PASSOS PARA APRENDER CSS:

- TERMINAR E ANOTAR O VÍDEO
- FLEX FROGGY
- PROJETOS
- UDEMY

Html5

Estrutura básica

```
<!DOCTYPE html>
<html>
    <head>
        <title> Quiz para programadores </title>
        <meta name="description" content="Um quiz para testar seus
conhecimentos">
    </head>
    <body>

    </body>
</html>
```

`!DOCTYPE html`: Indica que é um arquivo html
`head`: Recebe metadados e elementos não visíveis
`body`: Parte visivel do site
`meta`: Utiliza de atributos existem vários...

Estrutura semântica

Os sites geralmente possuem cabeçalho, corpo e rodapé onde temos tags para cada uma delas.
`<header>`: Cabeçalho do site.
`<main>`: Conteúdo principal ou corpo.
`<footer>`: rodapé do Site.
Todas são tags semânticas.

Dentro da `main` é adicionada várias tags `<section>` para dividir o nosso site e ele ficar menos poluído, por exemplo uma `section` para pergunta 1 outra pra pergunta dois e assim por diante. Ela também menciona tags como `<strong>`, `<em>`, `<abbr>`.

Comentários

No html5 são feitos da seguintes forma: <!--

Form

Sempre que o usuário for fazer uma entrada de alguma informação usamos a tag <form>, formatos de entrada do usuários, possuindo um <input> e no mesmo o <type> que define o comportamento:

Tipos de Inputs em HTML:

`<input type="text">`: Campo de texto para entrada de uma linha de texto. É utilizado com o atributo placeholder

```
<input type="text" name="username">
```

`<input type="password">`: Campo de texto onde os caracteres são ocultados, geralmente usado para senhas.

```
<input type="password" name="password">
```

`<input type="email">`: Campo de texto específico para email, com validação automática para formatos de email.

```
<input type="email" name="email">
```

`<input type="number">`: Campo para números, com opção de definir um valor mínimo e máximo.

```
<input type="number" name="age" min="1" max="100">
```

`<input type="tel">`: Campo para números de telefone

```
<input type="tel" name="phone">
```

`<input type="url">`: Campo para URLs, com validação para formato de endereço da web.

```
<input type="url" name="website">
```

`<input type="date">`: Campo para selecionar uma data.

```
<input type="date" name="birthday">
```

`<input type="checkbox">`: Caixa de seleção que permite marcar ou desmarcar uma opção. É utilizado com o `<label>`.

```
<input type="checkbox" name="subscribe" value="newsletter">
```

`<input type="radio">`: Botão de seleção para escolher uma única opção entre várias. É utilizado com o `<label>`.

```
<input type="radio" name="gender" value="male"> Male
<input type="radio" name="gender" value="female"> Female
```

`<input type="submit">`: Botão para enviar o formulário.

```
<input type="submit" value="Enviar">
```

`<input type="reset">`: Botão para limpar todos os campos do formulário e restaurar seus valores iniciais.

```
<input type="reset" value="Limpar">
```

`<input type="file">`: Campo para fazer upload de arquivos.

```
<input type="file" name="profilePicture">
```

`<input type="hidden">`: Campo oculto que envia um valor ao servidor sem que o usuário veja.

```
<input type="hidden" name="userId" value="12345">
```

`<input type="color">`: Campo que permite escolher uma cor.

```
<input type="color" name="favoriteColor">
```

`<input type="range">`: Controle deslizante para selecionar um valor dentro de um intervalo.

```
<input type="range" name="volume" min="0" max="100">
```

`<input type="button">`: Botão que não envia o formulário, geralmente usado com JavaScript para executar uma ação específica.

```
<input type="button" value="Clique Aqui" onclick="alert('Botão clicado!')">
```

Select

No form temos a tag `<select>` onde dentro dela são colocados várias tags `<options>` e são usadas para criar um menu dropdown:

```
<form action="/submit-data" method="post">
  <label for="country">Escolha seu país:</label>
  <select id="country" name="country">
    <option value="brasil">Brasil</option>
    <option value="argentina">Argentina</option>
    <option value="eua">Estados Unidos</option>
    <option value="japao">Japão</option>
    <option value="alemanha">Alemanha</option>
  </select>

  <input type="submit" value="Enviar">
</form>
```

Atributos do <form>

action: Define para onde os dados do formulário devem ser enviados (uma URL). Por exemplo:

```
<form action="https://meuservidor.com/processar-dados">
```

method: Define o método HTTP para enviar os dados, que pode ser:
GET: Envia os dados na URL (utilizado quando os dados não são confidenciais).
POST: Envia os dados no corpo da requisição (mais seguro, usado para enviar dados sensíveis).

```
<form action="/submit-data" method="post">
```

