Pós-Graduação Lato Sensu

Curso de Especialização em Tecnologia Java

Frameworks Web

CSS

Prof. Esp. Hugo Baker Goveia

1. O que é CSS?

CSS significa Cascading Style Sheets (Folhas de Estilo em Cascata) e é uma linguagem de estilo usada para descrever a apresentação visual de um documento HTML. Em outras palavras, o CSS é responsável por controlar a aparência e o layout dos elementos HTML em uma página da web.

Separação de Conteúdo e Estilo
Uma das principais vantagens do CSS é a capacidade de separar o conteúdo da página que seria o HTML, da apresentação, que seria o estilo dela.
Isso permite que os desenvolvedores apliquem estilos consistentes em todo o site, alterando apenas um arquivo CSS, sem precisar modificar o HTML.

Seletor e Propriedades
O CSS funciona com base em seletores e propriedades.
Os **seletores** são usados para identificar os elementos HTML aos quais o estilo será aplicado, enquanto as **propriedades** são usadas para definir como esses elementos serão estilizados.

Cascata e Especificidade
A "Cascata" em Cascading Style Sheets refere-se à maneira como as regras de estilo são aplicadas e priorizadas. Quando várias regras conflitam, a especificidade das regras e a ordem de declaração determinam qual estilo será aplicado.

Reutilização de Estilos
O CSS permite a reutilização eficiente de estilos em todo o site, usando classes e IDs para aplicar estilos consistentes a diferentes elementos.
Isso promove uma manutenção fácil e consistência visual em todo o site.

Responsividade e Layouts Flexíveis
Com CSS, os desenvolvedores podem criar layouts responsivos e flexíveis que se adaptam a diferentes tamanhos de tela e dispositivos. Isso é essencial para garantir uma experiência de usuário consistente em dispositivos móveis, tablets e desktops.

O CSS desempenha um papel crucial no design e na aparência de uma página da web, permitindo que os desenvolvedores controlem cores, fontes, espaçamento, layout e outros aspectos visuais para criar uma experiência de usuário atraente e intuitiva.

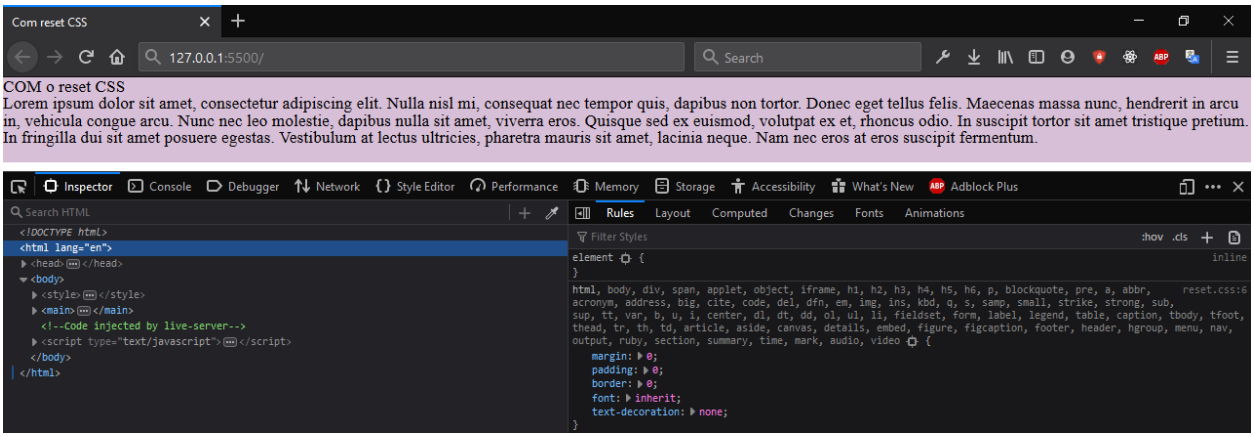
2. Reset CSS

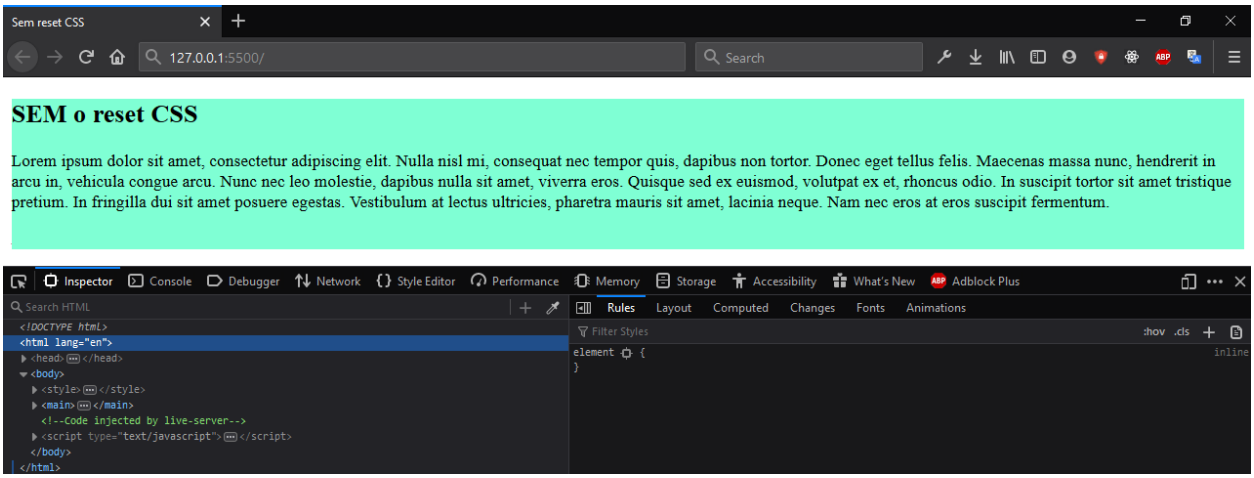
Ao desenvolver uma página da web, é comum encontrar inconsistências na aparência e no comportamento entre diferentes navegadores. Isso ocorre porque cada navegador possui suas próprias configurações de estilo padrão, o que pode resultar em elementos renderizados de maneira diferente, mesmo que o código HTML seja o mesmo. Para resolver esse problema e garantir uma experiência consistente para todos os usuários, os desenvolvedores recorrem ao uso de ferramentas como o Reset CSS e Normalize CSS.

O Reset CSS e Normalize CSS são arquivos CSS pré-construídos que visam criar uma base comum de estilos para todos os elementos HTML, removendo ou substituindo as configurações de estilo padrão dos navegadores. Isso permite que os desenvolvedores comecem de uma posição mais consistente ao estilizar sua página da web, reduzindo a necessidade de lidar com inconsistências entre navegadores.

Reset CSS

O Reset CSS é uma abordagem radical que redefine todos os estilos padrão dos elementos HTML, zerando todos os valores de margem, preenchimento, borda e outros atributos. Isso cria uma "tela limpa" na qual os desenvolvedores podem começar a aplicar seus estilos personalizados sem interferência dos estilos padrão do navegador.





Exemplo de arquivo reset.css

```css
/* Reset CSS */
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
```

Normalize CSS

Por outro lado, o Normalize CSS adota uma abordagem mais suave, mantendo alguns estilos padrão dos navegadores, mas garantindo que eles sejam consistentes em todos os navegadores. Em vez de zerar todos os estilos, o Normalize CSS aplica estilos padronizados para garantir que todos os elementos HTML sejam renderizados de maneira uniforme.

Como implementar o Normalize?

