
# CSS

PROF. VINICIUS VON GLEHN DE FILIPPO

VINICIUS.FILIPPO@ACADEMICO.DOMHELDER.EDU.BR

# O que é o CSS?

- Desenvolvido pelo W3C (World Wide Web Consortium) em 1996, o CSS (Cascading Style Sheets) é uma linguagem de folhas de estilo em cascata que permite alterar a aparência (estilo) padrão de páginas HTML utilizados nos navegadores
- O CSS separa o conteúdo (estrutura do documento em HTML) da representação visual do site

# Estilo padrão de um navegador

## Browser defaults

The browser will style HTML documents using an internal stylesheet. This ensures that headings are larger than normal text, links are highlighted and structures such as lists and tables are understandable.

Paragraphs are spaced out. List items get a bullet or number, Links are highlighted and underlined.

- Item One
- Item Two

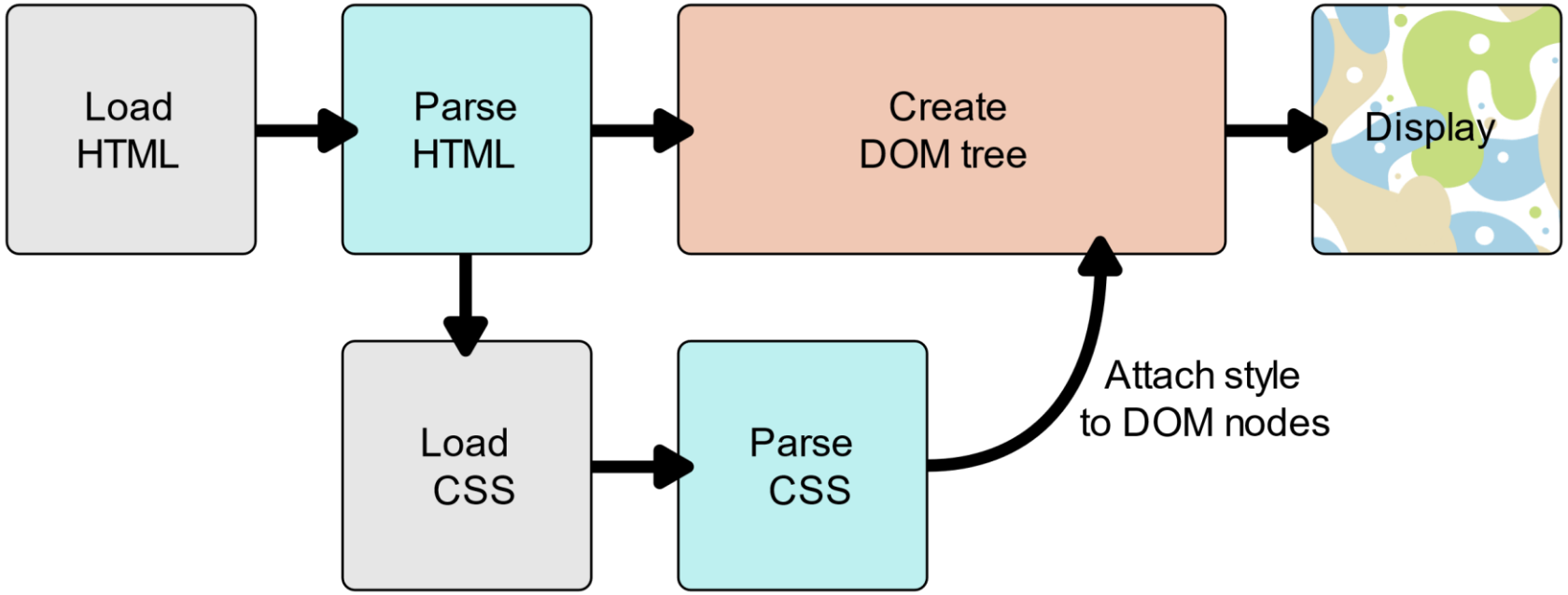
### A level 2 heading

You can change all of this with CSS.

# Como o CSS funciona?

1. O navegador carrega o HTML
2. O HTML é convertido e salvo na memória do computador como um DOM (*Document Object Model*)
3. O navegador requisita a maioria dos recursos que estão lincados no documento HTML como imagens incorporadas, vídeos, folhas de estilo CSS, etc. Os códigos JavaScript são manipulados um pouco mais tarde durante o processo
4. O navegador analisa o CSS e interpreta as diferentes regras por meio de seus diferentes tipos de seletores encontrados e insere as regras de estilização que devem ser aplicadas para cada nó no DOM, anexando os estilos para os elementos como foram especificados nas folhas de estilização (este processo intermediário é chamado de render tree ou árvore de renderização)
5. A árvore de renderização é organizada na estrutura para que das regras de estilo possam ser posteriormente aplicadas ao documento
6. O visual final da página é por fim apresentado na tela (este estágio é chamado de *painting* ou pintura)

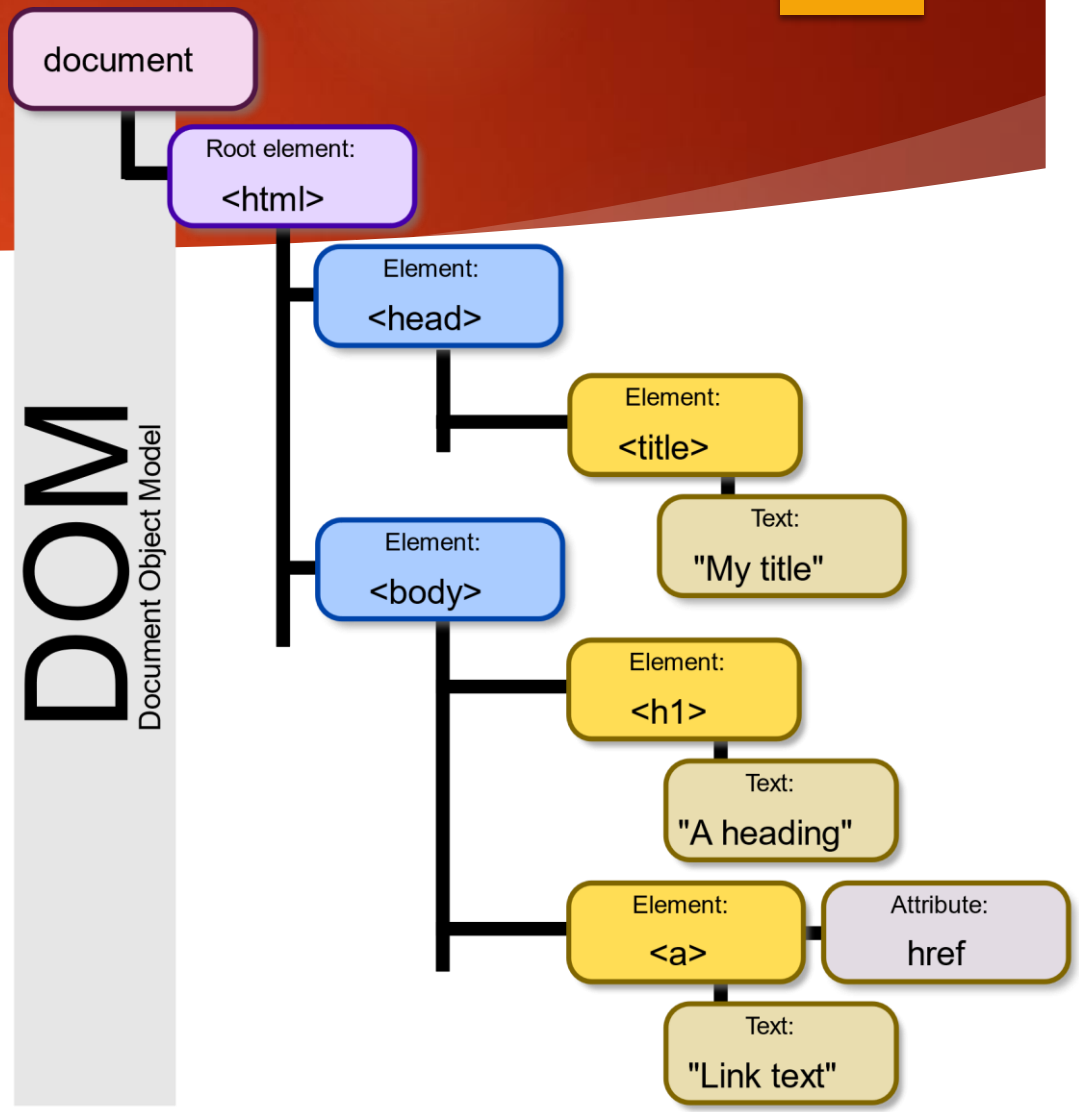
# Como o CSS funciona?



