FUNDAMENTOS DE HTML5 E CSS3

com Kristopher Mazzini

Sumário

Módulo 1.....5
Lição 1: Objetivos do módulo.....6
Lição 2: Redes de internet e desenvolvimento *web*.....6
O que é a internet.....6
Protocolos de internet.....7
O caminho da internet.....8
Aplicações *web*.....9
Lição 3: Estrutura do HTML e marcações.....10
A sintaxe do HTML.....10
Principais *tags* e estruturas de HTML.....10
Lição 4: Construindo a primeira página.....12
Lição 5: Definindo estilos com CSS.....12
Lição 6: Estilizando elementos textuais.....13
Lição 7: Aprendendo sobre *box-model* e *box-sizing*.....14
Lição 8: Criando listas e usando *links*.....15
Lição 9: Navegando entre páginas.....16
Lição 10: Classes de CSS.....17
Lição 11: Elementos semânticos e divisão de conteúdo.....17
Lição 12: Posicionamento de elementos.....18
Lição 13: Finalizando nossa primeira página.....18
Lição 14: Analisando páginas da web com *DevTool*.....18
Lição 15: Desafio *Tech*.....19
Lição 16: 5 passos práticos para aplicar o que você aprendeu.....19
Módulo 2.....20
Lição 1: Objetivos do módulo.....21
Lição 2: Dispondo elementos com display *flex*.....21
Lição 3: Criando novos *layouts* com Flexbox.....23
Lição 4: Introdução ao CSS *Grid Layout*.....23
Lição 5: Avançando em Grid.....24
Track sizing.....24
Células e áreas do grid.....25
Lição 6: Aplicando Grid.....26
Lição 7: Qual utilizar: Grid ou Flexbox?.....26
Lição 8: Entendendo a entrada de dados.....27
Lição 9: Criando o primeiro formulário.....27

Lição 10: Elementos avançados do formulário 28
Lição 11: Estilização de formulário 28
Lição 12: Interatividade com pseudo-classes de CSS 28
Lição 13: Criando a primeira tabela 29
Lição 14: Estilização de tabelas 30
Lição 15: Inserindo elementos externos 30
Lição 16: Desafio *Tech* 30
Lição 17: 5 passos práticos para aplicar o que você aprendeu 31
Módulo 3 32
Lição 1: Objetivos do módulo 33
Lição 2: *Design* responsivo 33
Lição 3: Trabalhando com responsividade 34
Lição 4: A metodologia *mobile First* 34
Lição 5: Identificando e reproduzindo elementos: Parte 1 34
Lição 6: Identificando e reproduzindo elementos: Parte 2 34
Lição 7: Estilizando a página para *tablet* 34
Lição 8: Estilizando a página para *desktop* 34
Lição 9: Desafio *Tech* 35
Lição 10: 5 passos práticos para aplicar o que você aprendeu 35

Supervisão editorial: Thiago Coutinho de Oliveira

Apresentação do curso: Kristopher Mazzini

Produção de conteúdo: Ana Beatriz Baldow Almeida e Bartolomeu Henrique Lopes

Kristopher Mazzini, tem 24 anos, é formado em Engenharia Elétrica pela Universidade Federal de Juiz de Fora, tendo feito parte da graduação na Wrocław University of Science and Technology (Polônia), mas começou a atuar profissionalmente com engenharia de software antes mesmo de se formar. Desde o início da minha jornada profissional, já atuei desenvolvendo softwares para diversas empresas dos setores industrial, automobilístico, bancário, logístico, entre outros. Graças a toda essa experiência, hoje atua como líder técnico de equipes.

Módulo 1
Criando uma página da web

Lição 1: Objetivos do módulo

Neste primeiro módulo você aprenderá sobre **como a *internet* funciona**, o que são as **linguagens de HTML e CSS**, como **construir sua primeira página da *web* do zero** e ainda estará preparado para realizar um desafio especial.

Responderemos perguntas do tipo:

- ✓ O que é a internet e como ela funciona?
- ✓ O que é o HTML e o CSS e qual a importância deles no desenvolvimento?
- ✓ Como a estrutura da página impacta na indexação do site?
- ✓ Quais os elementos fundamentais de HTML e CSS?

Lição 2: Redes de internet e desenvolvimento *web*

O que é a internet

Neste curso, aprenderemos os primeiros passos do desenvolvimento *web*, uma importante área que permite a criação e manutenção de todas as interfaces que acessamos na nossa rotina. No entanto, precisamos entender alguns **conceitos fundamentais sobre o funcionamento da internet**, servidores e clientes antes de avançar no conteúdo.

Para começar, uma percepção muito comum e incorreta é de que internet e *World Wide Web* são a mesma coisa. A internet é **uma estrutura física**, composta por milhares de computadores interligados pelo mundo inteiro de forma integrada, enquanto a *World Wide Web* (WWW) é um protocolo que utiliza a estrutura da internet para troca de dados.

Para entender melhor, imagine que os aparelhos da sua casa, computadores, tablets e celulares, estejam interligados e enviem dados entre si em uma rede local. Para enviar uma mensagem ao seu vizinho, sua rede local deve se conectar à dele. É neste momento que entra a internet, uma rede de redes compartilhadas, e essa conexão é feita através dos serviços dos provedores de internet.

A ***World Web Wide***, ou apenas *web*, é um **protocolo de comunicação**, um sistema de hipermídia interligado de arquivos e informações que são executados através da internet. Esses dados podem estar na forma de hipertexto (ou seja, texto exibidos em formato digital), imagens, vídeos e áudios.

A *web* utiliza três elementos principais para funcionar: a **URL**, que dá um endereço único a cada página da internet e indica como ela será localizada; o **HTTP/ HTTPS**, protocolo que de

comunicação que veremos com mais detalhes a seguir; e o **HTML** (Hypertext Mark Language), **linguagem de marcação de texto** que codifica as páginas da internet e que é o foco deste curso.

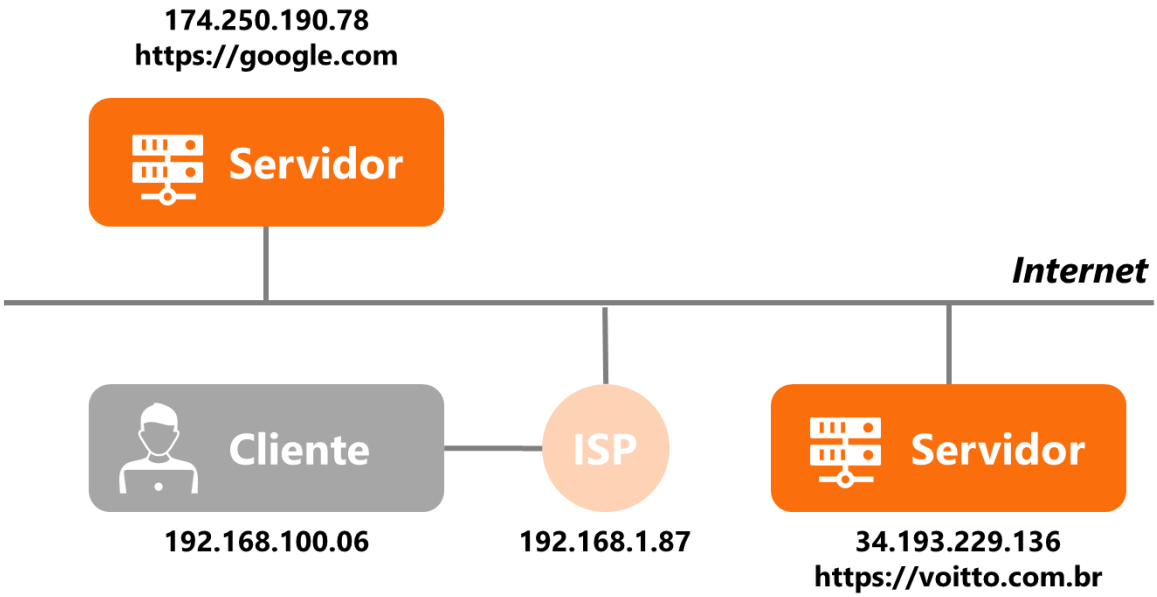
Protocolos de internet

Para que os computadores consigam interagir e essa rede funcione é necessário implementar uma série de regras de utilização, os protocolos de internet. Eles funcionam como uma linguagem universal que pode ser interpretada por computadores de todo o mundo.

