# Padrões CSS

Quando você começa a trabalhar com grandes projetos de front-end, os arquivos CSS vão crescendo, e os problemas também. Você começou a usar muitos important! no projeto? Os arquivos CSS estão ficando muito grandes? Você não sabe onde encontrar uma regra CSS específica? Você não é o único a passar por isso, relaxa. A partir desses e vários outros problemas, foram criados padrões CSS.

Eu vou aqui dividir os padrões CSS em dois tipos, os padrões de nomenclatura e padrões de estrutura.

* [Padrões CSS de nomenclatura](http://www.andrefelizardo.com.br/blog/padroes-css/#padroes-nomenclatura)
  + [BEM](http://www.andrefelizardo.com.br/blog/padroes-css/#bem)
    - [Block](http://www.andrefelizardo.com.br/blog/padroes-css/#block)
    - [Element](http://www.andrefelizardo.com.br/blog/padroes-css/#element)
    - [Modifier](http://www.andrefelizardo.com.br/blog/padroes-css/#modifier)
  + [OOCSS](http://www.andrefelizardo.com.br/blog/padroes-css/#oocss)
  + [RSCSS](http://www.andrefelizardo.com.br/blog/padroes-css/#rscss)
* [Padrões CSS de estrutura](http://www.andrefelizardo.com.br/blog/padroes-css/#padroes-estrutura)
  + [SMACSS](http://www.andrefelizardo.com.br/blog/padroes-css/#smacss)
    - [base](http://www.andrefelizardo.com.br/blog/padroes-css/#base)
    - [module](http://www.andrefelizardo.com.br/blog/padroes-css/#module)
    - [state](http://www.andrefelizardo.com.br/blog/padroes-css/#state)
    - [themes](http://www.andrefelizardo.com.br/blog/padroes-css/#themes)
  + [ITCSS](http://www.andrefelizardo.com.br/blog/padroes-css/#itcss)
    - [settings](http://www.andrefelizardo.com.br/blog/padroes-css/#settings)
    - [tools](http://www.andrefelizardo.com.br/blog/padroes-css/#tools)
    - [generic](http://www.andrefelizardo.com.br/blog/padroes-css/#generic)
    - [elements](http://www.andrefelizardo.com.br/blog/padroes-css/#elements)
    - [objetcs](http://www.andrefelizardo.com.br/blog/padroes-css/#objects)
    - [components](http://www.andrefelizardo.com.br/blog/padroes-css/#components)
    - [trumps](http://www.andrefelizardo.com.br/blog/padroes-css/#trumps)

## Padrões CSS de nomenclatura

Os padrões de nomenclatura tem o objetivo de facilitar a hierarquia de CSS, conseguindo evitar o uso do important!, além de permitir a reutilização de classes, diminuindo a quantidade de CSS escrita para cada página.

### BEM

Muito provavelmente o padrão CSS mais conhecido e mais utilizado. Focado em três principais pontos **B**lock (bloco), **E**lement (elemento) e **M**odifier (modificador).

#### Block

É algo significativo e independente no seu HTML e é possível (e provável) ter blocos aninhados (um dentro do outro). Qualquer elemento HTML pode ser um bloco.

<div class="bloco"></div>

#### Element

É um elemento que semanticamente está vinculado a algum bloco e não tem significado sozinho. É separado do nome do bloco com \_\_.

<div class="bloco">  
  
      <p class="bloco\_\_elemento"></p>  
  
</div>

#### Modifier

Modificador, usado para alterar a aparência, comportamento ou estado de um bloco ou de um elemento.

<div class="bloco">  
  
      <p class="bloco\_\_elemento"></p>  
  
      <p class="bloco\_\_elemento--modificador"></p>  
  
</div>

Agora um código real para finalizar:

[Código de exemplo usando o padrão CSS BEM](http://www.andrefelizardo.com.br/blog/wp-content/uploads/2019/09/padroes_css_bem.png)

O padrão BEM apesar de muito famoso é bastante antigo, não possui uma forma de fazer categorias, então eu considero que já existem várias evoluções desse padrão que se adequam melhor aos códigos feitos atualmente. [Veja mais detalhes do BEM](http://getbem.com/introduction/).