# DOM

**DOM (Document Object Model)**

- Estrutura arborea (*tree-like*)
- Cada elemento, atributo, ou fragmento de texto na linguagem de marcação (markup language) torna-se um **DOM node** (nó ou ponto de intersecção) na estrutura de árvore
- Os nodes são definidos por meio do relacionamento com outros nodes presentes no DOM
- Alguns elementos são pais ou superiores a outros elementos chamados *child nodes*, em português, nós filhos ou nós secundários
- Child nodes podem possuir elementos irmãos



# Aplicando CSS no seu HTML

- **Estilos inline**
  - É péssimo para manutenção uma vez que pode ser necessário atualizar a mesma informação diversas vezes em cada documento
  - Mistura informação de estilização do CSS com a estrutura do HTML, tornando o código de difícil leitura e compreensão
  - Manter diferentes tipos de código separados torna o trabalho muito mais fácil para todos

# Aplicando CSS no seu HTML

- **Folha de estilos interna**
  - O CSS é inserido dentro do elemento **<style>** localizado no **<head>** do documento HTML

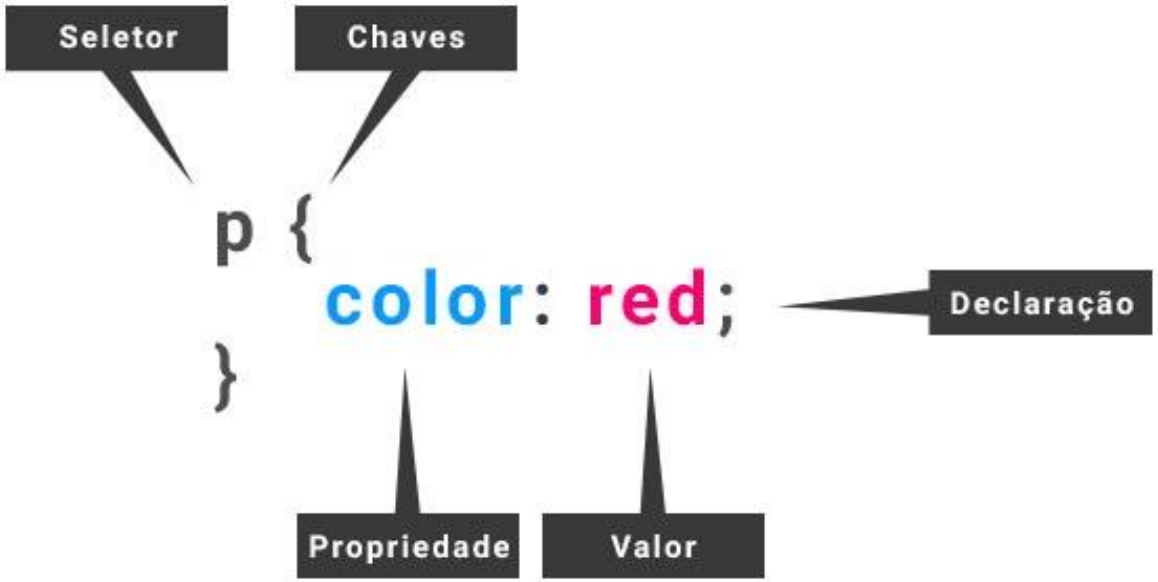
- **Folha de estilos externa**
  - O CSS é escrito em um arquivo separado com uma extensão **.css**, sendo sua ligação realizada dentro de um elemento **<link>** no documento HTML

# Sintaxe

Os blocos básicos de construção do CSS são:

- **Propriedade**: identificador que possui um nome legível e define o que será considerado ou editado. Cada propriedade possui um conjunto de valores válidos, definida por uma gramática formal, bem como um significado semântico

- **Valor**: descreve como o recurso será tratado pelo mecanismo de renderização do navegador



# Sintaxe

**Declaração CSS (propriedade + valor)**
propriedade: **color**
valor: **blue**

**Seletor**

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

**Bloco de declarações CSS**

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

**Regra CSS (seletor + bloco de declarações)**

# Seletores

- Um seletor é o modo pelo qual é possível apontar para algum elemento no documento HTML aplicando estilos à esse elemento

- Cada regra CSS inicia com um seletor ou uma lista de seletores que informam ao navegador em quais elementos estas regras deverão ser aplicadas

- Alguns exemplos de regras:

```
h1

a:link

.manythings

#onething

*

.box p

.box p:first-child

h1, h2, .intro
```

# Seletores

- Um seletor CSS é a primeira parte de uma regra CSS
- É um padrão de elementos e outros termos que informam ao navegador quais elementos HTML devem ser selecionados para que os valores de propriedade CSS dentro da regra sejam aplicados a eles
- O elemento ou elementos que são selecionados pelo seletor são referidos como o assunto do seletor

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

# Tipos de seletores

- **Seletores de tipo**

(https://developer.mozilla.org/en-US/docs/Web/CSS/Type_selectors)

```
h1 { }
p { }
```

- **Seletores de classe**

(https://developer.mozilla.org/pt-BR/docs/Web/CSS/Class_selectors)

```
. column
. row
```

- **Seletores de id**

(https://developer.mozilla.org/pt-BR/docs/Web/CSS/ID_selectors)

```
# menu
# unique
```

- **Seletores de atributos**

(https://developer.mozilla.org/pt-BR/docs/Web/CSS/Attribute_selectors)

```
a[title] { }
a[href="https://example.com"] { }
```

# Tipos de seletores

- **Pseudo-elementos** (https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-elements)

  Selecionam uma determinada parte de um elemento em vez do próprio elemento. No exemplo a seguir é realizada uma seleção em que somente a primeira linha de texto dentro de um elemento será afetada: `p::first-line { }`

- **Pseudo-classes** (https://developer.mozilla.org/pt-BR/docs/Web/CSS/Pseudo-classes)

  Diferentemente dos pseudo-elementos, pseudo-classes podem ser utilizadas para estilizar um elemento baseado em seu estado