Imagens

As imagens são colocadas através da tag `<img>` podemos adicioná-las da seguinte forma:

- **Com o link**:
  `<img src=”linkDaImagem" alt=”textoQueApareceráCasoaImagemNãoCarregue”`
- **Com a imagem no projeto**:
  `<img src=”caminhoDaImagem"alt=”textoQueApareceráCasoaImagemNãoCarregue”`

Figure

A tag `<figure>` é usada para legendar a foto/imagem e colocamos a tag `<img>` dentro da tag `<figure>` juntamente com a `<figcaption>`:

```
<figure>
      <img src=”caminhoDaImagem"alt=”textoQueApareceráCasoaImagemNãoCarregue”
      <figcaption>Figura 1: logotipo de uma linguagem de programação
</figcaption>
</figure>
```

Tabelas

As tabelas são iniciadas através da tag `<table>` (recebe o atributo `border=""`), dentro dela está a `<tr>` e, dentro dela, está a `<td>`. Além disso, a `<table>` pode ser dividida em cabeçalho, corpo e rodapé. Também é possível utilizar o `colspan="2"` e o `rowspan="2"`, que servem para que você decida quantas colunas e linhas serão ocupadas.

```
    <table boder=”1”> <!--Define a borda-->
        <thead> <!--Cabeçalho-->
            <th>Pontuação</th> <!--th é uma tag específica para cabeçalhos-->
            <th>Avaliação</th>
        </thead>
        <tbody> <!--Define o corpo da tabela-->
            <tr> <!--Cria uma nova linha-->
                <td>Produtos</td> <!--Cria um novo dado/informação-->
            </tr>
        </tbody>

        <tfoot> <!--Define o rodapé-->
            <tr>
```

```
                <td colspan=”2”> <!--Faz com que uma coluna ocupe dois
colunas--
                    Boa sorte na próxima
                </td>
            </tr>
        </tfoot>
    </table>
```

Listas

Temos as listas não ordenadas `<ul>` e as ordenadas `<ol>` os itens dentro de ambas as listas serão dados pela tag `<li>`:

```
<ul>
    <li>item</li>
</ul>
```

OU

```
<ol>
    <li>item</li>
</ol>
```

Details e Summary

Caso coloquemos a lista dentro da tag `<details>` e dentro da mesma a tag `<summary>` a lista só será exibida se o usuário realizar um clique.

```
<details>
    <summary> O que aparecerá como texto</summary>
    <ul>
        <li>item</li>
    </ul>
</details>
```

Formulários

Esse ela fala que é diferente do que nós vimos antes, pois aqui vemos a tag `<textarea>` onde é uma área de texto bem maior que o texto normal e podemos personalizar seu tamanho, além disso vimos a `<fieldset>` que cria uma área envolvendo o `<form>` e a tag `<legend>` adiciona uma legenda a esse `<fieldset>`:

```
<form>
  <label for=”comentarios”>blablabla</label>
  <input type="text" name="username">
  <label for=”comentarios”>blablabla</label>
  <textarea rows=”4” cols=”50”></textarea>
 <button>Enviar</button>

</form>
```

Footer

Como colocar **caracteres especiais**, pois geralmente usamos o © para o copyright do site que no html5 seria o `&copy`

links

Para isso usamos a tag <a> onde a tag possui um atributo href="" que indica para onde iremos quando clicarmos naquele link, o target="" indica onde abriremos esse link.

parei no minuto 56:29

CSS

Seletores

Seletores são o que usamos no css3 para dizer qual tag html5 queremos estilizar, possuindo seletor, chave, propriedade e valor:

```
p{
    color: white;
}
```

p - seletor;
{ - chave;
color - propriedade;
white - valor;
; - declaração

Mais sobre seletores

- Seletor universal: *{}.
- Seletor de texto: h1{}, h2{} e etc.
- Seletor ID: #id{}.
- Seletor filho: seleciona um elemento filho de outro ex: li>a{} (links que são filhos de li).
- Seletor descendente: p a{}. Esse seletor seleciona quaisquer elemento <a> que esteja dentro de <p> mesmo se houver outros elementos aninhados entre eles.

- Seletor de irmão adjacente: h1+p{}. Seleciona o primeiro elemento <p> depois de qualquer elemento <h1> (mas não outros elementos <p>).

