Curso completo de HTML e CSS

Crie os seus próprios sites!

O que é HTML?

- HTML não é uma linguagem de programação, e sim de marcação;
- Utilizamos para estruturar páginas web, criando elemento de texto, inserindo imagens, listas e formulários;
- É o esqueleto de qualquer aplicação web;
- Nós não precisamos de um software para compilar HTML;
- Podemos apenas abrir um arquivo .html no navegador e executar a linguagem;

O que é CSS?

- CSS é a linguagem que utilizamos para estilizar um site;
- Usamos em conjunto de HTML, a integração é super simples;
- Sem CSS todas as páginas seriam iguais, com apenas diferença no conteúdo;
- As regras de CSS são aplicadas aos elementos do HTML;
- Podemos adicionar cores, mudar o tamanho de uma fonte, adicionar bordas aos elementos e muito mais!

Editor de código

- Não precisamos de um editor para criar código de HTML e CSS;
- Porém os editores são ótimas ferramentas para programação;
- Nos ajudam com: estrutura de pastas, erros de sintaxe e highlight de sintaxe;
- Além de terem recursos extras, como terminal integrado e extensões que podem ser adicionadas;
- O meu preferido é o Visual Studio Code;

Nosso primeiro site

- Primeiramente precisamos criar um arquivo com a extensão .html;
- Podemos abrir o arquivo no navegador, sugiro o Chrome;
- Neste arquivo podemos escrever código HTML ou simplesmente texto;
- Vamos criar um e ver os resultados no navegador!

O que aconteceu?

- Nós criamos um arquivo com HTML, o navegador tem ferramentas para ler, entender e executar este tipo de arquivo;
- Ele coloca tamanhos diferentes para elementos diferentes, e os posiciona na página;
- Isso acontece porque os navegadores vêm pré-configurados para este fim;
- Vamos inspecionar o código que criamos no navegador!

Arquivos do curso

- Todos os arquivos estão no GitHub;
- É possível também fazer o download de todo o código do curso;
- Neste site podemos guardar os nossos projetos de programação de graça;
- Alguns programadores criam seu portfólio no GH, então é interessante você já criar a sua conta;
- Me segue lá também =)
- Vamos explorar o GitHub e o repositório do curso!

Curso de HTML e CSS

Fim da primeira seção

Fundamentos do HTML

Introdução

A anatomia das tags

- O elemento principal do HTML é a tag;
- Toda tag tem um nome e um propósito, a tag p serve para parágrafos;
- Nós precisamos envolver uma tag com sinais de menor e maior: assim:
- E no meio colocar um conteúdo: Texto
- Isso vai criar um elemento de parágrafo, quase todas as tags funcionam assim;

A estrutura do HTML

- Todos os projetos de HTML tem uma estrutura base, ou seja, precisamos criar algumas tags;
- **DOCTYPE**: Esta tag declara a versão do HTML;
- **html**: Esta tag envolve todo o código de HTML;
- **head**: Nesta colocamos todas as configurações de um site, como a importação de CSS e o título da página (meta tags);
- **body**: É onde todos os elementos visíveis estão;

Títulos

- Os títulos são conhecidos como headings;
- Utilizamos principalmente para separar seções;
- O nome da tag é h*, onde * pode ser um valor de 1 a 6;
- O maior título é o h1, e também é o mais importante, colocamos apenas um por página;
- A importância deve estar conectada com o propósito da nossa página, este assunto faz parte do HTML semântico, um tópico mais avançado;

Parágrafos

- Utilizamos os parágrafos para inserir textos maiores;
- A tag é <p>
- Cada parágrafo começa uma nova linha, e este comportamento acontece com todas as tags de bloco;
- Temos várias tags de bloco, os títulos fazem parte deste grupo;
- Vamos ver na prática!

Tags sem conteúdo

- Nós temos em HTML tags sem conteúdo;
- Elas possuem recursos geralmente, como quebrar linha;
- Para este fim podemos utilizar a tag
- Para uma linha horizontal temos <hr/>
- Eles também introduzem o conceito de self closing tags, onde a tag não possui uma outra tag de fechamento;
- Vamos ver na prática!

Comentários no HTML

- Comentários são utilizados para descrever como algo funciona no nosso código;
- Ou explicar a outros programadores o que fizemos;
- Os comentários não são exibidos na página;
- Mas qualquer um que inspecionar o seu código, terá acesso aos comentários;
- Vamos ver na prática!