A internet é dividida em diversas camadas de utilização e os protocolos são distribuídos entre elas de acordo com suas funções. Existem diversos protocolos de internet que valem a pena serem estudados, mas aqui abordaremos apenas os principais e relevantes para este curso:

- ✓ **IP – *Internet Protocol***, ou endereço do protocolo de internet, é um dos mais importantes protocolos da web, é como um documento de identidade virtual do seu dispositivo, que o identifica dentre os demais;
- ✓ **TCP – *Transmission Control Protocol***, ou protocolo de controle de transmissão, está localizado na capada de transporte da internet e é responsável por dividir as informações em pacotes menores;
- ✓ **TCP/IP** – a combinação dos dois protocolos anteriores dá origem ao protocolo responsável pela base de envio e recebimento de dados por toda a internet, e está dividido entre as camadas de aplicação (receber e enviar dados), transporte (transporta os dados recebidos na camada de aplicação), rede (anexa os arquivos empacotados no IP correspondente) e interface (executa o recebimento ou envio de dados na página da web);
- ✓ **HTTP/HTTPS – *Hypertext Transfer Protocol***, protocolo de transferência de texto, é um importante elemento da *web* para navegação, funcionando como uma ponte entre o cliente (browser) e servidor. O *Hyper Text Transfer Secure*, protocolo de transferência de hipertexto seguro, funciona da mesma forma, porém com uma camada a mais de segurança, normalmente oferecida por serviços pagos;

Na imagem abaixo temos um exemplo que aborda esses protocolos:



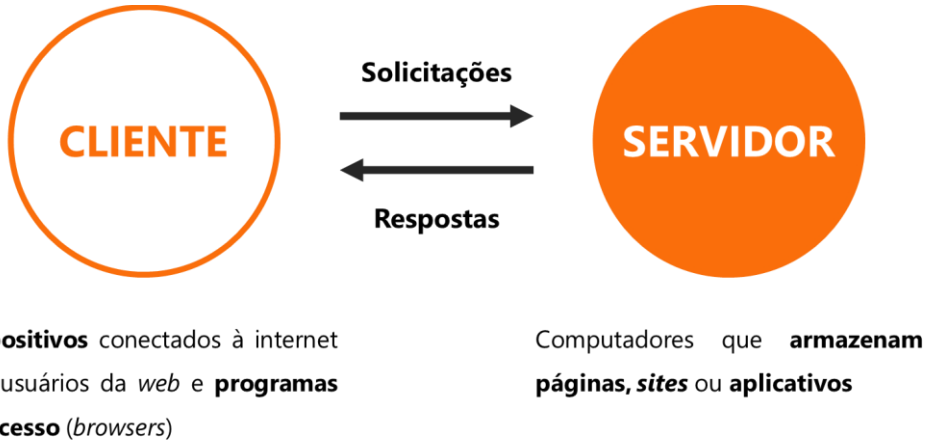
O caminho da internet

Agora que já fomos apresentados aos elementos principais da internet, vamos entender a organização da internet e o caminho que ela percorre até nossos aparelhos pessoais.

Os elementos físicos da internet não estão centralizados em um supercomputador. São diversos elementos, como modens, roteadores e *backbones*, conectados através de quilômetros de cabos e ondas de rádio.

A internet é uma **grande malha de interações cliente-servidor**. O cliente, dispositivos conectados à internet dos usuários da *web* e programas de acesso (*browser*), sempre inicia o processo de comunicação com um servidor. Quando acessamos um site, por exemplo, o navegador envia uma requisição de acesso ao nosso serviço de internet, que procura o computador em que o dado site está hospedado. Este computador é chamado de servidor e ele pode armazenar sites, páginas, aplicativos e outros arquivos.

O servidor *web* recebe a requisição, envia uma resposta à operadora e essas informações são repassadas para nosso computador. Todo esse processo de comunicação é feito por **meio do protocolo HTTP/HTPPS**.



Os *backbones*, espinhas dorsais, são os responsáveis por ligar as operadoras de internet aos servidores externos, nacionais ou internacionais. São elementos essenciais para que as informações circulam pela internet, em grande velocidade e volume.

Aplicações *web*

Já sabemos como a internet é ligada e como as informações circulam. Agora vamos entender melhor sobre as aplicações *web*, uma das áreas mais promissores para desenvolvedores. Aplicações *web* são soluções executadas diretamente pelo navegador, sem a necessidade de instalação no aparelho, que atendem a alguma necessidade do usuário. Elas são desenvolvidas para serem acessadas em qualquer computador ou aparelho mobile, através de um navegador, e podem variar de páginas simples e estáticas a programas mais complexos.

Quando queremos acessar uma aplicação *web*, nossa solicitação é enviada até o servidor com o método HTTP/HTTPS, a página que será acessada e os parâmetros do formulário. O servidor então encontra a página solicitada e retorna o conteúdo em formato HTML, que será interpretado e compilado em formato visual pelo nosso navegador.

Para desenvolver uma aplicação *web*, podemos dividir as áreas de desenvolvimento em duas: ***frontend* e *backend***. As tecnologias de ***frontend*** operam do lado do usuário, ou seja, na máquina pessoal dele, criando interfaces e lógicas de conexão com o servidor, e as principais são **HTML, CSS e JavaScript**. Já as tecnologias de ***backend***, como **PHP e Python**, operam do lado do servidor, tratando dados e outras estruturas importantes para o funcionamento da aplicação.

A partir de todas as informações, podemos concluir que um website e uma aplicação web são a mesma coisa? A resposta é não. Apesar de estarem no mesmo ambiente, existem algumas diferenças práticas entre ambos.

O que chamamos de sites são tradicionalmente páginas que carregam informações específicas de determinado produto ou serviço e a navegação se dá por meio de links. Já as aplicações web, de forma mais generalista, são sistemas completos hospedados em determinado

servidor, que permite que o usuário realize mais ações do que em um website, como fazer login, baixar documentos, cadastrar informações, entre outros.

Lição 3: Estrutura do HTML e marcações

Já entendemos como a internet funciona, os protocolos necessários para que ela exista, a função do HTML em aplicações *web* e muito mais. Neste capítulo vamos começar, de fato, nossos estudos em HTML.

Hypertext Markup Language, ou simplesmente HTML, linguagem de marcação de hipertexto, é o elemento básico da composição de sites e aplicações *web*, utilizado para organizar o conteúdo da página.

Hipertexto é um conjunto de palavras, imagens, vídeos e documentos conectados e organizados em uma rede de dados e informações. O **HTML é o esqueleto da sua página**, o que o navegador recebe do servidor, lê e renderiza na sua tela.

Principais tags e estruturas de HTML
A sintaxe do HTML

A sintaxe do HTML

O HTML é formado, basicamente, por ***tags***. *Tags* são códigos que definem toda a **estrutura** da página, a **funcionalidade** e **hierarquia** semântica de dado elemento, definidos pelos caracteres < e >, desta forma: <tag>. Um elemento de HTML, geralmente, é composto pela *tag* de abertura, conteúdo e tag de fechamento, desta forma: <tag> Conteúdo </tag>. No entanto, existem *tags* que não possuem um conteúdo por si só e sim carregam elementos externos, e por isso fogem da estrutura padrão.

Podemos também adicionar informações extras dentro da *tag*, os chamados atributos. Eles aparecem na forma **atributo = ”valor”** e em breve veremos os principais atributos dentro do HTML.