### OOCSS

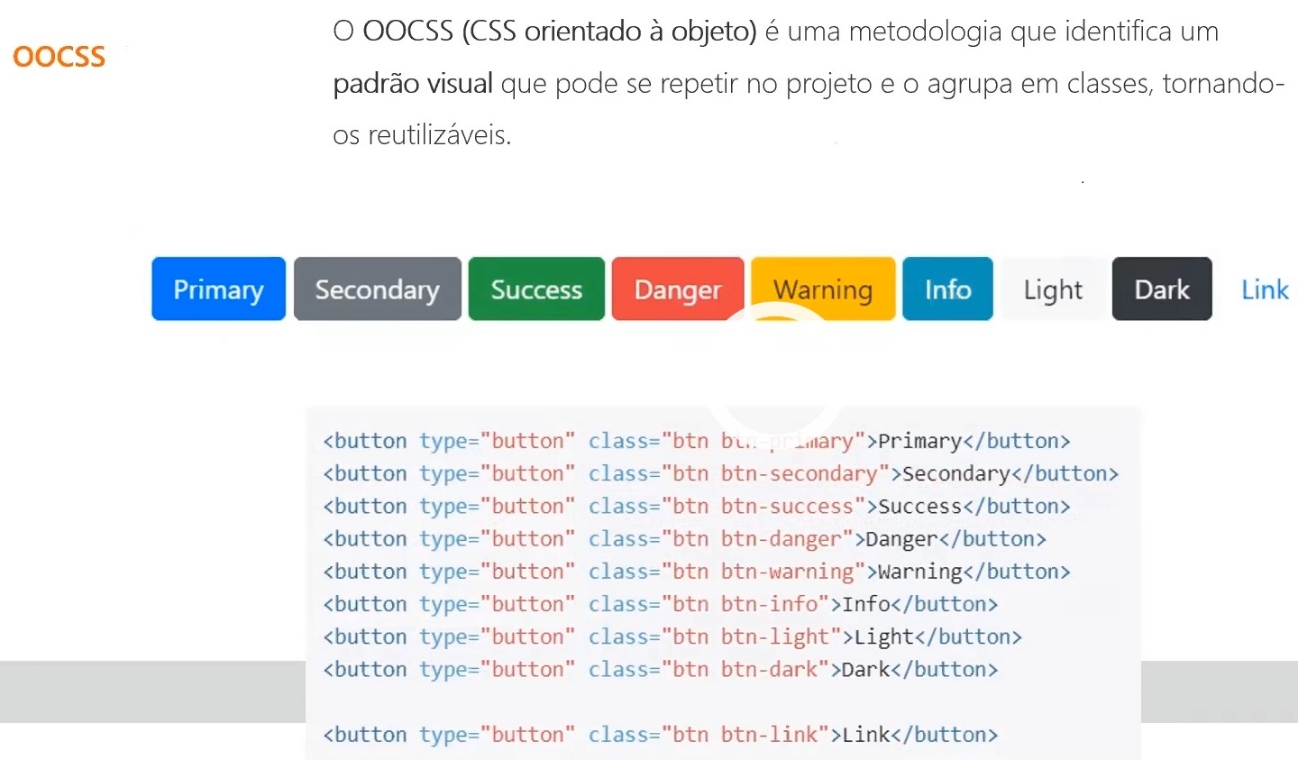
**O**bject **O**riented **CSS** (CSS Orientado a Objeto) é um padrão um pouco mais jovem que o BEM, mas também muito conhecido. O nome não diz muito quais são os objetivos desse padrão, mas eles são separar a estrutura do visual e o container do conteúdo.

A principal prática desse padrão é criar classes estruturais que possam ser reutilizadas em vários locais da página HTML, independente do contexto.

Como características visuais entenda como cores, títulos, bordas, gradientes e outros. Já as características de estrutura são tamanhos, espaçamentos.

