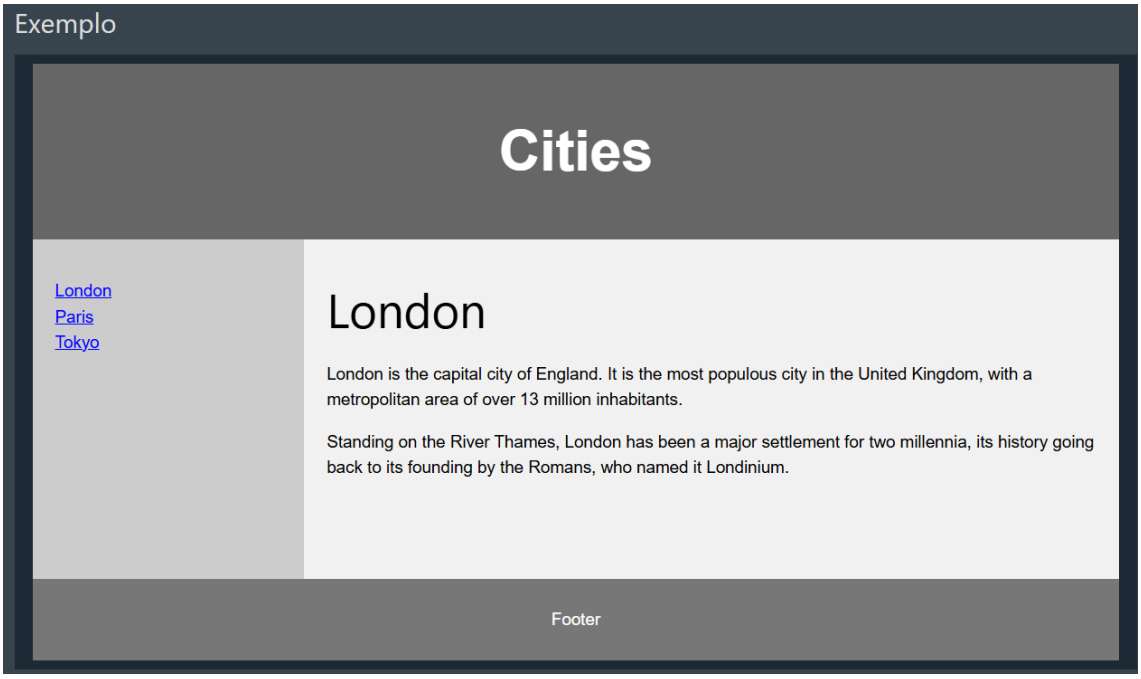
Introdução

Elementos e técnicas de layout HTML

Os sites geralmente exibem conteúdo em várias colunas (como uma revista ou um jornal).

Exemplo



Elementos de layout HTML

O HTML possui vários elementos semânticos que definem as diferentes partes de uma página web:

- <header>- Define um cabeçalho para um documento ou uma seção
- <nav>- Define um conjunto de links de navegação
- <section>- Define uma seção em um documento
- <article>- Define conteúdo independente e autocontido
- <aside>- Define o conteúdo separadamente do conteúdo (como uma barra
- <footer>- Define um rodapé para um documento ou uma seção
- <details>- Define detalhes adicionais que o usuário pode abrir e fechar sob
- <summary>- Define um título para o <details>elemento

Você pode ler mais sobre elementos semânticos em nosso capítulo Semântica HT

Técnicas de layout HTML

Existem quatro técnicas diferentes para criar layouts com várias colunas. Cada técnica tem seus prós e contras:

- CSS framework (você pode usar um framework CSS, como W3.CSS ou Bootstrap .)
- **CSS float property**
- **CSS flexbox**
- **CSS grid**

Layout float CSS

É comum criar layouts web inteiros usando a floatpropriedade CSS. Float é fácil de aprender — você só precisa lembrar como as propriedades floate funcionam. **Desvantagens:** Elementos flutuantes estão vinculados ao fluxo do documento, o que pode prejudicar a flexibilidade. Saiba mais sobre float em nosso capítulo sobre Float e Clear em CSS . clear

Layout Flexbox CSS

O uso do flexbox garante que os elementos se comportem de forma previsível quando o layout da página precisa acomodar diferentes tamanhos de tela e diferentes dispositivos de exibição.

Saiba mais sobre flexbox em nosso capítulo CSS Flexbox .

Layout de grid CSS

O módulo CSS Grid Layout oferece um sistema de layout baseado em grade, com linhas e colunas, facilitando o design de páginas da web sem precisar usar floats e posicionamento.

Saiba mais sobre grades CSS em nosso capítulo Introdução à Grade CSS .

Referências

https://www.w3schools.com/html/html_layout.asp

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_float

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_layout_flexbox

https://www.w3schools.com/css/tryit.asp?filename=trycss_grid_layout_named

Roteiro de Ensino: CSS Layout Float

1. O Problema e a Solução Clássica (20 minutos)

- **Introdução:** Explique que o float foi a primeira técnica para criar colunas.
- **O "Clearfix":** Mostre o problema do float e como o clearfix resolve.