Principais *tags* e estruturas de HTML

O HTML é dividido em várias *tags*, com funcionalidades distintas. Vamos conhecer agora as *tags* e estruturas mais utilizadas e importantes do HTML, suas definições e exemplos de aplicação:

- ✓ **<!DOCTYPE html>** é o elemento por onde informamos ao navegador qual versão de HTML estamos usando. Apesar da sintaxe parecida, não é uma *tag* e deve ser escrita na primeira linha de código, antes de toda a estrutura HTML;
- ✓ **<html>** é o elemento raiz de HTML, que define e delimita o documento HTML;

✓ **<head>** é onde declaramos as informações que não são exibidas na página, como título, metadados, importação de estilos, fontes, e outros elementos;

✓ **<body>** é a *tag* onde se encontra os elementos que compõem o corpo da página, os títulos, artigos, barras de navegação, imagens, listas, entre outros;

```
<!DOCTYPE html>
<html>
        <head>
            <!- -METADADOS- ->
        </head>

        <body>
            <!- - CONTEÚDO - ->
        </body>
</html>
```

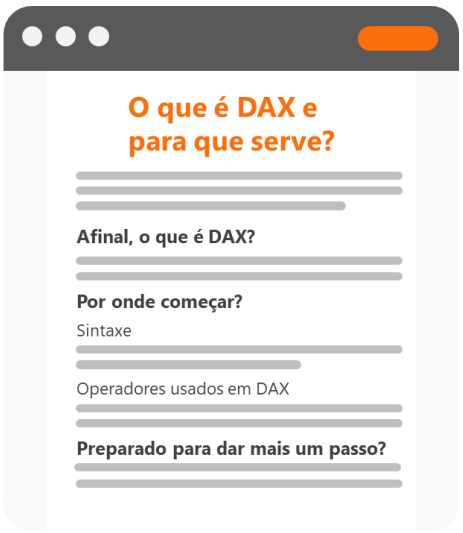
No <head> podemos encontrar diversos tipos de metadados, como:

✓ **<meta charset =" ">** é o que indica ao navegador qual conjunto de caracteres queremos utilizar. A tag <meta> é capaz de definir metadados, informações, ao arquivo HTML, enquanto o atributo charset é o que define o formato de codificação. De todas as codificações, a UTF-8 é a mais completa e utilizada, abrangendo o maior número de caracteres e idiomas;

✓ **<title>** é onde definimos o título da página;

No <body> é onde colocaremos os conteúdos, ao longo do curso iremos ver diversas *tags* mais específicas, de acordo com a funcionalidade de cada um. Também é possível organizar conjuntos destes elementos em blocos, tanto para fins de organização semântica quanto estética. Na imagem abaixo temos um exemplo da hierarquia semântica dessas *tags*:

```
<h1>O que é DAX e para que serve?</h1>
<h2>Afinal, o que é DAX?</h2>
<h2>Por onde começar?</h2>
<h3>Sintaxe</h3>
<h3>Operadores usados em DAX</h3>
<h2>Preparado para dar mais um passo?</h2>
```



Lição 4: Construindo a primeira página

Acompanhe nossa aula prática em vídeo.

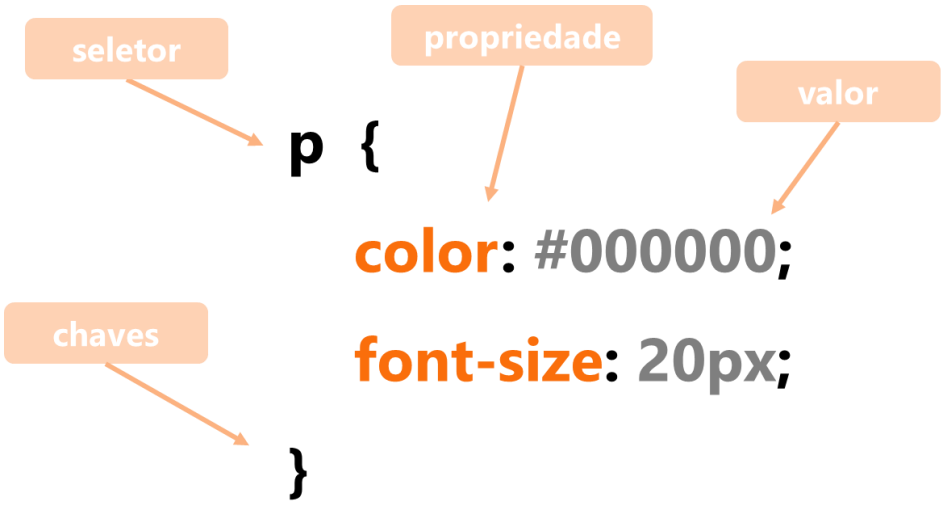
Lição 5: Definindo estilos com CSS

A *Cascading Style Sheets* (CSS) é uma linguagem de estilo utilizada para descrever a apresentação de documentos escritos em HTML ou XML. É usada para **controlar a formatação, layout e cores de um site**.

A estrutura básica de um arquivo CSS consiste em três elementos principais: **seletores, propriedades e valores**.

- ✓ **Seletores:** São os elementos HTML aos quais você deseja aplicar estilos. Pode ser uma tag HTML específica (como "p" ou "h1"), uma classe CSS específica (começando com um ponto ".") ou um identificador único (começando com um sinal de #).
- ✓ **Propriedades:** São as características que você deseja definir para o seletor. Exemplos incluem cor de fundo, tamanho da fonte, espaçamento entre linhas, etc.
- ✓ **Valores:** São os valores que você deseja atribuir às propriedades. Por exemplo, o valor da propriedade de cor de fundo pode ser "vermelho" ou "# ff0000".

A sintaxe básica de uma regra CSS é a seguinte:



CSS permite que você aplique estilos de forma organizada e eficiente para todo o seu site, o que significa que você pode fazer mudanças de estilo em larga escala rapidamente e sem precisar alterar cada página individualmente.

É importante entender a prioridade de CSS para garantir que seus estilos sejam aplicados da maneira desejada. A prioridade em CSS é determinada por diversos fatores, incluindo:

1. **Especificidade:** Quanto mais específico for o seletor, maior será a sua prioridade. Por exemplo, um identificador único (#id) tem prioridade sobre uma classe (.class) que, por sua vez, tem prioridade sobre uma *tag* HTML (p).
2. **Ordem de declaração:** As regras CSS que aparecem mais tarde no arquivo CSS terão prioridade sobre as regras que aparecem mais cedo. Isso significa que uma regra CSS declarada depois de outra regra CSS para o mesmo seletor sobrescreverá a primeira regra.
3. ***Inline styles:*** Estilos específicos que são incluídos diretamente no elemento HTML têm prioridade máxima sobre todas as outras regras CSS.

4. **Importância (!important):** Se você adicionar a declaração "!important" ao final de uma regra CSS, essa regra terá prioridade sobre todas as outras regras, incluindo inline styles. No entanto, é uma boa prática evitar o uso excessivo de !important, pois pode tornar o código CSS difícil de manter e entender.

Em caso de conflito de prioridade, o navegador seguirá essas regras para decidir qual regra aplicar.

Lição 6: Estilizando elementos textuais

Agora que já começamos a estilizar nossa página, é interessante conhecer um pouco mais sobre o conceito de cores.

Os alfabetos para computação são um pouco diferentes do alfabeto comum que conhecemos. Por exemplo, temos o binário, onde trabalhamos com 0 e 1. Temos o numérico, que vai do 0 ao 9. E aí conseguimos representar todos os números com isso. Ainda temos o **dicionário hexadecimal**, que é a mesma coisa que o dicionário numérico adicionando as letras abcdef.

Com esse dicionário conseguimos representar muito mais coisas do que só com o dicionário numérico. É ele que usamos para marcar cores. Uma outra forma de representar as cores é através da representação **RGB (*Red Green Blue*).**

Ambas as representações são baseadas na ideia de que uma cor é composta pelos componentes primários vermelho, verde e azul (RGB), mas elas representam esses componentes de maneiras diferentes.

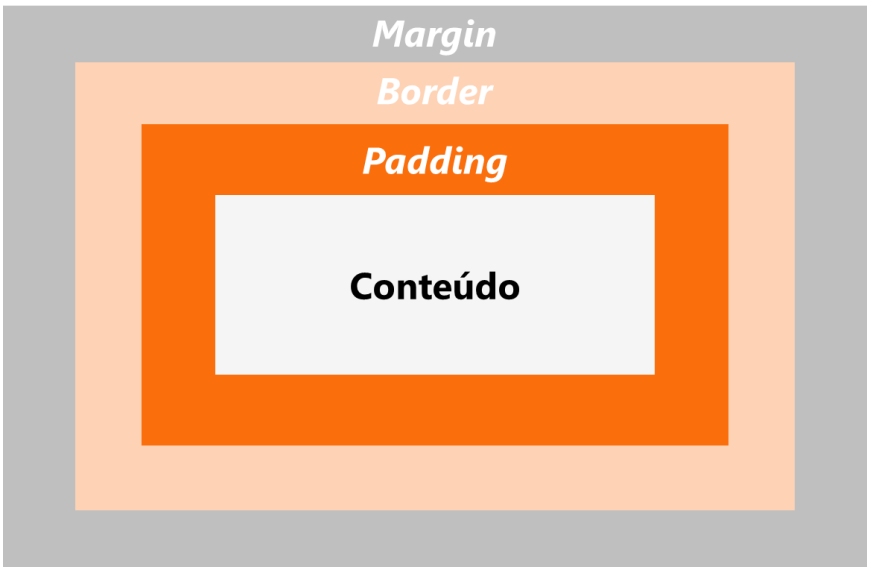
A representação RGB especifica cada componente como um valor decimal entre 0 e 255. Por exemplo, a cor vermelha mais intensa seria representada como RGB(255, 0, 0).

