Web

CSS

Catarina Oliveira

DCT DEPARTAMENTO CIÊNCIA E TECNOLOGIA

UPT UNIVERSIDADE PORTUCALENSE

IMP.GE.190.0

CONTENT

1. Context
2. Syntax
3. Integrating CSS into HTML
4. Selectors
    1. By type of element
    2. By identifier (#)
    3. Por class (.)
    4. By attributes
    5. Based on combiners
    6. By pseudo-classes
    7. By pseudo-elements
5. Cascading
6. Inheritance
7. CSS grid
8. Best practices

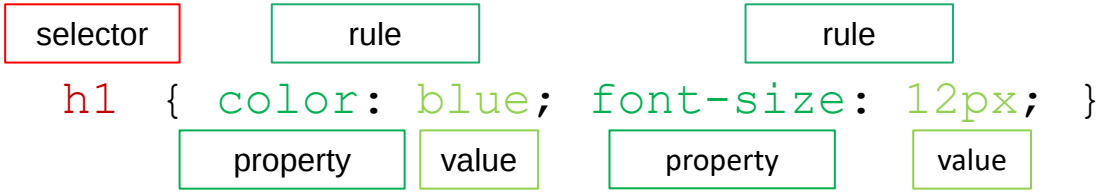
Context

CSS: *Cascading Style Sheets*

- Styling language
- Created and maintained by the W3C consortium
- Current version: 4 (we will be looking at version 3: https://www.w3schools.com/css/)
- Used to add styling to a web page content
- Styles in a separate file
    - More readability, easier to maintain, allows code modularisation
- Advantages
    - Saves time: code can be reused in other web pages
    - Fast loading: does not need HTML attributes for each *tag*
    - Easier to maintain: the web page style can be altered by changing only the CSS file
    - Platform independent: CSS is a standard and supports all browsers

Syntaxe

The styles are defined as a set of rules



Components

- **Selector**: selects the HTML elements to style
- **Set of rules**:
  - one or more style rules surrounded by curly braces ({ and }).
  - Each rule has a `property: value` pair and ends with a semicolon (`;`).
  - Comments are made with `/*` and `*/`

Integrating CSS into HTML

- **Local**

```
<h1 style="color:blue;margin-left:30px;">I am the header</h1>
```

- **Internal**

```
<html>
  <head>
    <style>
      body {
        background-color: red;
      }
      …
    </style>
  </head>
  <body>
    <h1> I am the header </h1>
  </body>
</html>
```

- **External**

```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Selector by type of element

C:\css\index1.html - Notepad++

Ficheiro Editar Procurar Visualização Codificação Linguagem
Configuração Tools Macro Executar Plugins Janela ?

index1.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo1.css">
    </head>

    <body>
        <h1>Cabeçalho 1</h1>
        <h2>Cabeçalho 2</h2>
        <h3>Cabeçalho 3</h3>
        <h4>Cabeçalho 4</h4>
        <h5>Cabeçalho 5</h5>
        <h6>Cabeçalho 6</h6>
        <p>Parágrafo</p>
    </body>
</html>
```

