Folhas de Estilo em Cascata: Introdução

UFJF - DCC202 - Desenvolvimento Web

Prof. Igor Knop igor.knop@ufjf.br

https://bit.ly/3x4tZh2

Responsabilidades separadas

- HTML
  - Linguagem descritiva
  - Usa marcação de texto (markup)
  - Descreve a estrutura e semântica de um documento
- CSS
  - Linguagem descritiva
  - Possui sintaxe e regras próprias
  - Descreve como desenhar os elementos de um documento
- JavaScript
  - Linguagem imperativa
  - Interpretada no navegador
  - Altera o comportamento padrão dos elementos de um documento

Folhas de Estilo

O navegador recebe o seu documento em uma linguagem de marcação como HTML, XML, ou SVG e tem que realizar uma representação gráfica dele na tela.

As **folhas de estilo** definem como cada elemento será posicionado, colorido, quais fontes e espaçamentos ele terá.

Cada navegador já traz a chamada folha de estilo do agente de usuário que de define uma representação padrão, com tudo o necessário para a página ser legível no maior número de dispositivos possível.

Entretanto, os autores do documento podem e querem alterar esse estilo para melhorar a experiência do usuário final, reforçar a sua marca e criar experiências únicas.

Eles fazem isso criando novas folhas de estilo, que sobrescrevem as regras da folha de estilo do agente de usuário.

Uma página padrão

Toda página apresentada no navegador usa a chamada folha de estilo do agente de usuário.

Ela define todas as regras de posicionamento, desenho e até alguns comportamentos e animações.

Uma página, por mais simples que seja, tem o estilo padrão associado a cada um dos elementos conhecidos e desconhecidos do HTML.

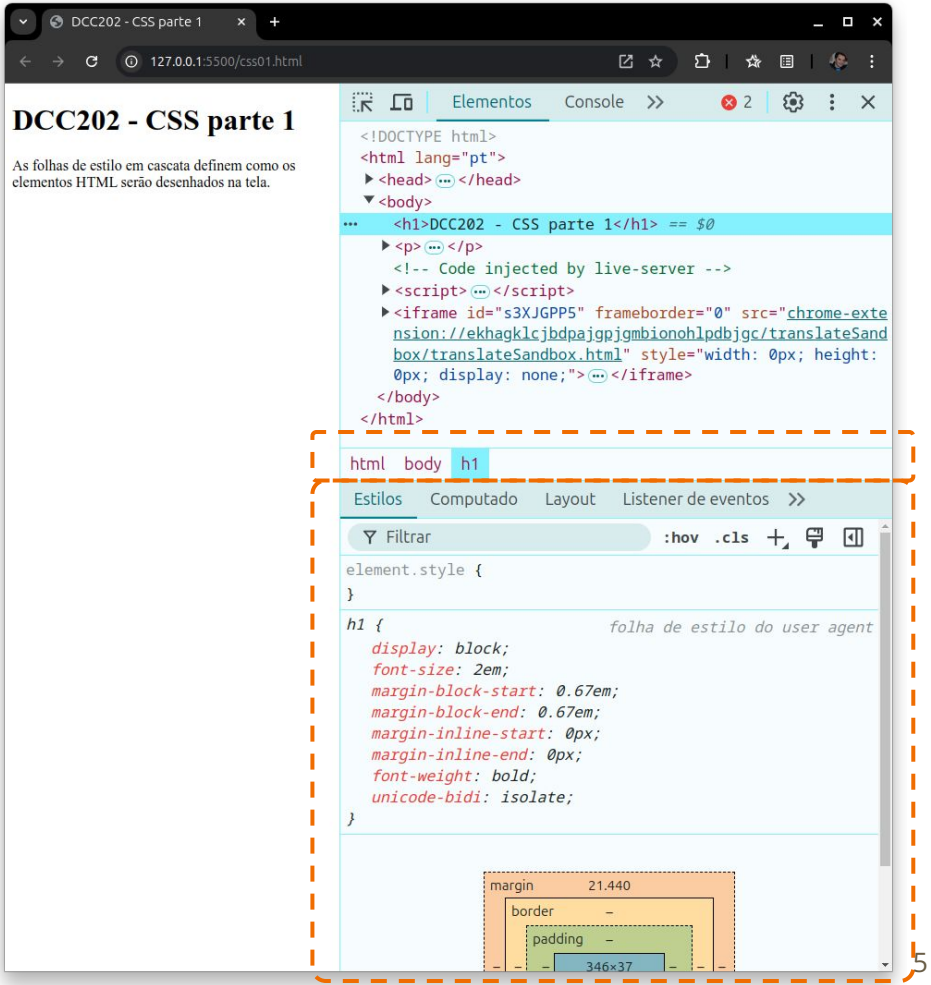
```
<!DOCTYPE html>
<html lang="pt">

<head>
 <meta charset="UTF-8">
 <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
 <title>DCC202 - CSS parte 1</title>
</head>

<body>
 <h1>DCC202 - CSS parte 1</h1>
 <p>As folhas de estilo em cascata definem
como os elementos HTML serão desenhados na
tela.</p>
</body>

</html>
```