A representação hexadecimal, por outro lado, especifica cada componente como dois dígitos hexadecimais. Cada dígito hexadecimal representa um valor numérico entre 0 e 15, e esses valores são combinados para produzir uma ampla gama de cores. Por exemplo, a cor vermelha mais intensa seria representada como "#ff0000".

Ambas as representações são amplamente usadas em desenvolvimento *web* e são suportadas pelos principais navegadores. A representação **RGB é mais fácil de entender** para pessoas que têm conhecimento de programação, enquanto a representação **hexadecimal é mais fácil de ler e digitar**. Ao escolher entre as duas representações, você pode levar em consideração suas preferências pessoais e suas necessidades específicas de projeto.

Lição 7: Aprendendo sobre *box-model* e *box-sizing*

Cada elemento em uma página *web* é modelado como uma "caixa", com seu conteúdo no interior e espaçamentos (margens, bordas e *paddings*) em torno dele. O tamanho total de uma caixa é calculado como a soma de todas essas áreas.



O ***box-model*** e o ***box-sizing*** são conceitos fundamentais em CSS que controlam como o tamanho de um elemento é calculado e como as propriedades de estilo, como largura, altura, margens, bordas e paddings, afetam o tamanho total de uma caixa.

O *box-model* define que o tamanho de um elemento inclui não apenas seu conteúdo (como texto ou imagens), mas também sua largura, altura, margens, bordas e *paddings*. Cada elemento em uma página *web* é modelado como uma "caixa", com seu conteúdo no interior e espaçamentos (margens, bordas e *paddings*) em torno dele.

Já a propriedade *box-sizing* permite **controlar como o *box-model* é aplicado a um elemento**, oferecendo a possibilidade de incluir ou não as bordas e os *paddings* no cálculo do tamanho total da caixa. Isso pode ser útil em situações onde é necessário controlar com precisão o tamanho de um elemento, sem que as bordas e *paddings* afetem suas dimensões.

Se o *box-sizing* for definido como ***"content-box"***, o tamanho da caixa será calculado como descrito acima, incluindo a largura, altura, margens, bordas e *paddings*. Se o *box-sizing* for definido como ***"border-box",*** o tamanho da caixa será calculado de uma maneira diferente: a largura e a altura incluirão não apenas o conteúdo, mas também as bordas e os *paddings*. Isso significa que, se você definir uma largura e uma altura para um elemento, o tamanho real do elemento não será afetado pelas bordas ou *paddings* adicionados.

Aqui está um exemplo de como usar a propriedade *box-sizing* para mudar o comportamento do *box-model*:

```
elemento {
  box-sizing: border-box;
  width: 200px;
  height: 100px;
  border: 10px solid black;
  padding: 20px;
}
```

Neste exemplo, mesmo que tenhamos adicionado uma borda de 10 pixels e um *padding* de 20 pixels, o tamanho total do elemento será de 200 pixels de largura e 100 pixels de altura, pois estamos usando o *box-sizing "border-box"*.

Lição 8: Criando listas e usando *links*

Listas e *links* são elementos importantes em HTML que permitem apresentar informações de maneira organizada e oferecer navegação para outras páginas ou recursos. A *tag* **<ul>** define uma **lista não ordenada**, enquanto a *tag* **<ol>** define uma **lista ordenada**. Dentro de cada lista, cada item é definido com a *tag* **<li>**.

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>

<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ol>
```

Já para a criação de *links*, utilizamos a *tag* <a>. O atributo **href** define o **destino do *link***, que pode ser uma página *web*, arquivo ou qualquer outro recurso na internet, como vídeos no YouTube ou alguma localização no Google Maps. O conteúdo dentro da *tag* **<a>** define o texto exibido como link.

```
<a href="https://www.voitto.com.br">Link para o site da Voitto</a>
```

Lembrando que, além de *links* externos, também é possível criar *links* internos na sua página, usando o atributo href com o endereço da seção desejada na sua página, como por exemplo:

```
<a href="#page1">Ir para a página 1</a>

...

