Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (DFE_1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

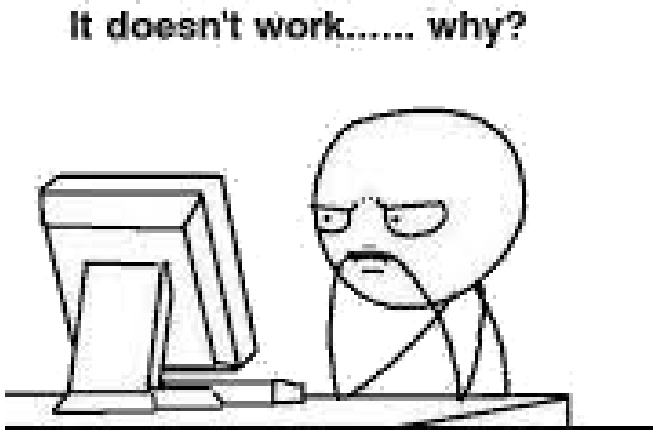
CSS
Especificidade, Herança e Efeito cascata

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas

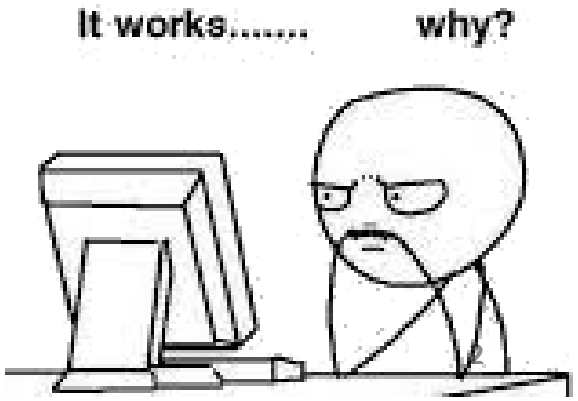
Problema

Não raramente um desenvolvedor enfrenta a situação em que está tentando aplicar um estilo CSS a um elemento, **mas ele parece não funcionar...**

A página simplesmente parece ignorar o código CSS...



Nestas situações, pode-se recorrer ao emprego de `!important` ou mesmo à inserção de estilos *inline* como medidas extremas...



É provável que a dificuldade enfrentada esteja relacionada à precedência no CSS.

Quem controla essa precedência?

Compreender quais estilos de CSS têm precedência sobre quais pode causar **menos frustração**, **código mais limpo** e **mais organizado**.

Para tanto é interessante a compreensão de três coisas que controlam **qual regra CSS se aplica a um determinado elemento HTML**:

1) Cálculo de especificidade;
2) Herança;
3) O efeito cascata;

Cálculo de especificidade

Cálculo de especificidade

Considere um código HTML que inclui um parágrafo ao qual foi atribuída a classe "`biografia`".

```html
<!-- HTML -->
<p class="biografia">
  Lorem ipsum dolor sit
  optio eaque dolores, ipsum
  tenetur maiores.
</p>
```

Existem duas regras CSS aplicáveis:

```css
/* CSS*/
p {
  font-size: 12px;
}

p.biografia {
  font-size: 14px;
}
```

Qual regra é aplicada?

É provável que, neste cenário, o resultado seja de 14px.

A razão para isso é que a segunda regra CSS (`p.biografia`) **possui uma especificidade maior do que a primeira**, visto que ela se inicia com um parágrafo da classe "`biografia`".

Cálculo de especificidade

Contudo, às vezes, a especificidade **não é tão simples de se detectar**.

Considere esses códigos:

```
<!-- HTML -->
<div id="sidebar">
  <p class="biografia">
    Lorem ipsum dolor sit
    optio eaque dolores, ipsum
    tenetur maiores.
  </p>
</div>
```

```
/* CSS */
div p.biografia {
  font-size: 14px;
  color: blue;
}

#sidebar p {
  font-size: 20px;
  color: magenta;
}
```

Em uma análise inicial, a primeira regra CSS parece mais específica...

No entanto, isso não é verdade.

Neste caso, a **segunda regra que determina a estilização** do parágrafo.

Mas, por que isso acontece?

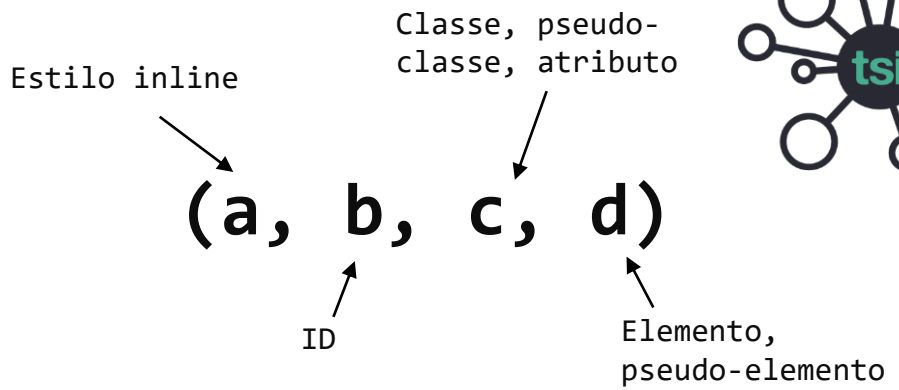
Regras de especificidade

A especificidade é calculada ao se contar diversos componentes do CSS, expressando-os sob a forma **(a, b, c, d)**.

