

CSS

Treinamento: E21-2023 React
Instrutor: Ivan J. Borchardt
©2023

CSS Cascading Style Sheets

- ❖ Documentos CSS são empregados para estilizar o conteúdo de um documento HTML, ou seja, controlar a apresentação de documentos de marcação.
- ❖ O HTML controla a estrutura de uma página, ao passo que o CSS determina sua aparência.
- ❖ Uma folha de estilos é uma coleção de regras de formatação aplicadas a um documento HTML.

CSS – Principais Funções

1. Estilização de elementos HTML

- ◊ Definir cores de texto, fundos e bordas
- ◊ Controlar o tamanho da fonte, a família da fonte e o espaçamento entre linhas
- ◊ Estilizar links e botões
- ◊ Personalizar a aparência de listas e tabelas

CSS – Principais Funções

2. Controle de layout

- ◊ Posicionar e alinhar elementos na página
- ◊ Controlar o dimensionamento e a proporção dos elementos
- ◊ Criar layouts responsivos para diferentes dispositivos e tamanhos de tela

CSS – Principais Funções

3. Animações e transições

- ◊ Criar animações suaves e efeitos de transição para elementos
- ◊ Definir animações baseadas em eventos, como :hover e :active

CSS – Principais Funções

4. Pseudoclasses e pseudoelementos

- ◊ Estilizar elementos com base em estados específicos, como :hover, :focus e :active
- ◊ Adicionar conteúdo ou estilização adicional usando ::before e ::after

CSS – Principais Funções

5. Estilos de formulários

- ◊ Personalizar a aparência de elementos de formulários, como caixas de texto, botões e caixas de seleção
- ◊ Estilizar campos obrigatórios, campos inválidos, entre outros estados

CSS – Principais Funções

6. Transformações e efeitos 2D/3D

- ◊ Aplicar transformações, como rotação, escala e translação em elementos
- ◊ Adicionar efeitos 2D/3D, como sombras e reflexos

CSS – Principais Funções

7. Controle de impressão

- ◊ Definir estilos específicos para a impressão de páginas, garantindo uma formatação adequada para materiais impressos

CSS – Como e onde adicionar

- ❖ CSS Inline – estilos aplicados diretamente em um elemento HTML, dentro do documento.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

CSS – Como e onde adicionar

- ❖ CSS Interno/Incorporado – O elemento style permite aplicar várias regras de uma vez no próprio documento HTML

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

CSS – Como e onde adicionar

- ❖ CSS Externo: um documento com extensão .css permite aplicar regras a um website inteiro de uma vez.

mystyle.css

```
body {  
    background-color: lightblue;  
}  
  
h1 {  
    color: navy;  
    margin-left: 20px;  
}
```

```
<!DOCTYPE html>  
<html>  
<head>  
<link rel="stylesheet" href="mystyle.css">  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>
```

CSS – Ordem em Cascata

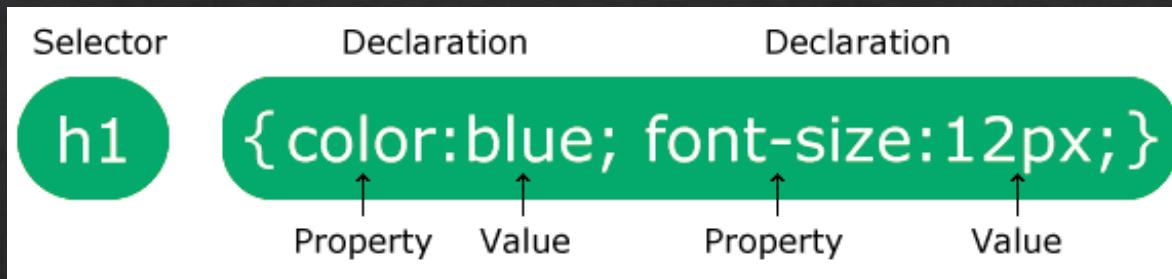
Qual estilo será usado quando houver mais de um estilo especificado para um elemento HTML?

Todos os estilos em uma página serão "cascateados" em uma nova folha de estilo "virtual" pelas seguintes regras, onde o número um tem a prioridade mais alta:

- ◊ Estilo embutido (dentro de um elemento HTML)
- ◊ Folhas de estilo externas e internas (na seção principal)
- ◊ Padrão do navegador

Portanto, um estilo embutido tem a prioridade mais alta e substituirá os estilos externos e internos e os padrões do navegador.

CSS – Sintaxe



Exemplo:

```
p {  
    color: red;  
    text-align: center;  
}
```

CSS – Seletores

Os seletores CSS são usados para "encontrar" (ou selecionar) os elementos HTML que você deseja estilizar.

Podemos dividir os seletores CSS em cinco categorias:

- ◊ Seletores simples (selecione os elementos com base no nome, id, classe)
- ◊ Seletores combinadores (selecione os elementos com base em uma relação específica entre eles)
- ◊ Seletores de pseudoclasse (selecione os elementos com base em um determinado estado)
- ◊ Seletores de pseudoelementos (selecione e estilize uma parte de um elemento)
- ◊ Seletores de atributos (selecione os elementos com base em um atributo ou valor de atributo)

***Sempre declare seus seletores do mais abrangente ao mais específico de cima para baixo!!!.

CSS – Seletores Simples

```
/*----Seletor universal          */
*{
    margin: 0;
    padding: 0;
    border: 0;
}

/*----Seletor de elementos CSS*/
p{
    text-align: center;
    color: blue;
}

/*----Seletor de Classe          */
.titulos {
    text-align: center;
}

h1.destaque{
    background-color: aquamarine;
}

/*----Seletor de ID              */
#cabecalho {
    text-align: center;
}
```

CSS – Seletores combinadores

/*

Elementos de combinação de seletores são usados para separar dois ou mais seletores simples que compõem um seletor combinado. Os elementos de combinação disponíveis são: espaço em branco (qualquer quantidade de espaço, tabulação ou caracteres de espaçamento), o sinal de maior “>” e o sinal de adição “+”. A função de cada um destes elementos de combinação dos seletores será descrita adiante.

*/

CSS – Seletores combinadores

```
/* Seletores descendentes
```

Um seletor descendente é uma combinação de dois ou mais seletores simples separados por um espaço em branco. Caso com elementos que sejam descendentes do primeiro elemento simples declarado no seletor. Por exemplo, na regra a seguir o seletor casa com todos os elementos p que sejam descendentes do elemento div.

```
*/  
div p {  
    color:#f00;  
}
```

CSS – Seletores combinadores

```
/*
```

Cada um dos seletores que compõem um seletor descendente pode ser um seletor simples de qualquer natureza. Na regra a seguir o seletor casa com todo o elemento p da classe info contido em um elemento li que esteja contido em um elemento div cuja id seja myid.

```
*/
```

```
div#myid li p.info {  
    color:#f00;  
}
```

CSS – Seletores combinadores

```
/* Seletores filho
   Um seletor filho tem como alvo um filho imediato de um elemento.
   O seletor filho consiste de um ou mais seletores simples separados por
   um sinal de maior ">". O elemento pai fica à esquerda do sinal ">", e
   é permitido deixar espaço em branco entre o elemento de combinação e
   os seletores. A regra a seguir aplica-se a todos os elementos strong
   que sejam filhos de um elemento div.
*/
div > strong {
    color:#f00;
}
```

CSS – Seletores combinadores

*/*Seletores irmãos adjacentes (sibling selectors)*

Um seletor irmão adjacente consiste de um ou mais seletores simples separados por um sinal de adição, “+”. É permitido deixar espaço em branco entre o elemento de combinação e os seletores. O seletor tem como alvo um elemento que seja irmão e adjacente ao primeiro elemento. Os elementos devem ter o mesmo pai e o primeiro elemento deve ser imediatamente anterior ao segundo.

**/*

```
p + p {  
    color:#f00;  
}
```

CSS – Seletores combinadores

/*Grupando

Para aplicar uma mesma regra a diferentes elementos alvo casados por diferentes seletores você pode agrupar os seletores em uma lista e separando-os por uma vírgula no lugar de escrever repetidamente a mesma regra para cada um dos seletores.

*/

```
div#news h3,div#news ul {  
    margin:0 2em;  
}
```

CSS – Seletores de pseudo-classe

```
/*
  Pseudo-classes casam elementos baseadas em características outras que não;
  seu nome, seus atributos ou seu conteúdo.
*/
/*:first-child
  Esta pseudo-classe casa com o elemento que é o primeiro filho de um outro
  elemento. Suponha que você quer estilizar diferenciadamente o primeiro
  parágrafo de um artigo. Se tal artigo estiver contido dentro de um elemento
  div ao qual foi atribuido a classe “article”, a regra a seguir casa com o
  primeiro elemento parágrafo p no artigo
*/
div.article p:first-child {
  font-style:italic;
}
```

CSS – Seletores de pseudo-classe

```
/*
  Para casar com todos os elementos p que sejam filhos de qualquer elemento
  você poderia usar o seletor da regra a seguir
*/
p:first-child {
  font-style:italic;
}
```

CSS – Seletores de pseudo-classe

`/*:link e :visited`

As pseudo-classes link afetam o estado dos links visitados e não visitados. Estes dois estados são mutuamente exclusivos – um link não pode ser visitado e não visitado ao mesmo tempo.