<h2 id=“page1">Página 1</h2>
```

Neste exemplo, ao clicar no link "Ir para a página 1", o usuário será redirecionado para a seção com o id "page1" na mesma página.

Lição 9: Navegando entre páginas

Acompanhe nossa aula prática em vídeo.

Lição 10: Classes de CSS

Nessa lição conheceremos o conceito de classes dentro do CSS. Classes em CSS são uma **maneira de aplicar estilos a elementos HTML de maneira seletiva**. Ao invés de aplicar estilos a todos os elementos de uma determinada *tag*, você pode usar classes para aplicar estilos apenas a elementos específicos que tenham a classe em questão.

Por exemplo, imagine que você deseja aplicar uma fonte diferente apenas aos títulos de uma seção em sua página. Em vez de alterar a fonte de todos os títulos na página, você pode usar uma classe para aplicar a fonte apenas aos títulos desejados.

Além disso, é possível usar classes em vários elementos diferentes na página, permitindo aplicar estilos de maneira consistente e organizada. É comum usar classes para aplicar estilos comuns a elementos semelhantes, como botões, ícones, etc.

Lição 11: Elementos semânticos e divisão de conteúdo

Elementos semânticos são elementos HTML que oferecem **significado semântico** ao conteúdo, ao invés de simplesmente definir a aparência. Eles ajudam a tornar o **código HTML mais estruturado, claro e acessível**, facilitando a compreensão do conteúdo para o usuário e para os mecanismos de busca.

Alguns exemplos de elementos semânticos incluem:

- ✓ **<header>:** usado para definir o cabeçalho de uma página ou seção.
- ✓ **<nav>:** usado para definir a navegação de uma página.
- ✓ **<main>:** usado para definir o conteúdo principal da página.
- ✓ **<article>:** usado para definir um conteúdo autônomo, como uma postagem de blog ou notícia.
- ✓ **<section>:** usado para definir uma seção distinta do conteúdo.
- ✓ **<aside>:** usado para definir um conteúdo secundário, como uma barra lateral ou bloco de anúncios.
- ✓ **<footer>:** usado para definir o rodapé de uma página ou seção.

Ao usar elementos semânticos, você pode tornar o conteúdo da sua página mais claro e fácil de ser entendido, além de ajudar a melhorar a acessibilidade e a otimização para mecanismos de busca.

A divisão de conteúdo é a técnica de separar o conteúdo em seções lógicas, utilizando elementos semânticos, como mencionados acima, ou divisões ** **, que não possuem significado semântico. Ao separar o conteúdo em seções, fica mais fácil aplicar estilos e organizar a estrutura da página, além de tornar o conteúdo mais claro e acessível.

Lição 12: Posicionamento de elementos

As propriedades de posição do CSS permitem controlar a posição de elementos em uma página. Há quatro valores principais para a propriedade position:

- ✓ **static:** é o valor padrão e significa que o elemento é posicionado na página da forma padrão, seguindo a ordem de fluxo do conteúdo.
- ✓ **relative:** significa que o elemento é posicionado de forma relativa ao seu local original no fluxo de conteúdo. Você pode usar as propriedades *top*, *right*, *bottom* e *left* para ajustar a posição do elemento em relação ao seu local original.
- ✓ **absolute:** significa que o elemento é posicionado fora do fluxo de conteúdo e pode ser ajustado usando as propriedades *top*, *right*, *bottom* e *left*. Se um elemento pai tiver posição relative, o elemento filho com posição absolute será posicionado em relação a esse elemento pai.
- ✓ **fixed:** significa que o elemento é posicionado fora do fluxo de conteúdo e mantém sua posição na tela mesmo quando a página é rolada. Você pode usar as propriedades *top*, *right*, *bottom* e *left* para ajustar a posição do elemento na tela.

Além disso, a propriedade ***float*** pode ser usada para **flutuar elementos na página**, permitindo que outros elementos sejam posicionados ao redor deles.

É importante ter cuidado ao usar as propriedades de posição, pois elas podem afetar a forma como o conteúdo é exibido na página e pode levar a problemas de *layout* se não forem usadas corretamente.

Lição 13: Finalizando nossa primeira página

Acompanhe nossa aula prática em vídeo.

Lição 14: Analisando páginas da web com *DevTool*

O DevTools é um **conjunto de ferramentas** de desenvolvimento integradas ao navegador web, que permite aos desenvolvedores inspecionar, depurar e testar o código de uma página web.

Com a ferramenta DevTools, os desenvolvedores **podem visualizar e modificar o HTML, CSS e JavaScript** de uma página em tempo real, sem precisar recarregar a página. Isso facilita o processo de desenvolvimento e otimização de páginas, pois os desenvolvedores podem ver imediatamente as alterações que eles fazem no código.

Além disso, ela também fornece recursos avançados, como a capacidade de inspecionar elementos da página, verificar o console de erros, depurar JavaScript, testar a performance de uma página e muito mais.

Em resumo, a ferramenta DevTools é uma ferramenta **essencial para qualquer desenvolvedor *web***, pois permite acesso rápido e fácil a informações importantes sobre o código de uma página, além de fornecer recursos avançados para testar e otimizar a performance de uma página *web*.

Lição 15: Desafio *Tech*

Para fixar seus conhecimentos, propomos um desafio. Você pode conferir na aula em vídeo e o gabarito será disponibilizado ao final do módulo.

Lição 16: 5 passos práticos para aplicar o que você aprendeu

1. Domine os **conceitos da *internet***: esse conhecimento será de grande valia para sua carreira;
2. Aprenda bem a **estrutura fundamental** de uma página;
3. Entenda a importância da utilização de **marcações semânticas** para seu código;
4. Seja familiarizado com os **espaçamentos internos e externos** de elementos;
5. Utilize **classes** e **hierarquização de CSS** para facilitar a estilização da sua página**.**

Módulo 2

Criando e posicionando novos elementos

Lição 1: Objetivos do módulo