Ln : 20 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS

estilo1.css

```css
h1 {
    text-align: center;
    color: red;
}
```

Página de teste

file:///C:/css/index1.html

Cabeçalho 1

Cabeçalho 2

Cabeçalho 3

Cabeçalho 4

Cabeçalho 5

Cabeçalho 6

Parágrafo

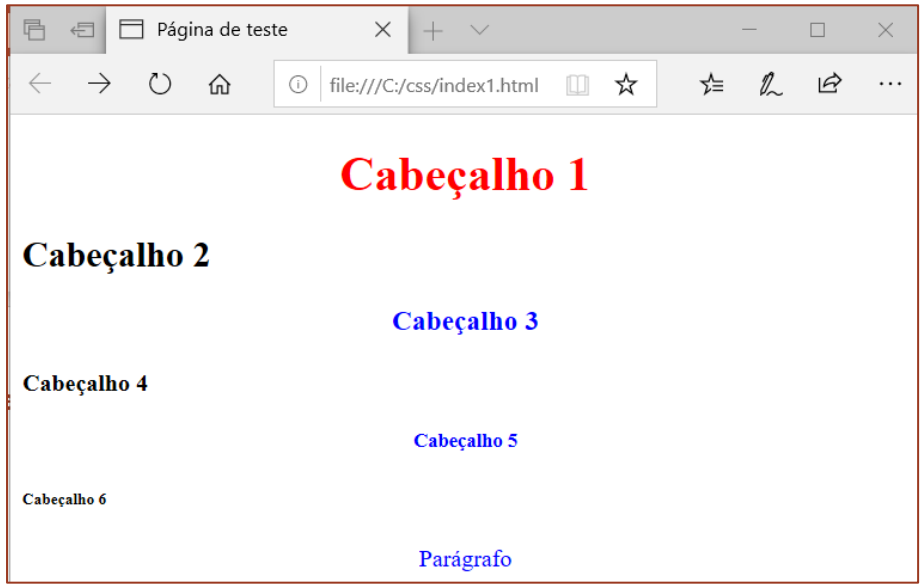
Selector by several types of elements

index1.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo1.css">
    </head>

    <body>

</html>
```

estilo1.css

```css
h1 {

h3, h5, p {
    text-align: center;
    color: blue;
}
```

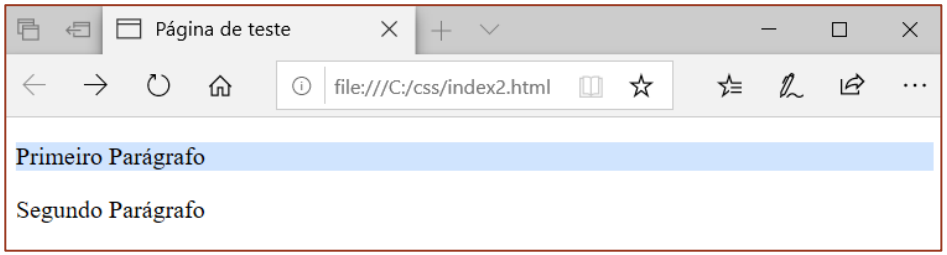


Selector by identifier (#)

Selects HTML elements based on their `id`

index2.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo2.css">
    </head>

    <body>
        <p id="p1">Primeiro Parágrafo</p>
        <p id="p2">Segundo Parágrafo</p>
    </body>

</html>
```



estilo2.css

```css
#p1 {
    background-color: #d0e4fe;
}
```

Choose hexadecimal colors:
https://www.w3schools.com/colors/colors_picker.asp

before the `id`

Selectors by class (.)

Selects HTML elements based on their `class`

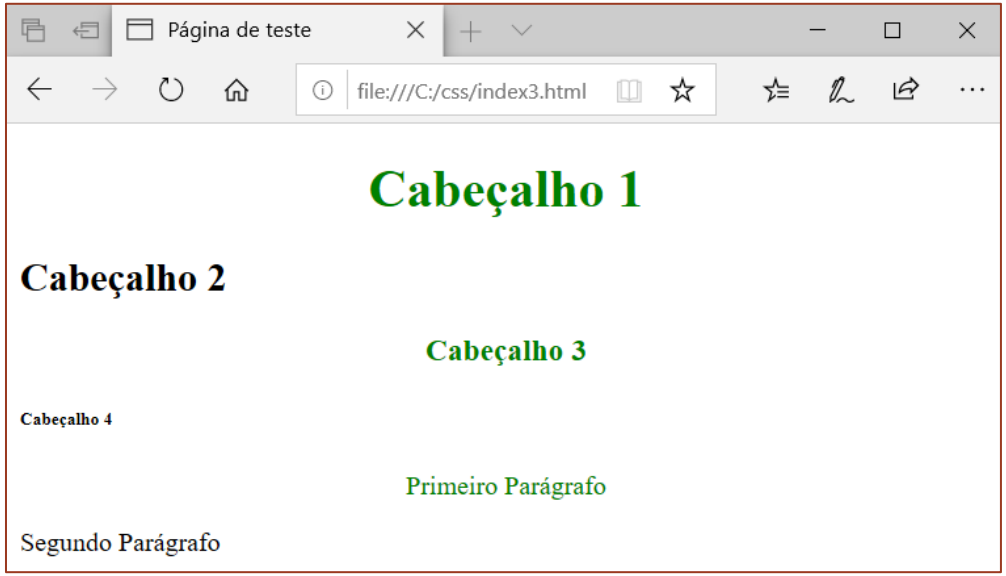
index3.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo3.css">
    </head>