Estas pseudo-classes aplicam-se somente a hyperlinks e âncoras definidas na linguagem de marcação do documento. Em HTML, isto é válido para elementos com o atributo href

`*/`

CSS – Seletores de pseudo-classe

`/*:hover, :active, e :focus`

As pseudo-classes dinâmicas podem ser usadas para controlar a apresentação de determinados elementos na dependência de ações do usuário.

`:hover` aplica-se para quando o usuário coloca um dispositivo apontador em um elemento mas não o ativa. O uso mais comum é quando da ação de usuário de apontar o cursor do mouse sobre o elemento.

`:active` aplica-se para quando o usuário ativa um elemento. Para ação de mouse, equivale a pressionar o botão e mantê-lo pressionado até soltar.

`:focus` aplica-se para quando um elemento recebe foco, ou seja, enquanto aceita eventos de teclado.

Um elemento pode ser casado a várias pseudo-classes ao mesmo tempo. Um elemento pode receber foco e ter o cursor do mouse sobre ele ao mesmo tempo.

`*/`

CSS – Seletores de pseudo-classe

```
input[type=text]:focus {  
    color:#000;  
    background:#ffe;  
}
```

```
input[type=text]:focus:hover {  
    background:#fff;  
}
```

CSS – Seletores de pseudo-classe

```
input[type=text]:focus {  
    color:#000;  
    background:#ffe;  
}  
  
input[type=text]:focus:hover {  
    background:#fff;  
}  
  
/*  
   A primeira regra casa com o elemento input e tem o foco, a segunda regra casa com  
   o mesmo elemento quando tem o ponteiro do mouse sobre ele.  
*/
```

CSS – Seletores de pseudo-classe

```
/*:lang
```

A pseudo-class para linguagem (idioma) pode ser usada para estilizar elementos cujo conteúdo está escrito em uma determinada linguagem (idioma - uma língua para humanos e não uma linguagem de marcação). A regra a seguir define que tipo de aspas usar para textos inline que estão escritos no idioma da Suécia.

```
*/
```

```
q:lang(sv) {  
    quotes: "\201D" "\201D" "\2019" "\2019";  
}
```

```
/*
```

A linguagem para humanos (idioma) de um documento, normalmente é especificada pelo atributo lang em HTML e pelo atributo xml:lang em XHTML.

```
*/
```

CSS – Seletores de pseudo-elementos

```
/*
  Os pseudo-elementos permitem acessar e formatar partes do documento que não estão
  disponíveis como nós da árvore do documento.
*/

/*:first-line
  O pseudo-elemento :first-line afeta a primeira linha de texto de um parágrafo.
  Aplica-se somente a elementos nível de bloco, blocos inline, table-caption ou table-cell.
  O comprimento da primeira linha depende obviamente de uma série de fatores, ai incluido
  o tamanho da fonte e a largura do elemento container do texto.

  A regra a seguir aplica-se à primeira linha do texto de um parágrafo
*/
p:first-line {
  font-weight:bold;
  color: #600;
}
```

CSS – Seletores de pseudo-elementos

```
/*:first-letter
```

Este pseudo-elemento permite casar a primeira letra ou primeiro caractere de um elemento e aplica-se a elementos nível de bloco, list-item, table-cell, table-caption e bloco inline.

A regra a seguir aplica-se ao primeiro caractere de um elemento cuja classe denomina-se “preamble”.

```
*/
```

```
.preamble:first-letter {  
    font-size:1.5em;  
    font-weight:bold;  
}
```

CSS – Seletores de pseudo-elementos

```
/*:before e :after
Entre uma das mais discutidas funcionalidades das CSS os pseudo-elementos
:before e :after podem ser usados para gerar conteúdos antes e depois do conteúdo
de um elemento
*/
.cbb:before {
    content:"";
    display:block;
    height:17px;
    width:18px;
    background:url(top.png) no-repeat 0 0;
    margin:0 0 0 -18px;
}
```

CSS – Seletores de pseudo-elementos

```
/*
  Um exemplo do uso de :after para inserir a URL logo após o texto de um link.
*/
a:link:after {
  content: " (" attr(href) ")";
}
```

CSS – Seletores de atributos

```
/* [att]
   O seletor na regra a seguir casa com todos os elementos p que
   tenham o atributo title, independentemente do valor do atributo.
*/
p[title] {
    color: blueviolet;
}

/* [att=val]
   No próximo exemplo o seletor casa com todos os elementos div
   que tem um valor para o atributo class igual a error
*/
div[class=error] {
    color:#f00;
}
```

CSS – Seletores de atributos

```
/* [att~=val]
   Para atingir todos os elementos td cujo atributo headers
   contenha o valor “col1”, podemos usar o seguinte seletor.
*/
td[headers~=col1] {
  color:#f00;
}

/* [att|=val]
   O seletor seguinte atinge todo elemento p cujo atributo
   lang comece com en.
*/
p[lang|=en] {
  color:#f00;
}
```

CSS – Seletores de atributos

```
/*
```

Múltiplos seletores de atributos podem ser usados em um mesmo seletor. Isto possibilita atingir vários diferentes atributos para o mesmo elemento. A regra a seguir aplica-se a todos os elementos `blockquote` que tenham o atributo `class` de valor igual a “`quote`”, e mais o atributo `cite` (independentemente do seu valor)

```
*/
```

```
blockquote[class=quote][cite] {  
    color:#f00;  
}
```

CSS – Reset

```
/*===== CSS Reset =====*/
*, :after, :before {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    text-decoration: none;
}
body{
    font-size: 100%;
    list-style-type: none;
}
```

Outros modelos de CSS Reset:

<http://meyerweb.com/eric/tools/css/reset/>

CSS – Especificidade de Seletores

- Descreve a hierarquia do seletor.
- Seletores mais específicos se sobrepõem aos seletores menos específicos.

Do mais específico ao menos específico:

id -> class -> element -> *

- Atenção aos seletores combinadores.

Calculadora de Especificidade de seletores:

<https://specificity.keegan.st/>

- Se houver 2 seletores para o mesmo elemento com o mesmo nível de especificidade, prevalence a ordem inversa de declaração.

CSS – Cores

RGB – Red; Green; Blue

- ✓ Cada canal é representado por um valor numérico que pode variar de 0 à 255.
- ✓ É possível formar 16.777.216 cores.

```
div[class=error] {  
    color: #rgb(255, 0, 0);  
}
```

rgb(255, 0, 0)

rgb(0, 0, 255)

rgb(60, 179, 113)

rgb(238, 130, 238)

rgb(255, 165, 0)

rgb(106, 90, 205)

CSS – Cores

HEX – Hexadecimal

É idêntico ao sistema RGB, porém em hexadecimal.

```
td[headers~=col1] {  
    color: #f00;  
}
```

```
td[headers~=col1] {  
    color: #ff0000;  
}
```



