppState>) {}
}
```

编译后

```
var TodosComponent = /** @class */ (function () {
  function TodosComponent(store) {/* store component */}
  TodosComponent.ngComponentDef = defineComponent({
    type: TodosComponent,
    selectors: [["todos-cmp"]],
    factory: function TodosComponent_Factory(t) {
      return new (t || TodosComponent)(directiveInject(Sto
    },consts: 2,vars: 3,
    template: function TodosComponent_Template(rf, ctx) {
      if (rf & 1) { // create dom
        pipe(1, "async");
        template(0, TodosComponent_div_Template_0, 2, 1, n
      } if (rf & 2) { // update dom
        elementProperty(0, "ngForOf", bind(pipeBind1(1, 1,
      }
    },encapsulation: 2
  });
  return TodosComponent;
}());
```

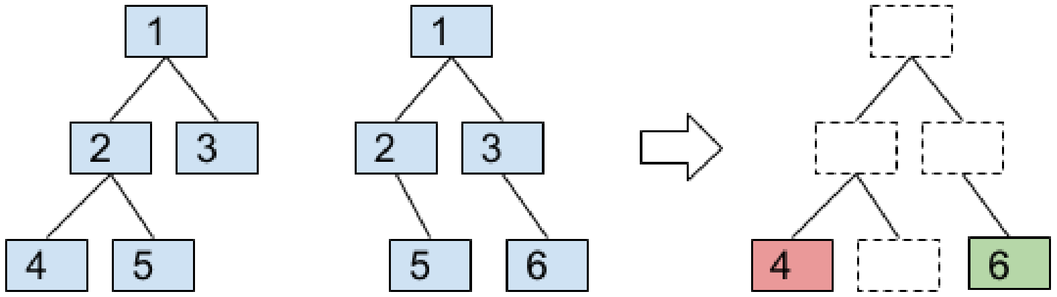
技术选型 - Virtual DOM

> 每个组件每次重新渲染时都会根据需要创建和更新新的虚拟 DOM 树，然后对浏览器 DOM 应用一系列转换以匹配新的虚拟 DOM 树。

ORIGINAL DIV DOM

```
> var div = document.createElement('div')
  var str = ''
  for (var key in div){
    str += key + ' '
  }
< "align title lang translate dir dataset hidden tabIndex accessKey draggable spellcheck contentEditable isContentEditable
  offsetParent offsetTop offsetLeft offsetWidth offsetHeight style innerText outerText webkitdropzone onabort onblur
  oncancel oncanplay oncanplaythrough onchange onclick onclose oncontextmenu oncuechange ondblclick ondrag ondragend
  ondragenter ondragleave ondragover ondragstart ondrop ondurationchange onemptied onended onerror onfocus oninput
  oninvalid onkeydown onkeypress onkeyup onload onloadeddata onloadedmetadata onloadstart onmousedown onmouseenter
  onmouseleave onmousemove onmouseout onmouseover onmouseup onmousewheel onpause onplay onplaying onprogress onratechange
  onreset onresize onscroll onseeked onseeking onselect onshow onstalled onsubmit onsuspend ontimeupdate ontoggle
  onvolumechange onwaiting click focus blur onautocomplete onautocompleteerror namespaceURI prefix localName tagName id
  className classList attributes innerHTML outerHTML shadowRoot scrollTop scrollLeft scrollWidth scrollHeight clientTop
  clientLeft clientWidth clientHeight onbeforecopy onbeforecut onbeforepaste oncopy oncut onpaste onsearch onselectstart
  onwheel onwebkitfullscreenchange onwebkitfullscreenerror previousElementSibling nextElementSibling children
  firstElementChild lastElementChild childElementCount hasAttributes getAttribute getAttributeNS setAttribute
  setAttributeNS removeAttribute removeAttributeNS hasAttribute hasAttributeNS getAttributeNode getAttributeNodeNS
  setAttributeNode setAttributeNodeNS removeAttributeNode closest matches webkitMatchesSelector getElementsByTagName
  getElementsByTagNameNS getElementsByClassName insertAdjacentElement insertAdjacentText insertAdjacentHTML
  createShadowRoot getDestinationInsertionPoints requestPointerLock getClientRects getBoundingClientRect scrollIntoView
  scrollIntoViewIfNeeded animate remove webkitRequestFullScreen webkitRequestFullscreen querySelector querySelectorAll
  ELEMENT_NODE ATTRIBUTE_NODE TEXT_NODE CDATA_SECTION_NODE ENTITY_REFERENCE_NODE ENTITY_NODE PROCESSING_INSTRUCTION_NODE
  COMMENT_NODE DOCUMENT_NODE DOCUMENT_TYPE_NODE DOCUMENT_FRAGMENT_NODE NOTATION_NODE DOCUMENT_POSITION_DISCONNECTED
  DOCUMENT_POSITION_PRECEDING DOCUMENT_POSITION_FOLLOWING DOCUMENT_POSITION_CONTAINS DOCUMENT_POSITION_CONTAINED_BY
  DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC nodeType nodeName baseURI isConnected ownerDocument parentNode parentElement
  childNodes firstChild lastChild previousSibling nextSibling nodeValue textContent hasChildNodes normalize cloneNode
  isEqualNode isSameNode compareDocumentPosition contains lookupPrefix lookupNamespaceURI isDefaultNamespace insertBefore
  appendChild replaceChild removeChild addEventListener removeEventListener dispatchEvent "
```

Virtual DOM DIFF



主要优点:

- 对真实 DOM 进行抽象描述，除了核心属性外可以用来扩展，灵活性极强。甚至可以跨平台，渲染到DOM（web）之外的平台。比如ReactNative，Weex.
- 拥有非常高的运行时性能，可以减少直接操作DOM，指定特定的组件进行重新渲染。避免整体更新DOM带来的高代价计算。

技术选型 - React or Vue ?

> 我们当然选择渐进式JavaScript 框架。 – 子可能曰过

历史因素

从Jquery时代过渡

学习曲线平滑

性能优势

数据驱动

灵活性

规模化向上/向下扩展

运行时性能

真香

未来发展

Github上升最快

专职团队维护

社区活跃性高

未来可期

当然，如果现场有Angular用户，上面的当我没说。😉

初始化项目 - 初始化工程

> Talk is cheap. Show me the code. —— Linus Torvalds

1. 使用脚手架 Vue Cli