Em seu projeto HTML, dentro da tag head, utilize a tag link, para chamar o arquivo normalize.css, como no exemplo**: <link rel="stylesheet" href="normalize.css">**

Ele deve ser o primeiro arquivo CSS a ser importado no projeto. É importante “limpar” primeiro o projeto antes de começar a desenvolver.

Confira abaixo, as principais diferenças entre Reset CSS e Normalize:

- O Reset CSS limpa TODOS os padrões dos navegadores. É uma forma muito agressiva de “limpeza”, enquanto o Normalize ainda mantém padrões que são úteis.
- O Normalize CSS corrige alguns bugs que existem nas estilizações padrões dos navegadores.
- O Normalize é modular. O código todo é separado por seções específicas, então você pode procurar por mudanças de forma mais direta.
- O Normalize tem uma documentação detalhada sobre o uso e justificativas das mudanças feitas por ele.

3. Atributo STYLE (estilo inline)

O atributo style é uma ferramenta poderosa no arsenal de um desenvolvedor web, permitindo a aplicação de estilos diretamente a elementos HTML individualmente.
Esse método de estilização é conhecido como **"estilo inline"** e oferece uma maneira rápida e direta de personalizar a aparência de elementos específicos em uma página da web.

O estilo inline é definido dentro da própria tag HTML do elemento que você deseja estilizar. O atributo style é adicionado à tag e contém uma lista de propriedades de estilo CSS, cada uma com seu valor correspondente.

** **Este é um parágrafo estilizado inline** **
Neste exemplo, a cor do texto é definida como azul e o tamanho da fonte é definido como 16 pixels diretamente na tag ** **. Esses estilos serão aplicados apenas a este parágrafo específico, sem afetar outros elementos na página.

<h1 style="color: red;">Título em Vermelho**</h1>**
** **Parágrafo em Verde** **
Nos exemplos acima, tanto o <h1> quanto o estão sofrendo alterações de cores.

** **Este texto possui uma fonte maior.** **
No exemplo acima o conteúdo da está recebendo uma definição de tamanho de fonte para 20px.

****Este texto possui um fundo amarelo.****
No exemplo acima está sendo feita aplicação de cor de fundo.

Vantagens

- **Simplicidade:** O estilo inline é fácil de implementar e entender, especialmente para iniciantes.
- **Especificidade:** Os estilos inline têm alta especificidade, o que significa que eles podem substituir estilos definidos em arquivos CSS externos ou incorporados.

Desvantagens

- **Dificuldade de Manutenção:** O estilo inline pode tornar o código HTML mais difícil de manter, especialmente em páginas com muitos estilos aplicados dessa maneira.
- **Falta de Reutilização:** Os estilos inline não podem ser reutilizados em outros elementos, o que pode resultar em código redundante e inchado.

Podemos concluir então que o estilo inline oferece uma maneira rápida e direta de aplicar estilos a elementos HTML específicos, mas deve ser usado com moderação para evitar problemas de manutenção e reutilização.

4. Estilo Incorporado

O estilo incorporado é uma técnica de estilização em que as regras CSS são definidas na própria página HTML, dentro da tag **<style>** no cabeçalho do documento.
Esse método oferece uma maneira conveniente de aplicar estilos a elementos específicos ou a toda a página, mantendo-os separados do conteúdo HTML, mas ainda dentro do mesmo arquivo.

Para aplicar estilos incorporados a uma página HTML, os desenvolvedores podem usar a tag <style> no cabeçalho do documento. Dentro desta tag, as regras CSS são escritas da mesma maneira que em um arquivo CSS externo, utilizando seletores e propriedades de estilo.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de CSS Incorporado</title>
    <style>
        h1 {
            color: blue;
            font-size: 24px;
        }
        p {
            color: green;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <h1>Título Estilizado USANDO O ESTILO Incorporado</h1>
    <p>Parágrafo Estilizado USANDO O ESTILO Incorporado</p>
</body>
</html>
```

O exemplo acima demonstra a implementação do estilo incorporado.
E o resultado segue na imagem abaixo:

Neste exemplo acima, os estilos para os elementos **<h1>** e **<p>** são definidos dentro da tag **<style>** no cabeçalho do documento HTML. Esses estilos serão aplicados apenas a esses elementos específicos na página.

```
<style>
  h2 {
      color: red;
      font-size: 20px;
  }
</style>
```

No exemplo acima, é aplicado uma estilização ao título h2 de uma página HTML, mudando a cor para vermelho e o tamanho para 20px.

```
<style>
  a {
      color: blue;
      text-decoration: none;
  }
  a:hover {
      text-decoration: underline;
  }
</style>
```

No exemplo acima é aplicado estilo aos links, de forma que o link fique na cor azul e remove o sublinhado do link.
E quando o link entrar em estado de **hover**, que é quando o mouse está sobre o link, ele deverá colocar o sublinhado no link, pois está definindo o text-decoration para **underline**

Vantagens

- **Simplicidade:** O estilo incorporado é fácil de implementar e entender, especialmente para pequenas páginas ou protótipos.
- **Escopo Local:** Os estilos incorporados são aplicados apenas à página específica em que estão definidos, evitando conflitos com estilos de outras páginas.

Desvantagens

- **Dificuldade de Manutenção:** Assim como o estilo inline, o estilo incorporado pode tornar o código HTML mais difícil de manter, especialmente em páginas com muitos estilos aplicados dessa maneira.
- **Falta de Reutilização:** Os estilos incorporados não podem ser facilmente reutilizados em outras páginas, o que pode resultar em código redundante e inchado.

Sendo assim, o estilo incorporado oferece uma maneira conveniente de aplicar estilos a elementos HTML dentro da própria página, mantendo-os separados do conteúdo e permitindo uma estilização mais granular.

5. Estilo com arquivo externo

Ao criar páginas da web com estilos mais complexos e extensos, é comum utilizar arquivos CSS externos para organizar e manter o código de estilo de forma mais eficiente.

Em vez de incluir estilos diretamente no documento HTML, os desenvolvedores podem criar um arquivo separado com extensão **.css** contendo todas as regras de estilo para a página. Esse arquivo CSS pode ser vinculado ao documento HTML usando a tag **<link>** no cabeçalho do documento.

```html
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo de Estilo com Arquivo Externo</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Título Estilizado com Arquivo Externo</h1>
    <p>Parágrafo Estilizado com Arquivo Externo</p>
</body>
</html>
```

Neste exemplo acima, o arquivo de estilo **styles.css** está vinculado ao documento HTML usando a tag **<link>.** Todas as regras de estilo contidas nesse arquivo serão aplicadas à página HTML, permitindo uma organização mais clara e uma manutenção mais fácil dos estilos.

Vantagens do Estilo com Arquivo Externo:

- **Organização do Código:** Manter os estilos em um arquivo CSS separado ajuda a manter o código HTML limpo e focado no conteúdo, enquanto o código de estilo é mantido em um local separado e organizado.
- **Reutilização de Estilos:** Arquivos CSS externos podem ser facilmente reutilizados em várias páginas da web, promovendo consistência de estilo em todo o site e evitando a duplicação de código.
- **Facilidade de Manutenção:** Ao centralizar todos os estilos em um arquivo externo, as atualizações e modificações podem ser feitas de forma rápida e eficiente em um único local, em vez de ter que editar cada página individualmente.
- **Cache do Navegador:** Os navegadores podem armazenar em cache arquivos CSS externos, o que pode resultar em tempos de carregamento mais rápidos para páginas subsequentes do mesmo site.

O estilo com arquivo externo é uma prática recomendada para projetos de desenvolvimento web, pois promove a organização, reutilização e manutenção eficiente dos estilos.

6. Seletores de Tipo, Classe e ID

Os seletores CSS desempenham um papel fundamental na aplicação de estilos a elementos HTML específicos. Entre os mais comuns estão os seletores de **tipo**, **classe** e **ID**, cada um com sua própria sintaxe e aplicação.

Seletores de Tipo

Esses seletores aplicam estilos a todos os elementos de um determinado tipo na página. Eles são representados pelo nome do elemento HTML.

Exemplo:

```css
p {
  color: blue;
}
```

Neste exemplo acima, todos os parágrafos ** ** terão sua cor definida como azul.

Seletores de Classe

Os seletores de classe são usados para aplicar estilos a elementos que possuem uma classe específica atribuída. Eles são representados por um ponto seguido pelo nome da classe.