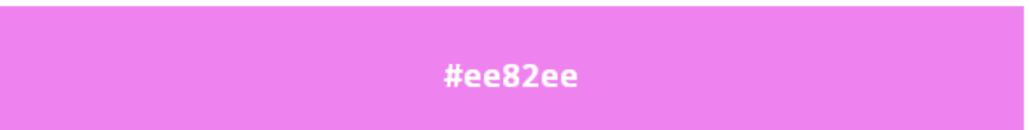
#ff0000



#0000ff



#3cb371



#ee82ee



#ffa500

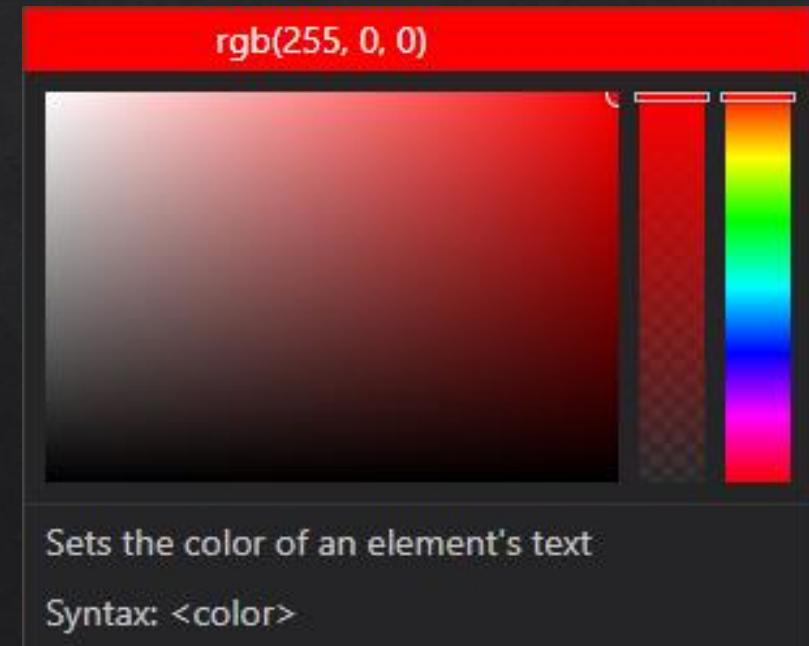
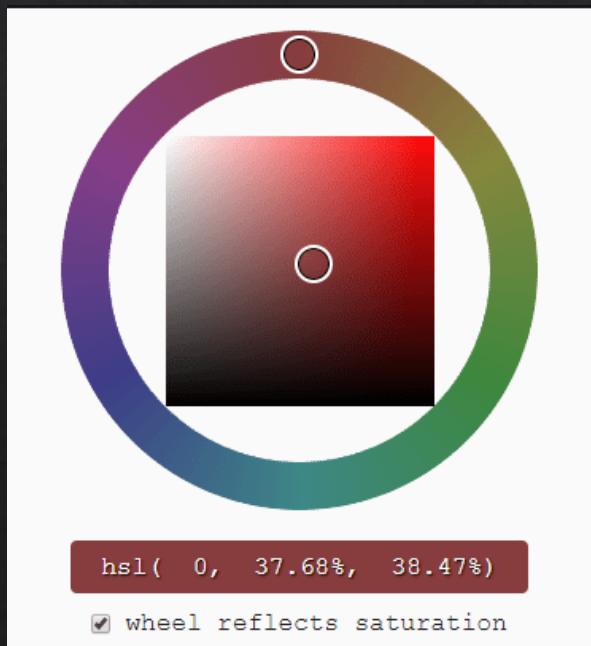
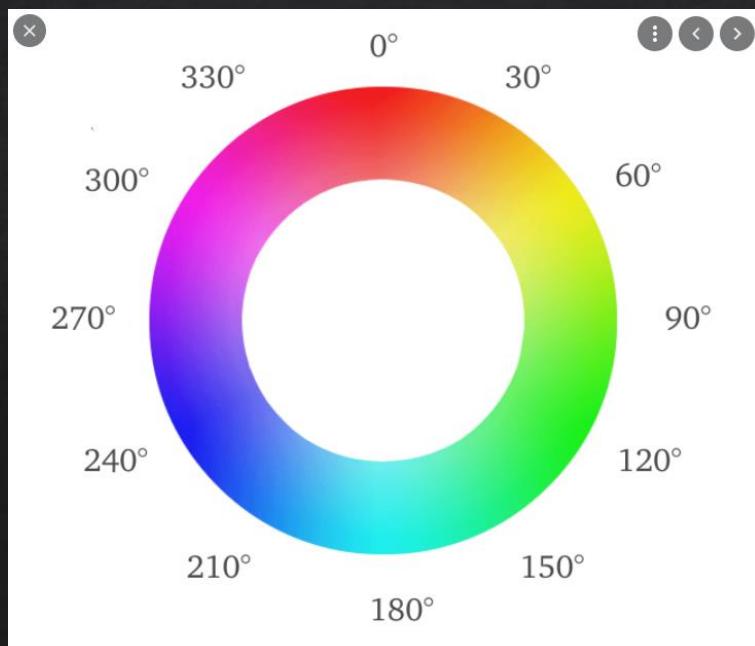


#6a5acd

CSS – Cores

HSL – Hue; Saturation; Lightness

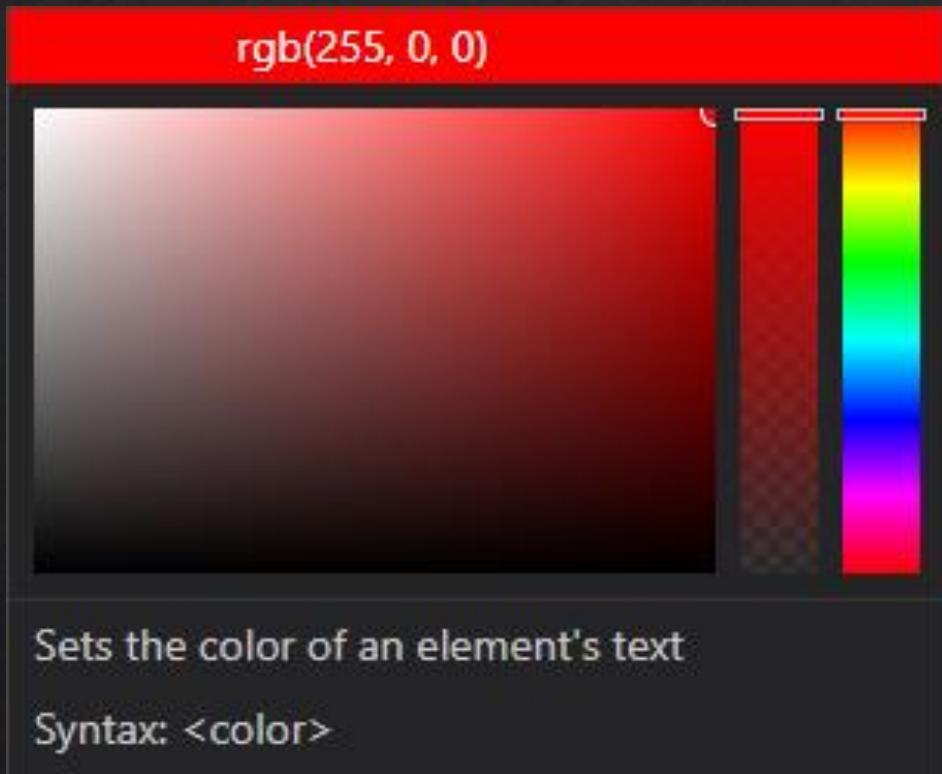
- Matiz é um grau na roda de cores de 0 a 360. 0 é vermelho, 120 é verde e 240 é azul.
- A saturação é um valor percentual, 0% significa um tom de cinza e 100% é a cor total.
- A luminosidade também é uma porcentagem, 0% é preto, 50% não é nem claro nem escuro, 100% é branco.



CSS – Cores

RGBA - Red; Green; Blue; Alpha

HSLA - Hue; Saturation; Lightness; Alpha



HWB – Hue; Whiteness Blackness

CMYK – Cyan; Magenta; Yellow; black

Ncol – Natural Colors

CSS Color Names:

<https://www.w3.org/wiki/CSS/Properties/color/keywords>

https://www.w3schools.com/colors/colors_names.asp

CSS – Backgrounds

- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background (shorthand property)

```
div {  
    background-color: green;  
    opacity: 0.3;  
}  
  
div {  
    background: rgba(0, 255, 0, 0.3) /* Green background with 30% opacity */  
}
```

CSS – Textos

- color
- text-transform
- text-shadow
- text-decoration-line
- text-decoration-color
- text-decoration-style
- text-decoration-thickness
- text-decoration
- text-indent
- letter-spacing
- line-height
- word-spacing
- white-space
- text-align
- text-align-last
- direction
- vertical-align

CSS – Fontes

Font Family

Sans-serif vs. Serif



- `font-style`
- `font-variant`
- `font-weight`
- `font-size`
- `font-family`
- `font`

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	<i>Brush Script MT</i> <i>Lucida Handwriting</i>
Fantasy	Copperplate <i>Papyrus</i>

CSS – Fontes

- **Font Web Safe & Fallbacks**

```
#titulo2{  
    font-family: 'Alfa Slab One', cursive;  
}
```

- **Font Size (unidades de medida)**

- px
- em (1em = 16px)
- vw (viewport width – responsivo)