Classes e IDs

Classes e s id são dois tipos de identificadores, contudo o **ids são únicos** e não podem ser usados mais de uma vez, diferentemente da **classe que possui essa versatilidade de ser usada em várias tags**. Para que possamos chamar um id no html5 usamos a "**#**" e para a classe o "**.**":

```
<p class="Classe">Azul</p>
<p id="Identificador">Verde</p>

#Identificador{
    background: green;
}
.Classe{
    background: blue;
}
```

Geralmente utilizamos a **classe para as exceções** e o ID para as coisas mais gerais ou únicas, ademais o **ID já foi usado para a estruturalização dos sites, contudo com as tags header, main, section, footer se tornou algo menos usado**.

Divs e span

`div` vem do inglês *division* (divisão). É utilizada para separar partes do site dentro de uma caixa, principalmente quando se deseja agrupar itens e aplicar estilos em conjunto. Normalmente, atribui-se uma classe à `div`. A `div` e o elemento `<p>` (parágrafo) são elementos do tipo *bloco* (*block*). Isso significa que eles ocupam toda a largura disponível do seu contêiner pai, quebrando a linha e empurrando os elementos seguintes para baixo.

O elemento **span**, diferentemente da `div`, ocupa apenas a área do conteúdo e posiciona os elementos lado a lado, por ser um elemento *em linha* (*inline*)."

Explicação mais detalhada e exemplos:

- **`div` (Elemento de Bloco):**
    - Cria um bloco de conteúdo.
    - Ocupa toda a largura disponível.
    - Força uma quebra de linha antes e depois do elemento.
    - Usado para estruturar o layout da página, agrupar outros elementos e aplicar estilos a um conjunto deles.

span (Elemento em Linha):

- Ocupa apenas o espaço necessário para o seu conteúdo.
- Não força quebras de linha.
- Usado para aplicar estilos a pequenas porções de texto dentro de um parágrafo ou outro elemento de bloco.

Bordas

É uma propriedade que adiciona um contorno em torno de um elemento HTML, podendo ser aplicada em todos os lados do elemento da seguinte forma: **border: espessura(ex: px), estilo(ex: solid), cor (ex: red);**. Existem muitos estilos de bordas em CSS3, caso queira saber mais, pesquise. Também é possível utilizar cada um dos tipos de bordas separadamente, como: `border-color`, `border-width`, `border-style`, onde usamos isso para definir especificamente cada elemento. Por exemplo, com o `color` podemos definir a borda de cada canto, começando por cima, direita, embaixo e esquerda, mas caso você defina apenas dois valores, o primeiro será aplicado na parte superior e inferior, e o segundo na parte direita e esquerda. Para criar bordas arredondadas, utilize a propriedade `border-radius`.

Fontes e Cores e Estilos de Textos

As cores no CSS3 podem ser expressas em código hexadecimal, RGB, nome da cor e outros formatos. Algumas propriedades usadas são **color**, **background-color**, entre outras. As fontes podem ser utilizadas tanto as do próprio CSS3 quanto de sites externos, como o Google Fonts, que podem ser aplicadas ao código HTML5 ou ao CSS3. O **font-size** define o tamanho, **font-family** é usado para trocar a fonte (Arial, Verdana) e existem diferenças entre as famílias *sans-serif* e *serif*. O **font-weight** define a espessura da fonte, com números ou palavras como *bold* e *normal*. O **font-style** define o estilo da fonte (normal, itálico ou oblíquo). O **text-decoration** controla decorações aplicadas ao texto, como sublinhado, tachado ou sobrelinhado. Existem outras, mas essas são as mais utilizadas.

Cor e Imagem de Fundo

As propriedades de **background-color** e **color** já foram citadas acima. Podemos aplicar uma imagem de fundo com o **background-image: url('caminho/para/a/imagem');**. Caso a imagem não seja grande o suficiente para preencher toda a área, ela irá se repetir tanto verticalmente quanto horizontalmente. Para controlar a repetição, usaríamos o **background-repeat**, e, caso quisermos controlar o scrolling desse fundo, usamos o **background-attachment**. O **background-position** controla o posicionamento, sendo **center**, por exemplo, uma opção, e pode receber dois valores: o primeiro alinha na horizontal e o segundo na vertical.