```css
.destaque {
  font-weight: bold;
}
```

No exemplo acima, todos os elementos com a classe **destaque** terão seu peso da fonte definido como negrito.

Como por exemplo aqui: ** **Este parágrafo está em destaque** **

Seletores de ID

Os seletores de ID aplicam estilos a um elemento HTML específico que possui um ID único atribuído. Eles são representados por uma hashtag seguida pelo nome do ID.

```css
#cabecalho {
  background-color: #f0f0f0;
}
```

No exemplo acima, o elemento com o **ID cabecalho** terá sua cor de fundo definida como cinza claro.

** **Conteúdo do Cabeçalho** **

Os seletores de **tipo**, **classe** e **ID** são ferramentas essenciais para estilizar elementos HTML de forma precisa e eficiente.

7. Agrupando Seletores

Agrupar seletores CSS é uma técnica que permite aplicar estilos a vários elementos de uma vez, simplificando o código e facilitando a manutenção.

Ao agrupar seletores CSS, você pode aplicar as mesmas regras de estilo a múltiplos elementos, reduzindo a redundância e melhorando a legibilidade do código. Essa técnica é útil quando você deseja aplicar estilos semelhantes a elementos diferentes ou quando deseja evitar repetições de código.

Dessa forma é possível agrupar seletores em CSS, fazendo que propriedades comuns sejam compartilhadas entre os mesmos.

No exemplo acima, o **ID** #tidulo e a **classe** .missao estão compartilhando a mesma configuração para fonte = Arial.
Já a classe CSS **introducao** e **missao** estão compartilhando a definição de cor vermelha.
E como introdução tinha uma propriedade única, que só ela iria receber, foi declarado embaixo novamente a classe **introducao** com a propriedade de font que ela possuía diferente dos demais.

Vantagens do Agrupamento de Seletores

- **Redução da Redundância:** Evita a repetição de estilos semelhantes em diferentes regras de estilo.
- **Facilidade de Manutenção:** Permite fazer alterações em vários elementos de uma vez, simplificando a manutenção do código CSS.
- **Melhor Legibilidade:** Agrupa estilos relacionados em um único bloco de código, facilitando a compreensão do que está sendo estilizado.

O agrupamento de seletores CSS é uma técnica muito boa para estilizar múltiplos elementos de forma eficiente e organizada.

8. Seletores Descendentes e de Filhos Diretos

Os seletores CSS permitem estilizar elementos HTML de maneira precisa e granular.
Os **seletores descendentes** e os **seletores de filhos diretos** são duas técnicas usadas para aplicar estilos a elementos aninhados dentro de outros elementos.

Seletores Descendentes

Os seletores descendentes permitem estilizar elementos que são descendentes de um elemento específico, independentemente do nível de aninhamento. Eles são representados pela combinação de seletores separados por um espaço.

```css
div p {
  color: blue;
}
```

No exemplo acima, todos os parágrafos **<p>** que estão dentro de elementos ** ** terão sua cor de texto definida como azul. E os outros parágrafos **<p>** da página não sofrerão essa alteração para cor azul.

Seletores de Filhos Diretos

Os seletores de filhos diretos são semelhantes aos seletores descendentes, mas eles estilizam apenas os elementos que são filhos diretos de um elemento específico, sem levar em consideração elementos aninhados mais profundamente. Eles são representados pela combinação de seletores separados por um sinal de maior **>**.

```css
div > p {
  text-transform: uppercase;
}
```

```html
<div>
  <p>Parágrafo dentro de uma div.</p>
  <span>
      <p>Parágrafo dentro de um span dentro de uma div.</p>
  </span>
</div>
```

Considerando o exemplo acima, apenas o parágrafo diretamente dentro da div terá seu texto transformado em maiúsculas, enquanto o parágrafo dentro do span não será afetado.

Vantagens dos Seletores Descendentes e Seletores de Filhos Diretos:

- **Precisão na Estilização:** Permitem aplicar estilos de forma precisa a elementos específicos dentro de uma estrutura HTML complexa.
- **Flexibilidade:** Podem ser combinados com outros seletores para criar regras de estilo mais refinadas e específicas.
- **Melhoria na Organização:** Facilitam a organização e a manutenção do código CSS, permitindo estilos específicos para elementos aninhados.

9. Pseudo-elementos e Pseudo-classes

Pseudo-elementos e pseudo-classes são recursos avançados do CSS que permitem aplicar estilos a **partes específicas** de um elemento HTML, proporcionando maior flexibilidade e controle sobre o design da página.

Pseudo-elementos

Os pseudo-elementos permitem estilizar partes específicas de um elemento HTML que não são representadas diretamente no código HTML. Eles são representados por dois pontos :: seguidos do nome do pseudo-elemento.

```
p::first-line {
  font-weight: bold;
}
```

No exemplo acima, a primeiro linha de todos os parágrafos terá seu peso de fonte definido como negrito.

Considerando o exemplo dos parágrafos abaixo:
A Lorem ipsum dolor sit amet, consectetur adipiscing elit.
O Lorem ipsum dolor sit amet, consectetur adipiscing elit.
E Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Se tivermos o Código CSS abaixo, iremos aplicar uma alteração no tamanho da fonte de todas as primeiras letras, de todos os parágrafos acima, que seriam as letras ‘A’, ‘O’ e ‘E’.

```
p::first-letter {
  font-size: 24px;
}
```

Quando usamos o pseudo-elemento **before** a gente consegue adicionar alguma coisa antes do conteúdo.

```
H1::before{
  Content: "Título: ";
}
```

No exemplo acima estou adicionando o texto “**Título**” antes de cada elemento **<h1>** que aparecer na página.

Quando usamos o pseudo-elemento **after** a gente consegue adicionar alguma coisa após o conteúdo.

```
H1::after{
  Content: “-Artigo“;
}
```

No exemplo acima estou adicionando o texto **“-Artigo”** após cada elemento **<h1>** que aparecer na página.

Pseudo-classes
As pseudo-classes permitem aplicar estilos a um elemento HTML com base em seu estado ou interação do usuário. Elas são representadas por um único **dois pontos :** seguido do **nome** da pseudo-classe.

```
a:hover {
  color: red;
}
```

No exemplo acima, a cor do texto de todos os links **<a>** mudará para vermelho quando o cursor do mouse estiver sobre eles.