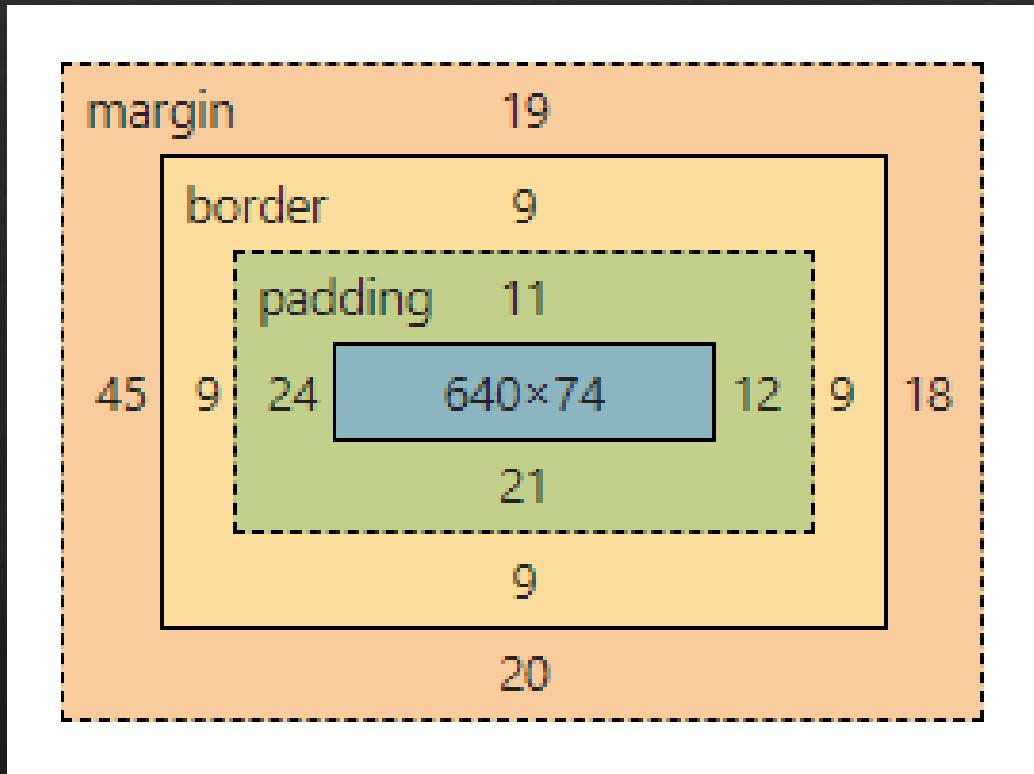
```
#titulo2{  
    font-size: 1.5em;  
}
```

CSS – Fontes

- Google Fonts

```
<head>
  <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
  <style>
    body {
      font-family: "Sofia", sans-serif;
    }
  </style>
</head>
```

CSS – Box Model



```
h1{  
    margin: 5px;  
    margin: 19px 18px 20px 21px ;  
    margin-left: 45px;  
  
    border-style: solid;  
    border-width: 9px;  
  
    padding: 10px;  
    padding: 11px 12px 21px 24px;  
}
```

CSS – Height/Width

`height: 35vh; /* Determina a Altura do elemento*/`

`width: 35vh; /* Determina a Largura do elemento*/`

`max-height: 35vh; /* Determina a Altura Máxima do elemento*/`

`max-width: 35vh; /* Determina a Largura Máxima do elemento*/`

`min-height: 35vh; /* Determina a Altura Mínima do elemento*/`

`min-width: 35vh; /* Determina a Largura Mínima do elemento*/`

Valores possíveis:

Auto - Este é o padrão. O navegador calcula a altura e a largura

Length - Define a altura/largura em px, cm, etc.

% - Define a altura/largura em porcentagem do bloco que o contém

Initial - Define a altura/largura para seu valor padrão

Inherit - A altura/largura será herdada de seu valor pai

CSS – Display

Display é a propriedade CSS mais importante para controlar o layout.

A propriedade display especifica se/como um elemento é exibido.

Cada elemento HTML tem um valor de exibição padrão, dependendo do tipo de elemento. O valor de exibição padrão para a maioria dos elementos é block ou inline.

display: inline; height: 35vh;

display: block; width: 35vh;

display: none;

visibility: hidden;

CSS – Display

- **Elementos de Bloco**

Elementos bloco ocupam todo o espaço horizontal disponível e iniciam uma nova linha no documento. Novos elementos irão começar na próxima linha livre.

Exemplos de elementos bloco:

div, h1 até h6, p, ul, ol, form, header, footer, section...

- **Elementos de Linha**

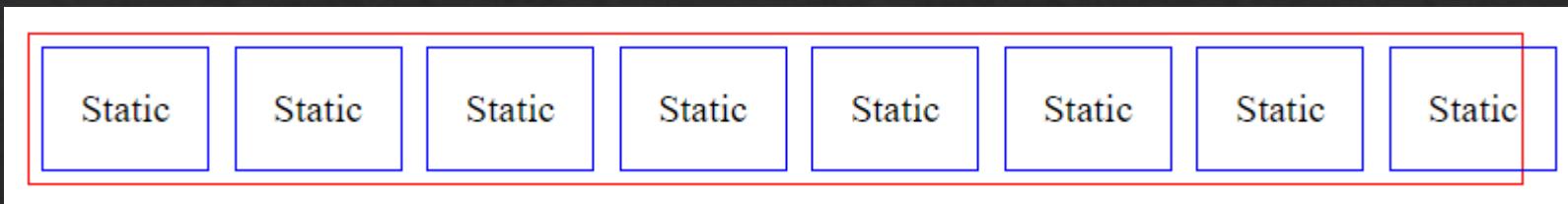
Elementos em linha ocupam apenas o espaço necessário e não iniciam uma nova linha. São chamados elementos em linha justamente por aparecer na mesma linha que outros elementos, caso seja possível.

Exemplos de elementos em linha:

span, Strong, em, a, img...

CSS – Display – Flexbox

Flex Container (Elemento pai)



```
<div class="linha">
  <div class="coluna"> Static </div>
  <div class="coluna"> Static </div>
</div>
```

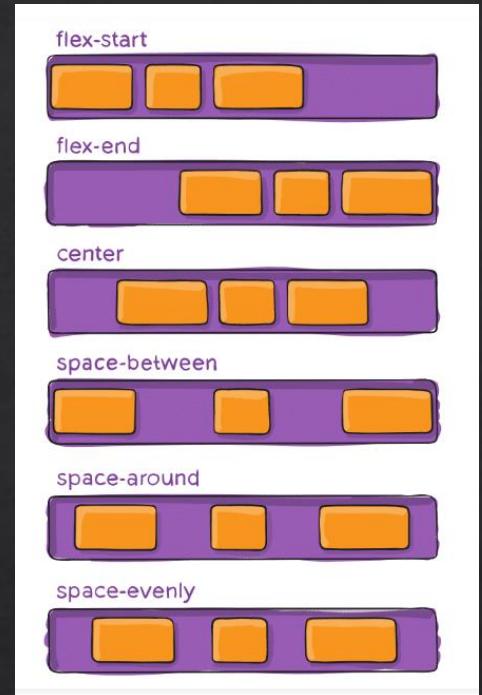
```
.linha{
  border: 1px solid red;
  margin: 15px;
  display: flex;
}

.coluna{
  border: 1px solid blue;
  margin: 5px;
  padding: 15px;
}
```

CSS – Display – Flexbox

Flex Container (Elemento pai)

```
.linha{  
    border: 1px solid red;  
    margin: 15px;  
    height: 300px;  
  
    display: flex;  
    flex-direction: row; /*column; column-reverse; row; row-reverse;*/  
  
    /*Alinhamento Horizontal*/  
    justify-content: space-between; /*center; space-around; space-between; space-evenly;*/  
  
    /*Alinhamento Vertical*/  
    align-items: center; /*center; flex-start; flex-end; stretch; baseline;*/  
    /*align-content: flex-end; /*center; space-between; space-around; stretch; flex-start; flex-end;*/  
  
    /*Modo de Encapsulamento dos itens (quebra de linha...)*/  
    flex-wrap: wrap; /*wrap; wrap-reverse; nowrap;*/  
  
}  
/*Propriedade abreviada para definir as propriedades flex-direction e flex-wrap.*/  
flex-flow: row wrap;
```



CSS – Display – Flexbox

Flex Items (Elementos filho)