Neste módulo vamos aprofundar em novos elementos, como **tabelas** e **formulários**, utilizando **entrada de dados** e estilizações mais refinadas, além de explorar as possibilidades de **posicionamento** de elementos com ***grid*** e ***flexbox***.

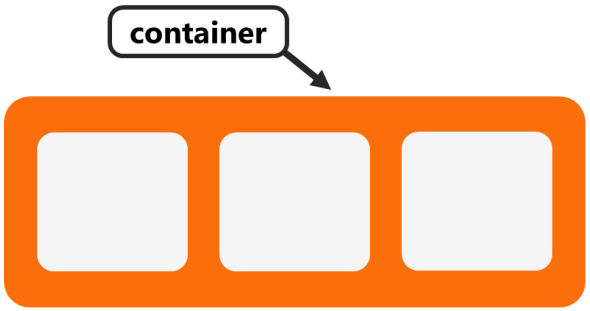
Responderemos perguntas do tipo:

- ✓ Quais tipos de entradas dentro do HTML e como o navegador interpreta cada uma delas?
- ✓ Como tornar o site mais dinâmico sem se aprofundar em JavaScript?
- ✓ Como posicionar elementos de forma mais prática e otimizada?

Lição 2: Dispondo elementos com display *flex*

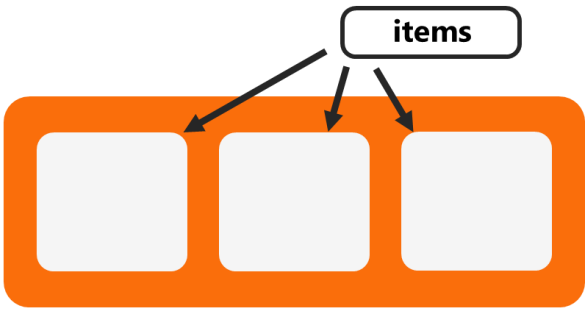
O display flexbox é uma propriedade de layout do CSS que permite aos desenvolvedores **criar *layouts* flexíveis e responsivos**, especialmente para dispositivos móveis. Ele permite que os elementos filhos de um container sejam dispostos em linhas ou colunas, com controle sobre a distribuição de espaço e a ordem dos elementos.

A propriedade **display: flex** é aplicada ao elemento pai, ou container, e define que os elementos filhos são itens flexíveis.



Propriedades do elemento pai

(*flex container*)

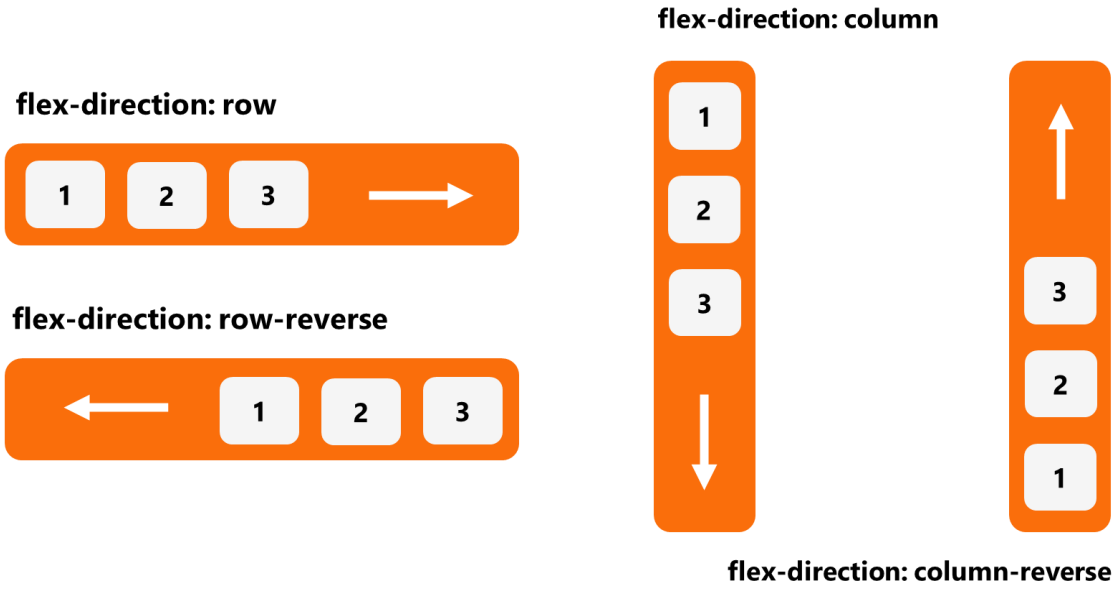


Propriedades do elemento filho

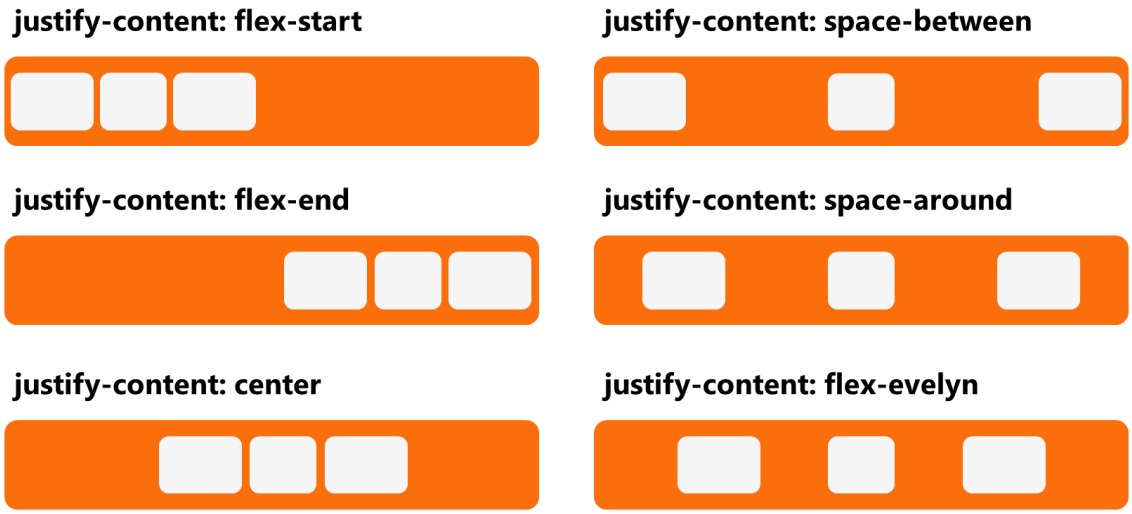
(*flex items*)

Além disso, existem outras propriedades que podem ser usadas em conjunto com o display flexbox para controlar o comportamento dos elementos filhos, como:

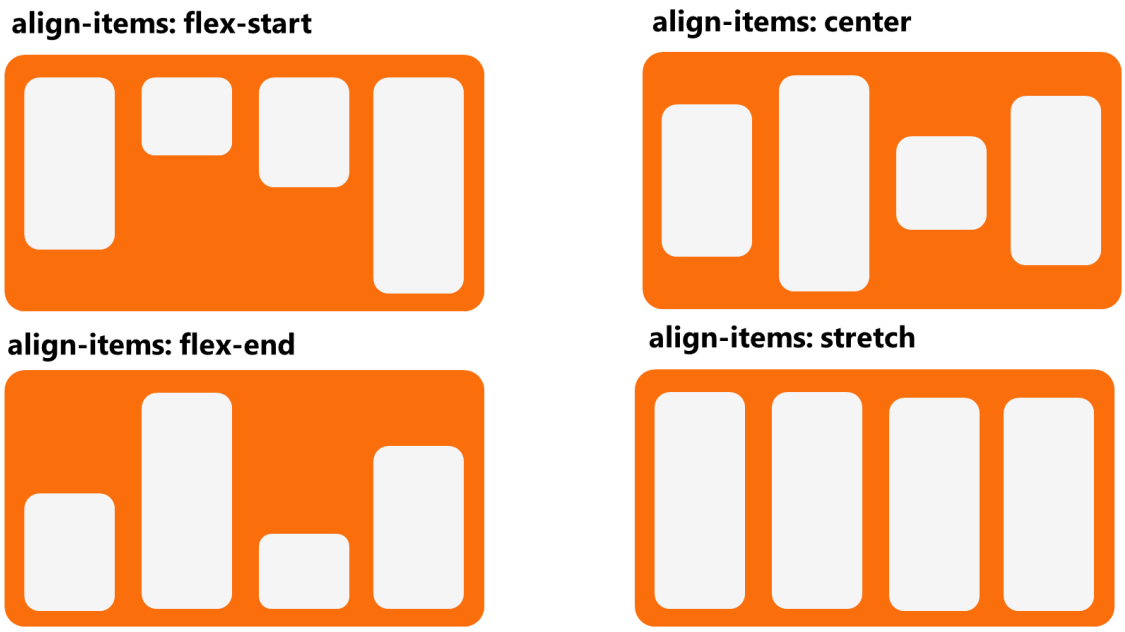
- ✓ **flex-direction:** define a direção dos itens flexíveis, se em linha ou coluna.



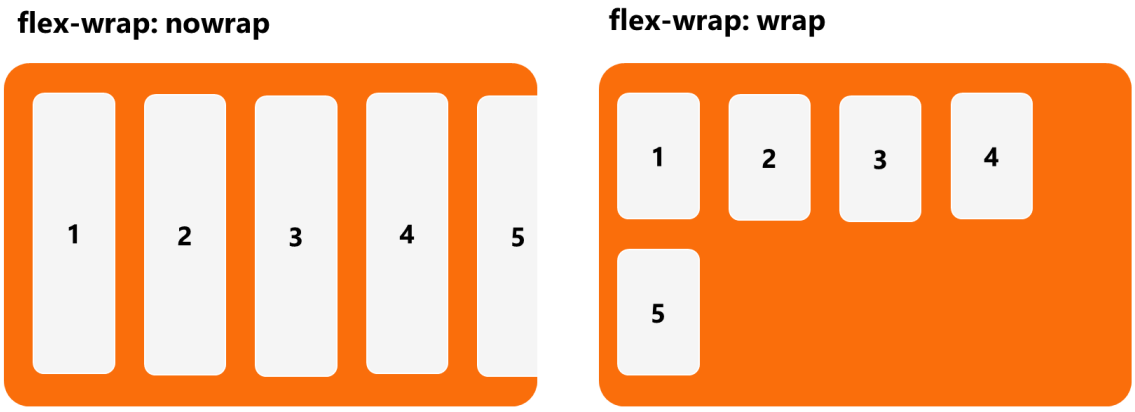
✓ **justify-content:** define como os itens flexíveis são distribuídos ao longo do eixo principal.



✓ **align-items:** define como os itens flexíveis são alinhados ao longo do eixo secundário.



✓ **flex-wrap:** define se os itens flexíveis devem ser divididos em mais de uma linha ou coluna quando não há espaço suficiente na direção principal.



Lição 3: Criando novos *layouts* com Flexbox

Acompanhe nossa aula prática em vídeo.

Lição 4: Introdução ao CSS *Grid Layout*

O display Grid Layout é uma propriedade de *layout* do CSS que permite aos desenvolvedores criar *layouts* de grade **complexos e estruturados**. Ele permite que os elementos filhos de um container sejam dispostos em linhas e colunas, com controle sobre o posicionamento e o tamanho dos elementos, como mostrado no exemplo abaixo:

A propriedade **display: grid** é aplicada ao elemento pai, ou container, e define que os elementos filhos são itens na grade. Além disso, existem outras propriedades que podem ser usadas em conjunto com o display Grid Layout para controlar o comportamento dos elementos filhos, como:

- ✓ **grid-template-columns:** define o tamanho das colunas da grade.
- ✓ **grid-template-rows:** define o tamanho das linhas da grade.
- ✓ **grid-template-areas:** define áreas nomeadas na grade, permitindo que os elementos filhos sejam posicionados de forma mais fácil e intuitiva.
- ✓ **grid-column-start:** define a coluna de início de um item na grade.
- ✓ **grid-column-end:** define a coluna de fim de um item na grade.
- ✓ **grid-row-start:** define a linha de início de um item na grade.
- ✓ **grid-row-end:** define a linha de fim de um item na grade.

O Grid é uma opção mais avançada do que o display flexbox e permite a criação de *layouts* mais complexos e estruturados. No entanto, também é mais difícil de aprender e usar do que o display flexbox. Além disso, o suporte ao display Grid Layout não é tão amplo quanto o suporte ao display flexbox, então é importante verificar se ele é compatível com as plataformas e navegadores que você precisa antes de usá-lo.

Lição 5: Avançando em Grid

Além das propriedades básicas descritas acima, existem algumas outras propriedades do display Grid Layout que permitem ainda mais flexibilidade e controle sobre a apresentação dos elementos filhos em um container.

Track sizing

Track sizing é um termo que se refere ao **tamanho das colunas e linhas** no *grid*. Em outras palavras, é como as colunas e linhas são dimensionadas e ajustadas para caber o conteúdo. No CSS Grid Layout, existem três tipos diferentes de *track sizing*:

- ✓ **fr (*fraction unit*):** essa unidade permite definir o tamanho de uma coluna ou linha como uma fração da grade disponível. Por exemplo, se você tiver uma grade com duas colunas, uma com o tamanho 1fr e outra com o tamanho 2fr, a primeira coluna ocupará 33,33% da grade e a segunda ocupará 66,66%.
- ✓ **px (pixels):** essa unidade permite definir o tamanho de uma coluna ou linha em pixels. Por exemplo, se você definir uma coluna com o tamanho 100px, ela sempre terá 100 pixels de largura, independentemente do tamanho da grade ou do conteúdo.
- ✓ **% (porcentagem):** essa unidade permite definir o tamanho de uma coluna ou linha como uma porcentagem da grade disponível. Por exemplo, se você definir uma coluna com o tamanho 50%, ela ocupará 50% da grade disponível.

O tipo de *track sizing* que você escolhe dependerá do tipo de *layout* que você está criando e das suas necessidades específicas. Por exemplo, se você quiser que uma coluna tenha sempre o mesmo tamanho, independentemente do tamanho da grade ou do conteúdo, você pode usar pixels. Se, por outro lado, você quiser que uma coluna tenha um tamanho proporcional ao tamanho da grade, você pode usar frações ou porcentagens.

Células e áreas do grid

As células de uma grade são as unidades de espaço na grade, cada uma ocupando uma coluna ou linha específica. Cada item dentro do grid é posicionado em uma ou mais células.

As áreas do grid são **regiões específicas na grade**, definidas pelo desenvolvedor, onde os itens podem ser posicionados. As áreas são criadas usando a propriedade **grid-template-areas**. Essa propriedade permite especificar o nome de uma área, que pode ser referenciado posteriormente quando posicionamos os itens na grade.

Por exemplo, se você quiser criar uma grade com duas colunas e três linhas, com as áreas "header", "sidebar" e "main", você pode escrever o seguinte:

```
.grid {
  display: grid;
  grid-template-columns: 1fr 2fr;
  grid-template-rows: auto;
  grid-template-areas:
    "header header"
    "sidebar main"
    "sidebar main";
}
```

Em seguida, você pode posicionar itens na grade usando a propriedade **grid-area**. Essa propriedade permite especificar em que área um item deve ser posicionado.

Por exemplo, se você quiser posicionar um item na área "header", você pode escrever o seguinte:

```
.header {
  grid-area: header;
}
```

Em conclusão, o Display Grid Layout é uma das ferramentas mais poderosas e versáteis para criar *layouts* na *web*. Ele permite posicionar elementos de maneira **precisa e controlada**, independentemente do tamanho da tela ou da resolução.

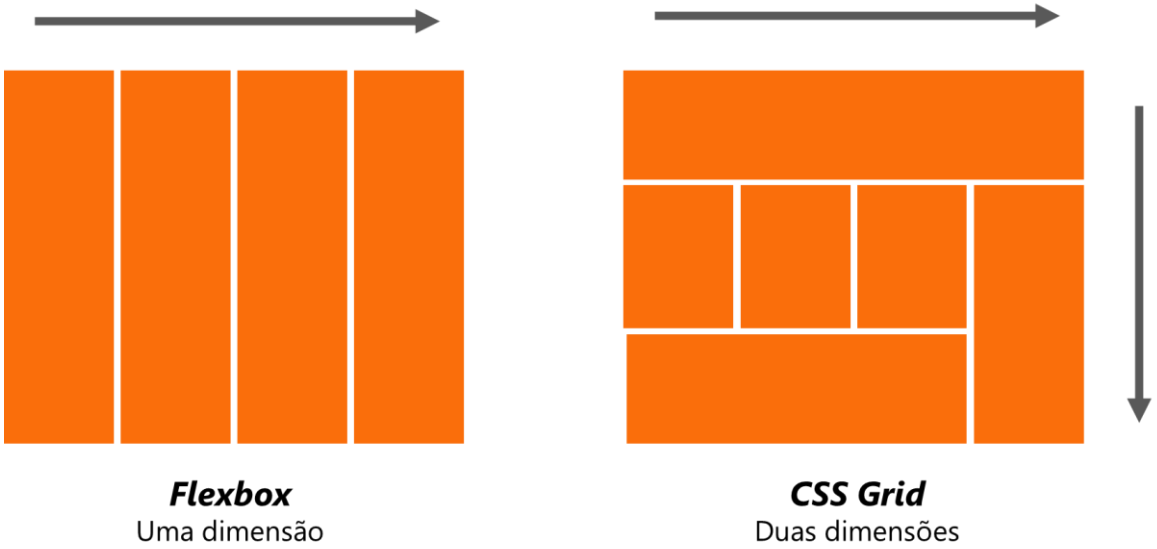
Lição 6: Aplicando Grid

Acompanhe nossa aula prática em vídeo.

Lição 7: Qual utilizar: Grid ou Flexbox?

A escolha entre Flexbox e Grid depende das necessidades específicas de cada projeto. Aqui estão alguns fatores a serem considerados ao escolher entre as duas ferramentas:

- ✓ ***Layout*:** Se você precisa de um *layout* simples e em linha ou coluna, a Flexbox é a escolha ideal. Se você precisa de um *layout* mais complexo e bidimensional, o Grid é a melhor opção.
- ✓ **Alinhamento:** Se você precisa de alinhamento em uma única linha ou coluna, a Flexbox é uma boa escolha. Se você precisa de alinhamento em duas dimensões, o Grid é a melhor opção.



Flexbox
Uma dimensão

CSS Grid
Duas dimensões

- ✓ **Direção:** Se você precisa de uma direção definida para os itens, a Flexbox é uma boa escolha. Se você precisa de uma direção mais flexível, o Grid é a melhor opção.
- ✓ **Espaçamento:** Se você precisa de espaçamento simples entre itens, a Flexbox é uma boa escolha. Se você precisa de espaçamento mais complexo, o Grid é a melhor opção.
- ✓ **Navegação:** Se você precisa de navegação simples entre itens, a Flexbox é uma boa escolha. Se você precisa de navegação mais complexa, o Grid é a melhor opção.

Em resumo, a escolha entre Flexbox e Grid depende das necessidades específicas de cada projeto e da complexidade do *layout* desejado. Ambas as ferramentas são excelentes e é importante considerar as vantagens e limitações de cada uma antes de escolher a ferramenta certa para o projeto.

Lição 8: Entendendo a entrada de dados

A entrada de dados no HTML pode ser feita através de diversos elementos, como **formulários**. Aqui estão alguns dos elementos mais comuns usados para entrada de dados no HTML:

- ✓ **<input>:** Este elemento é o mais comum para entrada de dados no HTML. Ele pode ser usado para receber entradas de texto, números, datas, arquivos, entre outros.
- ✓ **<textarea>:** Este elemento é usado para criar uma área de texto multilinha.
- ✓ **<select>:** Este elemento é usado para criar uma lista suspensa para seleção.
- ✓ **<option>:** Este elemento é usado dentro de um elemento <select> para definir as opções disponíveis na lista suspensa.
- ✓ **<button>:** Este elemento é usado para criar um botão que pode ser clicado pelo usuário.
- ✓ **<label>:** Este elemento é usado para fornecer uma descrição aos elementos de entrada de dados.
- ✓ **<fieldset>:** Este elemento é usado para agrupar elementos de entrada de dados relacionados.

Os elementos de entrada de dados são geralmente usados dentro de um elemento **<form>**, que é usado para criar um formulário na página. Os dados inseridos pelo usuário são enviados para o servidor através de um processo conhecido como envio de formulário. Para trabalhar com formulários, é importante ter conhecimento de HTML, CSS e JavaScript.

Lição 9: Criando o primeiro formulário

Acompanhe nossa aula prática em vídeo.

Lição 10: Elementos avançados do formulário

Além dos elementos básicos de entrada de dados descritos na lição anterior, existem alguns elementos avançados que podem ser usados em formulários HTML para fornecer uma melhor experiência de usuário e recursos adicionais. Aqui estão alguns deles:

- ✓ **<input type="radio">:** Este elemento é usado para criar botões de opção, onde o usuário pode selecionar apenas uma opção de uma lista.
- ✓ **<input type="checkbox">:** Este elemento é usado para criar caixas de seleção, onde o usuário pode selecionar várias opções de uma lista.
- ✓ **<input type="range">:** Este elemento é usado para criar um controle deslizante, onde o usuário pode selecionar um valor de uma faixa de valores.
- ✓ **<input type="date">:** Este elemento é usado para criar um campo de data, onde o usuário pode selecionar uma data usando um calendário.
- ✓ **<input type="time">:** Este elemento é usado para criar um campo de hora, onde o usuário pode selecionar uma hora usando um seletor de hora.
- ✓ **<input type="file">:** Este elemento é usado para criar um campo de arquivo, onde o usuário pode selecionar um arquivo para enviar ao servidor.
- ✓ **<datalist>:** Este elemento é usado para criar uma lista de sugestões para elementos de entrada de dados.
- ✓ **<output>:** Este elemento é usado para exibir o resultado de uma operação realizada pelo usuário em um formulário.

Estes elementos avançados de formulários combinados com validação de formulários usando JavaScript podem fornecer uma experiência de usuário mais rica e interativa aos visitantes do seu site.

Lição 11: Estilização de formulário

Acompanhe nossa aula prática em vídeo.

Lição 12: Interatividade com pseudo-classes de CSS

As pseudo-classes em CSS são classes especiais que **permitem a aplicação de estilos em elementos HTML** baseados em algum estado específico, como por exemplo, um estado ativo, um link visitado, um elemento com o mouse pairado, entre outros. Aqui estão algumas das pseudo-classes mais comuns:

- ✓ **:hover**: A pseudo-classe :hover é usada para aplicar estilos a um elemento quando o mouse estiver pairado sobre ele.

- ✓ **:active**: A pseudo-classe :active é usada para aplicar estilos a um elemento quando ele estiver ativo, ou seja, quando o usuário clicar nele.
- ✓ **:focus**: A pseudo-classe :focus é usada para aplicar estilos a um elemento quando ele estiver em foco, ou seja, quando o usuário estiver interagindo com ele.
- ✓ **:visited**: A pseudo-classe :visited é usada para aplicar estilos a um link depois que ele já foi visitado.
- ✓ **:first-child**: A pseudo-classe :first-child é usada para aplicar estilos ao primeiro elemento filho dentro de um elemento pai.
- ✓ **:last-child**: A pseudo-classe :last-child é usada para aplicar estilos ao último elemento filho dentro de um elemento pai.
- ✓ **:nth-child(n)**: A pseudo-classe :nth-child(n) é usada para aplicar estilos a um elemento filho específico dentro de um elemento pai, onde n é o número do elemento filho que deseja estilizar.

Estas pseudo-classes permitem uma interação mais sofisticada com os elementos HTML, e são uma ferramenta poderosa para criar interfaces mais atraentes e interativas para os visitantes do seu site.

Lição 13: Criando a primeira tabela

As tabelas em HTML são usadas para exibir dados em uma estrutura de linhas e colunas. Elas são uma das formas mais comuns de apresentar informações em tabela na *web*, e podem ser usadas para exibir dados como horários de eventos, preços de produtos, resultados de pesquisas, entre outros.

A estrutura básica de uma tabela em HTML é composta pelos seguintes elementos:

- ✓ **<table>**: Este elemento define o início da tabela.
- ✓ **<tr>**: Este elemento define uma linha da tabela.
- ✓ **<th>**: Este elemento define uma célula de cabeçalho.
- ✓ **<td>**: Este elemento define uma célula de dados.

Aqui está um exemplo de como você pode criar uma tabela simples em HTML:

```
<table>
 <tr>
  <th>Nome</th>
  <th>Idade</th>
  <th>País</th>
 </tr>
 <tr>
  <td>João</td>
  <td>25</td>
  <td>Brasil</td>
 </tr>
 <tr>
  <td>Maria</td>
  <td>30</td>

<td>Portugal</td>
 </tr>
 <tr>
  <td>Pedro</td>
  <td>35</td>

<td>Espanha</td>
 </tr>
</table>
```

O resultado seria uma tabela com três linhas e três colunas, com o cabeçalho "Nome", "Idade" e "País". Além disso, você pode usar o CSS para estilizar a tabela, como definir a largura das colunas, a cor de fundo, entre outros.

Lição 14: Estilização de tabelas

Acompanhe nossa aula prática em vídeo.

Lição 15: Inserindo elementos externos

Acompanhe nossa aula prática em vídeo.

Lição 16: Desafio *Tech*

Para fixar seus conhecimentos, propomos um desafio. Você pode conferir na aula em vídeo e o gabarito será disponibilizado ao final do módulo.

Lição 17: 5 passos práticos para aplicar o que você aprendeu

1. Aprenda as diferentes **entradas de dados** e como se comportam na interação com ***mobile***;
2. Crie **tabelas** e **formulários** de forma **semântica** e de fácil compreensão;
3. Utilize ***Flexbox*** e ***CSS Grid*** para posicionar de forma **intuitiva** os elementos da sua página;
4. **Pratique** as propriedades de ***Flexbox*** e ***CSS Grid***;
5. **Avance no CSS** para trazer mais **personalidade** para sua página.

Módulo 3

Trabalhando com Figma e diferentes telas

Lição 1: Objetivos do módulo

Neste módulo iremos aprender um pouco mais sobre o **Figma**, ferramenta de *design* amplamente utilizada no desenvolvimento *web* e **como utilizar seus recursos**. Também iremos aprender sobre ***design* responsivo** e **metodologia *mobile first***, aprofundando mais na estilização em CSS.

Responderemos perguntas do tipo:

- ✓ Como desenvolver o código de um *layout* do Figma?
- ✓ Como mudar o *layout* da página automaticamente ao diminuir o tamanho da tela?

Lição 2: *Design* responsivo

O ***design* responsivo** é uma abordagem de *design* de página que permite que o **conteúdo se adapte automaticamente a diferentes tamanhos de tela e dispositivos**. Isso significa que a página se ajusta automaticamente ao tamanho da tela do usuário, seja ele um computador de mesa, um *laptop*, um *tablet* ou um *smartphone*.

Com as ferramentas de CSS, você pode criar regras que especificam como o conteúdo deve ser exibido em **diferentes larguras de tela**. Por exemplo, você pode especificar que as colunas sejam reorganizadas em uma disposição vertical em dispositivos móveis, mas mantenham sua disposição horizontal em dispositivos de *desktop*.

Além disso, você também pode **usar medidas relativas**, como porcentagens, ao invés de medidas absolutas, como *pixels*, para garantir que o conteúdo se adapte corretamente a diferentes tamanhos de tela.

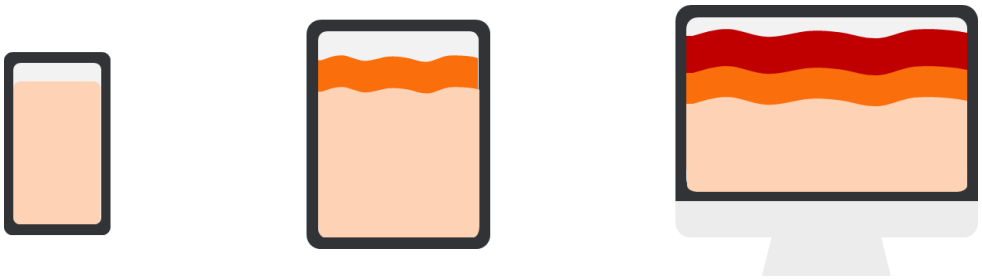
Em resumo, o design responsivo é importante porque permite que o conteúdo **seja acessível** e **fácil de ler em todos os dispositivos**, independentemente do tamanho da tela. Isso é especialmente importante com o aumento do uso de dispositivos móveis para acessar a Internet.

Lição 3: Trabalhando com responsividade

Acompanhe nossa aula prática em vídeo.

Lição 4: A metodologia *mobile First*

A metodologia ***Mobile First*** é uma abordagem de *design* de página que prioriza o *design* para dispositivos móveis e, em seguida, adiciona mais recursos e *design* para dispositivos de *desktop*. Em outras palavras, é uma abordagem que coloca **o *design* para dispositivos móveis como a prioridade máxima**, em vez de simplesmente adaptar o *design* para *desktop* para dispositivos móveis.



A razão para esta abordagem é que a maioria das pessoas está acessando a Internet em dispositivos móveis, e é importante ter um *design* responsivo que seja fácil de usar e acessível em todos os tipos de dispositivos. Além disso, a metodologia *Mobile First* incentiva a **simplificação do *design***, concentrando-se nas informações mais importantes e priorizando a experiência do usuário em dispositivos móveis.

Lição 5: Identificando e reproduzindo elementos: Parte 1

Acompanhe nossa aula prática em vídeo.

Lição 6: Identificando e reproduzindo elementos: Parte 2

Acompanhe nossa aula prática em vídeo.

Lição 7: Estilizando a página para *tablet*

Acompanhe nossa aula prática em vídeo.

Lição 8: Estilizando a página para *desktop*

Acompanhe nossa aula prática em vídeo.

Lição 9: Desafio *Tech*

Para fixar seus conhecimentos, propomos um desafio. Você pode conferir na aula em vídeo e o gabarito será disponibilizado ao final do módulo.

Lição 10: 5 passos práticos para aplicar o que você aprendeu

1. Sempre se preocupe com **como sua página ficará em diferentes telas**;
2. Identifique os **elementos** que precisam ser **adaptados** para outros aparelhos;
3. Utilize a metodologia do ***mobile first*** para desenvolver sua página;
4. Realize **testes** para averiguar **responsividade** de suas aplicações.
5. Tenha o **Figma** como **seu grande aliado** para o desenvolvimento *web*;