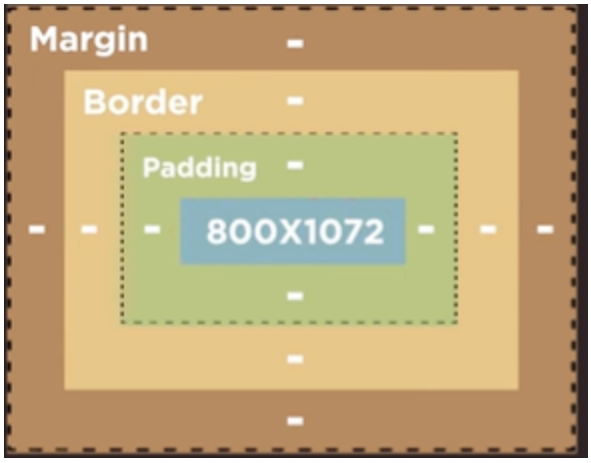
Podemos juntar todas essas propriedades na propriedade **background**. O primeiro valor é para a cor, o segundo para imagens, o terceiro para o repeat, o quarto para o scrolling e o quinto para o position.

CSS Externo

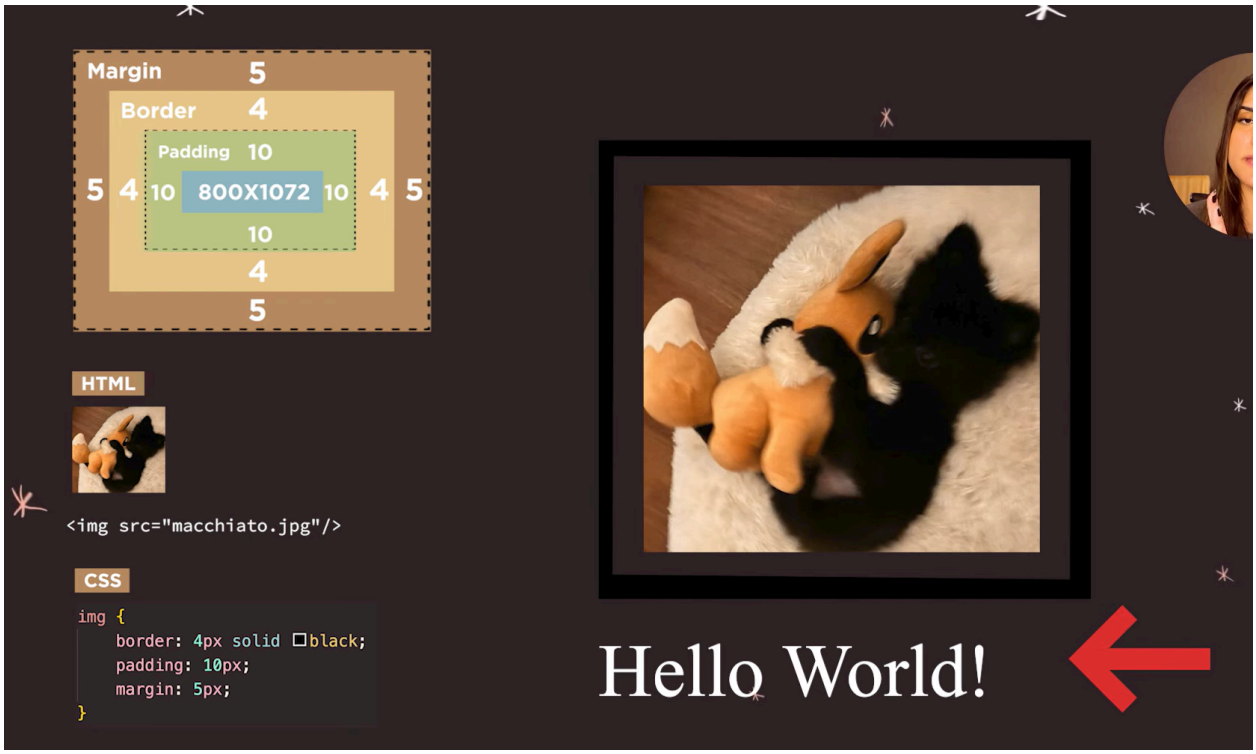
O css externo é o tipo de css que nós mais utilizamos, basta criar um arquivo com .css e fazer uma ligação a ele no arquivo .html que vc desejar com o <link>.