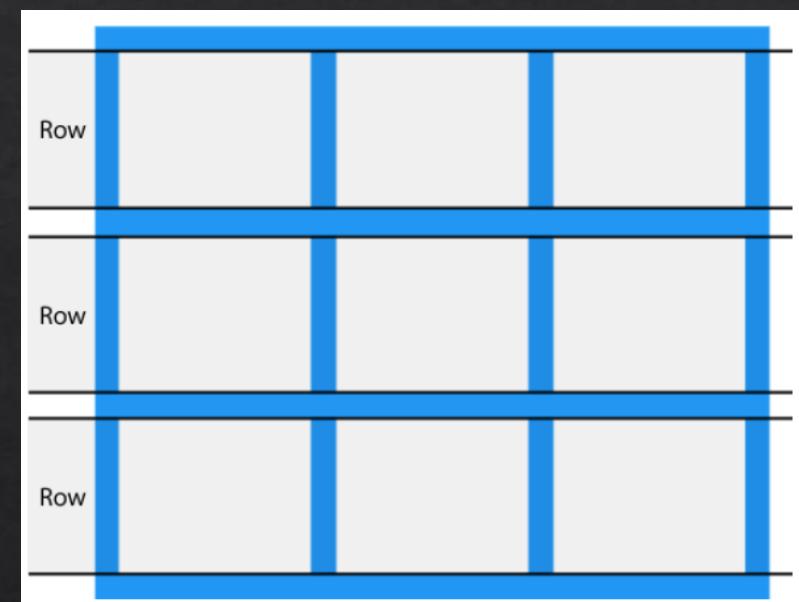
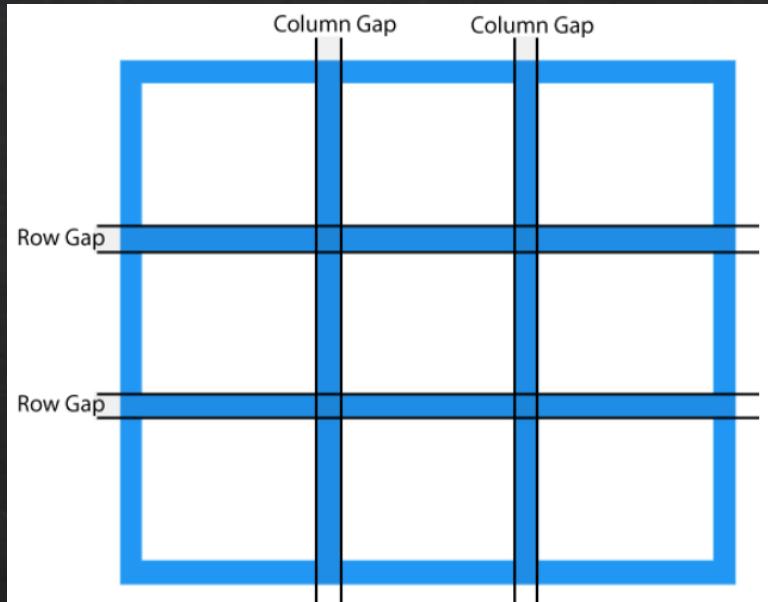
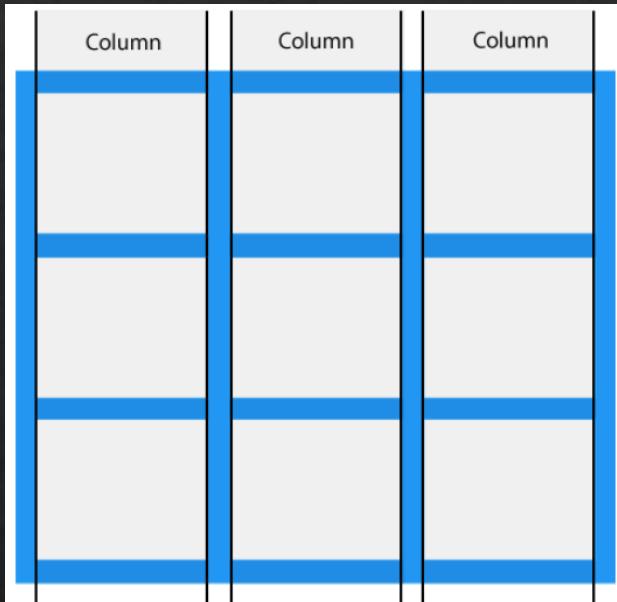
```
<div class="linha">
  <div class="coluna" id="item-1"> Static 1 </div>
  <div class="coluna" id="item-2"> Static 2 </div>
  <div class="coluna" id="item-3"> Static 3 </div>
</div>

#item-1{
  order: 3; /*manipula a ordem do elemento dentro do container*/
  flex-grow: 1; /*proporção de crescimento do elemento*/
  flex-shrink: 1; /*proporção de diminuição do elemento*/
}

#item-2{
  order: 2;
  flex-grow: 1;
  flex-basis: 200px; /*tamanho mínimo do elemento*/
  flex-shrink: 2;
}

#item-3{
  order: 1;
  flex-grow: 2;
  flex-shrink: 4;
}
```

CSS – Display – Grid Layout

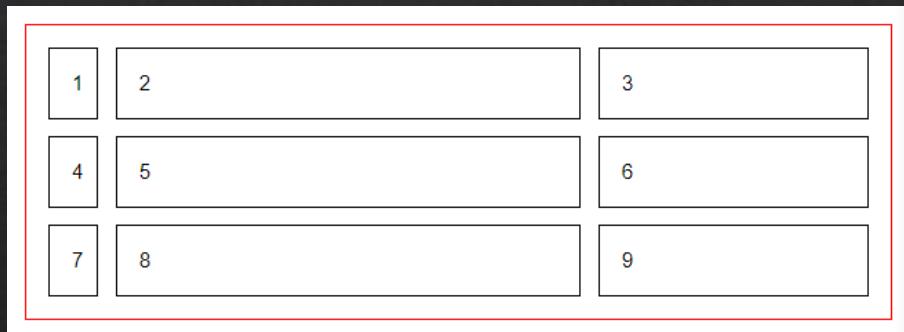


```
.grid-container {  
  display: grid;  
  grid-template-columns: 45px auto 33%;  
  gap: 5px;  
}
```

```
column-gap: 16px;  
row-gap: 15px;  
gap: 5px;
```

CSS – Display – Grid Layout

Template Column



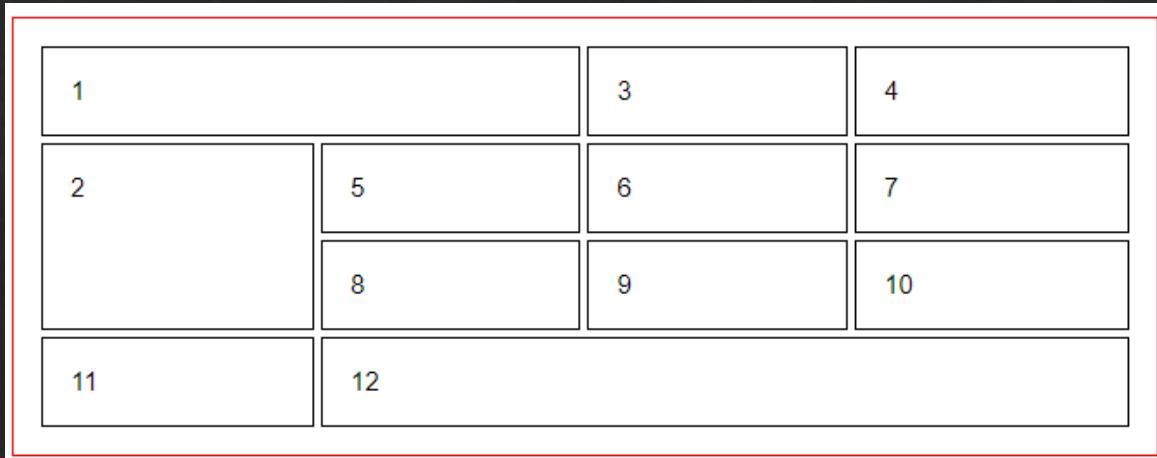
```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item">3</div>
  <div class="grid-item">4</div>
  <div class="grid-item">5</div>
  <div class="grid-item">6</div>
  <div class="grid-item">7</div>
  <div class="grid-item">8</div>
  <div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

```
.grid-container {
  padding: 20px;
  border: 1px solid red;
  display: grid;
  grid-template-columns: 45px auto 33%;
  gap: 5px;
}

.grid-item {
  padding: 20px;
  border: 1px solid black;
}
```

CSS – Display – Grid Layout

Trabalhando com Itens do Grid



```
<div class="grid-container">
  <div id="item1" class="grid-item">1</div>
  <div id="item2" class="grid-item">2</div>
  <div id="item3" class="grid-item">3</div>
  <div id="item4" class="grid-item">4</div>
  <div id="item5" class="grid-item">5</div>
  ...
  <div id="item12" class="grid-item">12</div>
</div>
```

```
.grid-container {
  padding: 20px;
  border: 1px solid red;
  display: grid;
  grid-template-columns: auto auto auto auto;
  gap: 5px;
}