```
a:hover {
  text-decoration: underline;
}
```

No exemplo acima, o texto de todos os links **<a>** ficará sublinhado quando o cursor do mouse estiver sobre eles.

Vantagens de Pseudo-elementos e Pseudo-classes

- **Personalização Avançada:** Permitem estilizar partes específicas de elementos HTML que não podem ser alcançadas diretamente com seletores normais.
- **Interatividade Aprimorada:** Permitem criar efeitos de estilo com base em interações do usuário, como hover, foco e visitado.
- **Melhoria na Usabilidade:** Permitem melhorar a experiência do usuário ao fornecer feedback visual durante a interação com elementos da página.

10. Formatação do texto em CSS

A formatação de texto é uma parte essencial do design de páginas da web, e o CSS oferece uma variedade de propriedades para personalizar a aparência do texto.

Fonte
A propriedade **font-family** permite especificar a família de fontes a ser usada para o texto.
Por exemplo:

```
p {
  font-family: Arial, sans-serif;
}
```

Isso define a fonte do texto em parágrafos <p> para **Arial** e caso ele não encontre a fonte Arial ele irá aplicar a font **sans-serif** genérica.

Tamanho
A propriedade **font-size** determina o tamanho do texto.
Por exemplo:

```
h1 {
  font-size: 24px;
}
```

Isso define o tamanho do texto em cabeçalhos **<h1>** para **24 pixels**.

Cor
A propriedade **color** define a cor do texto.
Por exemplo:

```
.destaque {
  color: #ff0000;
}
```

Isso define a cor do texto em elementos com a classe "destaque" para vermelho.

Espaçamento entre Linhas
A propriedade **line-height** controla a altura da linha do texto.
Por exemplo:

```
p {
  line-height: 1.5;
}
```

Isso define o espaçamento entre as linhas em parágrafos <p> como 1.5 vezes a altura da fonte.

Exemplo de estilização do texto do elemento título <h1>

```
h1 {
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  font-size: 36px;
  color: #333;
  font-weight: bold;
  text-transform: uppercase;
}
```

Exemplo de estilização do texto da classe “destaque”

```
.destaque {
  color: #ff9900;
  font-style: italic;
}
```

Importância da Formatação de Texto
A formatação de texto desempenha um papel crucial na experiência do usuário, afetando a legibilidade, o estilo e a aparência geral da página da web. Ao usar propriedades CSS para personalizar a fonte, tamanho, cor e espaçamento do texto, vocês podem criar conteúdo textual atraente e acessível, melhorando assim a usabilidade e a eficácia do design da página.

11. Propriedade Display

A propriedade display em CSS controla como um elemento é renderizado no layout da página da web. Entre os valores mais comuns estão inline, block e inline-block, cada um com características e usos específicos.

Inline
Elementos com "**display: inline**" são renderizados em linha com o conteúdo circundante, permitindo que outros elementos apareçam ao lado deles na mesma linha.
Eles não começam em uma nova linha e não ocupam toda a largura disponível.

Block
Elementos com **"display: block"** são renderizados em blocos retangulares distintos que começam em uma nova linha e ocupam toda a largura disponível.
Eles não permitem que outros elementos apareçam ao lado deles na mesma linha.

Inline-block
Elementos com **"display: inline-block"** são renderizados como elementos inline, permitindo que outros elementos apareçam ao lado deles na mesma linha, mas eles também respeitam propriedades de bloco, como largura e altura, tornando-os úteis para criar layouts mais complexos que requerem elementos alinhados horizontalmente.

None
A propriedade **"display: none"** em CSS é usada para ocultar completamente um elemento da página, removendo-o do fluxo de layout e tornando-o invisível para o usuário.

Quando Usar Cada Tipo de Display?

- **Inline:** Use para elementos que precisam aparecer no meio do texto, como palavras ou frases curtas, ou quando deseja que vários elementos apareçam lado a lado na mesma linha, como links de navegação.
- **Block:** Use para elementos que precisam começar em uma nova linha e ocupar toda a largura disponível, como parágrafos, divisores de seção e listas.
- **Inline-block:** Use para elementos que precisam aparecer inline, mas também têm propriedades de bloco, como largura e altura definidas, como imagens ou botões.
- **None:** Use quando precisar ocultar elementos da página da web de forma eficaz.

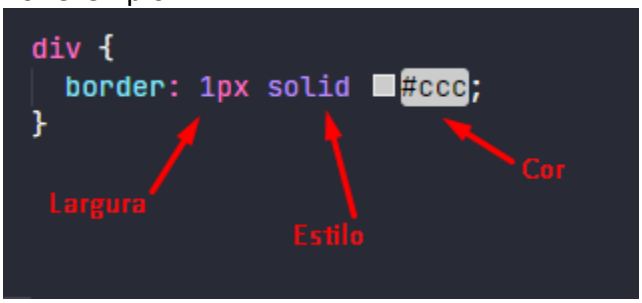
12. Borda, Padding e Margin

As propriedades CSS de borda, padding e margin desempenham um papel crucial no design e layout de páginas da web, permitindo aos desenvolvedores controlar o espaçamento entre elementos e estilizar as bordas.

Borda

A propriedade **border** em CSS é usada para definir a borda de um elemento. Ela consiste em três partes: largura, estilo e cor.

Por exemplo:

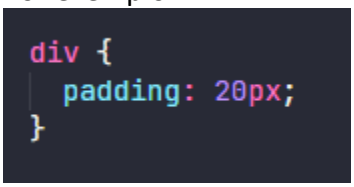


Neste exemplo acima, o elemento ** ** terá uma **borda de 1 pixel de largura**, **estilo sólido** e **cor cinza claro (#ccc)**.

Padding

A propriedade **padding** em CSS é usada para definir o espaçamento interno de um elemento, ou seja, o espaço entre o conteúdo do elemento e sua borda.

Por exemplo:



No exemplo acima, o elemento ** ** terá 20 pixels de espaçamento interno em todas as direções.

Quando coloco apenas um valor, como no exemplo acima **"Padding: 20px;"**, ele coloca nos quatro lados.

Mas se eu colocar dois valores como por exemplo:

Padding: 20px 50px;

Ele coloca o segundo valor nas laterais e o primeiro valor fica para cima e baixo. Pois ele é um short-hand, permitindo definir tudo um só e não ficar definindo padding-left, padding-bottom, etc, evitando assim que especifiquemos uma a uma.

Margin

A propriedade margin em CSS é usada para definir o espaçamento externo de um elemento, ou seja, o espaço entre o elemento e seus elementos vizinhos.

Por exemplo:

```
div {
  margin: 10px;
}
```

Neste exemplo, o elemento terá 10 pixels de espaçamento externo em todas as direções.

Importância das Propriedades de Espaçamento e Borda

As propriedades de **borda**, **padding** e **margin** são fundamentais para criar layouts bem estruturados e visualmente agradáveis em páginas da web.

Elas permitem controlar o espaço entre elementos e estilizar as bordas para criar designs mais atraentes e profissionais.

13. Propriedade Float

A propriedade CSS **float** é muito útil para criar layouts de várias colunas em páginas da web. Ela permite que os elementos sejam posicionados horizontalmente em relação ao contêiner pai, facilitando a criação de designs flexíveis e responsivos.

Funcionamento do Float

Quando um elemento é definido com **float: left** ou **float: right**, ele é movido para a esquerda ou para a direita do contêiner pai, respectivamente. Os elementos subsequentes fluem ao redor do elemento flutuante, criando assim um layout de várias colunas. É importante observar que elementos flutuantes são retirados do fluxo normal de layout, o que significa que o contêiner pai não leva em consideração a altura desses elementos ao calcular sua própria altura.

Considere o seguinte código HTML e CSS para criar um layout de duas colunas usando float:

```
<div class="container">
  <div class="coluna-esquerda">
      Conteúdo da Coluna Esquerda
  </div>
  <div class="coluna-direita">
      Conteúdo da Coluna Direita
  </div>
</div>
```

```css
.container {
  width: 100%;
  max-width: 960px; /* Definindo uma largura máxima para o contêiner */
  margin: 0 auto; /* Centralizando o contêiner na página */
  overflow: hidden; /* Clearfix para conter os elementos flutuantes */
}