    <body>
        <h1 class="centro">Cabeçalho 1</h1>
        <h2 class="esquerda">Cabeçalho 2</h2>
        <h3 class="centro">Cabeçalho 3</h3>
        <h6 class="esquerda">Cabeçalho 4</h6>
        <p class="centro">Primeiro Parágrafo</p>
        <p class="esquerda">Segundo Parágrafo</p>
    </body>
</html>
```

estilo3.css

```css
.centro {
    text-align: center;
    color: green;
}
```

. Before the `class`



Selectors by class (.)

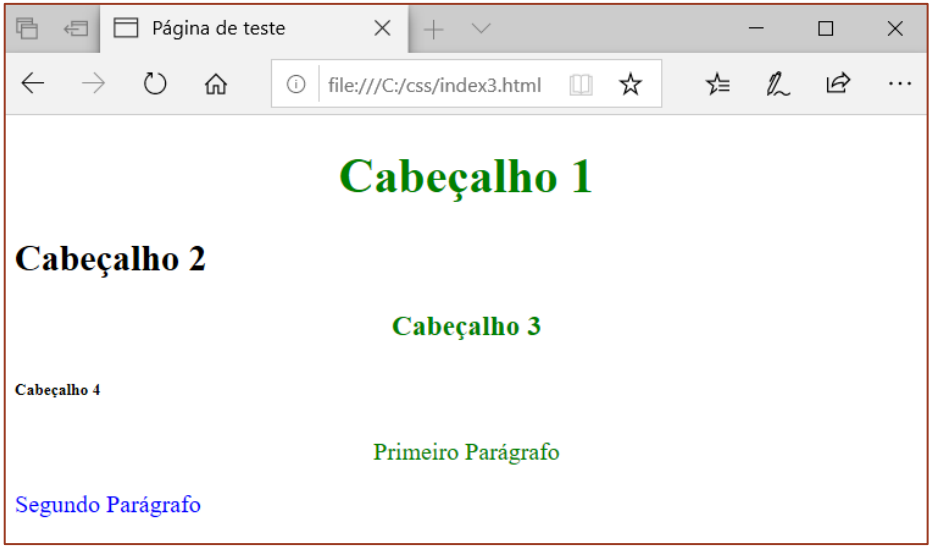
Selects specific HTML elements based on their `class`

index3.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo3.css">
    </head>

    <body>
        <h1 class="centro">Cabeçalho 1</h1>
        <h2 class="esquerda">Cabeçalho 2</h2>
        <h3 class="centro">Cabeçalho 3</h3>
        <h6 class="esquerda">Cabeçalho 4</h6>
        <p class="centro">Primeiro Parágrafo</p>
        <p class="esquerda">Segundo Parágrafo</p>
    </body>
</html>
```

estilo3.css

```css
.centro {

p.esquerda {
    color: blue;
}
```



- Between the `tag` and the `class`

Selectors by attributes

- Selects HTML elements based on two criteria:

index4.html

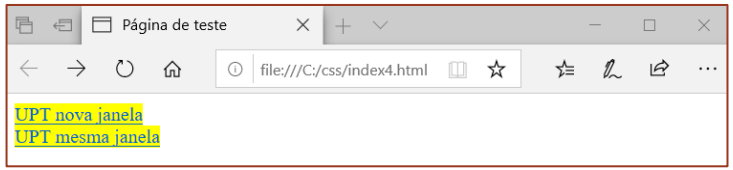
```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo4.css">
    </head>

    <body>
        <a href="http://www.upt.pt" target="_blank">UPT nova janela</a><br/>
        <a href="http://www.upt.pt" target="_self">UPT mesma janela</a>
    </body>
</html>
```

Selector	Description
`[attribute~="value"]`	Attribute value contains “value” (full word)
`[attribute|="value"]`	Attribute value starts with “value” (full word)
`[attribute^="value"]`	Attribute value starts with “value”
`[attribute$="value"]`	Attribute value ends with “value”
`[attribute*="value"]`	Attribute value contains “value”

- **Specific attributes**: `selector[attribute]` to select elements with a specific attribute.