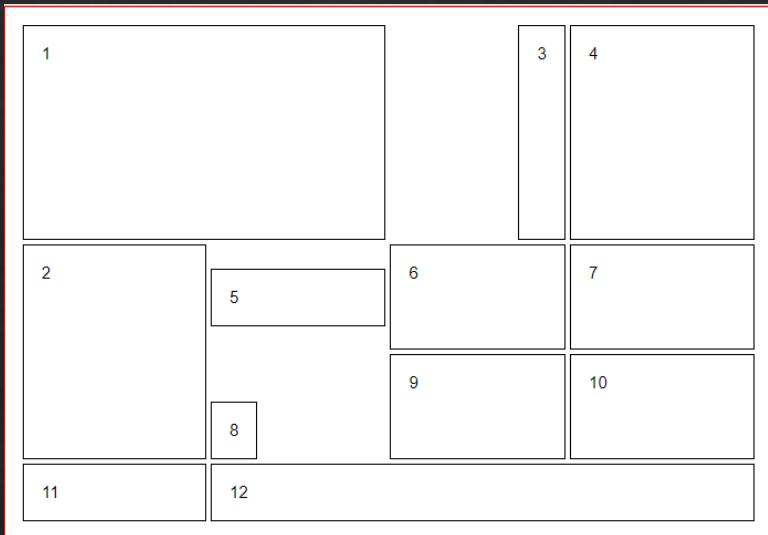
.grid-item {
  padding: 20px;
  border: 1px solid black;
}

#item1 {
  grid-column-start: 1;
  grid-column-end: 3;
}

#item2{
  grid-row-start:2;
  grid-row-end: 4;
  height: 200px;
}
```

CSS – Display – Grid Layout

Trabalhando com Itens do Grid - Alinhamentos



```
<div class="grid-container">
  <div id="item1" class="grid-item">1</div>
  <div id="item2" class="grid-item">2</div>
  <div id="item3" class="grid-item">3</div>
  <div id="item4" class="grid-item">4</div>
  <div id="item5" class="grid-item">5</div>
  ...
  <div id="item12" class="grid-item">12</div>
</div>
```

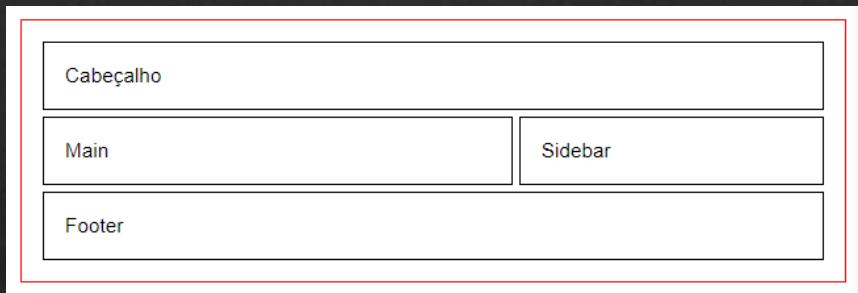
```
#item3 {
  justify-self: end;
  height: 200px;
}
```

```
#item5 {
  align-self: center;
}
```

```
#item8{
  justify-self: start;
  align-self: end;
}
```

CSS – Display – Grid Layout

Template Areas



```
<div class="grid-container">
  <header id="id_header" class="grid-item">
    <p>Cabeçalho</p>
  </header>
  <aside id="id_sidebar" class="grid-item">
    <p>Sidebar</p>
  </aside>
  <main id="id_main" class="grid-item">
    <p>Main</p>
  </main>
  <footer id="id_footer" class="grid-item">
    <p>Footer</p>
  </footer>
</div>
```

```
.grid-container {
  padding: 20px;
  border: 1px solid red;
  margin: 20px;
  gap: 6px;
  display: grid;
  grid-template-areas:
    'header header header'
    'main main sidebar'
    'footer footer footer';
}

.grid-item {
  padding: 20px;
  border: 1px solid black;
}

#id_header {
  grid-area: header;
}

#id_sidebar {
  grid-area: sidebar;
}
```

CSS – Display – Grid Layout

Alinhamento dentro do Grid

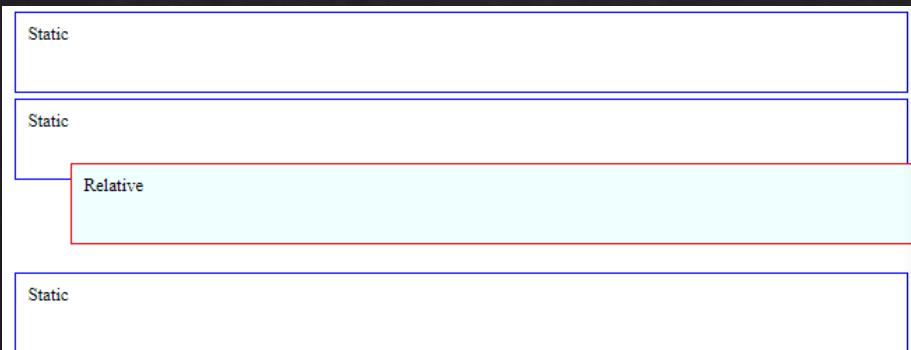
```
/*horizontal*/  
justify-content: center; /*start; center; end*/  
  
/*vertical*/  
align-items: center; /*start; center; end*/
```

CSS – Posicionamento

Static – É a posição padrão dos elementos.

Relative – O elemento é posicionado em relação à sua posição normal.

Definir as propriedades superior, direita, inferior e esquerda de um elemento relativamente posicionado fará com que ele seja ajustado para longe de sua posição normal. Outros conteúdos não serão ajustados para caber em qualquer lacuna deixada pelo elemento.



```
<body>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="relative"> Relative </div>
  <div class="static"> Static </div>
</body>
```

```
.relative {
  height: 50px;
  border: 2px solid red;
  margin: 5px;
  padding: 10px;
  background-color: azure;
}

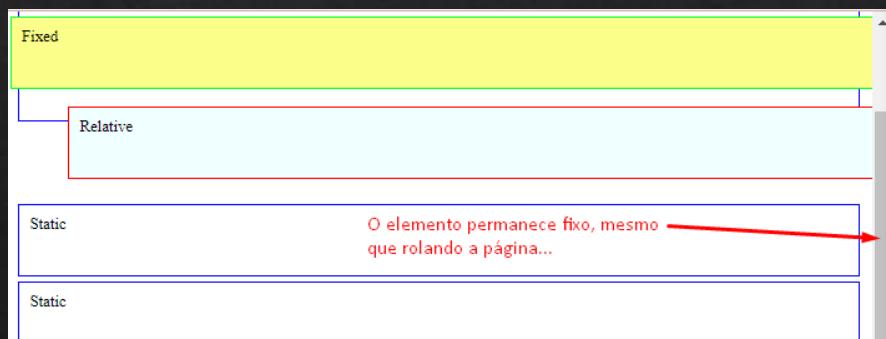
position: relative;
left: 50px; /*left, right*/
bottom: 20px; /*bottom, top*/
}
```

The code defines a CSS class named ".relative" with various styles: height, border, margin, padding, and background-color. It also includes position: relative;, left: 50px;, and bottom: 20px; properties. A large orange arrow points from the word "relative" in the CSS code to the "Relative" box in the diagram, indicating the relationship between the code and the visual representation.

CSS – Posicionamento

Fixed – Um elemento com position: fixed; é posicionado em relação à viewport, o que significa que ele sempre permanece no mesmo lugar, mesmo que a página seja rolada. As propriedades superior, direita, inferior e esquerda são usadas para posicionar o elemento.

Um elemento fixo não deixa uma lacuna na página onde normalmente estaria localizado.
*Muito usado para criar menus fixos.



```
<body>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="relative"> Relative </div>
  <div class="static"> Static </div>
  <div class="fixed"> Fixed </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
</body>
```

```
.fixed {
  height: 50px;
  border: 2px solid rgb(0, 255, 30);
  margin: 5px;
  padding: 10px;
  background-color: rgba(251, 255, 138);

  position: fixed;
  width: 100%;
  top: 0;
  left: 0;
}
```

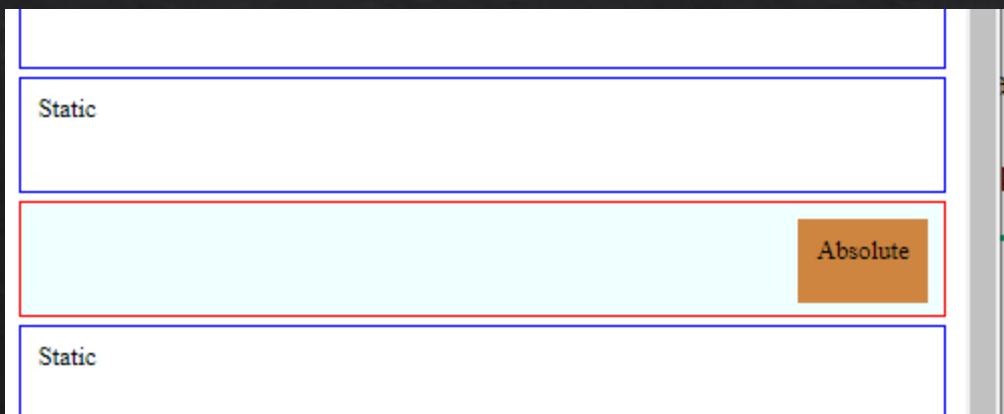


CSS – Posicionamento

Absolute - Um elemento com `position: absolute;` é posicionado em relação ao ancestral posicionado mais próximo (em vez de posicionado em relação à viewport, como fixo).

No entanto; se um elemento posicionado absoluto não tiver ancestrais posicionados, ele usará o corpo do documento e se moverá junto com a rolagem da página.

Nota: Elementos posicionados absolutos são removidos do fluxo normal e podem sobrepor elementos.