  Definem o estilo de certos estados de um elemento. No exemplo a seguir, é realizada uma seleção que afetará o elemento `<a>` somente quando o ponteiro do mouse for passado sobre ele: `a:hover { }`

- **Combinadores** (https://developer.mozilla.org/pt-BR/docs/Web/CSS/CSS_Selectors#combinadores)

  Selecionam os nós que seguem (não necessariamente imediatamente) o elemento especificado anteriormente. No exemplo a seguir serão selecionados os parágrafos que são filhos diretos de `<article>` utilizando o combinador filho `(>)`: `article > p { }`

# Especificidade CSS

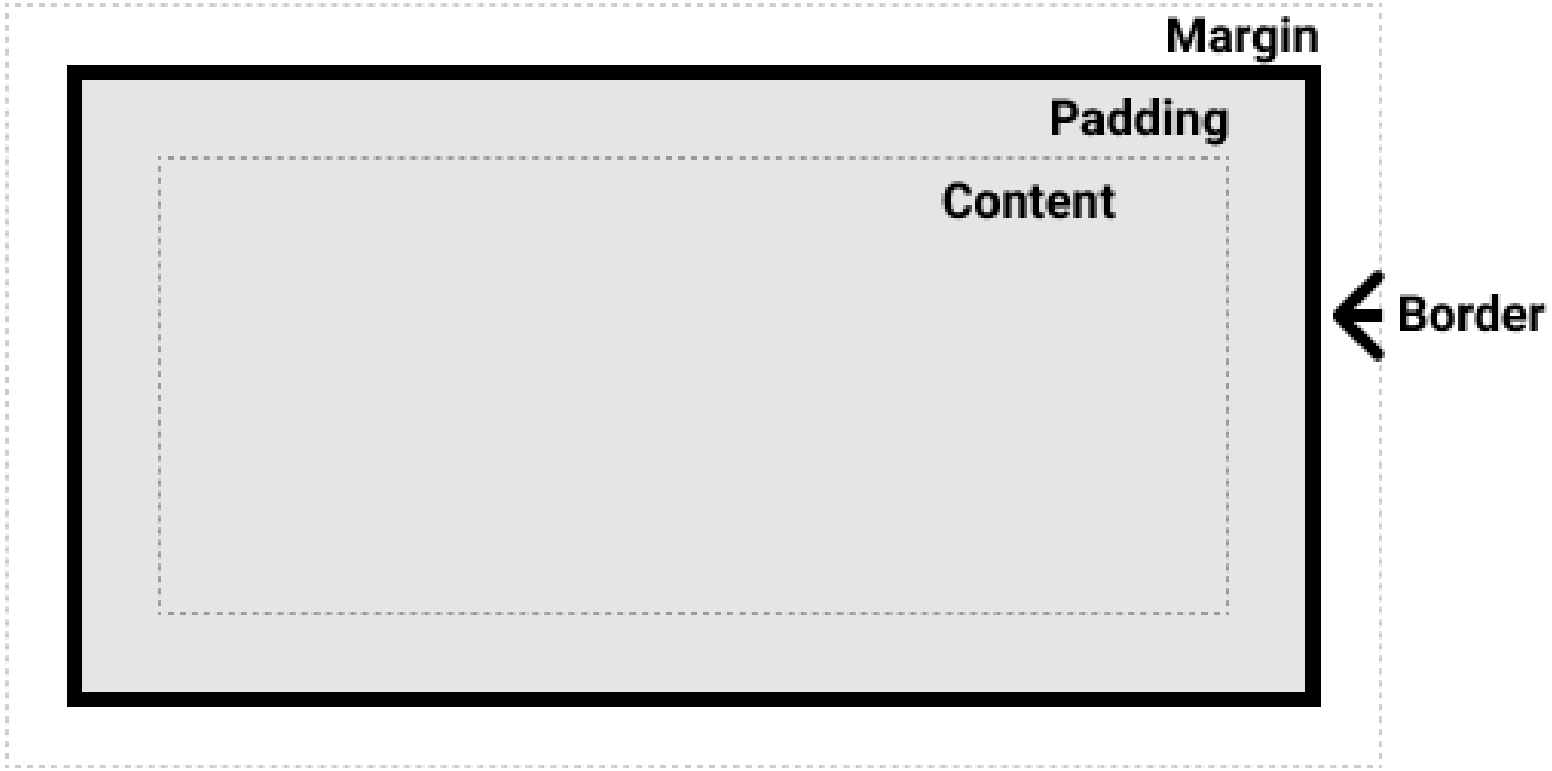
- Pode ser compreendida como se fosse um sistema de pesos que serve para determinar qual regra CSS tem precedência quando várias regras podem ser aplicadas a um mesmo elemento
- A especificidade apresenta um peso maior para o menor de acordo com o seguinte ordem:
  1. Estilos inline
  2. IDs
  3. Classes, pseudoclasses e atributos
  4. Elementos e pseudoelementos

# Eficiência de Seletores CSS

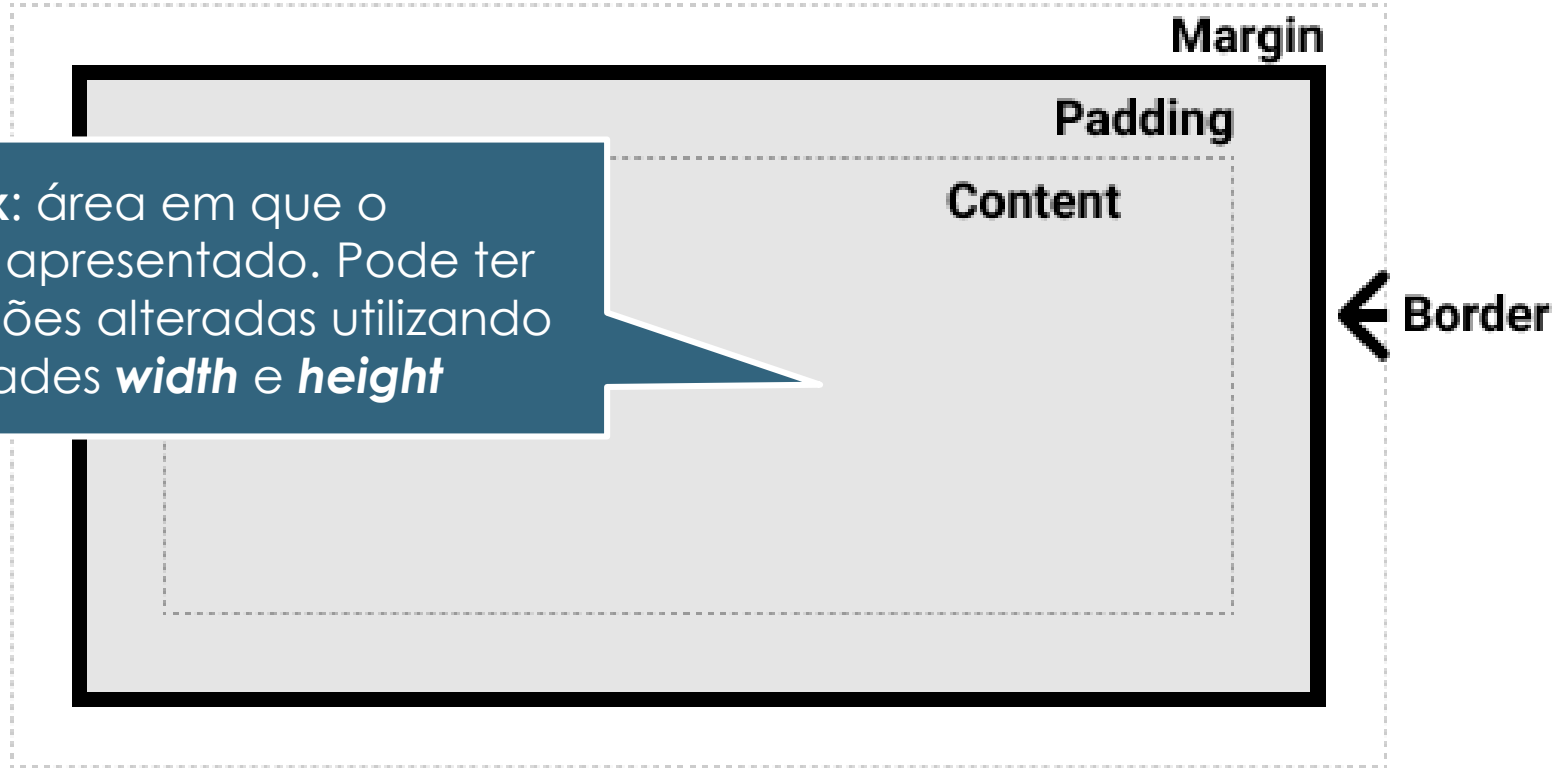
- A eficiência dos seletores CSS segue a seguinte ordem da maior para a menor eficiência:

```
#header { }          /* ID */
.promo { }           /* Classe */
div { }              /* Tipo */
h2 + p { }           /* Irmão adjacente */
li > ul { }          /* Filho */
ul a { }             /* Descendente */
* { }                /* Universal */
[type="text"] { }    /* Atributo */
a:hover { }          /* Pseudo-classe/Pseudo-elemento */
```