```
npm install -g @vue/cli
```

2. 创建一个项目

```
vue create fintech-financeRecon-portal
```

- 当然，前提是需要有nodejs环境，npm包管理工具（国内使用npm缓慢，可以选用国内的「镜像源」）。

初始化项目 - 规范的开始

> convention over configuration 约定优于配置

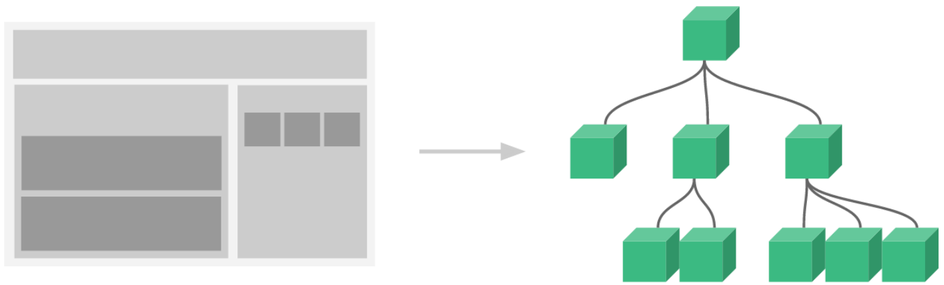
- 规范工程结构

```
├── README.md        //说明文件
├── package.json     // 打包配置文件
├── src
│   ├── App.vue      // 应用入口
│   ├── api          // api 定义
│   ├── assets       // 素材文件
│   ├── components   // 自定义组件
│   │   ├── Charts
│   ├── config      // 应用内部设置
│   │   └── defaultSettings.js
│   ├── main.js     // js入口
│   ├── router.js   // 路由入口
│   ├── store       // 缓存配置
│   ├── utils       // 工具
│   │   ├── axios.js
│   │   └── utils.js
│   └── views       // 页面
│       ├── 404.vue
├── .env.dev       // 开发环境配置
```

制造组件 - 组件的组织

> 不搞组件的前端不是个好前端。

- 组件的组织



例如，你可能会有页头、侧边栏、内容区等组件，每个组件又包含了其它的像导航链接、博文之类的组件。

为了能在模板中使用，这些组件必须先注册以便 Vue 能够识别。可以进行全局注册和局部注册。

- 简单举例：通用fintech后台管理系统结构

```
+---+---------------+
|   | hbTablePage   |
+---+---------------+
| M | search-bar    |
| E | action-bar    |
| N | data-table    |
| U |               |
+---+---------------+
```

制造组件 - 组件的诞生1

> 前端组件化开发，将需要独立某个页面或者模块，以黑盒的形式全部封装到起来，提供对外暴露简便的入口来调用。

一个 table 示例 - 数据绑定

- pros属性配置初始化数据：

```
/* TablePage.vue */
props: {
    dataTable: {
      columns: [],
      rowKey: "",
    },
    dataListFun: {
      type: Function,
    },
  },
```

- 数据加载：