São eles:

▪ Elemento, pseudo-elemento	**d = 1**	representado por (0, 0, 0, 1)
▪ Classe, pseudo-classe, atributo	**c = 1**	representado por (0, 0, 1, 0)
▪ ID	**b = 1**	representado por (0, 1, 0, 0)
▪ Estilo inline	**a = 1**	representado por (1, 0, 0, 0)

A **especificidade** é calculada contando cada uma das opções e **adicionando 1 a cada a, b, c, ou d**.



Exemplos:

p	1 elemento	**(0, 0, 0, 1)**
div	1 elemento	**(0, 0, 0, 1)**
#sidebar	1 id	**(0, 1, 0, 0)**
div#sidebar	1 elemento, 1 id	**(0, 1, 0, 1)**
div#sidebar p	2 elementos, 1 id	**(0, 1, 0, 2)**
div#sidebar p.bio	2 elementos, 1 classe, 1 id	**(0, 1, 1, 2)**

Os valores mais à esquerda têm mais peso.

Para comparar duas especificidades, compara-se valores no milhar, depois na centenas, dezena e assim por diante.

Se um valor em uma posição for maior que o correspondente na outra posição, a regra com o valor mais alto terá precedência.

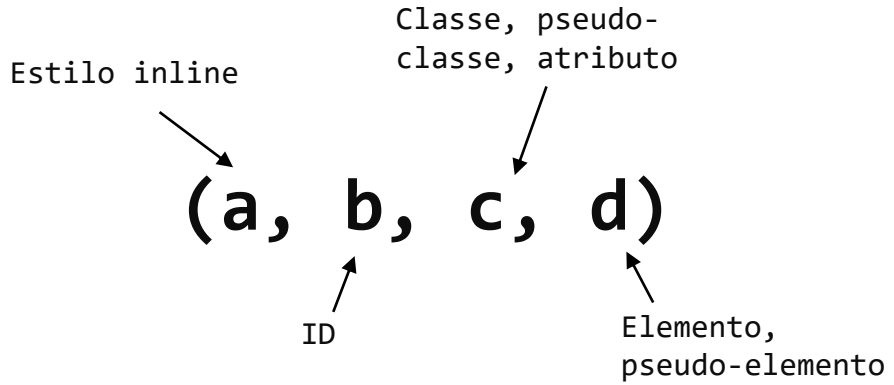
Olhando novamente o código


```css
/* CSS */
div p.biografia {
  font-size: 14px;
  color: blue;
}

#sidebar p {
  font-size: 20px;
  color: magenta;
}
```

(0, 0, 1, 2)

(0, 1, 0, 1)

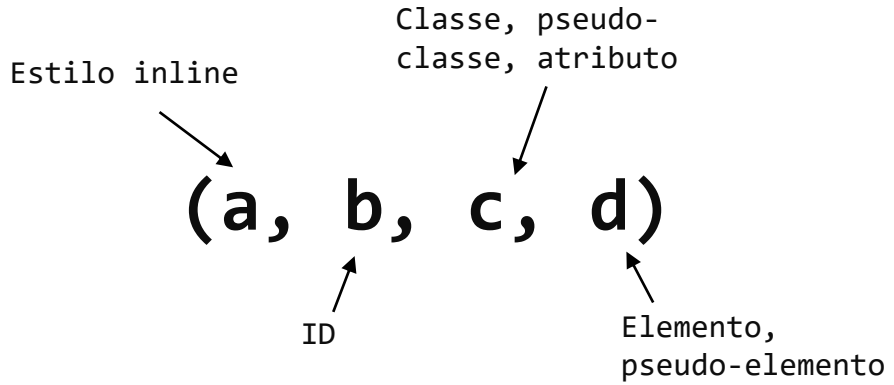


O segundo tem mais especificidade que o primeiro e assim tem precedência.

!important

Um último ponto... **Importância supera especificidade**.

- Ao usar a propriedade CSS com **!important** se substitui as regras de especificidade.




```css
/* CSS */
div p.biografia {
  font-size: 14px !important;
  color: blue;
}

#sidebar p {
  font-size: 20px;
  color: magenta;
}
```

Com o uso do **!important**, a primeira linha da regra CSS tem precedência sobre a segunda.

Vale ressaltar que **!important** é, em grande parte, **uma forma de contornar as regras fundamentais**.

Ou seja, idealmente, não deveria ser necessário **caso se compreenda como as regras funcionam**.

Herança em CSS

Herança

O conceito de herança em CSS é **bastante intuitivo**.

Estilos são transmitidos dos elementos pais aos filhos.

Assim, se a cor vermelha for atribuída à tag body, **todos os textos contidos nos elementos filhos de body também assumirão a cor vermelha**, a menos que exista uma regra específica que determine o contrário.