Observando os estilos

Nos navegadores modernos, ao clicar com o botão direito sobre um elemento e escolher “Inspecionar”, você abre as Ferramentas de Desenvolvimento.

Essa tela é importante para nós, pois mostra como o navegador interpreta o seu código e também outras duas informações importantes:

1. A hierarquia de elementos na árvore HTML;
2. As folhas de estilo aplicadas sobre o elemento.



Cascade Style Sheets

O Cascade Style Sheets (CSS) é a linguagem padrão utilizada pelos navegadores para a definição das folhas de estilo.

Seus principais conceitos são:

- **Chave de propriedade**
  - Nomes chave que representam uma característica de exibição. Ex.: cor da fonte, altura de linha, estilo de bordas, etc.
- **Valor de propriedade**
  - Qual é o valor associado à chave que tem efeito na exibição. Ex. vermelho, 18 pixels, tracejada, etc.
- **Regras ou agrupamento de propriedades**
  - Agrupa um conjunto de chaves-valor que vão mudar várias propriedades de exibição.
- **Seletores**
  - Especificam quais elementos serão alvo das regras com base na estrutura do documento HTML, por sua posição na árvore ou por atributos e identificadores.

```
/*    Comentário   */
seletor {
  chave1: valor1;
  chave2: valor2;
  chave3: valor3;
}

/*    Exemplo de estilo   */
h1 {
  color: red;
  line-height: 18px;
  border-style: dashed;
}
```

Cascade Style Sheets (2)

A sintaxe do CSS deve ser respeitada, por mais que não gere erros, os estilos podem não funcionar como planejado. As principais a serem observadas são:

- **Comentários**
  - Os comentários seguem o modelo de bloco barra-asterisco e asterisco-barra. Não há um comentário de linha.
- **Dois pontos e ponto e vírgula**
  - Use dois pontos (:) para separar as chaves dos valores e ponto e vírgula (;) após o valor para separar as propriedades alteradas.
- **Regras devem estar entre chaves**
  - É necessário ter o par de chaves para agrupar as propriedades.
- **Seletores múltiplos entre vírgulas**
  - Quando usamos mais de um seletor para o mesmo conjunto de regras, os separamos com vírgulas (,). Vamos ver isso a seguir:

```
/*    Comentário   */
seletor {
  chave1: valor1;
  chave2: valor2;
  chave3: valor3;
}

/*    Exemplo de estilo   */
h1 {
  color: red;
  line-height: 18px;
  border-style: dashed;
}
```

Cascade Style Sheets (3)

Se mais de uma regra seleciona um mesmo elemento, as propriedade de todas as regras são acumuladas no efeito final.

No exemplo ao lado, temos três regras que atingem o elemento **h1**. Então tanto sua cor, seu estilo de borda e sua altura de linha serão alterados.

É possível ter mais de um seletor para uma regra, separadas por vírgulas. No exemplo ao lado a altura de linha tanto do **h1** quanto do **p** são alteradas pela última regra.

```
seletorA {
  chave1: valor1;
}
seletorA {
  chave2: valor2;
}
seletorA, seletorB {
  chave3: valor3;
}


h1 {
  color: red;
}
h1 {
  border-style: dashed;
}
h1, p {
  line-height: 18px;
}
```

Atribuindo CSS ao HTML

Existem três formas de aplicar CSS ao HTML:

1. Em linha (inline) >:-(
2. Embutido (embedded) :-(
3. Em arquivos externos (external) :-)

Sempre depende do contexto, mas para projetos com múltiplas páginas, use arquivos externos. As vantagens imediatas de usar arquivos externos:

- Fácil manutenção
- Divisão de responsabilidades
- Cache

CSS em linha >:-(

O CSS é dito em linha quando vem aplicado diretamente nos elementos através do atributo **style**.