```
/* TablePage.vue */
loadData() {
  this.dataListFun(
    this.pagination.pageNum,
    this.pagination.pageSize,
    this.searchCondition
  ).then((res) => {
    if (res.code == RESULT_CODE.SUCCESS) {
      this.data = res.data.rows;
    }
  });
},
```

注意点：

- 所有的 prop 都使得其父子 prop 之间形成了一个单向下行绑定;每次父级组件发生变更时，子组件中所有的 prop 都将会刷新为最新的值。这意味着你不应该在一个子组件内部改变 prop。

制造组件 - 组件的诞生2

> 前端组件化开发，将需要独立某个页面或者模块，以黑盒的形式全部封装到起来，提供对外暴露简便的入口来调用。

一个 table 示例 - 事件绑定

- 传递绑定的事件：

```
/**
 * TablePage.vue
 * 列表选中一条数据事件
 * this.$emit('myEvent')
 */
onSelectChange(selectedRowKeys, selectedRows) {
  this.$emit('ON_SELECTED_DATA', this.selectedData);
}
```

- 事件监听：

```
/* Page.vue */
<template>
  <div>
    <hbTablePage
      @ON_SELECTED_DATA="onSelectedData"
    >
    </hbTablePage>
  </div>
</template>
```

注意点：

- 在有些情况下，我们可能需要对一个 prop 进行“双向绑定”。不幸的是，真正的双向绑定会带来维护上的问题；
- 这也是为什么我们推荐以 update:myPropName 的模式触发事件取而代之。

制造组件 - 组件的诞生3

> 前端组件化开发，将需要独立某个页面或者模块，以黑盒的形式全部封装到起来，提供对外暴露简便的入口来调用。

一个 table 示例 - 内容插槽

- 定义自由组合的灵活的内容插槽：

```
/* TablePage.vue */
<div v-if="$slots.searchBar">
  <slot name="searchBar"></slot>
  <a-divider dashed style="margin: 12px 0px" />
</div>
<div v-if="$slots.actionBar">
  <slot name="actionBar"></slot>
</div>
```

- 插槽填充：

```
/* Page.vue */
<template>
  <div>
    <hbTablePage>
      <template v-slot:searchBar>
        <hbTableSearchBar />
      </template>
      <template v-slot:actionBar>
        <hbTableActionBar />
      </template>
    </hbTablePage>
  </div>
</template>
```

生产构建 - 按需加载

> 上线问题千千万，打包问题一大半。

- 路由加载

```
/* router.js
* 异步加载*/
{
  path: '/userLogin',
  name: 'userLogin',
  component: resolve => require(['@/views/user/Login'],
  resolve),
}
```

```
/* router.js
* 懒加载*/
{
  path: '/userLogin',
  name: 'userLogin',
  component: =>import(
  /* webpackChunkName: "userLogin" */
  '../views/userLogin.vue')
}
```

- 组件按需引入：

```
/* babel-plugin-component
* .babelrc */
{
  "presets": [["es2015", { "modules": false }]],
  "plugins": [
    ["component",
        {"libraryName": "ant-design-vue",
        "styleLibraryName": "theme-chalk"}]
  ]
}
```

```
/* main.js or page.vue */
import Vue from 'vue';
import { Button, message } from 'ant-design-vue';

Vue.use(Button);
Vue.prototype.$message = message;
```

生产构建 - 压缩再压缩1

> 上线问题千千万，打包问题一大半。

- 图片压缩

```
/* image-webpack-loader
* vue.config.js */
{
  test: /\.(png|jpe?g|gif|svg)(\?.*)?$/,
  use:[{
    loader: 'url-loader',
    options: {
      limit: 10000,
      name: utils.assetsPath('img/[name].[hash:7].[ext]')
      }
    },{
      loader: 'image-webpack-loader',
      options: {
        bypassOnDebug: true,
      }
    }]
}
```

- 提取公共代码:

```
/* CommonsChunkPlugin
* vue.config.js
* package.json 里依赖的包，都会被打包进 vendor.js 文件中 */
new webpack.optimize.CommonsChunkPlugin({
  name: 'vendor',
  minChunks: function(module, count) {
    return (
      module.resource &&
      /\.js$/.test(module.resource) &&
      module.resource.indexOf(
        path.join(__dirname, '../node_modules')
      ) === 0
    );
  }
}),
/* 抽取出代码模块的映射关系 */
new webpack.optimize.CommonsChunkPlugin({
  name: 'manifest',
  chunks: ['vendor']
})
```

生产构建 - 压缩再压缩2

> 上线问题千千万，打包问题一大半。

- 去除 ES6 转为 ES5 的冗余代码

