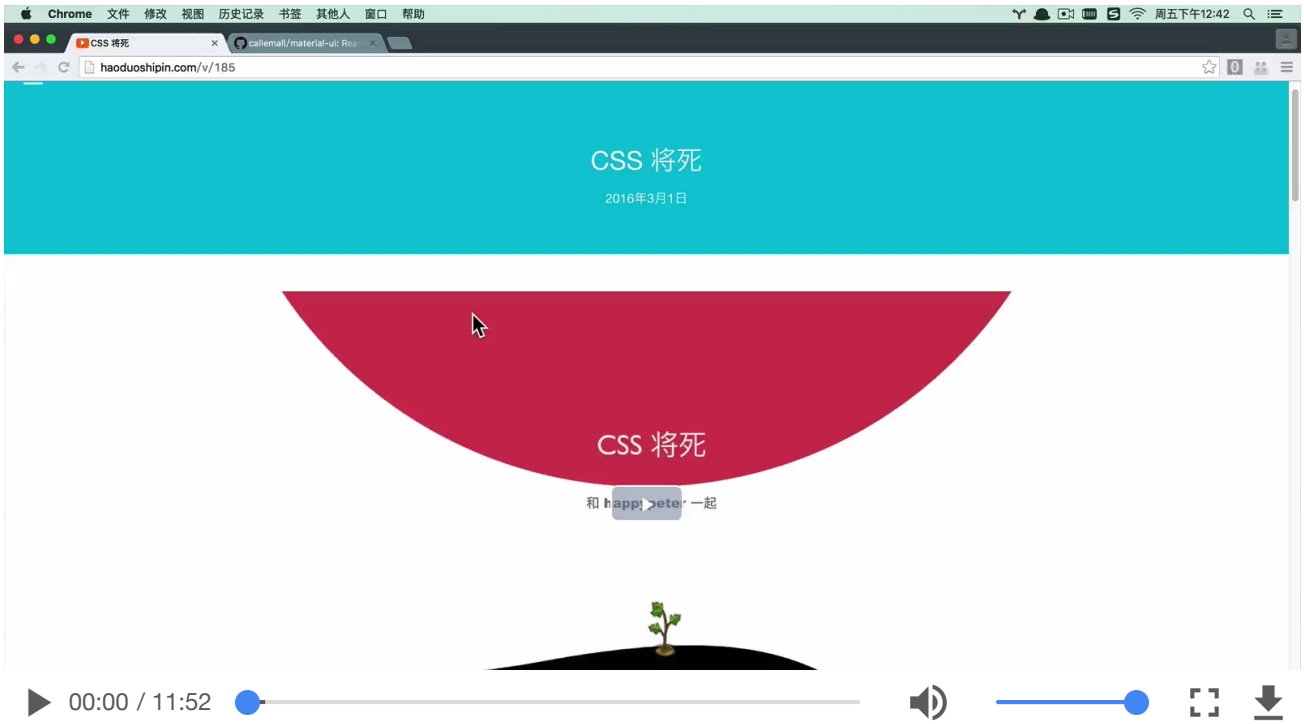

Inline Style 攻略

像 Peter 自己痴迷 React 的组件化思想也不是一天两天，朋友圈里面发了不少相关的文章了。组件化时代，我们需要忘记以前的很多老思路。对于样式这块儿，很早以前我就发布了 CSS 将死。我自己当时的思路就是尽快用 inline style 把 CSS 文件取代掉，所以

这几天继续搜罗资料，来摸索 inline style 的实际可操作的一些技巧。下面是一个阶段性的分享。

inline style 更友好

CSS 的特点是所有选择符默认都是全局变量，也就是说你对一个组件的样式设置是可能会影响到其他页面组件的，而 inline style 的默认思路是我的一个样式设置默认就只是会影响这一个组件而绝对不会影响到其他的组件。静下心来一想，还是后者对新手更友好，不是吗？毕竟把一个项目组织的各个元素样式都井井有条，互不影响是连老手都做不好的事情。

乍一接触，好像是 CSS 更简单，而 inline style 复杂一些。但是实际上也并不是这么回事。比如在 React Inline Styles and the Future of CSS 这样演讲中，作者就提到了，在他们做的沃尔玛的网站大项目中，Specificity Hell（选择符优先级竞争）就很明显，导致开发一团糟。

inline style 有一些具体的技术问题

Inline style 的实质就是放弃层叠，放弃样式表，用 JS 逻辑解决一切样式问题。这个思想听起来很 Cool，但是实际操作就会有一些问题，例如一个 React 组件内部我们甚至都不能用 Media 查询语句，等等这些技术问题。针对这些问题，Radium 来了。