Nele não há a necessidade de seletores, pois é realizado diretamente no elemento.

Tem um peso muito forte quando for resolvida a especificidade.

É útil para correções pontuais, específicas de um documento, mas não escala muito bem quando o número de elementos cresce.

Há uma sobreposição de linguagens HTML e CSS que pode confundir.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport"
content="width=device-width, initial-scale=1.0">
 <title>Exemplo de CSS em linha</title>
</head>

<body>
 <h1>Exemplo de CSS em linha</h1>
 <p>Cuidado ao usar usar CSS em linha, pois:</p>
 <ul>
   <li>Difícil de manter;</li>
   <li>Mistura estilo e estrutura;</li>
   <li>Não é possível usar o <i>Cascade</i> do
CSS.</li>
 </ul>
</body>
</html>
```

CSS em linha >:-(

O CSS é dito em linha quando vem aplicado diretamente nos elementos através do atributo **style**.

Nele não há a necessidade de seletores, pois é realizado diretamente no elemento.

Tem um peso muito forte quando for resolvida a especificidade.

É útil para correções pontuais, específicas de um documento, mas não escala muito bem quando o número de elementos cresce.

Há uma sobreposição de linguagens HTML e CSS que pode confundir.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport"
content="width=device-width, initial-scale=1.0">
 <title>Exemplo de CSS em linha</title>
</head>
<body style="font-family: sans-serif; width:
300px;">
 <h1>Exemplo de CSS em linha</h1>
 <p>Cuidado ao usar usar CSS em linha, pois:</p>
 <ul style="border: 1px solid black;">
   <li style="color: red;">Difícil de
manter;</li>
   <li style="background-color:
lightgrey;">Mistura estilo e estrutura;</li>
   <li>Não é possível usar o <i
style="color:blue;">Cascade</i> do CSS</li>
 </ul>
</body>
</html>
```

CSS embutido :-|

O CSS é dito embutido quando vem em um elemento **style**, em alguma parte do documento.

Se vem na cabeça do documento, se aplica a todo o documento. Se vem em um elemento e é marcado com o atributo **scoped**, ele se aplica apenas ao elemento pai.

Ele faz uso de seletores e define as regras a serem aplicadas aos elementos.

Ele vai seguir a ordem de especificidades, posição e origem para resolver conflitos nas regras.

É útil para envio de documentos únicos, autocontidos. Com os seletores é possível aplicar a mesma regra a vários elementos e permite a fácil alteração.

Não há uma sobreposição de linguagens: a área no HTML onde a linguagem muda para CSS é bem fácil de identificar.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport"
content="width=device-width, initial-scale=1.0">

 <title>Exemplo de CSS embutido</title>
</head>

<body>
 <h1>Exemplo de CSS embutido</h1>
 <p>Cuidado ao usar usar CSS embutido, pois:</p>
 <ul>
   <li>Limitado a um único documento;</li>
   <li>Muito código de estilo no documento, mas
isolado.</li>
 </ul>
</body>

</html>
```

CSS embutido :-|

O CSS é dito embutido quando vem em um elemento **style**, em alguma parte do documento.

Se vem na cabeça do documento, se aplica a todo o documento. Se vem em um elemento e é marcado com o atributo **scoped**, ele se aplica apenas ao elemento pai.

Ele faz uso de seletores e define as regras a serem aplicadas aos elementos.

Ele vai seguir a ordem de especificidades, posição e origem para resolver conflitos nas regras.

É útil para envio de documentos únicos, autocontidos. Com os seletores é possível aplicar a mesma regra a vários elementos e permite a fácil alteração.

Não há uma sobreposição de linguagens: a área no HTML onde a linguagem muda para CSS é bem fácil de identificar.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width,
initial-scale=1.0">
 <title>Exemplo de CSS embutido</title>
 <style>
   body {
     font-family: sans-serif;
     width: 300px;
   }

   ul {
     border: 1px solid black;
   }

   q {
     color: blue;
   }

   li {
     background-color: lightgrey;
   }
 </style>
</head>
<body>
 <h1>Exemplo de CSS embutido</h1>
 <p>Cuidado ao usar usar CSS embutido, pois:</p>
 <ul>
   <li>Limitado a um único documento;</li>
   <li>Muito código de estilo no documento, mas isolado.</li>
 </ul>
</body>
</html>
```

CSS externo :-)

O CSS é dito externo quando uma há uma ligação via elemento **link** com o atributo de relação **`rel="stylesheet"`** em alguma parte do documento.