Box Model (Modelo de Caixa)

É dividido em quatro partes: **conteúdo, padding, border e main** quando adicionamos uma imagem sem nenhuma estilização, ela ficará com seu tamanho original irá ser exibido o número do tamanho (largura e espessura) dela no conteúdo do Box Model.3

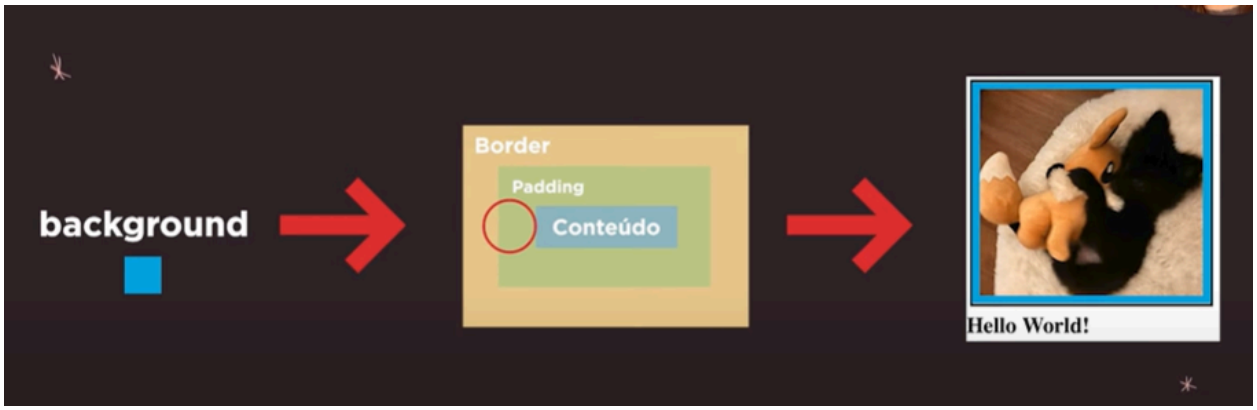


Caso adicionassémos uma borda de 4px o border ia ter o número 4, o padding é o espaço entre o conteúdo e a borda e o margin será o espaçamento entre a borda e o próximo conteúdo.

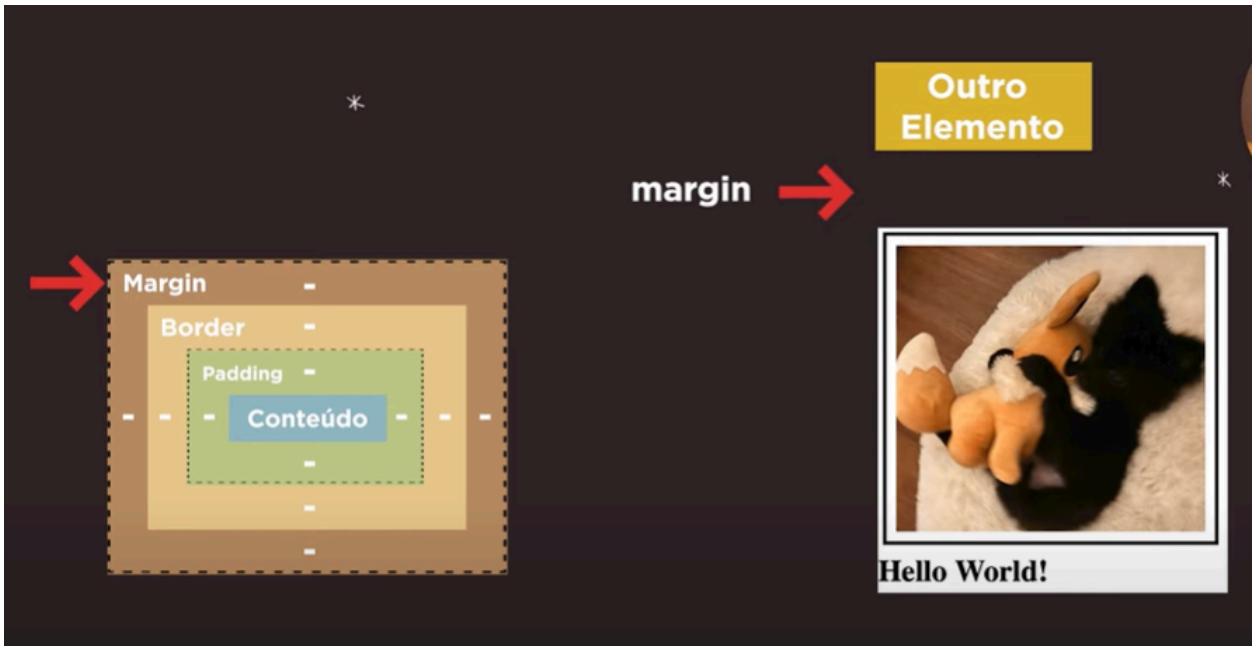


As propriedades width e height ajustam o tamanho do conteúdo de uma caixa, mas o tamanho total da caixa na página inclui também o padding, border e margin.

Se colocarmos um fundo na imagem (background) ele irá ficar apenas no padding.



A border ficará visualmente como você estilizar e a margin será apenas imaginária apenas para que você se distancie dos elementos do lado ou de outro elemento.



