escola britânica de artes criativas & tecnologia

Desenvolvedor Full Stack Python

Boas práticas de CSS

Neste módulo vamos conhecer alguns temas avançados de CSS, como a especificidade e metodologias para escrita de código CSS: **BEM e SMACSS.**

Você pode consultar o código escrito durante o módulo clicando aqui.

Especificidade CSS

Escrever código CSS é relativamente simples, porém precisamos ter alguns cuidados e entender alguns conceitos, como a especificidade.

A especificidade no CSS diz o quão específica, forte, um seletor é.

O efeito cascata do CSS pode fazer com que uma regra CSS sobrescreva outra, por exemplo:

```
h1 { color: red; }
h1 { color: blue; }
```

A cor aplicada ao elemento H1 será azul, por essa ser a última regra para este seletor.

Especificidade CSS

Porém podemos contornar esse comportamento deixando o seletor mais específico:

```
h1.titulo { color: red; }
h1 { color: blue; }
```

Com isso o H1 que tiver a classe “titulo” terá a cor vermelha, por essa regra ser mais especifica.

A aplicação de estilos no CSS são do mais especifico para o menos especifico:

1. Estilos inline;
2. Ids;
3. Classes, pseudoclasses e atributos;
4. Elementos e pseudo elementos.

Especificidade CSS

Existe uma regra para calcularmos a especificidade de uma regra CSS.

A especificidade é definida pelas contagens de Ids –classes e afins – elementos e afins.

Por exemplo:

h1 // 001 –possui 0 Ids, 0 classes e 1 elemento
h1.titulo // 011 possui 0 Ids, 1 classe e 1 elemento

A segunda seleção é mais especifica pois 011 é maior que 001.

Especificidade CSS

Existe uma exceção na especificidade, que é o uso da palavra !importante logo após o valor da propriedade, por exemplo:

```
h1 { color: red! importante; }
h1.titulo { color: blue; }
```

Neste exemplo a primeira regra será aplicada, por possuir o !importante.

Porém isso é uma má prática que deve ser evitada e contornada com a escrita de um código CSS escalável e de boa manutenibilidade.

Você pode acessar uma ferramenta que auxilia no cálculo da especificidade de um seletor clicando aqui.

Como o navegador entende o seletor CSS

O navegador lê um seletor CSS da direita para a esquerda, leve em consideração o seletor: nava.

Como a leitura é feita da direita para à esquerda, primeiro o navegador irá selecionar todos os elementos A que existirem na página, após isso ele irá filtrar por aqueles que estiverem dentro do elemento NAV.

Pode existir centenas de links na página e o navegador irá verificar cada um deles para ver se estão dentro do elemento NAV, isso faz com que o nosso seletor não seja performático.

A melhor abordagem seria criar uma classe .nav-link e aplica-la aos elementos que desejamos estilizar. Assim o navegador terá que verificar menos elementos no documento, sendo mais performático.

BEM

BEM –Block Element Modifier– é uma metodologia para escrita de código de estilos.

O BEM propõe separarmos os estilos entre:

Blocos: são containers, grandes ou pequenos, por exemplo: .form, .profile;

Elementos: elementos dentro dos containers, como: .form__input, .profile__avatar;

Modificadores: são estados, que alteram um elemento, por exemplo: .form__input—invalid, .profile__avatar_online.

BEM

O BEM nos sugere separar um bloco de um elemento utilizando o **underline** duas vezes:

BLOCO__ELEMENTO

Já para os modificadores usamos o traço duas vezes:

BLOCO__ELEMENTO--MODIFICADOR

Além disso temos algumas recomendações como:

- Não utilizar o nome das tags para estilização, por exemplo: div.input;
- Não utilizar Ids para estilizar um elemento, por exemplo: #form__input

SMACSS

O SMACSS (Scalable and Modular Architecture for CSS) é uma arquitetura para utilizada para construção de código CSS, nesta arquitetura temos a divisão do CSS nas categorias:

- Base;
- Layout;
- Módulos;
- Estados;
- Temas.

SMACSS - base

Nos arquivos base estilizamos as regras mais básicas da página, é essencialmente um reset, onde podemos remover as margens, espaçamentos e definir outras propriedades que seguirão um padrão na página.

Exemplo de conteúdo de um arquivo base:

```
h1, h2, h3, h4, h5, h6 {
    margin: 0;
}
body {
    font-family: Roboto, sans-serif;
}
```

SMACSS - layout

No SMACSS os componentes são divididos entre componentes maiores e menores, e um layout é um componente maior, que tem a função de conter outros componentes, podemos considera-los como containers.

Exemplo de conteúdo de um arquivo layout:

```
#header {
    padding: 16px;
    display: flex;
}
```

No exemplo anterior utilizamos o ID como seletor do elemento, na documentação do SMACSS vemos o uso de Ids na estilização de layouts, porém não há nenhum problema em se utilizar classes.

SMACSS - módulos

Um módulo é um componente estilizado. Geralmente estão localizados dentro de um layout, o módulo deve ser independente e ter a mesma aparência em qualquer lugar que seja inserido.

Por exemplo, um card de produto, deve possuir o mesmo estilo dentro de uma listagem de produtos ou quando utilizado dentro de uma página de post.

Um módulo pode conter outros módulos, como no caso de um card de produto, podemos ter um módulo de detalhes do produto.

SMACSS - estado

No SMACSS um estado é utilizado para sobre escrever um estilo, por convenção utilizamos o prefixo **"is-"** no nome da classe, por exemplo: um campo de formulário inválido:

```
.is-invalid{
    border-color: red;
}
```

```
<input class="form-input is-invalid" type="text"
    placeholder="Nome completo" />
```

SMACSS - temas

Um tema no SMACSS tem o objetivo de afetar a aparência das páginas, com cores e até mesmo diferentes fontes.

Exemplo:

```
.button{ // modules/button.css
    padding: 8px;
    font-weight: bold;
}

.button{  // themes/private.css
    background-color: #f0932b;
    color: #fff;
}
```

SMACSS - temas

Podemos ver esse caso de uso em sites de bancos tradicionais, onde há seguimentos de clientes, como convencional, private, exclusive, etc.

O site é o mesmo, a estrutura é a mesma, o que muda é o tema, a identidade visual.

Um exemplo prático são os sites da Globo, onde temos apenas a mudança do logo e cor de fundo nos cabeçalhos:

SMACSS - nomenclaturas

O SMACSS nos propõe a utilização de alguns prefixos para nomear as classes:

l-ou layout- para a construção de layouts;

is- para estados, como: is-open, is-close, is-hidden;

Não utilizar prefixos para o módulo, visto que essas classes serão a maioria, ficaria muito verboso escrever module-antes do nome da classe de qualquer módulo.

Links úteis

- Documentação do SMACSS
- Documentação do BEM
- Calculadora de especificidade CSS