Atributos

- Atributos podem ser utilizados para adicionar funcionalidades as tags;
- A tag a é responsável por nos direcionar a uma nova página ou site;
- Mas aonde vamos adicionar o endereço/URL?
- Colocamos no atributo chamado href;
- Um exemplo: Google
- Neste exemplo, ao clicar no link o usuário é redirecionado para o Google;

Abrir nova aba

- Com um atributo podemos fazer o link abrir em uma nova aba;
- Isso é utilizado frequentemente para redirecionar a outro site;
- Por exemplo: temos um link que leva a um e-commerce que não é nosso, não temos aquele domínio;
- Então utilizamos o atributo target com o valor _blank;
- Vamos ver na prática!

Imagens no HTML

- Nós podemos inserir imagens no nosso site com a tag img;
- O caminho relativo até a imagem é inserido no atributo src;
- Normalmente colocamos a imagem numa pasta chamada img ou assets, para fins de organização;
- Nota: a imagem é uma self closing tag;
- Vamos ver na prática!

O atributo alt

- Nas tags de imagem temos um atributo chamado alt;
- Nós inserimos nele um texto que descreve a imagem;
- Todas as imagens devem ter este atributo configurado;
- Este recurso é importante para acessibilidade, fazendo com que nosso site seja melhor rankeado pelo Google também;
- Vamos ver na prática!

Listas não ordenadas

- Listas são importantes para muitos fins nos sites;
- Podemos até criar um menu a partir de uma lista;
- As não ordenadas são criadas pela tag ul;
- Cada item na lista é representado pela tag li;
- Vamos ver na prática!

Listas ordenadas

- Listas ordenadas são interessantes para quando há um procedimento ou passos a serem seguidos;
- Exemplo: receita de algum alimento;
- Agora utilizamos a tag ol;
- E os itens continuam sendo a tag li;
- Vamos ver na prática;

Tabelas

- Nós usamos tabelas para exibir dados que podem ser categorizados em colunas;
- Tabelas são estruturas complexas no HTML, e não tão utilizadas;
- Precisamos da tag table, isso cria a tabela;
- E também um cabeçalho e um corpo;
- Cada linha é criada em uma tag tr, e os dados ficam em td;
- No cabeçalho utilizamos a tag th;

A tag div

- A tag div é utilizada para criar divisões/seções no nosso site;
- Podemos criar elementos menores também, como cards;
- O principal propósito é: encapsular elementos que estão conectados entre si;
- Vamos ver na prática!

Criando a estrutura com VS Code

- Podemos criar toda a estrutura básica do VS Code com um simples comando;
- Basta digitar ! e pressionar tab;
- Vamos ver na prática!

Desafio 1

1. Crie um novo arquivo HTML, nomeie como quiser;
2. Utilize a inicialização rápida de estrutura HTML;
3. Crie um título que descreva uma imagem;
4. Insira a imagem;
5. Crie um parágrafo que fale mais sobre o que está na imagem;
6. E por fim uma lista, com três itens com elementos presentes na imagem;

Fundamentos do HTML

Conclusão

Fundamentos do CSS

Introdução

Maneiras de adicionar CSS

- Temos algumas maneiras de adicionar CSS ao HTML;
- Inline: quando os estilos são adicionados por um atributo;
- Internal: quando o CSS é adicionado na tag head;
- External: quando o CSS é adicionado através de um arquivo externo, e depois importado no HTML;
- Vamos sempre optar pelo external CSS, quando houver a opção;
- Isso vai organizar melhor nosso projeto;

A anatomia do CSS

- Com CSS aplicamos CSS a um elemento;
- Primeiramente devemos selecionar o elemento, isso pode ser feito através da tag do elemento;
- Depois precisamos colocar as propriedades e os valores;
- Se quisermos mudar a cor de algo, utilizamos:

color: red;

- Nome da propriedade, dois pontos, valor, ponto e vírgula;

Inline CSS

- O CSS inline pode ser adicionado sem selecionar o elemento, porque é um atributo diretamente inserido no mesmo;
- O elemento já está selecionado!
- O atributo style nos permite escrever regras de CSS;
- Exemplo:

  style=”color: red;”

Múltiplas regras

- Nós podemos adicionar várias regras de CSS;
- Elas podem ser separadas por ponto e vírgula;
- Então é possível fazer uma união de estilos, dar um design melhor ao elemento;
- Vamos ver na prática!