Podemos definir como a largura e a altura de todo o Box Model serão calculadas usando a propriedade box-sizing. Por padrão, ele vem com o valor content-box, mas podemos alterá-lo para border-box para controlar o cálculo do tamanho do nosso elemento. Uma prática comum é criar um arquivo (como reset.css) para configurar padrões, como padding: 0; e box-sizing: border-box;.

Dentro das propriedades, temos o outline, que é um contorno fora da borda e não ocupa espaço no Box Model. A propriedade overflow define como o conteúdo que excede a área do elemento será mostrado, com opções como visible, hidden, auto, scroll, overflow-x, e overflow-y.

Existem dois tipos principais de caixas: inline-level (como span) e block-level (como div).

Unidades de Medidas

Unidades Relativas

1. **em**: Relativa ao tamanho da fonte do elemento pai.
```
.exemplo {
  font-size: 2em; /* 2 vezes o tamanho da fonte do elemento pai
*/
}
```

2. **rem**: Relativa ao tamanho da fonte do elemento raiz (geralmente <html>).

```
.exemplo {
  font-size: 1.5rem; /* 1.5 vezes o tamanho da fonte do elemento
raiz */
}
```

3. **%: Percentual relativo ao tamanho do elemento pai.**
```
.exemplo {
  width: 50%; /* 50% da largura do elemento pai */
}
```

Unidades Absolutas

1. **px**: Pixels, uma unidade fixa.
```
.exemplo {
  margin: 20px; /* 20 pixels de margem */
}
```

2. **cm, mm**: Centímetros e milímetros.
```
.exemplo {
  width: 10cm; /* 10 centímetros de largura */
}
```

3. **in**: Polegadas.
```
.exemplo {
  height: 2in; /* 2 polegadas de altura */
}
```

4. **pt, pc**: Pontos (1/72 polegada) e picas (1/6 polegada).
```
.exemplo {
  font-size: 12pt; /* Tamanho da fonte em pontos */
}
```

Unidades de Viewport

1. **vw**: Largura da viewport (janela de visualização).
```
.exemplo {
  width: 50vw; /* 50% da largura da viewport */
}
```

2. **vh**: Altura da viewport.
```
.exemplo {
  height: 100vh; /* 100% da altura da viewport */
}
```

3. **vmin, vmax**: O menor ou maior valor entre largura e altura da viewport.
```
.exemplo {
  font-size: 10vmin; /* 10% do menor valor entre largura e altura
da viewport */
}
```

Posicionamentos

Um truque para centralizar um contêiner é utilizar o `margin: 10px auto;` dessa forma a margin direita e esquerda ficarão automáticas.

1. Posicionamento Estático (static)

- **Padrão**: Todos os elementos HTML têm a posição estática por padrão.
- **Comportamento**: O elemento segue o **fluxo normal do documento**, ou seja, ele fica onde deveria ficar na página de forma natural.
- **Propriedades como top, left, right, bottom NÃO funcionam** com static.

Exemplo:

```
div {
    position: static; /* Posição padrão */
    top: 10px; /* Não terá efeito */
    left: 20px; /* Não terá efeito */
}
```

---

2. Posicionamento Relativo (relative)

- **Comportamento**: O elemento **mantém seu espaço original** no fluxo do documento, mas pode ser deslocado **em relação a si mesmo** usando as propriedades top, left, right e bottom.
- É útil quando queremos mover o elemento sem afetar os outros elementos ao redor.

Exemplo:

```
div {
    position: relative;
    top: 10px; /* Move 10px para baixo */
    left: 20px; /* Move 20px para a direita */
```

```
}
```

- O espaço original do elemento **continua reservado** onde ele estaria normalmente, mas ele é deslocado visualmente.

---

3. Posicionamento Absoluto (absolute)

- **Comportamento:** O elemento **é removido do fluxo normal do documento**, ou seja, ele não ocupa mais espaço onde estava.
- Ele se posiciona em **relação ao seu conteiner mais próximo que tenha position: relative ou position: absolute ou fixed**.
- Caso não haja um conteiner com position, ele será posicionado **em relação ao body (ou ao elemento raiz)**.
- Não afeta os outros elementos, pois fica como se ele não existisse.
- acompanha a rolagem da página.

Exemplo:

```
div {
    position: absolute;
    top: 0; /* 0px a partir do topo do elemento pai */
    left: 0; /* 0px a partir da esquerda do elemento pai */
}
```

- **Importante:** Se um elemento pai tiver position: relative, o elemento absoluto se posiciona em relação a esse pai.