estilo4.css

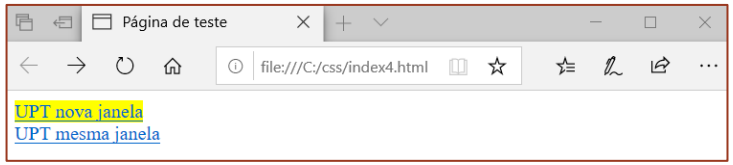
```css
a[target] {
    background-color: yellow;
}
```



- **Attribute values**: `selector[attribute=value]` select elements with specific attributes and values

estilo4.css

```css
a[target="_blank"] {
    background-color: yellow;
}
```



Selectors based on combiners

- We can combine several selectors

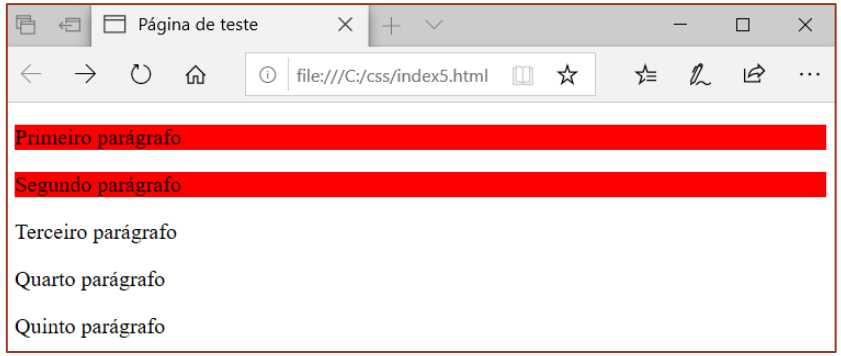
index5.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo5.css">
    </head>

    <body>
        <div>
            <p>Primeiro parágrafo</p>
            <p>Segundo parágrafo</p>
        </div>
        <span>
            <p>Terceiro parágrafo</p>
            <p>Quarto parágrafo</p>
        </span>
        <p>Quinto parágrafo</p>
    </body>
</html>
```

estilo5.css

```css
div > p {
    background-color: red;
}
```

Selectors	Description
Children ( )	All the element's children and their children
Right child (>)	The elements children
Right sibling (+)	The siblings after the element
Sibling (~)	All the siblings



Selectors by pseudo-classes

- **Pseudo-classes** define element status (e.g.: mouse-over, visited/not visited links, elements based on their location)
- Syntax:

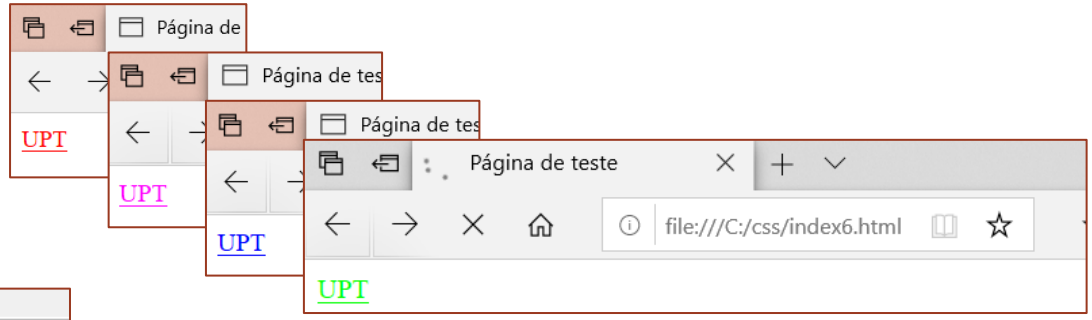
```
selector: pseudoclass{
    proprerty: value;
}
```

index6.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo6.css">
    </head>

    <body>
        <a href="http://www.upt.pt">UPT</a>
    </body>
</html>
```

estilo6.css

```css
/*link não visitado*/
a:link {
    color: #FF0000;
}

/*link visitado*/
a:visited {
    color: #00FF00;
}

/*mouse over link*/
a:hover {
    color: #FF00FF;
}

/*link selecionado*/
a:active {
    color: #0000FF;
}
```



Selectors by pseudo-elements

- Pseudo-elements: used to style parts of an element (e.g: element's first row or character, insert content before/after the element)
- Syntax:

```
selector:pseudoelement{
        proprerty: value;
}
```

index7.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo7.css">
    </head>

    <body>
        <h1>Cabeçalho</h1>
        <p>Parágrafo</p>
    </body>
</html>
```

estilo7.css