.coluna-esquerda {
  float: left;
  width: 50%;
  background-color: blue;
}

.coluna-direita {
  float: right;
  width: 50%;
  background-color: yellow;
}

/* Limpar o float para evitar comportamento indesejado */
.container::after {
  content: "";
  display: table;
  clear: both;
}
```

Neste exemplo, os elementos com as classes **“.coluna-esquerda”** e **“.coluna-direita”** estão definidos para flutuar para a esquerda e para a direita, respectivamente, ocupando cada um metade da largura do contêiner pai **“.container”**.

Quando Usar Float para Layouts de Múltiplas Colunas?

- **Layouts de Colunas:** O float é ideal para criar layouts de várias colunas, como barras laterais, menus e seções de conteúdo em páginas da web.
- **Design Responsivo:** Ao combinar o float com unidades de largura percentual ou media queries, é possível criar layouts que se adaptam a diferentes tamanhos de tela e dispositivos.

IMPORTANTE ao Usar Float:

- **Limpeza do Float:** É importante limpar os floats após os elementos flutuantes para evitar comportamentos indesejados, como elementos que não se estendem corretamente ou problemas de alinhamento.
- **Compatibilidade com Flexbox e Grid:** Embora o float seja uma técnica válida, é importante considerar alternativas mais modernas, como Flexbox e CSS Grid, que oferecem mais controle e flexibilidade no design de layouts.

14. Box-model: Modelo de caixa CSS

O modelo de caixa CSS é um conceito fundamental para o design e o layout de elementos HTML em uma página da web. Ele define como as dimensões (largura, altura), preenchimento (padding), bordas (border) e margens (margin) de um elemento são calculadas e aplicadas.

Componentes do Box-model:

- **Conteúdo (Content):** Refere-se ao espaço ocupado pelo conteúdo do elemento, como texto, imagens ou outros elementos HTML.
- **Preenchimento (Padding):** É a área ao redor do conteúdo do elemento. O preenchimento é definido pela propriedade padding em CSS e é usado para criar espaço entre o conteúdo e a borda do elemento.
- **Borda (Border):** É uma linha que envolve o conteúdo e o preenchimento do elemento. A borda é definida pela propriedade border em CSS e pode ter diferentes estilos, larguras e cores.
- **Margem (Margin):** É a área fora da borda do elemento. A margem é definida pela propriedade margin em CSS e é usada para criar espaço entre os elementos vizinhos.

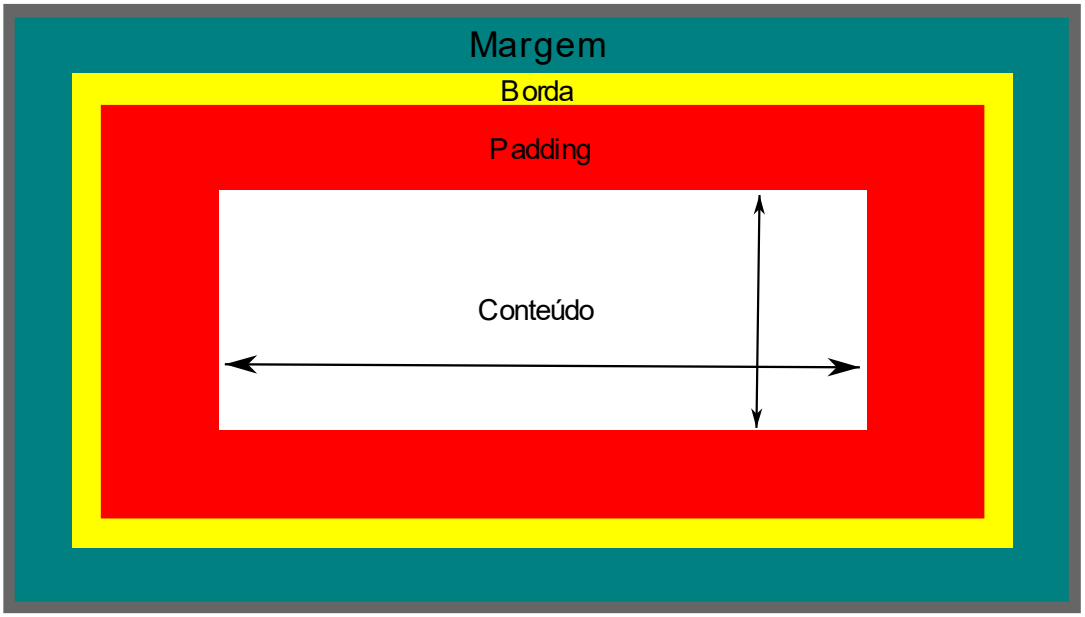
Box-sizing: content-box;

É o valor padrão, quando não especificamos nada. Na imagem abaixo, podemos observar como ele funciona.

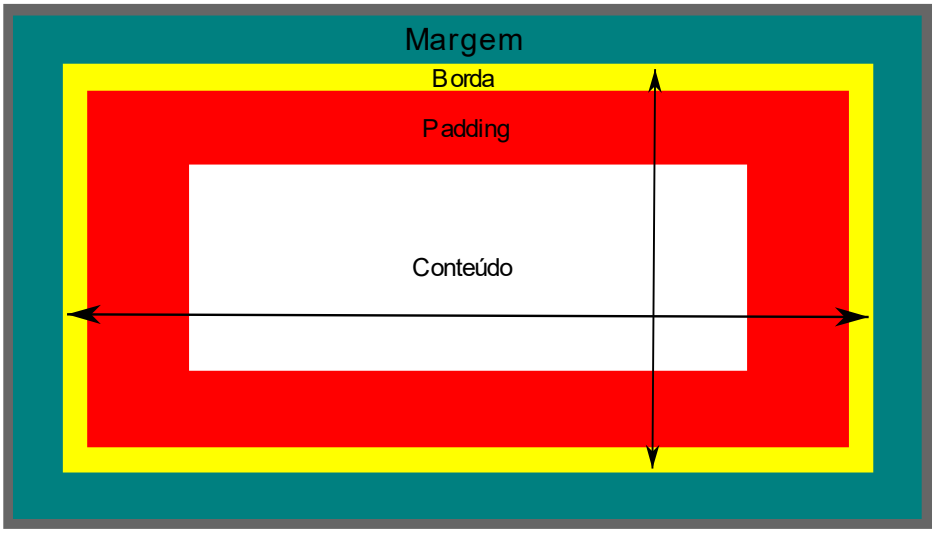
Se especificamos um tamanho de 500px de largura por exemplo, seria dessa parte branca da figura que estamos falando, seria ela que teria 500px.

Quando adicionamos um padding a borda o navegador irá jogar isso para fora do tamanho que definimos para “área de conteúdo”.

Muitas vezes nós desenvolvedores nos confundimos por causa da cor de fundo do conteúdo se expandir para o padding e ficamos achando que tudo é uma coisa só.



Box-sizing: border-box;
Usando essa formatação de border-box, se adicionamos uma borda, o tamanho da caixa continua exatamente o mesmo, ou seja, a borda vai pra dentro da caixa. E se coloamos um padding, também continua exatamente do mesmo tamanho.
Observe na imagem abaixo o indicador das setinhas pretas.
Observamos que a definição do tamanho envolve a borda, padding e conteúdo. Ou seja, a área de conteúdo é calculada com o que vai sobrar e a margem fica por fora.



Calculando as Dimensões Totais
Para calcular as dimensões totais da caixa, precisamos considerar o conteúdo, o preenchimento, a borda e a margem:

- **Largura Total =** largura do conteúdo + preenchimento esquerdo + preenchimento direito + borda esquerda + borda direita + margem esquerda + margem direita
- **Altura Total =** altura do conteúdo + preenchimento superior + preenchimento inferior + borda superior + borda inferior + margem superior + margem inferior

Importância do Box-model
Compreender o modelo de caixa CSS é essencial para criar layouts precisos e previsíveis em páginas da web. Ao controlar o conteúdo, o preenchimento, a borda e a margem de um elemento, nós podemos criar designs visualmente atraentes e funcionais que se adaptam às necessidades do usuário e às especificações de design.

15. Unidades de medida EM e REM

No desenvolvimento web, é essencial entender e escolher as unidades de medida adequadas para dimensionar elementos e criar layouts responsivos. Entre as unidades mais comuns estão o pixel (px), a porcentagem (%) e as unidades em (em) e rem (root em).

Pixel (px)

O “**pixel”** é a unidade de medida mais comum e simples em CSS. Um pixel representa um único ponto na tela do dispositivo.

Ele é uma unidade fixa, o que significa que o tamanho de um elemento definido em pixels não muda, independentemente do tamanho da tela ou do dispositivo.

Os pixels são amplamente utilizados para definir tamanhos de fonte, margens, preenchimentos e bordas de elementos.

Exemplo:

```
div {
  width: 200px;
  font-size: 16px;
}
```

Porcentagem (%)

A “**porcentagem”** é uma unidade de medida relativa, que representa uma proporção do tamanho do elemento pai. Ela é especialmente útil para criar layouts responsivos, onde os tamanhos dos elementos são relativos ao tamanho do contêiner pai. As porcentagens são comumente usadas para definir larguras, alturas, margens e preenchimentos de elementos.

```
div {
  width: 50%;
  margin-top: 10%;
}
```

Unidades em (em)

A unidade “**em”** é relativa ao tamanho da fonte do elemento pai. Um “**em”** é igual ao tamanho da fonte atual do elemento.

Essa unidade é útil quando se deseja criar layouts escaláveis, onde os tamanhos dos elementos se ajustam automaticamente com base na escala da fonte.

As unidades **“em”** são frequentemente usadas para definir tamanhos de fonte, margens e preenchimentos.

```
div {
  font-size: 1.2em;
  margin-left: 1em;
}
```

Unidades rem (root em)

A unidade “**rem”** é semelhante à unidade **“em”**, mas é relativa ao tamanho da fonte do elemento raiz (html).

Isso torna as unidades **“rem”** mais previsíveis e fáceis de controlar em comparação com as unidades **“em”**, especialmente em layouts complexos.

As unidades **“rem”** são amplamente recomendadas para definir tamanhos de fonte e espaçamentos globais.

```css
html {
  font-size: 16px;
}
div {
  font-size: 1.2rem;
  margin-left: 2rem;
}
```

No exemplo de código acima, o seletor está aplicando o tamanho de fonte de 16 pixels para o elemento <HTML>, ou seja, o documento HTML ficará com o tamanho da fonte especificado nesse seletor. Isso serve como ponto de referência para as unidades de medida relativas, como o **“rem”** e **“em”** por exemplo.
No seletor onde está a “div” é aplicado o tamanho de fonte (font-size) de “**1.2rem”** para todos os elementos dessa .
Como a unidade **“rem”** é relativa ao tamanho da fonte do elemento raiz (html), neste caso, o tamanho de fonte do será **1.2 vezes maior** que o tamanho de fonte definido para o elemento <html>.
Além disso, ele define uma margem à esquerda de **“2rem”** para todos os elementos . Isso significa que a margem será igual a duas vezes o tamanho da fonte definido no elemento raiz. Como resultado, a margem à esquerda será proporcional ao tamanho da fonte do documento, o que torna o layout mais escalável e responsivo.

16. Responsividade

O design responsivo é uma abordagem essencial no desenvolvimento web moderno, que visa criar layouts flexíveis e adaptáveis que se ajustam automaticamente a diferentes tamanhos de tela e dispositivos. Introduzir o conceito de design responsivo é fundamental para garantir uma experiência de usuário consistente e agradável em qualquer dispositivo, desde desktops até smartphones.

RECURSOS A SEREM APLICADOS PARA TRAZER RESPONSIVIDADE

Media Queries

As media queries são uma técnica poderosa para aplicar estilos CSS com base nas características do dispositivo, como largura da tela, altura da tela, orientação e resolução. Com as media queries, os desenvolvedores podem criar regras de estilo específicas para diferentes dispositivos e breakpoints, permitindo ajustes precisos do layout conforme necessário.

```css
@media (min-width: 800px) {
  Article {
        Width: 30%;
        Float: left;
        Margin-right: 5%;
  }
}
```

Acima estamos definindo uma largura mínima(800px) para que quando ela for verdadeira, as regras CSS que estão dentro dela sejam aplicadas.

Unidades de Medida Flexíveis

O uso de unidades de medida flexíveis, como porcentagem (%) e unidades rem (root em), é fundamental para criar layouts responsivos. Essas unidades permitem que os elementos se ajustem proporcionalmente ao tamanho da tela ou do contêiner pai, garantindo uma aparência consistente em todos os dispositivos.

Layout Fluido

Em um layout fluido, os elementos são dimensionados em unidades relativas, como porcentagem (%) ou unidades rem, em vez de unidades fixas, como pixels (px). Isso permite que os elementos se ajustem dinamicamente à largura da tela, expandindo e contraindo conforme necessário para preencher o espaço disponível.

Abaixo segue um exemplo de código HTML e CSS para demonstrar a responsividade:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Design Responsivo</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <header>
            <h1>Modelo de Site Responsivo para turma UTFPR</h1>
            <nav class="menu">
                <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Sobre</a></li>
                    <li><a href="#">Serviços</a></li>
                    <li><a href="#">Contato</a></li>
                </ul>
            </nav>
        </header>
        <main>
            <section class="conteudo">
                <h2>Bem-vindo ao Nosso Site!</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla eget neque eu arcu suscipit fermentum.</p>
            </section>
        </main>
    </div>
</body>
</html>
```

```
.container {
  width: 100%;
  max-width: 960px;
  margin: 0 auto;
  padding: 0 20px;
}

@media screen and (max-width: 768px) {
  .menu {
      display: none;
  }
}



@media (min-width: 800px) {
  Article {
      Width: 30%;
      Float: left;
      Margin-right: 5%;
  }
}
```

No exemplo acima, a largura máxima do contêiner é definida como 960 pixels, mas os elementos dentro do contêiner, como o menu de navegação, se adaptam automaticamente e são ocultos em telas menores que 768 pixels de largura. Se você fizer o teste dessa implementação e diminuir o tamanho do browser você verá que o menu irá ficar oculto à medida que vai diminuindo a tela e fica menor que 768px.

17. Layout Fixo e Layout Fluido

Ao criar um design para um site ou aplicativo da web, uma das primeiras decisões a serem tomadas é escolher entre um layout fixo e um layout fluido. Ambos os tipos de layout têm suas vantagens e desvantagens, e a escolha certa depende das necessidades específicas do projeto e das preferências do designer. Esse tipo de escolha pode ser aplicada para partes específicas de um projeto também.

Layout Fixo

Em um layout fixo, as dimensões dos elementos são definidas em unidades de medida fixas, como pixels (px). Isso significa que o tamanho dos elementos não muda, independentemente do tamanho da tela ou do dispositivo do usuário.