---

4. Posicionamento Fixo (fixed)

- **Comportamento**: O elemento é **removido do fluxo normal** e fica **fixo em relação à viewport** (a área visível da tela do navegador).
- Mesmo que você role a página para baixo ou para cima, o elemento **permanece fixo** na posição definida.
- É muito usado para menus fixos, cabeçalhos ou botões flutuantes.

Exemplo:

```
div {
    position: fixed;
    top: 0; /* Fixa no topo da página */
    left: 0; /* Fixa no lado esquerdo */
}
```

- O elemento **não se move** com o scroll da página.

Resumo Rápido:

Tipo	Ocupa Espaço Original?	Posição Baseada em:	Propriedades top, left, etc. Funcionam?
Static	Sim	Fluxo normal do documento	Não
Relative	Sim	Sua própria posição original	Sim
Absolute	Não	Ancestral posicionado ou body	Sim
Fixed	Não	Viewport (área visível do navegador)	Sim

Exemplos Práticos:

Se você tiver essa estrutura HTML:

```
<div class="parent">
    <div class="child">Filho</div>
</div>
```

E aplicar os seguintes estilos:

```
.parent {
    position: relative; /* Ancestor posicionado */
    width: 300px;
    height: 300px;
```

```
    background-color: lightblue;
}

.child {
    position: absolute; /* Posicionado em relação ao .parent */
    top: 50px;
    left: 50px;
    background-color: lightcoral;
}
```

Resultado:

- **O .child** se move para 50px do topo e 50px da esquerda, **dentro do .parent**.
- Isso acontece porque .parent tem position: relative.

5. Controle de Sobreposição de Elementos com z-index

Quando usamos posições **relativas**, **absolutas** ou **fixas**, os elementos podem se **sobrepor**. Para controlar qual elemento fica à frente, usamos a propriedade **`z-index`**. A ordem padrão, sem o uso do **`z-index`**, é definida pela estrutura do HTML. Ao aplicar o **`z-index`**, podemos alterar essa ordem: quanto maior o número, mais à frente o elemento ficará. Exemplo:

```
.conteudo1 {
    z-index: 2;
}
.conteudo2 {
    z-index: 1;
}
.conteudo3 {
    z-index: 1;
}
```

Se o **`z-index`** for o mesmo, **a ordem do HTML será seguida**. Essa propriedade é conhecida como **contexto de empilhamento**, pois permite controlar a sobreposição dos elementos.

Elementos Flutuantes Partes 1 e 2

Os elementos flutuantes são uma das formas de posicionar elementos na página, além do **flexbox** e do **grid**, embora não sejam tão utilizados atualmente. Essa técnica utiliza a propriedade **float**, que aceita os valores: `right`, `left` e `none`.

A propriedade **float** não leva em consideração os outros elementos. Portanto, se posicionarmos um elemento com **float** na mesma posição ocupada por outro elemento, ele será sobreposto. Para evitar esse comportamento de sobreposição, utilizamos a propriedade **clear** com o valor `both`. Essa propriedade pode ser aplicada em uma `<div>` (mesmo que vazia) ou na tag onde desejamos limpar a sobreposição, como um `<p>`. Geralmente, a `<div>` recebe uma classe para facilitar a organização. A propriedade **clear** aceita os valores: `right`, `left`, `none` e `both`.

Outro ponto importante: se houver uma caixa muito grande, o elemento com a propriedade **float** pode não conseguir aplicar o valor dependendo da situação. Além disso, sempre que utilizarmos elementos flutuantes, será necessário utilizar o **clear** no final do elemento.

Elementos Inline, Block e Inline-Block

Os elementos **block** ocupam toda a largura disponível na tela, e, por isso, são posicionados um abaixo do outro. Já os elementos **inline** ocupam apenas o espaço necessário para seu conteúdo, permitindo que fiquem alinhados lado a lado.

Por sua vez, os elementos **inline-block** combinam características dos dois: ocupam apenas o espaço necessário para seu conteúdo, como os elementos **inline**, mas podem ser manipulados em relação à largura e altura, como os elementos **block**. No entanto, eles também podem ficar posicionados lado a lado, diferentemente dos elementos **block**.