```
<div class="relative">
  <div class="absolute"> Absolute </div>
</div>

.absolute {
  height: 30px;
  border: 2px solid peru;
  padding: 10px;
  background-color: peru;

  position: absolute;
  right: 10px;
}
```

CSS – Posicionamento

Sticky - Um elemento com `position: sticky;` é posicionado com base na posição de rolagem do usuário.

Um elemento fixo alterna entre `relative` e `fixed`, dependendo da posição de rolagem. Ele é posicionado relativo até que uma determinada posição de deslocamento seja encontrada na viewport - então ele "gruda" no lugar (como `position:fixed`).



```
<body>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="relative"> Relative </div>
  <div class="static"> Static </div>
  <div class="fixed"> Fixed </div>
  <div class="sticky"> Sticky </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
  <div class="relative">
    <div class="absolute"> Absolute </div>
  </div>
  <div class="static"> Static </div>
  <div class="static"> Static </div>
</body>
```

```
.sticky{
  height: 50px;
  border: 2px solid rgb(0, 238, 255);
  margin: 5px;
  padding: 10px;
  background-color: bisque;
  position: sticky;
  top: 250px;
}
```

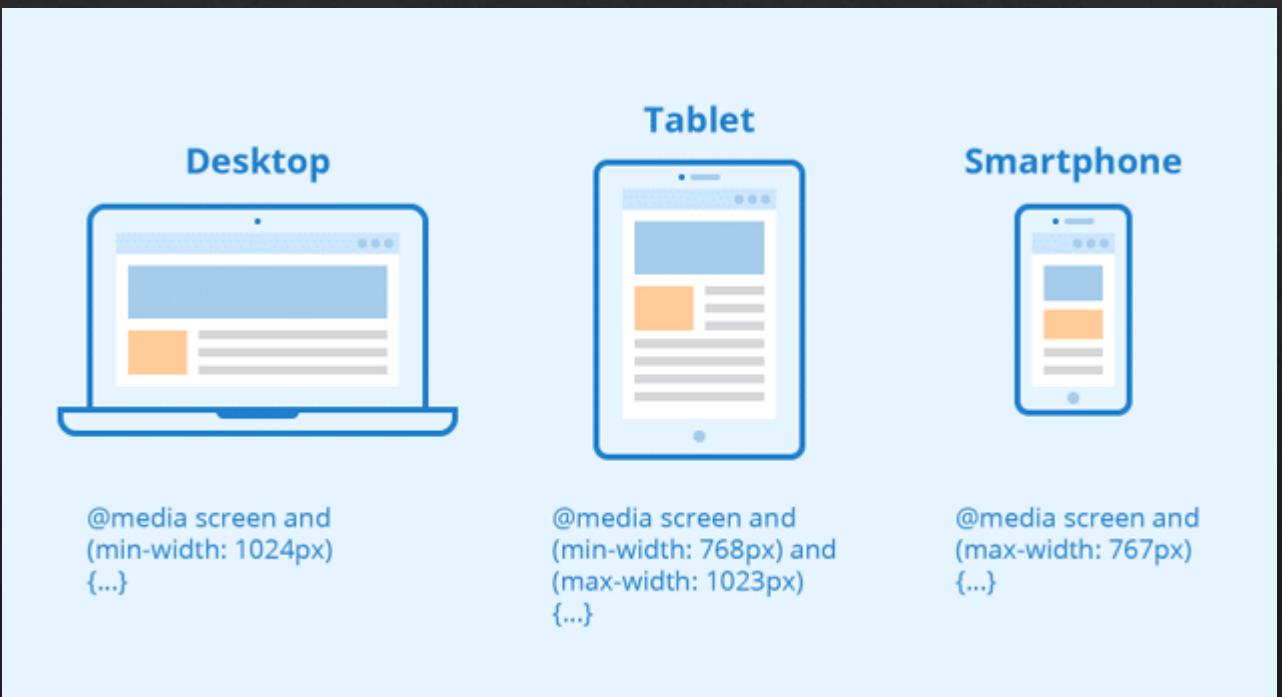
CSS – Responsividade

Media Queries

As consultas de mídia podem ser usadas para verificar características, como:

- largura e altura da viewport
- largura e altura do dispositivo
- orientação (o tablet/telefone está no modo paisagem ou retrato?)
- resolução

O uso de consultas de mídia é uma técnica popular para fornecer uma folha de estilo personalizada para desktops, laptops, tablets e telefones celulares (como iPhone e telefones Android).



CSS – Responsividade

Media Queries

```
<!-- Configuração do Media Query no Head HTML-->
<meta name="viewport" content="width=device-width, initial-scale=1.0">

/* --- Sintaxe --- */
@media not|only mediatype and (expressions) {
    CSS-Code;
}

@media not|only mediatype and (mediafeature
and|or|not mediafeature) {
    CSS-Code;
}

<link rel="stylesheet" media="mediatype and|not|only
(expressions)" href="print.css">
```

Media Types:

All - Usado para todos os tipos de media
Print - Usado para Impressoras
Screen - Usado para Telas
Speech - Usado para leitores de tela que "lê" a página em voz alta

Media Features:

max-width: 600px - largura igual ou menor que 600px
min-width: 600px - largura igual ou maior que 600px

orientation: portrait - retrato (vertical)
orientation: landscape - paisagem (horizontal)

min-resolution: 300dpi - resolução de tela mínima
max-resolution: 300dpi - resolução de tela máxima
resolution: 300dpi - resolução exata

min-aspect-ratio: 16/9 - proporção de tela mínimo
max-aspect-ratio: 16/9 - proporção de tela máximo
aspect-ratio: 16/9 - proporção de tela exata

CSS – Responsividade

Media Queries

Not Only e And:

not: A palavra-chave **not** inverte o significado de uma media query inteira.

only: A palavra-chave **only** impede que navegadores mais antigos que não suportam consultas de mídia com recursos de mídia apliquem os estilos especificados. Não tem efeito em navegadores modernos.

and: A palavra-chave **and** combina um recurso de mídia com um tipo de mídia ou outros recursos de mídia.

Eles são todos opcionais. No entanto, se você usar **not** ou **only**, também deverá especificar um tipo de mídia.

CSS – Responsividade

Media Queries - Exemplos

```
<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">
<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">

/* - Oculte um elemento quando a largura do navegador for de 600px ou menos - */
@media screen and (max-width: 600px) {
    div.example {
        display: none;
    }
}

/* Em telas com 992px de largura ou menos ...*/
@media screen and (max-width: 992px) {
    .column {...}
}

/* Em telas com 600px de largura ou menos ...*/
@media screen and (max-width: 600px) {
    .column {...}
}
```

CSS – Responsividade

Media Queries - Exemplos

```
/*Use uma cor de fundo azul claro se a orientação estiver no modo paisagem*/
@media only screen and (orientation: landscape) {
    body {
        background-color: lightblue;
    }
}
```

CSS – Responsividade

Media Queries - Exemplos

Lista separada por vírgula : adicione uma consulta de mídia adicional a uma já existente, usando uma vírgula (isso se comportará como um operador OR):