Vantagens

- **Consistência:** Um layout fixo pode garantir uma aparência consistente em todos os dispositivos, já que os elementos mantêm suas dimensões definidas.
- **Controle Preciso:** O designer tem um controle preciso sobre o posicionamento e o dimensionamento dos elementos, o que pode ser útil para criar designs detalhados e precisos.

Desvantagens

- **Falta de Flexibilidade:** Um layout fixo pode não se adaptar bem a diferentes tamanhos de tela, resultando em uma experiência de usuário ruim em dispositivos móveis ou telas de resolução variável.
- **Barreiras à Acessibilidade:** Em dispositivos com telas menores, os usuários podem encontrar dificuldades para visualizar e interagir com o conteúdo devido ao layout fixo.

Exemplo

```
.container {
  width: 960px; /* Largura fixa */
  margin: 0 auto; /* Centralizar o conteúdo */
}
```

Layout Fluido

Em um layout fluido, as dimensões dos elementos são definidas em unidades de medida relativas, como porcentagem (%) ou unidades rem (root em). Isso permite que os elementos se ajustem dinamicamente ao tamanho da tela ou do contêiner pai.

Vantagens

- **Adaptabilidade:** Um layout fluido se adapta de forma dinâmica a diferentes tamanhos de tela e dispositivos, proporcionando uma experiência de usuário consistente e otimizada em todas as plataformas.
- **Melhor Acessibilidade:** Os layouts fluidos tendem a ser mais acessíveis, pois permitem que os usuários redimensionem o conteúdo de acordo com suas preferências e necessidades.

Desvantagens

- **Menos Controle de Precisão:** O designer pode ter menos controle sobre o posicionamento e o dimensionamento dos elementos em um layout fluido, o que pode ser desafiador ao criar designs detalhados e precisos.
- **Possíveis Problemas de Visualização:** Em telas muito grandes ou muito pequenas, os elementos do layout fluido podem parecer distorcidos ou mal posicionados, se as proporções **não forem cuidadosamente** planejadas.

Exemplo

```css
.container {
  width: 100%; /* Largura relativa */
  max-width: 1200px; /* Largura máxima */
  margin: 0 auto; /* Centralizar o conteúdo */
}
```

Escolhendo o Layout Adequado

Ao escolher entre um layout fixo e um layout fluido, é importante considerar as necessidades e objetivos específicos do projeto. Se o design requer uma aparência consistente em todas as telas e dispositivos, um layout fixo pode ser mais apropriado. Por outro lado, se a acessibilidade e a adaptabilidade são prioridades, um layout fluido pode oferecer uma melhor experiência de usuário.

Além disso, é possível combinar elementos de layout fixo e fluido em um único design, utilizando técnicas como media queries e unidades de medida flexíveis para criar layouts híbridos que ofereçam o melhor dos dois mundos.

18. Sistema de Grid CSS

Os sistemas de grid CSS permitem criar layouts complexos de maneira eficiente e consistente em projetos web. Eles fornecem uma estrutura flexível e responsiva que permite organizar e posicionar elementos na página de forma precisa e escalável.

Estruturação da Página

Um sistema de grid CSS divide o espaço da página em uma grade imaginária de linhas e colunas. Essa estruturação ajuda a organizar o conteúdo da página de maneira lógica e coesa.

Flexibilidade de Layout

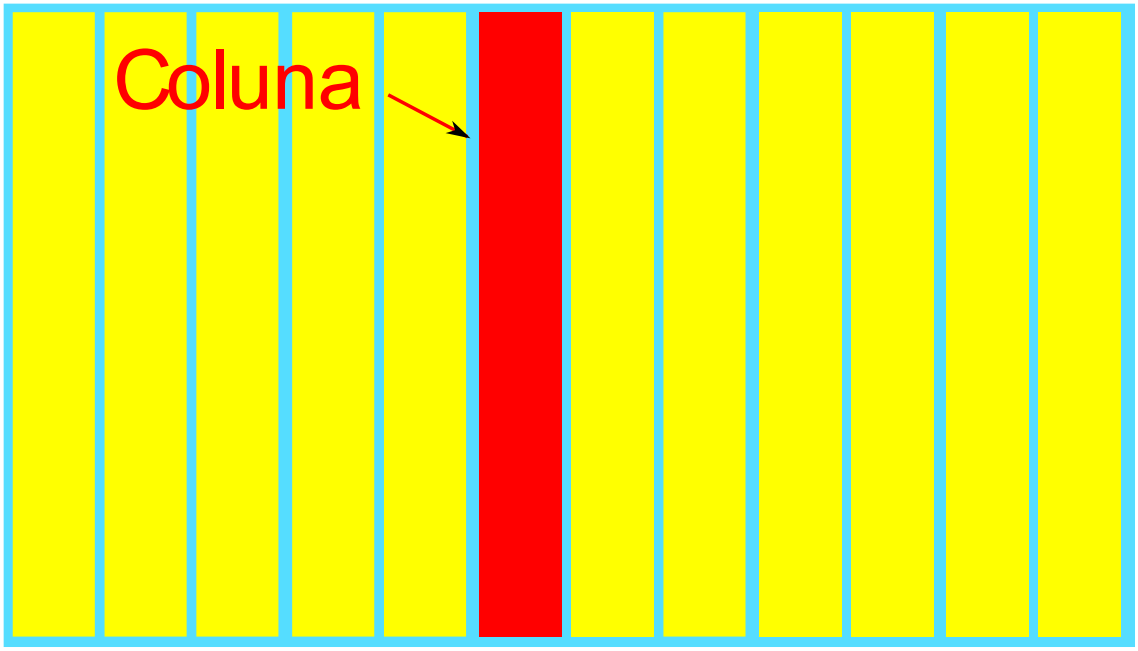
Os sistemas de grid oferecem flexibilidade para criar layouts variados, desde simples designs de uma coluna até layouts mais complexos com múltiplas seções e áreas de conteúdo diferenciadas.

Responsividade Integrada

Os sistemas de grid são projetados para serem responsivos, o que significa que eles se ajustam automaticamente para se adaptar a diferentes tamanhos de tela e dispositivos. Isso é alcançado através do uso de media queries e unidades de medida flexíveis.

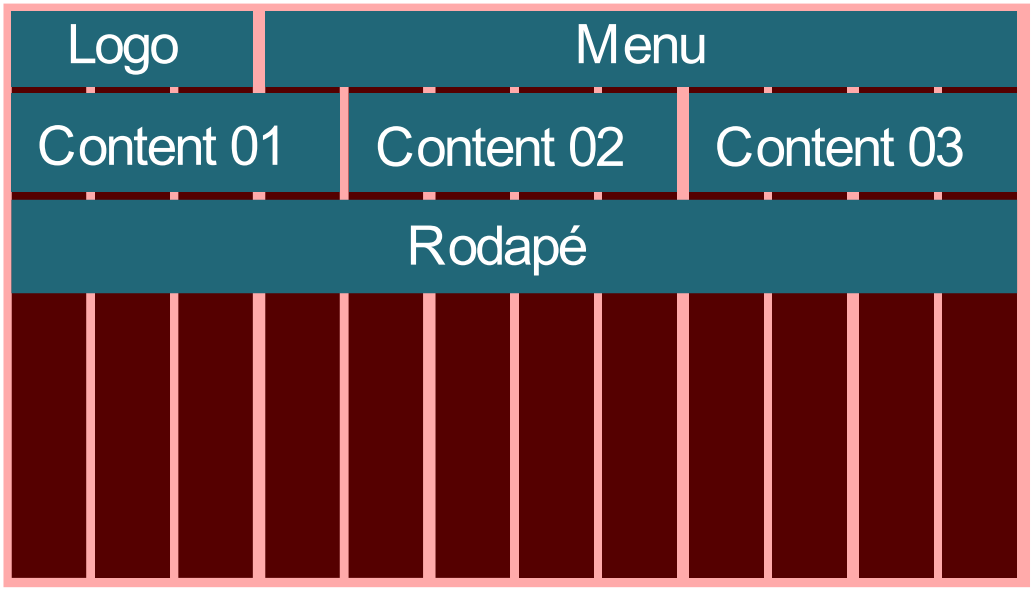
Como os Sistemas de Grid CSS São Usados

- **Divisão do Espaço:** Os sistemas de grid permitem dividir o espaço disponível na página em seções proporcionais, facilitando a criação de layouts equilibrados e visualmente atraentes.
  O sistema de grid é dividido em colunas, como no exemplo do imagem abaixo.