# CSS Box



# CSS Box

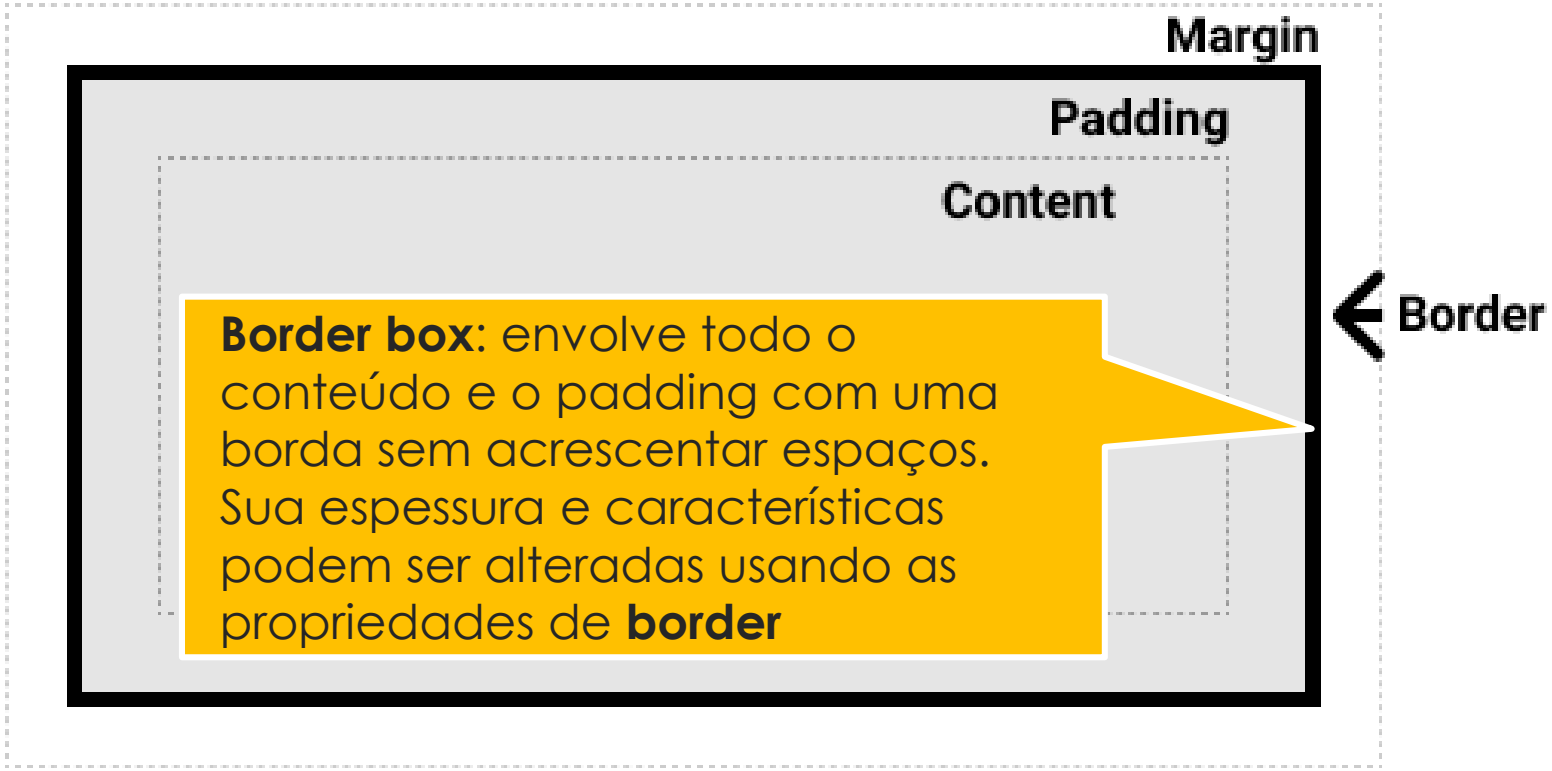



**Content box**: área em que o elemento é apresentado. Pode ter suas dimensões alteradas utilizando as propriedades ***width*** e ***height***