- **Código de Exemplo:** Use este código para demonstrar a criação das colunas e a solução do clearfix.

```
<!DOCTYPE html>
<html>
<head>
<style>
/* Reset básico e estilo de corpo */
* { box-sizing: border-box; }
body { font-family: Arial, Helvetica, sans-serif; }
/* Estilo do cabeçalho, rodapé, etc. */
header, footer { background-color: #666; padding: 10px; text-align: center; color: white; }
nav { float: left; width: 30%; background: #ccc; padding: 20px; }
article { float: left; padding: 20px; width: 70%; background-color: #f1f1f1; }
/* A mágica do clearfix */
section::after {
  content: "";
  display: table;
  clear: both;
}
</style>
</head>
<body>
<header><h2>Cabeçalho</h2></header>
<section>
  <nav><ul><li>Menu 1</li></ul></nav>
  <article><h1>Conteúdo Principal</h1></article>
</section>
<footer><p>Rodapé</p></footer>
</body>
</html>
```

2. Layout Responsivo com Float (15 minutos)

- **Introdução:** Explique a necessidade de adaptar o layout para telas menores.
- **Media Queries:** Mostre como o @media é usado para aplicar estilos específicos.
- **Código de Exemplo:** Adicione este bloco CSS ao exemplo anterior para demonstrar a responsividade.

```
/* Adicione esta regra ao bloco <style> do exemplo anterior */

@media (max-width: 600px) {

  nav, article {

    width: 100%;

    float: none; /* Desativa o float em telas pequenas */

  }

}
```

3. Revisão e Desafio (15 minutos)

Recapitulação: Resuma os pontos principais: float cria colunas, mas precisa do clearfix. @media é usado para a responsividade, resetando o float.

Mini-Desafio: Peça aos alunos para removerem float: left; da regra de nav no código e observarem como o layout se quebra.

Mini-Desafio: CSS Layout Float

- **Desafio:** Remova float: left; da regra de nav no código e observe como o layout se quebra.
- **Resolução e Explicação:**
    1. O aluno deve ir à regra CSS que estiliza o <nav> e comentar ou remover a linha float: left;.

```
nav {

  /* float: left;  <-- Remover ou comentar esta linha */

  width: 30%;

  background: #ccc;

  padding: 20px;

}
```

 Ao salvar e recarregar a página, o <nav> voltará para o fluxo normal do documento, comportando-se como um elemento de bloco. Isso significa que ele ocupará 100% da largura, empurrando o <article> para a linha de baixo.

 A explicação para os alunos é que o float é a propriedade que permite que elementos "flutuem" lado a lado. Quando a removemos, os elementos voltam ao seu comportamento padrão, que é ocupar uma linha inteira cada. Este exercício demonstra a dependência que o layout float tem dessa propriedade e como a sua ausência quebra a estrutura de colunas.

Roteiro de Ensino: CSS Layout Flexbox

1. A Solução Moderna e Flexível (20 minutos)

- **O Problema Resolvido:** Relembre a complexidade do float e apresente o **Flexbox** como a solução para layouts de uma dimensão.
- **Container e Itens:** Explique a diferença entre container (pai) e itens (filhos).
- **A "Flex-Magia":** Demonstre como display: flex; transforma o contêiner.
- **Código de Exemplo:** Use este código para mostrar a criação de um layout flexível e como as larguras são controladas.

```
<!DOCTYPE html>
<html>
<head>
<style>
/* Reset básico e estilo de corpo */
* { box-sizing: border-box; }
body { font-family: Arial, Helvetica, sans-serif; }
/* Estilo do cabeçalho, rodapé, etc. */
header, footer { background-color: #666; padding: 10px; text-align: center; color: white; }
/* O Container Flex */
section { display: flex; }
/* Os Itens Flex */
nav { flex: 1; background: #ccc; padding: 20px; }
```

```
article { flex: 3; background-color: #f1f1f1; padding: 10px; }
</style>
</head>
<body>
<header><h2>Cabeçalho</h2></header>
<section>
 <nav><ul><li>Menu 1</li></ul></nav>
 <article><h1>Conteúdo Principal</h1></article>
</section>
<footer><p>Rodapé</p></footer>
</body>
</html>
```

2. Responsividade Simples com Flexbox (15 minutos)

- **Introdução:** Explique que o Flexbox facilita a responsividade.
- **A Mágica do flex-direction:** Demonstre como apenas uma linha de código muda o layout.
- **Código de Exemplo:** Adicione este bloco CSS ao exemplo anterior para mostrar a responsividade.

```
/* Adicione esta regra ao bloco <style> do exemplo anterior */
@media (max-width: 600px) {
 section {
  flex-direction: column; /* Faz os itens se empilharem */
 }
}
```

3. Revisão e Desafio (15 minutos)

- **Recapitulação:** Resuma os pontos principais: **display: flex;** cria o contêiner, **flex** controla o tamanho e **flex-direction: column;** o torna responsivo.
- **Mini-Desafio:** Peça aos alunos para alterarem o valor de flex do nav e do article para flex: 2; e flex: 2; para que as colunas fiquem com tamanhos iguais.