OOCSS é um padrão que funciona bem até hoje, pois temos o costume de tentar realmente componentizar nosso código. Então ao invés de repetir um padding em vários e vários elementos, crie uma classe para esse padding. O mesmo para padrões de textos e outros.

Se forçar a pensar em código que pode ser reutilizado no CSS é uma prática excelente, que aconselho, caso você ainda não pratique. [Veja mais sobre OOCSS](http://oocss.org/).



### RSCSS

**R**easonable **S**ystem for **CSS** Stylesheet Structure (Sistema razoável para estrutura de CSS) é um padrão mais atual focado em componentes. Esse é meu padrão atual (a alguns anos) para projetos que demandam grandes CSS. E RSCSS funciona perfeitamente com pré-processador SASS, ou seja, bem atual.

Comece ele olhando para o seu layout e enxergando cada pedaço da tela como um componente. Tudo nele é componente. O nome dos componentes deve ter ao menos duas palavras:

.form-busca, .botao-compartilhar

Os componentes vão ter elementos. Nomeie esses elementos com classes de apenas uma palavra:

.form-busca {

> .campo {...}

> .acao {...}

}

Fazemos as variações com um traço:

.botao-compartilhar {

&.-desabilitado {...}

&.-pequeno {...}

}

Eu realmente gosto de RSCSS e acredito que ele funciona muito bem para os meus códigos. Ainda por cima a documentação é ótima, então dá uma olhada na [documentação](https://silvamateus.gitbooks.io/rscss/content/) e nas várias dicas e detalhes que tem lá além do que falei aqui.

## Padrões CSS de Estrutura

Os padrões CSS de estrutura surgiram com o objetivo de organizar melhor os arquivos CSS em diretórios com objetivos específicos para que seja mais simples saber onde encontrar uma regra CSS, por exemplo. Quando começamos um projeto, muitas vezes não temos total noção do tamanho do projeto, então começar o projeto de uma forma bem organizada é sempre o mais indicado.

### SMACSS

SMACSS é um styleguide que foi criado para funcionar em projetos de tamanhos variados. A separação de diretórios que o SMACSS indica é a seguinte:

#### base

O diretório base é para armazenar os códigos CSS que fazem parte da estrutura do projeto, como header, footer, classes de posicionamento, resets, plugins de layout e outros.

#### module

O diretório module é o diretório onde ficarão a maioria dos arquivos CSS. Neste diretório ficam os componentes como cores, botões, tipografia e etc.

#### state

Em aplicações complexas, o diretório state também é bastante utilizado pois esse diretório é para colocar os CSS que controlam os estados dos componentes como hover, disabled, active entre outros.

#### themes

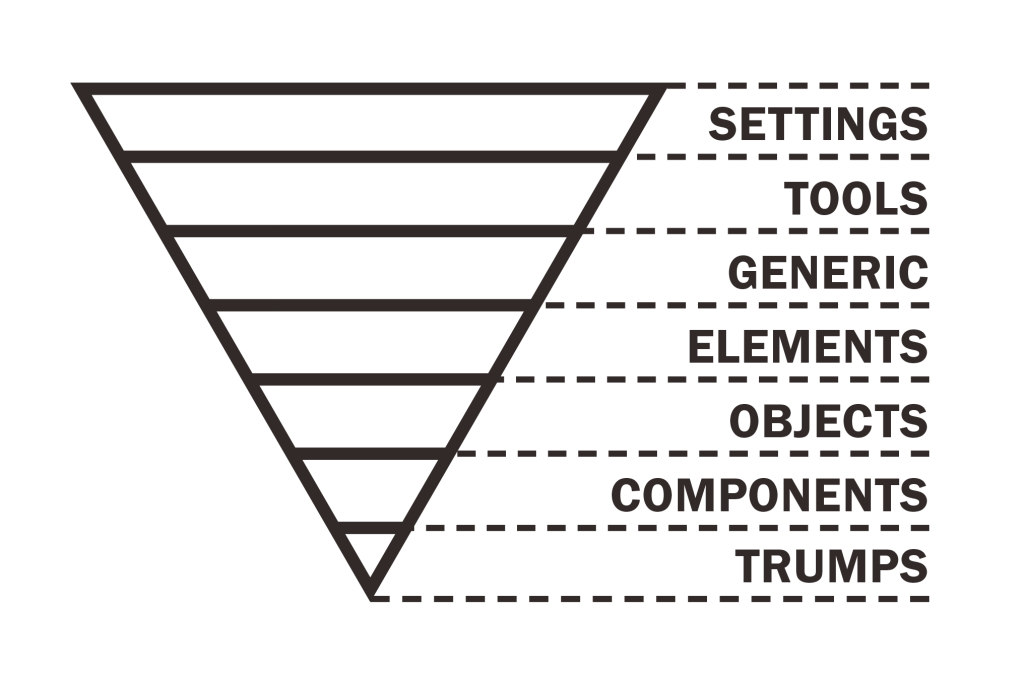
Esse diretório é para criação de temas, como variações de cores, ou até para a criação de variação de alto contraste, por exemplo.