Podemos alterar o comportamento padrão de um elemento usando a propriedade **display**. Por exemplo:

```
display: inline-block;
```

Formatando Links

Obviamente, existe a formatação normal:

```
a{
```

```
    color: white;
    background-color: green;
}
```

Contudo, existem outras formas de fazer formatação em links, pois eles possuem estados específicos:

Links não visitados - é o estado normal de um link.

```
a:link {
    color: #b9c941;
}
```

Links visitados - links que já foram visitados.

```
a:visited {
    color: #c0c0c0;
}
```

Links hover: quando você passa o cursor sobre o link.

```
a:hover {
    color: #6d790f;
}
```

Links ativos: quando você clica nesse link.

```
a:active {
    color: #e4f371;
}
```

Geralmente, usamos a mesma formatação para links visitados e para não visitados:

```
a:visited, a:link {
    color: #b9c941;
}
```

FlexBox

Trata-se de uma caixa flexível que permite organizar, alinhar e distribuir o espaço entre os itens de um contêiner (elemento pai). Usamos ele alterando a propriedade display de algum elemento. Os itens ficam dentro desse contêiner e algumas propriedades são passadas para o pai, e outras para o filho.

Propriedades do pai

Por padrão a largura do contêiner ocupa toda a tela e o item só o espaço que ele está contendo a menos que nós definissémos um valor diferente para altura e largura
O **display: flex;** torna o contêiner flexível, ou seja, faz com que os filhos possam se ajustar automaticamente, ocupando o espaço disponível de maneira mais eficiente e controlada.
O gap serve para colocar espaçamento horizontal e vertical entre os itens sem que nós precisamos ficar colocando espaçamento (margin) em cada item:

Diferenças principais:

Aspecto	`Padding`	`Gap`
Espaçamento	Dentro do elemento, entre o conteúdo e as bordas	Entre elementos filhos no contêiner
Onde atua	Afeta o espaço interno do elemento	Afeta apenas o espaço entre os filhos
Afeta a área total?	Sim, aumenta o tamanho do elemento	Não, não afeta a área total
Suporte geral	Funciona em qualquer elemento	Funciona apenas em `flex` e `grid`

Diferenças principais:

Aspecto	`Gap`	`Margin`
Espaçamento	Entre os elementos filhos em um contêiner	Fora do elemento, entre outros elementos
Onde atua	Internamente, apenas entre filhos	Externamente, ao redor do elemento
Afeta o layout externo?	Não	Sim, cria espaço ao redor do elemento
Compatibilidade	`flex` e `grid`	Funciona em qualquer elemento

O **flex direction**: ; é usado para definir a direção dos itens dentro de um contêiner flexível (flex container) ou qual eixo será o principal (x ou y). Ela controla como os elementos filhos (flex items) são dispostos, podendo ser horizontal ou vertical, e também define a ordem dos elementos. Possui os valores row (padrão), row-reverse, column e o column-reverse. row=linha e column=coluna.

O **flex-wrap** indica se deve quebrar uma nova linha ou não quando não há espaço suficiente para os itens na linha principal. Por padrão, todos os itens tentam caber em uma única linha. Possui os valores nowrap (padrão), wrap, wrap-reverse.

O **flex-flow**: column wrap; é uma junção do flex-direction e o flex-wrap onde primeiro definimos a direção e depois o wrap.

O **justify-content**: center; serve para colocar uma div no meio, ou seja, o alinhamento dos itens ao longo do eixo principal possuindo os seguintes valores flex-start (padrão), flex-end, center, space-between, space-around, space-evenly.

O **align-items** controla o alinhamento dos itens ao longo do eixo cruzado (perpendicular ao eixo principal) em um contêiner flexível. Eixo cruzado: Se o flex-direction for row (padrão), o eixo cruzado será vertical já Se o flex-direction for column, o eixo cruzado será horizontal. Possui os valores flex-start, flex-end, center, baseline, stretch (padrão),
justify-content: Controla o alinhamento dos itens ao longo do eixo principal.
align-items: Controla o alinhamento ao longo do eixo cruzado (perpendicular ao principal).

gap: Podemos colocar o espaçamento na linha e na coluna respectivamente da seguinte forma gap: 10px 20px; ou separadamente com o row-gap e o column-gap.

Propriedades do filho

flex-grow: Determina a quantidade de espaço que o item deve ocupa dentro do contêiner.

flex-shrink: Determina o quanto os outros itens devem diminua em relação aos outros itens.

flex-basis: Determina um tamanho específico para o elemento.

flex: Junta todas as anteriores em apenas uma na ordem grow, shrink e basis.

align-self: Alinha apenas um elemento no centro como se fosse o align-items, mas para um único elemento. o self possui o center, flex-end, flex-start, baseline, stretch.

order: Define qual elemento ficará em qual parte da página, a ordem opadrão é zero então no caso de algum item possuir a order 2 irá pro final e se algum item tiver a order 3 o de order 2 irá para a penúltima posição.

PAREI NO MINUTO 10:57 DA LARISSA KICH (CSS3).