Contudo, vale salientar que **nem todas as propriedades CSS são passíveis de herança.**

Por exemplo, propriedades relacionadas ao *box model*, não são herdadas.

Assim, definir uma margem ou um preenchimento para uma `div` **não implica** que os parágrafos internos adotarão essas mesmas propriedades.

Eles utilizarão, ao invés disso, os valores padrão estabelecidos pelo navegador.

Herança – propriedades não herdadas

A maioria das propriedades CSS **não é herdada**.

Em especial essa regra inclui as propriedades relacionadas ao ***layout* visual do elemento** (largura, altura, margem, preenchimento e borda), além de outras propriedades que especificam o comportamento ou a aparência específica de um elemento.

`margin`: cada elemento tem sua própria margem, que define o espaço ao redor do elemento. A margem de um elemento pai não é passada para os seus elementos filhos.

`border`: A borda é aplicada individualmente a cada elemento. A definição de uma borda para um elemento pai não aplica automaticamente essa borda aos elementos filhos.

`padding`: O preenchimento é o espaço entre o conteúdo de um elemento e sua borda. Assim como a margem e a borda, o padding definido em um elemento pai não é herdado pelos filhos.

`width` e `height`: As dimensões de um elemento são específicas para esse elemento. Definir a largura e a altura de um elemento pai não afeta diretamente as dimensões dos elementos filhos, **a menos que os filhos usem porcentagens para se basearem nas dimensões do pai indiretamente.**

Propriedades não herdadas

Box Model
- margin
- border
- padding
- width
- height

Background
- background-color
- background-image
- background-position
- background-size

Positioning
- position
- top
- right
- bottom
- left
- z-index

Flexbox e Grid
- flex-basis
- flex-direction
- flex-flow
- flex-grow
- flex-shrink
- flex-wrap
- grid-template-columns
- grid-template-rows
- grid-area
- grid-column
- grid-row

Outras Propriedades Visuais
- display
- overflow
- visibility

No geral, a regra é que propriedades que influenciam diretamente o layout, a estrutura ou o comportamento específico de um elemento são não herdadas, pois a **herança automática dessas propriedades** poderia levar a **resultados indesejáveis e imprevisíveis** na **apresentação da página**.

Herança

A herança é responsável por uma **parte da cascata**.

A vantagem da herança reside na capacidade de **efetuar alterações no design global do site com a modificação de apenas algumas linhas no CSS**.

Isso evita a necessidade de **ajustar as propriedades de cada elemento individualmente**, simplificando o processo de atualização e manutenção do estilo do site como um todo.

A herança é amplamente utilizada em frameworks de CSS ou até mesmo em técnicas para que a customização de layouts seja explorada de forma automática ou manual pelo cliente, como *homes* de portais, *e-commerce* etc.

Assim, o **controle fica mais genérico**, e é aí que entra a **especificidade** do CSS para controlar o detalhe.

Herança explícita

É possível especificar **explicitamente** que uma propriedade deve ser **herdada do elemento pai**.

Exemplos

```
p {
      margin: inherit;
      padding: inherit;
}
```

```
/* Essa declaração faz com que o elemento parágrafo
herde ambas as propriedades do contêiner pai. */
```

```
div {
  color: blue; /* Cor padrão para o texto dentro do div */
}

div p {
  color: inherit; /* Força a herança da cor do texto do elemento pai (div) */
}
```

Efeito Cascata

Efeito cascata

Um dos princípios fundamentais do CSS que controla toda a precedência das regras de estilo e funciona da seguinte forma:

1) Encontrar todas as declarações aplicáveis: O navegador identifica todas as declarações de estilo que se aplicam a um elemento e propriedade específicos.

2) Ordenar por origem e peso: As declarações são classificadas com base em sua origem (estilo do autor, estilo do usuário, padrão do navegador) e peso.

Peso se refere à **importância da declaração**.

- Por exemplo, estilos do autor têm mais peso do que estilos do usuário, que têm mais peso do que estilos padrão do navegador.
- A propriedade `!important` tem a maior importância e sobrepõe todas as outras declarações. tem mais peso que declarações normais.

Efeito cascata

Um dos princípios fundamentais do CSS que controla toda a precedência das regras de estilo e funciona da seguinte forma:

3) Calcular a **especificidade**: Se duas regras têm a mesma origem e peso, a especificidade é calculada para determinar qual regra tem precedência. A regra com a especificidade mais alta é aplicada.

4) Ú**ltima declaração prevalece**: Se duas regras são iguais em todos os aspectos acima, a última declaração no código CSS tem precedência e é aplicada.

Isso significa que estilos embutidos (*inline*) no HTML têm precedência sobre estilos definidos em arquivos externos, independentemente da ordem em que aparecem no HTML.

Curso Superior de Tecnologia em Sistemas para Internet
Desenvolvimento front-end I (DFE_1)
Rafael Cunha Cardoso
rafaelcardoso@ifsul.edu.br

CSS
Especificidade, Herança e Efeito cascata

INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Pelotas