Ele faz uso de seletores e define as regras a serem aplicadas aos elementos.

Ele vai seguir a ordem de especificidades, posição e origem para resolver conflitos nas regras.

É útil para manter o mesmo estilo entre vários documentos diferentes, permitindo ao navegador realizar cache e economizar recursos.

Isola completamente as linguagens: o documento HTML tem apenas marcação e o arquivo CSS apenas estilo.

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width,
initial-scale=1.0">
 <title>Exemplo de CSS externo</title>
</head>

<body>
 <h1>Exemplo de CSS externo</h1>
 <p>Motivos para usar CSS externo:</p>
 <ul>
   <li>Um mesmo arquivo de estilo serve para vários
documentos;</li>
   <li>Nenhuma mistura de linguagens: o documento fica
apenas em HTML.</li>
 </ul>
</body>

</html>
```

CSS externo :-)

O CSS é dito externo quando uma há uma ligação via elemento **link** com o atributo de relação **`rel="stylesheet"`** em alguma parte do documento.

O atributo **href** deve ter uma URL válida para um arquivo contendo apenas linguagem CSS.

Ele faz uso de seletores e define as regras a serem aplicadas aos elementos.

Ele vai seguir a ordem de especificidades, posição e origem para resolver conflitos nas regras.

É útil para manter o mesmo estilo entre vários documentos diferentes, permitindo ao navegador realizar cache e economizar recursos.

Isola completamente as linguagens: o documento HTML tem apenas marcação e o arquivo CSS apenas estilo.

```html
<!DOCTYPE html>
<html>
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width,
initial-scale=1.0">
 <link rel="stylesheet" href="estilo.css">
 <title>Exemplo de CSS externo</title>
</head>
<body>
 <h1>Exemplo de CSS e
 <p>Motivos para usa
 <ul>
   <li>Um mesmo arqu
documentos;</li>
   <li>Nenhuma mistu
apenas em HTML.</li>
 </ul>
</body>
</html>
```

```css
body {
 font-family: sans-serif;
 width: 300px;
}

ul {
 border: 1px solid black;
}

q {
 color: blue;
}

li {
 background-color: #CCC;
}
```

Seletores: universal e por tipo

Através dos seletores escolhemos quais elementos da página irão receber os estilos.

Combinações complexas podem ser criadas com os seletores básicos, permitindo inserir uma resposta visual aos dados apresentados no documento.

O primeiro que vamos apresentar é o **seletor universal**, o * pega todos os elementos da página. É usado normalmente para normalizar o estilo entre navegadores ou quando o novo estilo é muito radical em mudanças.

O próximo mais simples é o **seletor por tipo** do elemento HTML. Esse vai pegar todos os elementos na página do mesmo tipo. No exemplo, todas as âncoras do documento ficarão com texto vermelho e sem a linha sublinhado. Todas as listas ordenadas e não ordenadas ficarão com o texto verde escuro.

Todos os elementos, universal

```
* {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}
```

Por tipo de elemento

```
a {
 color: red;
 text-decoration: none;
}

ul, ol {
 color: darkgreen;
}
```

Seletores por id e classe

Elementos podem, via atributos em sua etiqueta de abertura, receberem uma identidade única no documento e também, famílias às quais pertence.

Um elemento com o atributo **`id="identificador"`** pode ser selecionado pelo seu identificador usando o **seletor por id #** (hashtag). Espera-se que esse identificador seja único na página, portanto ele é muito específico.

Um elemento que possua o atributo **`class="classeA classeB"`** pode ser selecionado usando o seletor **por classe .** (ponto final). É possível que vários elementos pertençam à mesma família, mesmo com etiquetas diferentes. Um elemento também pode pertencer a várias famílias ao mesmo tempo.

No exemplo, elementos pertencentes à classe insuficiente ficarão com o texto vermelho. Já os que pertencem às classes suficiente e infrequente, terão o texto riscado.

Seleciona o único elemento com atributo **`id="m202265059"`**

```
#m202265059 {
 text-decoration: double;
}
```

Seleciona todos os elementos que pertençam à classes insuficiente, ou suficiente e infrequente ao mesmo tempo