```css
p::first-letter {
    color: #ff0000;
    font-size: xx-large;
}

h1::before {
    content: url(smile.png);
}
```

Cascading

When more than one rule is defined for the same element, there are 3 “tiebreaker” methods:

- **Specificity**: the most specific “wins”

index8.html

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            body p {
                color: blue;
            }
            p {
                color: red;
            }
        </style>
    </head>
    <body>
        <p>Frase</p>
    </body>
</html>
```

C:\css\index8.html

file:///C:/css/index8.html

Frase

- **Order**: the last one “wins”

index9.html

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color: blue;
            }
            p {
                color: red;
            }
        </style>
    </head>
    <body>
        <p>Frase</p>
    </body>
</html>
```

C:\css\index9.html

file:///C:/css/index9.html

Frase

- The one with `!important` “wins”

index10.html

```
<!DOCTYPE html>
<html>
    <head>
        <style>
            p {
                color: blue !important;
            }
            p {
                color: red;
            }
        </style>
    </head>
    <body>
        <p>Frase</p>
    </body>
</html>
```

C:\css\index10.html

file:///C:/css/index10.html

Frase

Inheritance

- HTML has an hierarchy
  - An HTML document can be represented by its document tree (**DOM**: *Document Object Model*)
    - Specifies relationshis (parents, children, siblings, ancestors, descendants, …)
- Heritage allows elements to inherit properties
  - Less need to write code and load heavy documents
- Not all properties can be inherited (e.g: `background`, `border`, `width`)
- To determine inheritance:
  - inherit: defines that the value must be inherited
  - initial: defines that the element must assume what is defined to its predecessor as pre-defined by the browser. If not defined, it inherits its predecessor style
  - unset: reset the property to its natural value (inherits if it is naturally inherited)

index11.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo11.css">
    </head>

    <body>
        <div class="container">
            <p>Prágrafo 1</p>
            <p class="inherit">Parágrafo 2</p>
            <p class="initial">Prágrafo 3</p>
            <p class="unset">Parágrafo 4</p>
        </div>
    </body>
</html>
```

estilo11.css

```css
.container {
    color: red;
    border: 10px solid blue;
}
.inherit {color: inherit; border: inherit;}
.initial {color: initial; border: initial;}
.unset {color:unset: border: unset}
```

CSS grid

- A grid *layout* must have a parent element with the **`display`** property defined as **`grid`** or **`inline-grid`**.
- That element's direct children automatically become grid elements

index12.html

```html
<!DOCTYPE html>
<html>
    <head>
        <meta  charset="utf-8">
        <title>Página de teste</title>
        <link rel="stylesheet" href="estilo12.css">
    </head>

    <body>
        <h1>Elementos Grid</h1>
        <div class="grid-container">
            <div class="grid-item">1</div>
            <div class="grid-item">2</div>
            <div class="grid-item">3</div>
            <div class="grid-item">4</div>
            <div class="grid-item">5</div>
            <div class="grid-item">6</div>
            <div class="grid-item">7</div>
            <div class="grid-item">8</div>
            <div class="grid-item">9</div>
        </div>
    </body>
</html>
```

estilo12.css

```css
.grid-container {
  display: grid;
  grid-template-columns: auto auto auto;
  background-color: #2196F3;
  padding: 10px;
}
.grid-item {
  background-color: rgba(255, 255, 255, 0.8);
  border: 1px solid rgba(0, 0, 0, 0.8);
  padding: 20px;
  font-size: 30px;
  text-align: center;
}
```

Best practices

- Use CSS naming methodologies. Examples:
  - **OOCSS**: *Object Oriented CSS* http://oocss.org/
  - **BEM**: *Block, Element, Modifier* http://getbem.com/introduction/
  - **ACSS**: *Atomic CSS* http://github.com/nemophrost/atomic-css
  - **SMACSS**: *Scalable and Modular Architecture for CSS* http://smacss.com/
- Use external rules to:
  - Be able to reuse codebetween documents and keep consistency in websites with more that one page
  - Isolate the web page structure (HTML) from its styling (CSS)
    - More readability, easier to maintain, allows code modularisation
- Understand *cascading* and rule precedence
- Use inheritance to reduce the number of rules needed

UNIVERSIDADE
PORTUCALENSE

Do conhecimento à prática.