# CSS Box

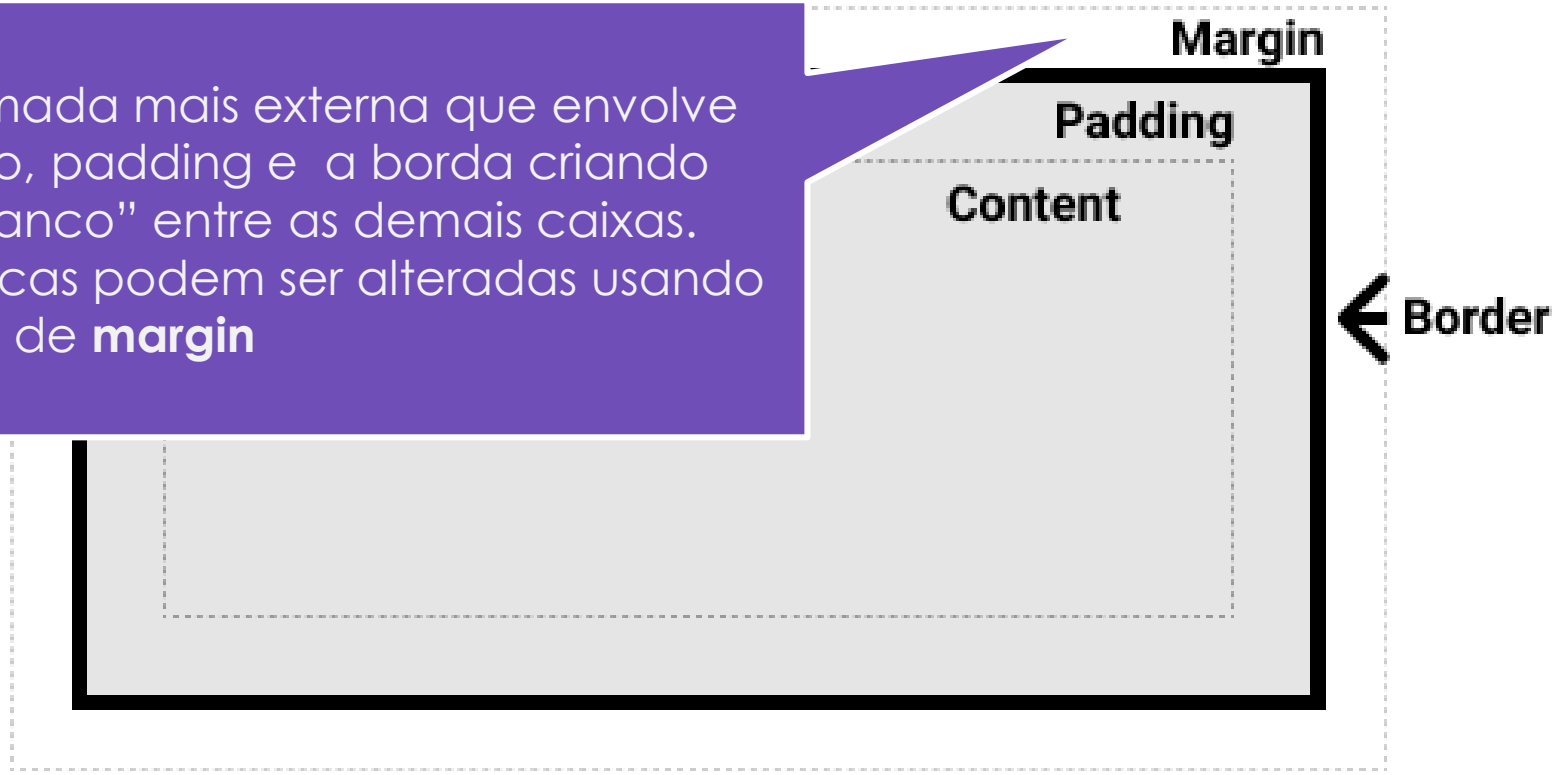
**Padding box**: área de preenchimento ao redor do conteúdo. Os espaços de preenchimento são controlados pelas propriedades de **padding**

# CSS Box


Margin
Padding
Content
Border
**Border box**: envolve todo o conteúdo e o padding com uma borda sem acrescentar espaços. Sua espessura e características podem ser alteradas usando as propriedades de **border**

# CSS Box

**Margin box**: camada mais externa que envolve todo o conteúdo, padding e  a borda criando "espaços em branco" entre as demais caixas. Suas características podem ser alteradas usando as propriedades de **margin**



# CSS Box – Padrão

Considerando o exemplo ao lado:

**Largura total:** **410px** (350 + 25 + 25 + 5 + 5)

**Altura total:** **210px** (150 + 25 + 25 + 5 + 5)


Como pode ser observado acima, as dimensões do padding e a espessura da borda não são contabilizadas na largura e altura totais da caixa. Entretanto esses valores afetam o espaço total que será ocupado na página por essa caixa

**Os navegadores utilizam este comportamento do CSS Box por padrão**

```css
.box {
  width: 350px;
  height: 150px;
  margin: 10px;
  padding: 25px;
  border: 5px solid black;
}
```

# CSS Box – Alternativo


- O CSS Box alternativo foi introduzido como uma opção para obtenção do tamanho real da caixa, conforme figura ao lado

- Utilizando o mesmo CSS mas incluindo a propriedade ***box-sizing: border-box***, é possível se obter as dimensões reais da caixa

- Para que todos os elementos da página utilizem esse método alternativo, a forma recomendada é a seguinte:

```
.box {
  box-sizing: border-box;
}
```

```
.box {
  width: 350px;
  height: 150px;
  margin: 10px;
  padding: 25px;
  border: 5px solid black;
}
```

```
html {
    box-sizing: border-box;
 }
 *, *::before, *::after {
    box-sizing: inherit;
 }
```

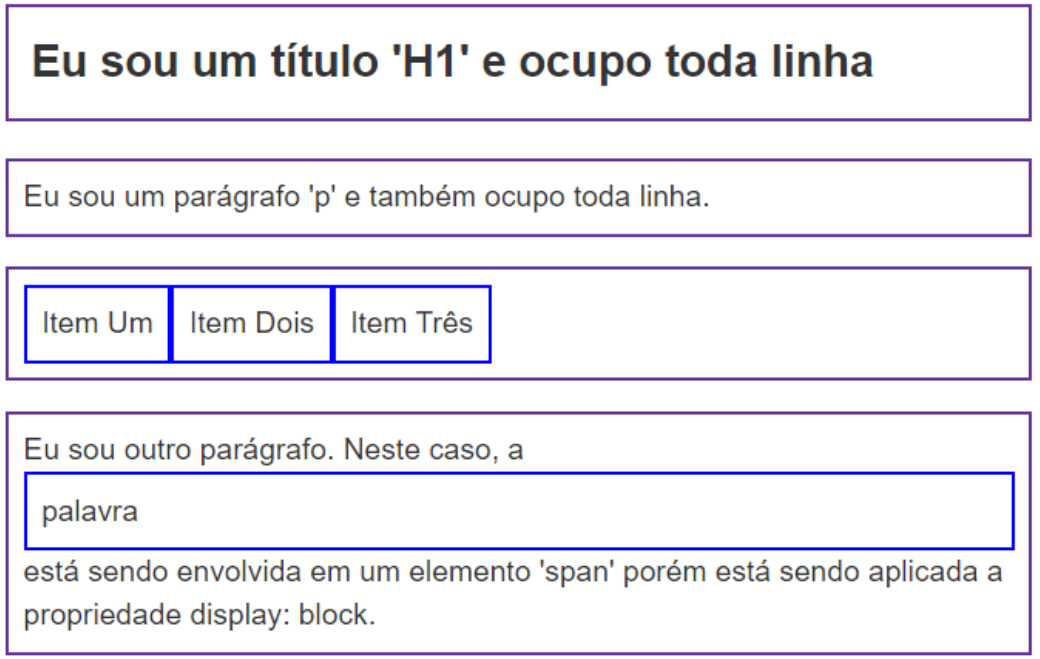
# CSS Box Model

Tudo em CSS são quadros que podem ser organizados tanto para criar layouts quanto para alinhar itens entre si constituindo dessa forma o modelo de caixas CSS (***CSS Box Model***)

Existem dois tipos principais de caixas: **block** e **inline**. A definição destas características afeta tanto o fluxo de exibição quanto a relação entre os blocos em uma página

O tipo de caixa a ser aplicado em elemento é definido pelos valores da propriedade ***display***:

- **display: block**
- **display: inline**



# Block Boxes

- Elementos como **<H1>** e **<p>** são do tipo **display: block** por padrão
- A caixa será quebrada em uma nova linha
- Cada caixa será sempre estendida na direção da linha para preencher todo o espaço disponível
- As propriedades ***width*** e ***height*** são respeitadas
- ***Padding***, ***margin*** e ***border*** podem ser aplicados sendo capazes de empurrar outros elementos pra fora da caixa

# Inline Boxes

- Elementos como **<a>**, ****, **** e **** são do tipo **display: inline** por padrão
- A caixa **não** será quebrada em uma nova linha
- As propriedades ***width*** e ***height*** não são respeitadas
- ***Padding***, ***margin*** e ***border*** **(na vertical)** podem ser aplicados mas não são capazes de empurrar outros elementos pra fora da caixa
- ***Padding***, ***margin*** e ***border*** **(na horizontal)** podem ser aplicados sendo capazes de empurrar outros elementos pra fora da caixa

# CSS – Unidades absolutas de medida

| Unit | Name | Equivalent to |
|---|---|---|
| `cm` | Centimeters | 1cm = 38px = 25/64in |
| `mm` | Millimeters | 1mm = 1/10th of 1cm |
| `Q` | Quarter-millimeters | 1Q = 1/40th of 1cm |
| `in` | Inches | 1in = 2.54cm = 96px |
| `pc` | Picas | 1pc = 1/6th of 1in |
| `pt` | Points | 1pt = 1/72th of 1in |
| `px` | Pixels | 1px = 1/96th of 1in |

# CSS – Unidades relativas de medida

| Unit | Relative to |
| --- | --- |
| `em` | Font size of the parent, in the case of typographical properties like `font-size`, and font size of the element itself, in the case of other properties like `width`. |
| `ex` | x-height of the element's font. |
| `ch` | The advance measure (width) of the glyph "0" of the element's font. |
| `rem` | Font size of the root element. |
| `lh` | Line height of the element. |
| `vw` | 1% of the viewport's width. |
| `vh` | 1% of the viewport's height. |
| `vmin` | 1% of the viewport's smaller dimension. |
| `vmax` | 1% of the viewport's larger dimension. |

# Unidades: em e rem

- Dentre as diversas unidades relativas de medida frequentemente utilizadas para dimensionamento de textos no CSS tem-se:
  - ***em*** *(my parent element's font-size)* e
  - ***rem*** *(the root element's font-size)*
- O valor base da altura de fontes utilizado pelos navegadores é **16px**

Dica!

```
html{
    font-size: 62.5%;
}

h1{
    font-size: 2.4rem;  /*equivalente a 24px*/
}

p{
    font-size: 1.2rem; /*equivalente a 12px*/
}
```

```
10px = 0.625rem
12px = 0.75rem
14px = 0.875rem
16px = 1rem
18px = 1.125rem
```

# Variáveis CSS

- Variáveis CSS são entidades definidas por desenvolvedores, para conter valores específicos a serem reutilizados ao longo do documento
- São configuradas usando a notação `--` como por exemplo:

```
--cor-principal: black;
```

- Podem ser acessadas usando a função `var()` como por exemplo:

```
color: var(--cor-principal);
```

# CSS Position

- **Positiong** permite alterar o fluxo normal de apresentação de um elemento no documento como por exemplo:
  - Posicionar um elemento no topo de outro elemento
  - Manter um determinado elemento em uma posição fixa na tela do navegador
  - Etc.

- Existem diferentes tipos de posicionamento que podem ser alterados utilizando a propriedade: **position**

# CSS Position

- **position: static**
  - comportamento padrão. Posiciona cada elemento no fluxo normal de exibição do documento
- **position: relative**
  - em conjunto com as propriedades ***Top***, ***Bottom***, ***Left*** e ***Right*** alteram o posicionamento do elemento. Todos os deslocamentos são realizados à partir da posição inicial do elemento no fluxo normal
- **position: absolute**
  - Permite posicionar qualquer elemento **de acordo com o elemento pai** que tenha um position diferente de ***static***
- **position: fixed**
  - Comporta-se de forma similar ao ***absolute***, deixando de fazer parte do fluxo comum da página. Entretanto a grande diferença é que o fixed passa a se referenciar ao viewport do navegador independentemente de barra de rolagem

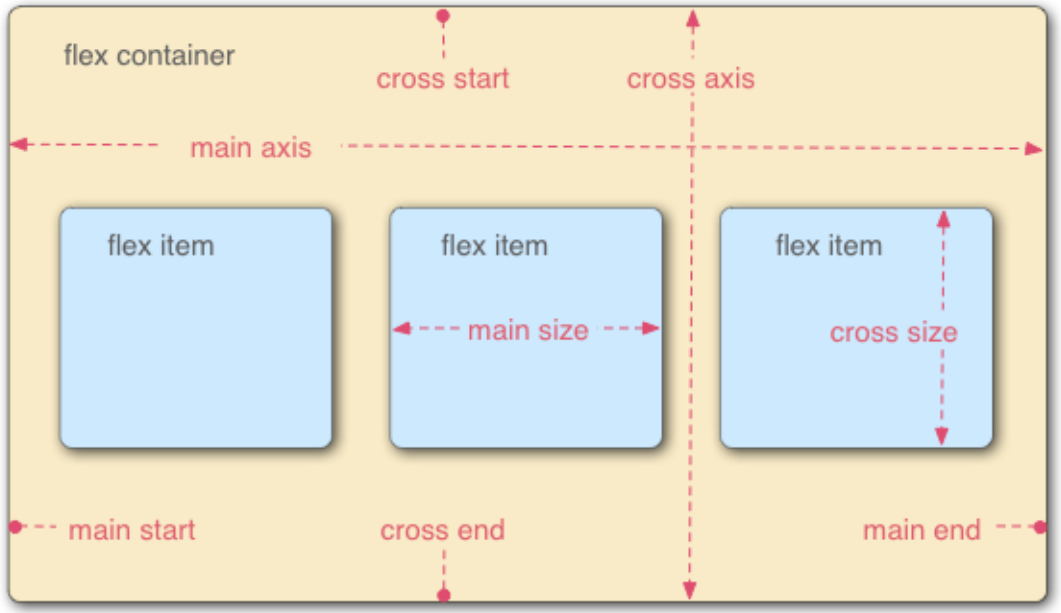
# Z-index

- A propriedade z-index é utilizada para alterar o posicionamento da profundidade dos elementos no documento
- Trabalha no eixo de coordenadas Z para determinar qual elemento estará mais próximo ou mais afastado da tela

# Flexbox

- Por um longo tempo, **floats** e **position** foram as únicas propriedades disponíveis no CSS, compatíveis entre browsers, para criação de layouts
- Apesar de flexíveis e muito utilizadas, em alguns casos apresentavam limitações
- O **Flexbox** foi então desenvolvido para preencher essas lacunas e resolver alguns problemas de layouts como por exemplo:
  - Centralizar um bloco de conteúdo verticalmente dentro de seu pai
  - Fazer com que os filhos de um container ocupem uma quantidade igual de largura/altura disponível, independentemente da quantidade de largura/altura disponível
  - Fazer com que todas as colunas de um layout com múltiplas colunas mantenham a mesma altura, mesmo que contenham conteúdos de tamanhos diferentes