```
.insuficiente {
 color: red;
}

.suficiente.infrequente {
 text-decoration: line-through;
}
```

Seletores por filho e descendência

Os elementos também podem ser encontrados pela sua posição na árvore do documento. Temos dois seletores que vão procurar pela descendência do elemento.

Podemos selecionar todos os elementos que sejam filhos direto de um elemento pai usando o **seletor de filhos >** (maior que). O elemento selecionado é sempre o mais à direita. Esta seleção é mais restrita e a ordem tem que ser respeitada à risca.

Um seletor que relaxa essa condição é **seletor de descendência**: ele espera que um elemento seja descendente de outro, mas não se importando quais e quantos outros elementos estejam no caminho. A descendência é especificada por um espaço em branco entre o elemento desejado (o mais à direita) e seu ancestral.

No segundo exemplo, itens de lista dentro de navs vão ter um espaçamento de 10px, não importando quais listas estejam. E que marcações dentro de um bloco de citação vão ter um fundo colorido, não importando quão profundos eles estejam

Seleciona todos os itens de listas filhos de listas não ordenadas e todos os h1, filhos de divs que sejam filhos de um header.

```
ul > li {
  color: blue;
}
header > div > h1 {
  color: white;
}
```

Seleciona todos os itens de lista que descendem de um nav. Seleciona todos as marcações descendentes de um bloco de citação.

```
nav li {
  padding: 10px;
}
blockquote mark {
  background-color: tan;
}
```

Seletores por irmãos

Os elementos também podem ser encontrados com base em seus irmãos. Temos dois seletores que vão procurar pelo irmão imediato ou não.

Podemos selecionar os elementos que sejam antecedidos por um determinado irmão. O seletor de **irmão adjacente** é representado pelo **+** (mais). O elemento selecionado é sempre o mais à direita e só se o elemento anteriormente adjacente a ele respeitar o seletor anterior. Esta seleção também bem restrita e a ordem tem que ser respeitada à risca.

Um seletor que relaxa essa condição é **seletor irmão ~** (til) e espera que um elemento tenha pelo menos um irmão respeitando o seletor anterior, não se importando com quais e quantos outros elementos estejam no caminho.

Seleciona todos os parágrafos que seguem imediatamente um h1.

```
h1 + p {
 font-variant: small-caps;
}
```

Seleciona todos os itens de lista que possuam pelo menos um item de lista irmão antes deles. Em uma lista normal, pega todos, menos o primeiro.

```
li ~ li {
 text-decoration: underline;
}
```

Seletores por atributos

Os elementos também podem ser selecionados pela presença e valor de seus atributos.

Colocando colchetes [], com um nome de uma chave, você especifica que seleciona um elemento com base na presença de um atributo.

Alternativamente, você pode especificar um valor. Assim, apenas os elementos com o valor exato serão selecionados.

Seleciona itens de lista com o atributo value presente. Seleciona âncoras que tenham a hiper referência configurada.

```
li[value] {
 text-decoration: underline;
}

a[href] {
 color: red;
}
```

Seleciona apenas o item de lista que tenha o atributo value com o valor 5. Altera o fundo da célula da tabela que ocupar duas colunas.

```
li[value="5"] {
 text-decoration: underline;
}

td[colspan="2"] {
 background-color: coral;
}
```

Seletores de atributos (2)

Temos ainda variações para seleção dos atributos, adicionando outros marcadores antes do igual.

- *= contém o fragmento de texto
- ~= contém o valor no conjunto
- |= contém o valor no conjunto começando com
- ^= começa com o texto
- $= termina com o texto

Seleciona elementos com lang pt ou começando com pt-

```
a[lang|="pt"] {
 color: green;
}
```

Seleciona elementos com o valor começa com mailto

```
a[href^="mailto"] {
 color: orange;
}
```

Seleciona elementos que o destino termina com pdf

```
a[href$="pdf"] {
 color: blueviolet;
}
```

Seleciona se a palavra ufjf aparece no meio do valor

```
a[href*="ufjf"] {
 color: crimson;
}
```

Seleciona se ufjf está em uma lista de valores separada por espaços

```
a[data-organizers~="ufjf"] {
 color: crimson;
}
```

Seletores por pseudo classes

As pseudo classes representam informações que estão fora da árvore do documento, como situações específicas ou enumerações.

Elas sempre começam com dois pontos (:) e podem ser combinadas com outros seletores, desde que não sejam outras pseudo classes que representam ideias contraditórias.

As pseudo classes podem ser:

- Dinâmicas com base no cursor/mouse do usuário;
- Controles de formulário;
- Estruturais.

Se o mouse está sobre o link ou o cursor é ativado

```
a:hover{
 background-color: blue;
 color: white;
}
a:active{
 background-color: red;
}
```

Seleciona um a cada três dos itens de lista.