```
/* babel-plugin-transform-runtime
* .babelrc */
"plugins": [
    "transform-runtime"
]
```

- BABEL 插件会在将 ES6 代码转换成 ES5 代码时会注入一些辅助函数，多次引用会造成多次重复生成。
- 通过REQUIRE('BABEL-RUNTIME/HELPERS/CREATECLASS') 的方式导入，做到只生成一次。

- gzip压缩：

```
/* compression-webpack-plugin@5.0.1
* vue.config.js
* 会打包生成.gz格式文件 */
const CompressionPlugin = require("compression-webpack-plu
configureWebpack: {
    plugins: [
      new CompressionPlugin({
        filename: "[path].gz",
        algorithm: "gzip",
        test: /\.js$|\.css$|\.html$/,
        threshold: 10240, // //压缩超过此大小的文件,以字节为单位
        minRatio: 0.8,
        deleteOriginalAssets: false
      })]
  }
```

```
/* nginx.conf
* nginx开启gzip */
gzip  on;  #开启gzip
...
```

做个小结

> 来！该做个小结了！

- 至此，我们完成了最基础的从0到1的应用建造过程；

- 小结一下：

1. 通过vue cli初始化了一个工程；
2. 编写了一个「看起来简单的」组件；
3. 利用各种手段在生产环境中更加「顺畅的」加载；

更多VueJs的细节可以查看官网 https://vuejs.org/.

Where to Go From Here?

> 就这？是不是意犹未尽？
> 技术栈系列接下来可能会讨论更多...

跨端技术

大前端的演进

性能指标

测试与安全

Flutter实战

前端智能化

可视化技术

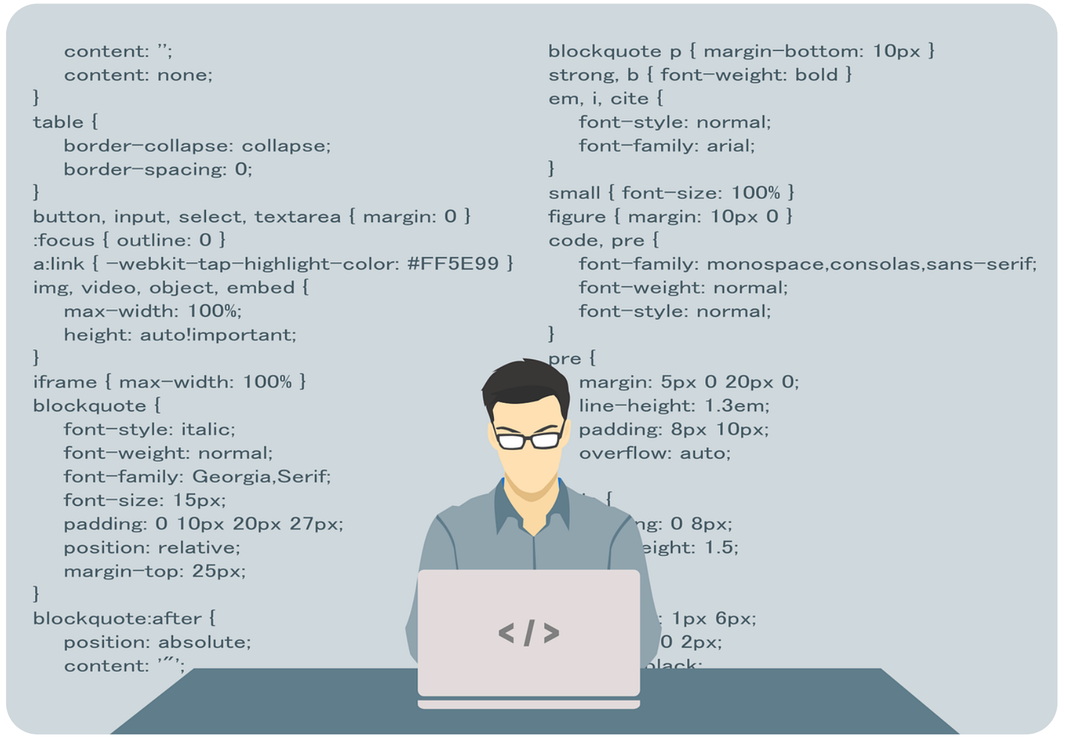
.....

总结与后续思考

- 前端工程化的特点是?
- 现代前端框架(reactjs、vuejs、angularjs)与jquery的明显区别是?
- 单页面（SPA）应用的优缺点?

作业与答疑

- 从0初始化一个基于vuejs的web应用；
- 利用vuejs的特性定制一个日历组件；



THANK YOU !

智造高质量的现代前端应用

蜂泰科技 | 徐健

One More Thing

本次演示所用的代码基于：https://sli.dev