# Flexbox



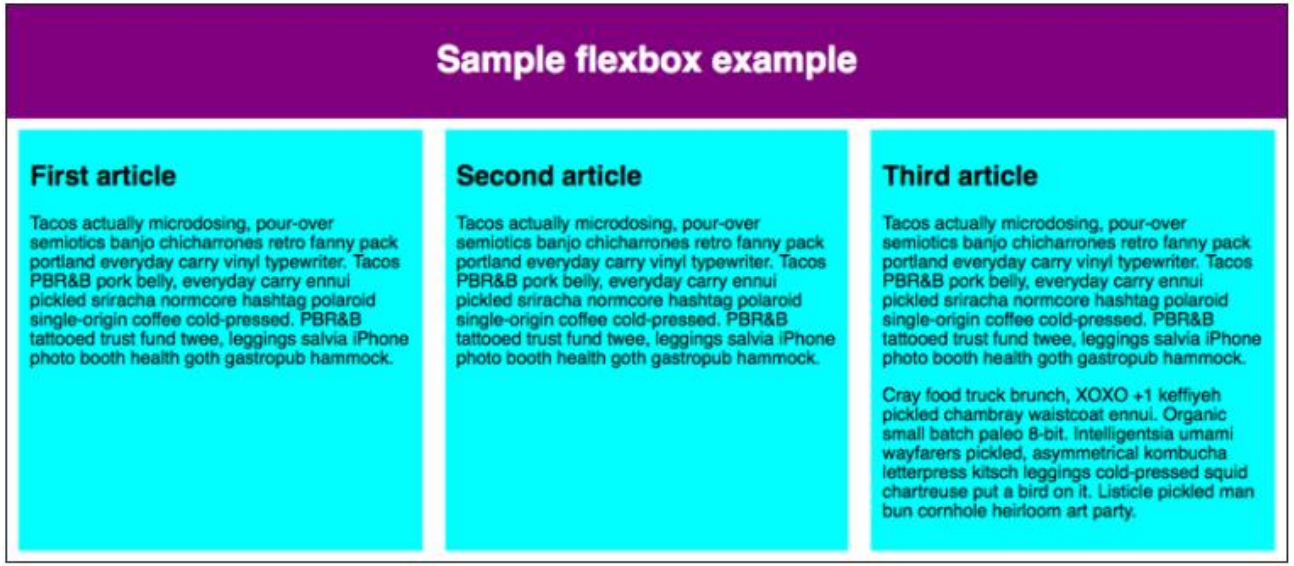

- **main axis:** eixo que corre na direção em que os flex items estão dispostos. O início e o fim do eixo são chamados: **main start** e **main end**
- **cross axis:** é o eixo perpendicular que corre na direção em que os flex items são dispostos**.** O início e o fim deste eixo são chamados: **cross start** e **cross end**
- **flex container:** elemento pai que possui a propriedade **display: flex** configurada
- **flex items**: itens iniciados como flexible boxes dentro do flex container

# Flexbox

- Para que um elemento se transforme em um ***Flexbox***, ou seja, uma **caixa flexível** é necessário inicialmente definir um valor especial de ***display*** no elemento pai dos elementos que se pretende que sejam alterados

```
section {
  display: flex;
}
```

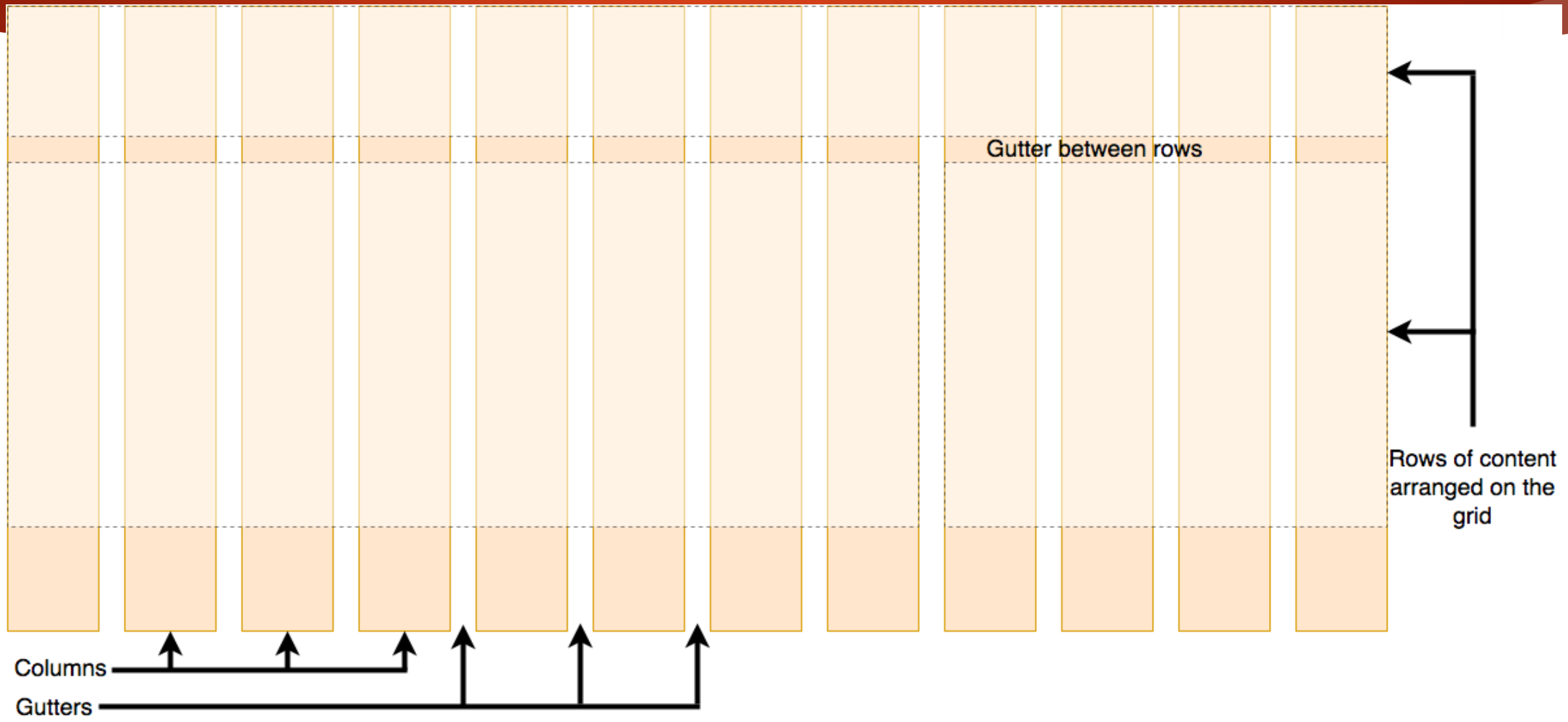
- No exemplo a seguir o objetivo é afetar os três elementos ***<article>.*** Para isso é necessário que o elemento pai ***<section>*** se transforme em um ***flex container***:



# Flexbox – Principais Propriedades

- display: flex
- flex-direction
- flex-wrap
- flex-flow
- flex-grow
- flex-basis
- align-items
- justify-content
- order

# Grids


Gutter between rows
Rows of content arranged on the grid
Columns
Gutters

# Grids

- Grids (ou grades) são coleções de **linhas horizontais e verticais** que constituem um padrão com objetivo de alinhar os elementos de design

- Os grids possibilitam a criação de designs mais consistentes em que os elementos não saltam ou alteram suas larguras durante a navegação entre páginas