```
li:nth-child(3n){
 color: brown
}
```

Seleciona o segundo parágrafo disponível no elemento pai.

```
p:nth-last-of-type(2){
 color: bisque
}
```

Seletores por pseudo elementos

Os pseudo elementos são abstrações sobre a árvore do documento que representam informações não acessíveis ou não existentes na estrutura do documento.

Eles sempre começam com dois dois pontos (**::**) e podem ser combinadas com outros seletores.

Os pseudo elementos podem, entre outros, permitir estilizar as primeiras linhas, primeiras letras e criar elementos virtuais antes e depois de um elemento existente.

A primeira linha dos parágrafos é convertida em maiúsculas.

```
p::first-line {
 text-transform: uppercase;
}
```

Cria elementos virtuais antes e depois de cada quinto item de lista.

```
li:nth-child(5)::before {
 content: "👉";
}

li:nth-child(5)::after {
 content: "👈";
}
```

Inheritance - Herança

Como os elementos em um documento são dispostos em uma árvore, parte dos estilos são passados para os descendentes como uma herança.

Isso evita ter que definir para cada um dos elementos e melhora a repetição de estilos, que dá um aspecto mais coeso.

Por via de regra, as principais características de herança são:

- Propriedades de texto, como família de fontes, tamanho e cor são passadas para os descendentes;
- Propriedades de caixa e posicionamento não são herdadas. Entretanto, podem haver a impressão que são simplesmente por um elemento estar posicionado dentro do outro.

Você pode controlar a herança usando três valores especiais para as propriedades:

- **inherit** define explicitamente que você quer herdar o valor;
- **initial** define o valor para o padrão do elemento;
- **unset**
  - se comporta como inherit se herdou automaticamente; ou
  - como initial se foi herdado explicitamente.

Cascade - Cascateamento

Teremos várias regras ativas em um mesmo documento e, mais de uma vez, as regras serão conflitantes. A própria redefinição dos estilos de agente de usuário são um conflito de regras sobre o mesmo elemento!

Portanto, existem regras de como as resolver esses conflitos e ver quais são de fato aplicadas nos elementos. A seguinte ordem é utilizada para o cascade, a cascata de folhas de estilo:

1. Importância;
2. Especificidade;
3. Posição no código fonte.

Cascade nível 1 - Importância

A importância que as regras são resolvidas é definida, da menor para a maior, na seguinte ordem:

1. Estilo do agente de usuário (menos importante);
2. Declarações do autor;
3. Declarações do usuário;
4. Declarações importantes do autor;
5. Declarações importantes do usuário (mais importante).

Cascade nível 2: Especificidade

Quando duas regras de mesma importância entram em conflito (frequentemente em nossa própria folha de estilo), é definido um sistema de pontuação de três posições que podemos fazer um equivalente a um sistemas de pontos:

- +1 ponto para cada tipo elemento ou pseudo elemento no seletor;
- +10 pontos para cada classe ou pseudo classe no seletor;
- +100 pontos para cada identificador no seletor.

```
p em {...} = 2
div.aviso p em {...} = 13
#mensagens p em {...} = 102
```

Cascade nível 3: Posição

Se mesmo após aplicar a importância e a especificidade, persistir o empate, olhamos a posição da regra do quem vem por último:

- No mesmo CSS, a última declaração tem preferência;
- No último CSS aplicado.

Para saber mais...

- MOZILLA. **Getting Start with CSS**. 2024. Available on Internet: <https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics>
- WEB.DEV. **Selectors**. 2021. Available on Internet: <https://web.dev/learn/css/selectors>
- WEB.DEV. **The Cascade**. 2021. Available on Internet: <https://web.dev/learn/css/the-cascade>
- WEB.DEV. **Inherit**. 2021. Available on Internet: <https://web.dev/learn/css/inheritance>
- GRIFFTTHS, Patrick. 2024. **CSS Beginner Tutorial**. Available on Internet: <http://htmldog.com/guides/cssbeginner/>