Para que a hierarquia de CSS funcione corretamente a ordem dos diretórios acima tem que ser respeitada.

### ITCSS

**I**nverted **T**riangle for **CSS** é uma outra maneira de organizar os diretórios de arquivos CSS, e particularmente é a minha preferida. Além de ser a minha organização preferida, funciona muito bem com meu padrão de nomenclatura preferido, o RSCSS.

Ele tem esse nome de triângulo invertido pois é formado por várias camadas organizadas da cama mais genérica para a camada mais específica, da base para o topo, formando assim uma representação de triângulo invertido.

[](https://raw.githubusercontent.com/szaranger/szaranger.github.io/master/seanamarasinghe.com/images/itcss/triangle.png)

Um padrão um pouco mais recente também pensado para ser usado com pré-processadores. As duas primeiras camadas: settings e tools são para serem usadas caso o projeto tenha pré-processadores.

#### settings

Settings são as configurações do projeto como cores, fontes, breakpoints e onde ficam as variáveis do projeto referente a estes itens. Um projeto com um design bem definido, começará por essa camada.

#### tools

Mais uma vez citando os pré-processadores, essa camada tem o objetivo de armazenar as funções, mixins e outras funções globais do projeto. Essa camada não costuma ter saída de CSS, ela apenas facilita a reutilização dessas funções, entre outros.

#### generic

Nesta camada são os estilos mais genéricos do projeto, como reset, normalize e classes bem genéricas como classes para border-box e outros.

#### elements

Aqui ficam os estilos para elementos HTMLs crus como h1, p, img etc. Nesta camada é interessante criar um arquivo para cada tipo de elemento, por exemplo: headings.scss, images.scss, links.scss.

#### objects

Objetos são pedaços pequenos do design que se repetem por todo o projeto como button, list etc. Mas a ideia aqui é usar seletores baseados em classes, e não crus, ou seja .buttons, .ui-list. Essa camada faz muito sentido com o conceito de orientação a objetos dos padrões CSS de nomenclatura que falamos acima como OOCSS e RSCSS.

#### components

Essa é a camada onde a maior parte do código CSS ficará. Os componentes específicos do design, serão desenvolvidos nessa camada. Quanto mais abstrato e reutilizável o código, menos a camada de components será utilizada, mas ainda assim ela será muito utilizada no projeto.

#### trumps

Essa camada serve para criarmos classes que tem o objetivo de sobrescrever muitas propriedades do CSS escrito. Como por exemplo a classe .hidden (que particularmente eu uso em todos os projetos) deverá ficar nesta camada. É aceitável que as classes destas camadas utilizem !important, mas use com sabedoria.

Como você pode perceber, são vários os padrões CSS e cada um tem o seu objetivo. Lendo assim, parecem ser complicados de aplicar num projeto. É claro que em um projeto em andamento, já com muitos arquivos, fica difícil aplicar um padrão diferente do que estava sendo feito. Mas, não tem forma melhor de aprender como funcionam e entender qual se adapta melhor aos tipos de projeto que você costuma fazer a não ser realmente colocando a mão na massa.

Fonte: http://www.andrefelizardo.com.br/blog/padroes-css/