Mini-Desafio: CSS Layout Flexbox

- **Desafio:** Altere o valor de flex do nav e do article para que eles tenham tamanhos iguais.
- **Resolução e Explicação:**
    1. O aluno deve alterar a propriedade flex nas regras do nav e do article para o mesmo valor. flex: 1; é a opção mais clara.

```
nav {
  flex: 1; /* Alterar de '1' para '1' */
  background: #ccc;
  padding: 20px;
}
article {
  flex: 1; /* Alterar de '3' para '1' */
  background-color: #f1f1f1;
  padding: 10px;
}
```

 Ao salvar e recarregar a página, as duas colunas terão exatamente a mesma largura (50% cada).

 A explicação para os alunos é que a propriedade flex distribui o espaço disponível no contêiner de forma proporcional. Ao darmos o mesmo valor (1) para os dois itens, estamos dizendo ao Flexbox para dividir o espaço igualmente entre eles. Isso demonstra a flexibilidade e a facilidade de controle de largura do Flexbox em comparação com a necessidade de recalcular porcentagens.

Roteiro de Ensino: CSS Grid Layout

1. O Sistema de Grade (20 minutos)

- **Grid vs. Flexbox:** Inicie a aula comparando o Grid ao Flexbox, focando em layouts **bidimensionais**.
- **Mapa da Grade:** Explique o conceito de "mapa visual" do grid-template-areas.
- **Conectando os Itens:** Mostre como as regras grid-area ligam os elementos ao mapa.
- **Código de Exemplo:** Use este código para mostrar como as áreas são definidas e conectadas.

```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "menu content"
    "footer footer";
  grid-template-columns: 1fr 3fr; /* Divide o espaço em 1/4 e 3/4 */
  gap: 5px; /* Espaçamento entre as células */
  padding: 5px;
}
.header { grid-area: header; background: #666; }
.menu { grid-area: menu; background: #ccc; }
.content { grid-area: content; background: #f1f1f1; }
.footer { grid-area: footer; background: #777; }

/* Estilos de conteúdo para o exemplo */
.container > div { padding: 10px; color: white; }
.container .content { color: black; }
</style>
</head>
<body>
<h1>CSS Grid Layout</h1>
<div class="container">
  <div class="header"><h2>Meu Cabeçalho</h2></div>
  <div class="menu"><ul><li>Menu 1</li><li>Menu 2</li></ul></div>
  <div class="content"><h3>Conteúdo Principal</h3></div>
  <div class="footer"><h4>Rodapé</h4></div>
</div>
```

</body>

</html>

2. Tamanho e Espaçamento (15 minutos)

- **Unidades de Medida:** Foque na unidade **fr** e explique como ela divide o espaço disponível de forma flexível.
- **Espaçamento Simples:** Demonstre como a propriedade gap adiciona espaçamento sem complicação.

3. Revisão e Desafio (15 minutos)

- **Recapitulação:** Resuma os pontos principais: **display: grid;**, **grid-template-areas** para o mapa, **fr** para o tamanho e **gap** para o espaçamento.
- **Mini-Desafio:** Peça aos alunos para alterarem o grid-template-columns para **100px 1fr**, fazendo com que a coluna do menu tenha uma largura fixa.

Mini-Desafio: CSS Grid Layout

- **Desafio:** Altere o grid-template-columns para 100px 1fr, fazendo com que a coluna do menu tenha uma largura fixa e a coluna de conteúdo ocupe o restante.
- **Resolução e Explicação:**
    1. O aluno deve ir à regra .container e modificar o valor da propriedade grid-template-columns.

```
.container {
 display: grid;
 grid-template-areas:
  "header header"
  "menu content"
  "footer footer";
 grid-template-columns: 100px 1fr; /* Alterar de '1fr 3fr' para '100px 1fr' */
 gap: 5px;
 padding: 5px;
}
```

 Ao salvar e recarregar a página, a coluna menu terá uma largura fixa de 100 pixels, e a coluna content irá ocupar todo o espaço horizontal restante.

 A explicação para os alunos é que o CSS Grid permite misturar diferentes unidades de medida (pixels, porcentagem, frações) para definir o tamanho das colunas. Neste caso, a unidade 100px cria uma coluna com largura fixa, e a unidade 1fr diz para a segunda coluna ocupar "uma fração" do espaço que sobrou. Isso mostra a versatilidade do Grid, que pode ser usado para layouts fluidos e fixos ao mesmo tempo.

@douglasabnovato

01/21/2025