```
/*Quando a largura estiver entre 600px e 900px OU acima de 1100px altere a aparência de ...*/  
@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {  
    div.example {  
        font-size: 50px;  
        padding: 50px;  
        border: 8px solid black;  
        background: yellow;  
    }  
}
```

CSS – Responsividade

Media Queries

Mobile First - min-width

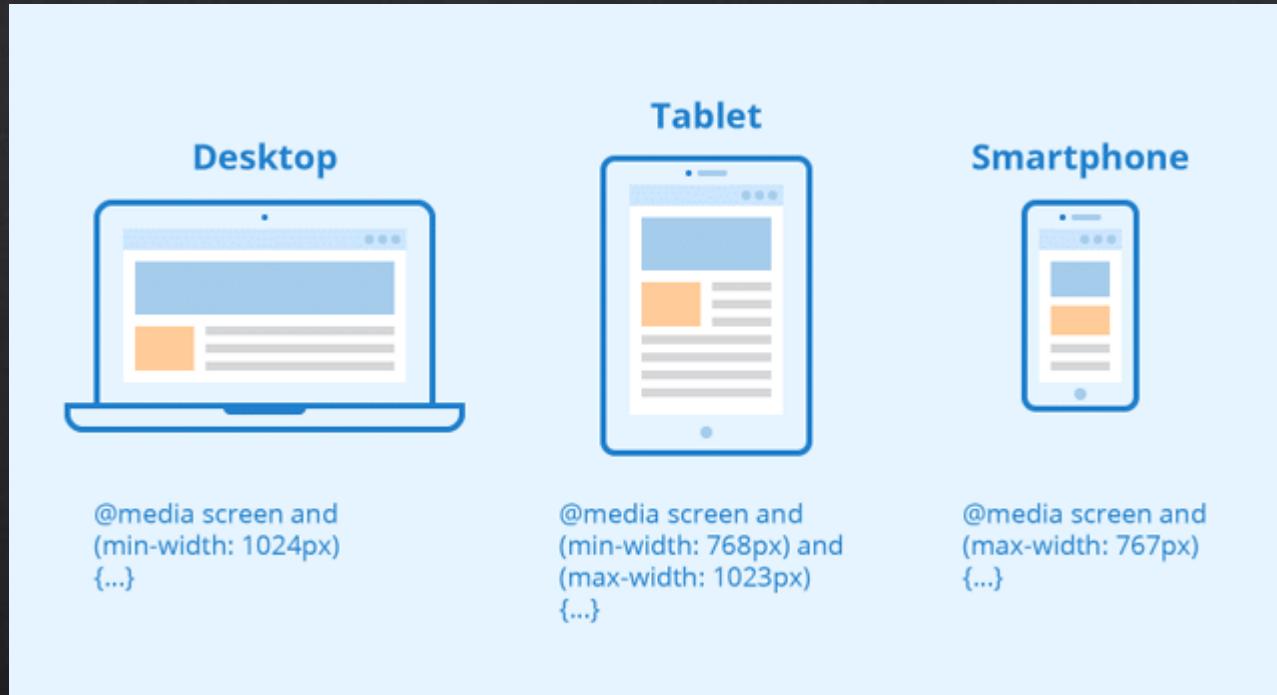
```
/*max-width, min-width, orientation:portrait */
@media screen and (min-width: 350px){
    /* all, screen, print, speech*/
    h1 {
        color: green;
    }
}

/*max-width, min-width, orientation:portrait */
@media screen and (min-width: 450px){
    /* all, screen, print, speech*/
    h1 {
        color: red;
    }
}
```

CSS – Responsividade

Media Queries Breakpoints

```
/*max-width, min-width, orientation:portrait */  
@media screen and (min-width: 350px){  
    /* all, screen, print, speech*/  
    h1 {  
        color: green;  
    }  
  
    /*max-width, min-width, orientation:portrait */  
    @media screen and (min-width: 450px){  
        /* all, screen, print, speech*/  
        h1 {  
            color: red;  
        }  
    }
```



Padrão do Bootstrap:

- 576px para telefones retrato
- 768px para comprimidos
- 992px para laptops
- 1200px para grandes dispositivos

CSS – Responsividade

Fluid Layout

Um layout fluido se baseia em valores dinâmicos como uma porcentagem da largura do viewport.



Esta abordagem aumentará ou diminuirá dinamicamente os diferentes tamanhos de elementos de recipientes com base no tamanho da tela.

CSS – Responsividade

Layout do Flexbox

Embora um layout baseado em porcentagem seja fluido, muitos designers e desenvolvedores web sentiram que ele não era dinâmico ou flexível o suficiente. Flexbox é um módulo CSS projetado como uma forma mais eficiente de disposição de múltiplos elementos, mesmo quando o tamanho do conteúdo dentro do contêiner é desconhecido.

Um recipiente flexível expande os itens para preencher o espaço livre disponível ou os encolhe para evitar o transbordamento. Estes recipientes flexíveis têm uma série de propriedades únicas, como conteúdo justificado, que você não pode editar com um elemento HTML regular.

CSS – Responsividade

Imagens - layout fluido vs. html srcset

```
img { width: 100%; } O Tamanho da imagem será renderizado com base na tela, porém a imagem sempre será carregada em seu tamanho máximo na memória.
```

Essa abordagem resultará em um carregamento mais otimizado...

```

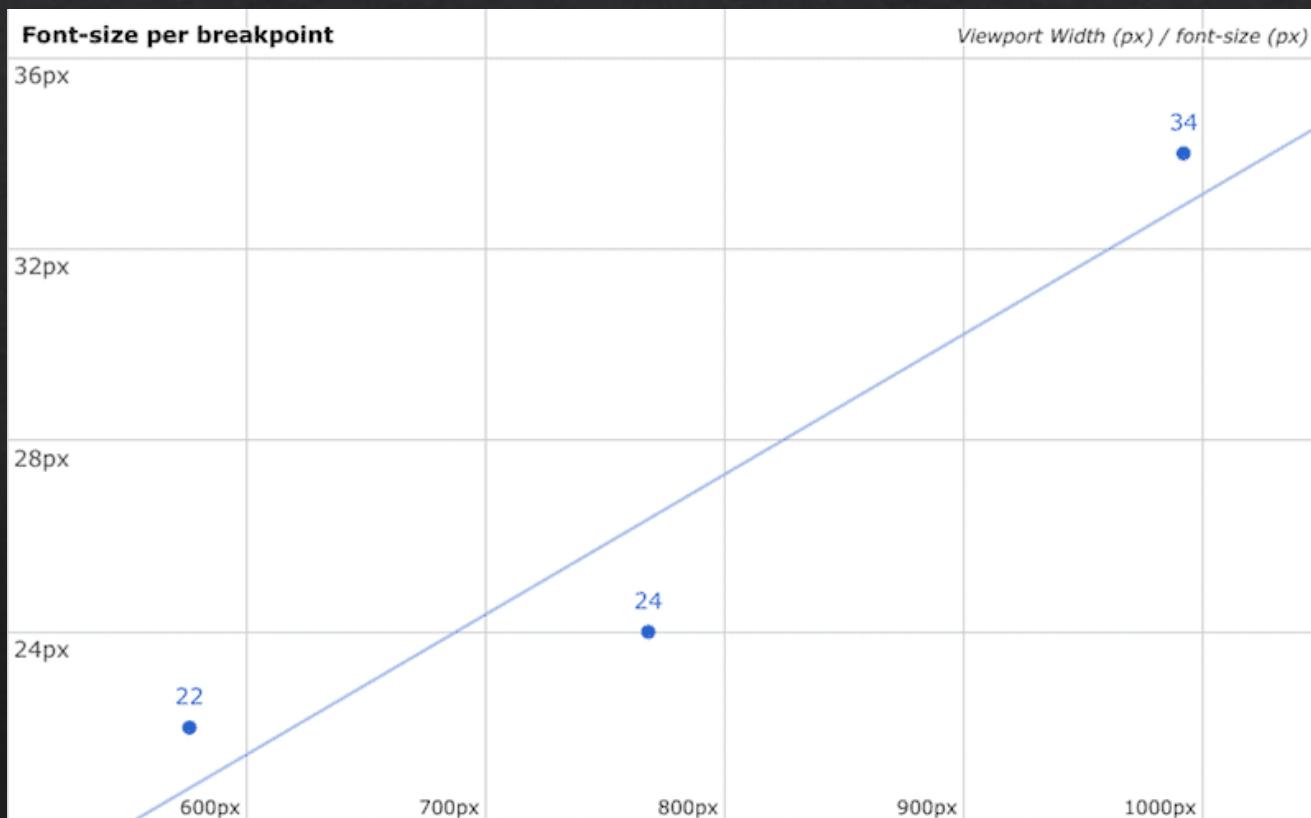
```

Essa abordagem também resultará em um carregamento mais otimizado...

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  
</picture>
```

CSS – Responsividade

Tipografia Responsiva



CSS – Responsividade

O Site é Mobile Friendly?

<https://search.google.com/test/mobile-friendly>

CSS – Unidades de Medida

Comprimentos Absolutos

As unidades de comprimento absoluto são fixas e um comprimento expresso em qualquer um deles aparecerá exatamente com esse tamanho.

Unidades de comprimento absoluto não são recomendadas para uso na tela, porque os tamanhos de tela variam muito. No entanto, eles podem ser usados se a mídia de saída for conhecida, como para layout de impressão.

Unit	Description
cm	centimeters
mm	millimeters
in	inches ($1\text{in} = 96\text{px} = 2.54\text{cm}$)
px *	pixels ($1\text{px} = 1/96\text{th of 1in}$)
pt	points ($1\text{pt} = 1/72 \text{ of 1in}$)
pc	picas ($1\text{pc} = 12 \text{ pt}$)

CSS – Unidades de Medida

Comprimentos Relativos

As unidades de comprimento relativo especificam um comprimento relativo a outra propriedade de comprimento. As unidades de comprimento relativo escalam melhor entre diferentes mídias de renderização.

Unit	Description
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element