Radium 的各个底层机制有了，更高级的组织还是没有思路，比如如何控制一些全局复用的样式。这个 Peter 目前是通过学习 material-ui 的源码来解决，里面的各种思路确实非常齐全。

Michael Chan 的演讲

Michael Chan 的关于 React 的几个 youtube 演讲都很不错，例如 Styling React Components in JavaScript 。里面他说

> *CSS is the New JS*

基本思路就是放弃那些 CSS 的传统的技巧，用 JS 编程的思路来控制样式。比如 BEM 应该说是非常优秀的写 CSS 的思路了，但是视频中 Chan 用大量的时间展示了一个小 App，首先用 BEM 来写，然后就会发现写得很乱，可见 CSS 还是为 document 而生

的，即使是非常简单的 App ，但是只要有一点点互动操作，有 state 的这个概念进来，BEM 方案就会写得很乱了。然后，大家可以自己到视频中去看，Chan 给出了 inline style 的思路，看了之后我们发现原来的把互动逻辑和 style 分离的那中传统 CSS 的思路是多么傻，inline style ，也就是把 style 直接放到组件中，由 js 来控制样式的状态是多么的自然。

Chan 还有一个隆重推介 Radium 的视频，也很赞的。

一个大大大的收获

使用 inline style 还是要集成原来 CSS 的一些优秀思想的，例如 BEM 。其实 BEM 的一个 block 的概念跟 React 的一个组件概念对应的非常好。

我觉得用 inline style 和组件的概念也很容易把一些通用的传统 style 样式设置的优秀思想很清晰的学起来。来自 November London React User Group- Inline Styles, Music Glue and React Router 1.0 24分钟30秒的地方就有一个非常好的例子。

问题：我的 button 需要根据它父元素的不同而改变自己的样式，怎么做?

以前用 CSS 的时候，我们可以这样定义

```
.button {
  color: red;
}
.sidebar .button {
  font-weight: bold
}
```

很多人都这么做，但是实际上这种做法是非常错误的。但是这种方法在 inline style 这里依然是非常容易实现的，注意，这个还是错误的思路，稍后会给出正确的思路。

```
const button = {
  color: 'red'
};
class Button extends Component {
  render() {
    const style = Object.assign({}, button, this.props.style);
```

```
    return <button style={style}>Click me!</button>;
  }
}
```

上面我给 button 设置了一个基本样式，然后留出一个 style 属性出来，button 的父组件 sidebar 给它传递属性

```
class Sidebar extends Component {
  render() {
    return <Button style={{fontWeight: 'bold'}} />;
  }
}
```

这样一样可以达成 CSS 实现的那种效果。但是思路是错误的，因为 Button 的样式现在是由 Sidebar 组件来决定了，这样就打破了封装原则。

更好的一个解决方案是:

```
const styles = {
  button: {color: 'red'},
  sidebar: {fontWeight: 'bold'}
};

class Button extends Component {
  render() {
    const {type} = this.props;
    const style = Object.assign({}, styles.button, styles[type]);
    return <button style={style}>Click me!</button>;
  }
}
```

这样，Button 组件自己知道自己各种状态下应该长成什么样子。在 Sidebar 中

```
class Sidebar extends Component {
  render() {
    return <Button type="sidebar" />;
```

```
    }
  }
```

总结

inline style 的思路其实目前即使在 React 社区也还没有占据主流。 干掉全局 CSS，The End of Global CSS 这里面聊了使用 Webpack+React 来进行工作的思路，依然还是有 css 文件的，而且这种做事情的方式在 React 社区是非常流行的。但是思路还是跟传统的一切皆全局的默认 CSS 的思路有很大不同，里面融入了组件化开发的很多先进的理念。在 inline style 这里优秀的理念都可以得到很好的开展，同时也可以很有效的避免大家去学习那些错误的过时的 CSS 技巧，所以 Peter 最近在自己的课程中是全力推 Inline style 的。

欢迎添加 Peter 的微信：happypeter1983

冀ICP备15007992号-3