Stoyard 前端代码风格指南

`Version 0.0.1-alpha`

> 本指南旨在方便和规范多人合作的项目，减少沟通和代码重构成本。
> 本指南应是一个供参考的规范和风格建议，而不是硬性的限制。原因一是学习一种风格最重要的是学习其思维方式和设计理念，而不是严守其规范；其二是前端技术迭代速度很快，稳定的意思就是过时（🤷‍♀️所以本规范永远不会有正式版）。因此，风格或者说规范应会随着使用的技术变化，这也是思维更重要的原因。
> 适用范围：HTML、CSS、Vue、小程序

模块化

BEM

即：模块(Block)、元素(Element)、修饰符(Modifier)
是一种 CSS 模块化方法，不同的 CSS 风格有很多（比如原子类），每一种都有其优劣；之所以选择 BEM 是因为其模块化理念，思路如果适当延展，也可帮助 Vue 或者小程序的组件化实现
参考网站(英文)

Block

- 可以独立存在的 (Standalone entity that is meaningful on its own)
- 可以嵌套其他 Block，可与其他 Block 交互
- 例：`header`, `list`, `menu`, `checkbox`, `input`

Element

- 不能独立存在，只能作为 Block 的一部分 (A part of a block that has no standalone meaning and is semantically tied to its block)
- 例：`header title`, `list item`, `menu item`, `checkbox caption`

Modifier

- 用来修饰 Block 或 Element，代表某种状态，以改变其表现或行为 (A flag on a block or element. Use them to change appearance or behavior)
- 例：`disabled`, `highlighted`, `checked`, `fixed`, `size big`, `color yellow`

BEM 命名指南

- 使用：英文字母（不区分大小写），数字和横线 `-`
- 不使用：其他文字，下划线 `_`
- block 与 element 以双下划线 `__`
- element 与 modifier 以双横线连接 `--`

```
block-name__element-name--modifier-value
```

Block

- 建议使用名词命名

```
/* HTML */
<div class="block">...</div>

/* CSS */
.block { ... }
```

Element

- 建议使用名词命名

```
/* HTML */
<div class="block>
  ...
  <span class="block__elem">...</span>
  ...
</div>

/* CSS - Good */
.block__elem { ... }

/* CSS - Bad */
.block .block__elem { ... }
span.block__elem { ... }
```

Modifier

- 建议使用形容词、动词过去分词、动词现在分词

```
/* HTML - Good */
<div class="block block--mod-1 block--mod-2">
  ...
  <span class="block__elem block__elem--mod">...</span>
  ...
</div>

/* HTML - Bad */
<div class="block--mod">
  ...
  <span class="block__elem--mod">...</span>
  ...
</div>

/* CSS */
.block { ... }
.block--mod-1 { ... }
.block--mod-2 { ... }
.block__elem { ... }
.block--mod-1 .block__elem { ... }
.block__elem--mod { ... }
```

优缺点（个人观点）

优：

- 模块、层级结构清晰
- Block 命名空间，不易混淆、污染
- 可复用性好
- 基本只使用单一的 class 选择器，浏览器效率最高

劣：

- 写法相当复杂，类名很长，增加文件体积

解决方案：

- 使用编辑器的代码提示功能，可减少输入，方便输入长类名
- 用预处理器来处理模块化和命名空间问题
  如上面的列子用 Less/Sass 可写成：

```
.block {
  ...
  &.mod-1 { ... }
  &.mod-2 {
    ...
    & .elem { ... }
  }
  & .elem {
    ...
    &.mod { ... }
  }
}

/* 编译后 */
.block { ... }
.block.mod-1 { ... }
.block.mod-2 { ... }
.block.mod-2 .elem { ... }
.block .elem { ... }
.block .elem.mod { ... }
/* 但这样应该会牺牲一定的渲染效率，不过依然比 Vue 里面的 style scoped 效率高很多 */
```

- 另一个方案：
  使用 PostCSS + postcss-bemed (还没有具体测试)
  使用 @block，@element，@modifier 三个自定义修饰符来声明，编译器自动组合类名

```
@block .block {
  ...
  @modifier .mod-1 { ... }
  @modifier .mod-2 {
    ...
    @element .elem { ... }
  }
```

```
  @element .elem {
    ....
    @modifier .mod { ... }
  }
}

/* 编译后 */
.block { ... }
.block--mod-1 { ... }
.block--mod-2 { ... }
.block--mod-2 .block__elem { ... }
.block__elem { ... }
.block__elem--mod { ... }
```

Bem methology in Vue

- 在 vue 中每个 component 应该是一个可以独立复用的部分，对应 block
- 组件名（单文件组件文件名）应为 block 名的 PascalCase 形式

```
/* BlockName.vue */
<template>
  <div class="block-name">...</div>
</template>
```

- Component 内部的 HTML 即为该 block 的 element

```
/* BlockName.vue */
<template>
  <div class="block-name">
    ...
    <span class="block-name__element-name">...</span>
  </div>
</template>
```

- 对于 component 嵌套使用的情况，嵌套的 component 自身既是 block 也是父 component 的 element，HTML中表现应为：

```
/* BlockAlpha.vue */
/* 该组件必须也在其他地方复用，否则不必单独封装成组件 */
<template>
  <div class="block-alpha">...</div>
</template>

/* BlockBeta.vue */
<template>
  <div class="block-beta">
    ...
    <BlockAlpha class="block-beta__elem">...</BlockAlpha>
    ...
  </div>
</template>

/* 渲染后 DOM */
...
  <div class="block-beta">
    ...
    <div class="block-alpha block-beta__elem">...</div>
    ...
  </div>
...
```