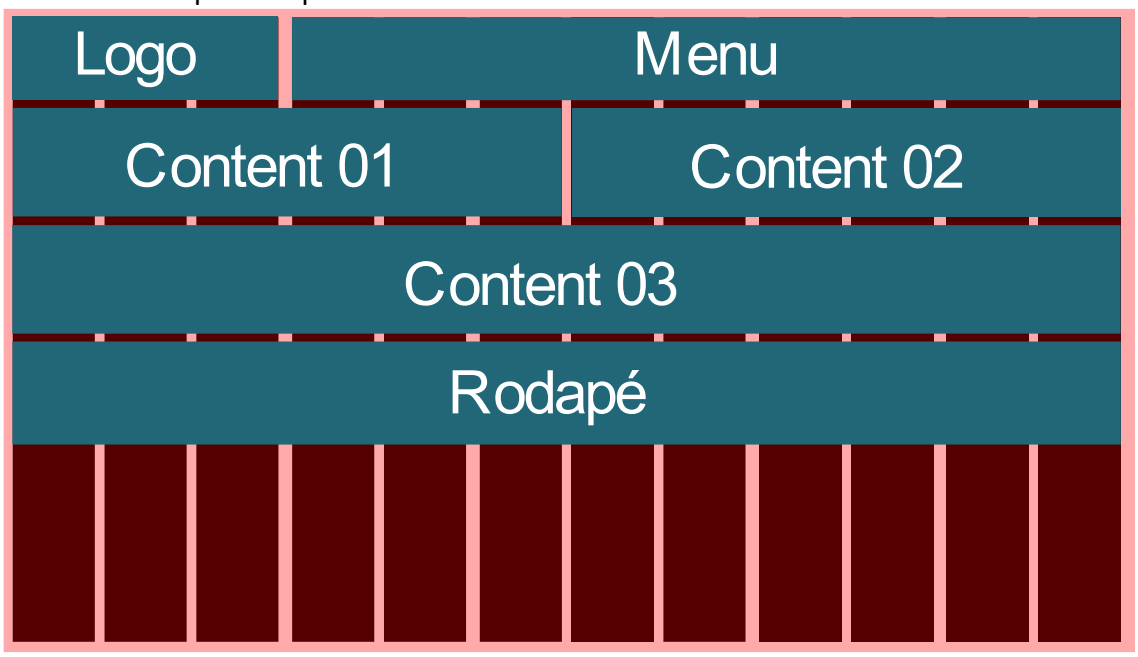


- **Alinhamento e Posicionamento:** Com a grade CSS, é fácil alinhar e posicionar elementos dentro do layout, garantindo uma apresentação consistente e organizada do conteúdo.
- **Reutilização e Manutenção:** Ao utilizar um sistema de grid CSS, os desenvolvedores podem reutilizar classes e estilos predefinidos em todo o projeto, o que simplifica a manutenção e promove a consistência visual.

Por exemplo na imagem abaixo, eu tenho o logo ocupando 3 colunas na primeira linha e o menu ocupando 9 colunas na primeira linha.
Os contents estão na segunda linha e cada um ocupando 4 colunas
E o rodapé ocupando as 12 colunas.



Agora imaginemos que o mesmo site seja aberto em um dispositivo de tela menor, como na imagem abaixo, podemos ver que há o redimensionamento para se adequar ao viewport do usuário com aquele dispositivo.



19. Especificidade do CSS

A especificidade do CSS é um conceito fundamental que determina qual estilo será aplicado a um elemento quando várias regras CSS entram em conflito. Essa especificidade é calculada com base nos seletores e na ordem em que as regras são declaradas, e é crucial para garantir que os estilos sejam aplicados de forma consistente e previsível.

Calculando a Especificidade

A especificidade do CSS é determinada pela combinação de seletores usados em uma regra CSS. Quanto mais específico o seletor, maior é sua especificidade.
Por exemplo, um seletor de ID (#id) é mais específico do que um seletor de classe (.classe).
Esse cálculo é realizado através da concatenação dos valores.

Priorização de Estilos

Quando duas ou mais regras CSS entram em conflito e se aplicam ao mesmo elemento, a regra com a maior especificidade é priorizada e seus estilos são aplicados. Se duas regras têm a mesma especificidade, a que aparece por último no código CSS prevalece.

Especificidade Inline

Estilos inline (definidos diretamente no elemento HTML usando o atributo style) têm a maior especificidade e sempre substituem estilos definidos em regras CSS externas ou internas.

Por exemplo:

```
#meu-elemento {
  color: red;
}
```

Na imagem acima, vemos o seletor de ID, ele tem peso = 100.

```
.minha-classe {
  color: blue;
}
```

Na imagem acima vemos o seletor de classe, que tem peso = 010.

```
p {
  color: green;
}
```

Na imagem acima temos o seletor de elemento, que tem peso = 001.

Considerando as imagens acima, se um elemento **<p id="meu-elemento" class="minha-classe">** estiver presente no HTML, a cor do texto será vermelha, pois a regra com o seletor de ID tem a maior especificidade.

Mas, como chegamos ao valor 100?

```
#meu-elemento {
  color: red;
}
```

Vamos lá, vamos debugar essa instrução acima e com base nas respostas preenchemos a tabela abaixo.

1º Perguntamos, tem um elemento de ID? Se sim, contamos quantos tem e colocamos o valor na coluna ID da tabela. Se tivéssesmos dois para ID, como por exemplo #meu-elemento e #titulo, nós deveríamos colocar o valor 2 na coluna ID.

2º Perguntamos, tem classe? Como no exemplo acima não tem, colocamos 0(zero) na coluna classe.

3º Perguntamos, tem Elemento? Como no exemplo acima não tem, também colocamos zero.

	ID	Classe	Elemento
#meu-elemento	1	0	0

Portanto o valor para essa regra é => 100

Vamos ao cálculo para o segundo exemplo:

	ID	Classe	Elemento
.minha-classe	0	1	0

Portanto o valor para essa regra é => 010

Vamos ao cálculo para o terceiro exemplo:

	ID	Classe	Elemento
p	0	0	1

Portanto o valor para essa regra é => 001

E se eu tivesse assim?

```
p span{
  color: green;
}
```

Vamos ao cálculo para o terceiro exemplo:

	ID	Classe	Elemento
p span	0	0	2

Portanto o valor para essa regra é => 002
Isso porque encontramos dois elementos na definição dessa regra CSS.

E se eu tivesse assim?

```
#meu-elemento .minha-classe{
  color: black;
}
```

Vamos ao cálculo para o terceiro exemplo:

	ID	Classe	Elemento
#meu-elemento .minha-classe	1	1	0

Portanto o valor para essa regra é => 110
Isso porque encontramos um id e uma classe na definição dessa regra CSS.