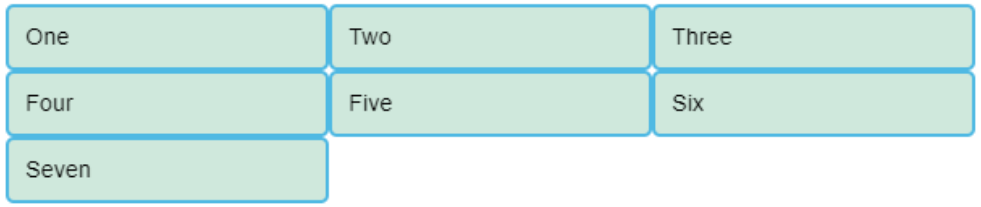
- Um grid é composto basicamente de linhas e colunas com gaps ou espaçamentos entre elas, comumente chamados de **gutters**

# Grids

- Para que um elemento se transforme em um ***Grid***, é necessário inicialmente definir um valor especial de ***display*** e além disso, a largura para as colunas

```
.container {
    display: grid;
    grid-template-columns: 200px 200px 200px;
}
```

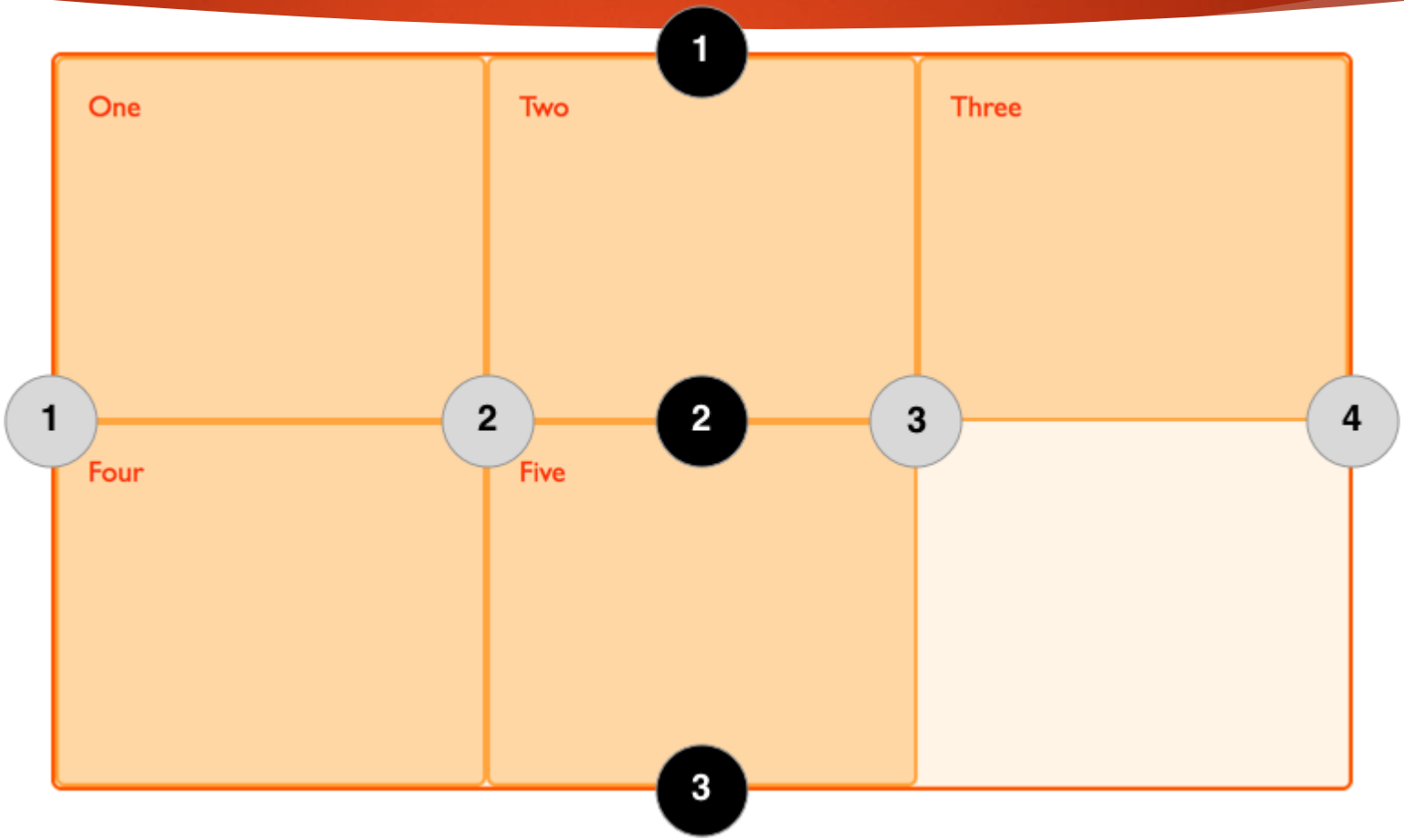
- Para definição das larguras das colunas é possível utilizar qualquer unidade de medida



# Grids – Posicionamento baseado em linha

- As linhas de um grid iniciam em 1 e estão relacionadas ao modo de escrita do documento
- Em idiomas como português e inglês o sentido da leitura se inicia na linha 1 e coluna 1, diferentemente de idiomas como o árabe em que o sentido da escrita é realizado da direita para esquerda
- Para que seja possíel alterar essas características podem ser utilizadas as seguintes propriedades:
  - grid-column-start
  - grid-column-end
  - grid-row-start
  - grid-row-end
- Como atalho, estas propriedades podem ser substituídas por números

# Grids – Posicionamento baseado em linha



# Grid Template Areas

- Uma forma alternativa de distribuir itens em um grid é usar a propriedade ***grid-template-areas*** dando nomes aos vários elementos do design
- Regras para utilização:
  - Todas as células do grid precisam ser preenchidas
  - Para abranger (mesclar) duas células, o nome deve ser repetido
  - Para deixar uma célula vazia, use o caractere "." (ponto)
  - As áreas devem ser retangulares não sendo possível a criação de áreas em forma de L, por exemplo.
  - As áreas não podem ser repetidas em locais diferentes.

# CSS Media Queries

- Media queries são consultas detectar diferentes resoluções de tela fornecendo um visual diferente de acordo com cada resolução detectada
- Definem condições para utilização de estilos CSS
- Caso o dispositivo de acesso do usuário esteja de acordo com as condições determinadas nas consultas, estilos são aplicados
- Surgem então os conceitos de **breakpoints** responsáveis por definir a partir de qual ponto serão aplicados os estilos CSS

# CSS Media Queries

**CSS:**

```css
/* Estilos aplicados a larguras até 480 pixels */

body {
    background-color: #fc344c;
    margin: 0;
    padding: 0;
    font-family: sans-serif;
    font-size: 1.5em;
    color: #fff;
}

/* Estilos aplicados a larguras acima 480 pixels */

@media screen and (min-width: 481px) {
    body {
        background-color: #05a4e7;
        font-size: 2em;
    }
}
```

**HTML:**

```html
<!DOCTYPE html>
<html lang="pt-br">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Testando Media Queries</title>
        <link rel="stylesheet" href="testando-media-queries.css">
    </head>
    <body>
    Se você estiver vendo esta página em um dispositivo com resokução superior a 480 pixels a cor de fundo dela é azul. Entretanto se você estiver vendo em um dispositivo com resolução igual ou menor que 480 pixels a cor de fundo é vermelha.
    </body>
</html>
```