Internal CSS

- O CSS interno é uma técnica melhor que o inline, vamos colocar todos os estilos na tag head;
- As regras precisam também estar entre a tag style;
- E através desta maneira, é necessário selecionar o elemento alvo:

```
p {
    color: red;
}
```

External CSS

- Para adicionar CSS com esta técnica precisamos criar um arquivo .css;
- Geralmente eles ficam numa pasta chamada css;
- E nós o importamos através da tag link;
- As regras que estão no arquivo são aplicadas no HTML;
- Vamos ver na prática!

Ordem do CSS

- O CSS é carregado a partir de uma ordem;
- Se temos estilos misturados (inline, internal e external), qual será aplicado?
- Todos eles, mas com a seguinte ordem: inline > internal = external > padrão do navegador;
- Esta regra funciona quando temos estilos em um mesmo elemento;
- Interno e externo tem a mesma prioridade, a última regra ganha a ‘corrida’;

Múltiplos arquivos de CSS

- É possível ter mais de uma folha de estilo no nosso projeto;
- Precisamos apenas importar todas elas na tag head;
- Os arquivos importados por último tem mais prioridade;
- Esta é uma boa prática, pois possibilita a divisão de CSS por páginas, por exemplo;
- Vamos ver na prática!

Desafio 2

1. Crie um arquivo de CSS chamado titles.css;
2. Importe este arquivo no HTML;
3. Estilize todos os h4 em alguma regra de CSS;
4. Crie um h4 no HTML e veja as mudanças de CSS;

Comentários no CSS

- Comentários do CSS são como os de HTML, utilizamos para descrever algo no código;
- Caso o código seja inspecionado, os comentários também são exibidos;
- A sintaxe é: /* algum comentário */
- Vamos ver na prática!

Classes e ids

- Classes e ids são atributos de tags do HTML, mas estão diretamente relacionados ao CSS;
- Podemos especificar elementos específicos com eles;
- Ids são utilizados para elementos únicos;
- E classes servem para um ou mais elementos, geralmente utilizadas em conjuntos de elementos;
- Veremos a utilização dos mesmos nos nossos projetos também;

Classes

- As classes são inseridas através de um atributo de HTML;
- O valor do atributo é o nome da classe, e também uma escolha nossa;
- Por exemplo: temos um botão que aparece x vezes no nosso projeto, podemos colocar uma classe btn nele;
- Ou seja, os padrões de estilo desses botões podem ser transmitidos através desta classe para os demais;
- O seletor fica: .<nome_da_classe>

IDs

- Os ids também são atributos do HTML;
- Podemos escrever qualquer coisa como valor, será o nome do id;
- Ids são únicos, ou seja, não repetimos o mesmo nome na mesma página;
- O HTML não nos proíbe disso, mas é uma má prática e deve ser evitada;
- O seletor fica: #<nome_do_id>
- Vamos ver na prática!

A ordem dos seletores

- Nós aprendemos sobre o seletor de id e classe;
- E se a tag estiver com id e uma classe, o que acontece?
- Como nas maneiras de adicionar CSS, temos também uma ordem;
- Que é: id > class > seletor de tag;
- Então o id vai vencer de todos os outros, utilize isso ao seu favor;
- Regras que não entram em conflito serão aplicadas normalmente;

As cores do CSS

- Em CSS as cores são divididas em grupos, temos:
- Nomes de cor: como red ou blue, não são muito utilizados;
- RGB: configuramos as tonalidades de red, green e blue;
- Hexadecimal: uma união de letras e números que podem criar uma cor, a maneira mais utilizada;
- HSL: hue, saturation e lightness, mudando estes valores, temos uma cor;
- Nas próximas aulas, abordaremos todos detalhadamente;

Nomes das cores

- Nós utilizamos muito essa maneira até agora, mas em projetos reais não é tão empregada;
- Pois ela nos limita a apenas as cores com nomes existentes;
- No mundo real precisamos de mais possibilidades, para não limitar os designers;
- O nome da cor consiste na utilização do nome real da cor como valor da propriedade;

HEX

- HEX ou Hexadecimal é a abordagem mais utilizada;
- Basicamente temos que inserir 6 dígitos, precedidos de uma #;
- Os dois primeiros representam o tom de vermelho, depois o de verde e por fim o azul;
- Os valores vão de 0 a 9 e A a F;
- 0 é o mais escuro e F o mais claro;
- O valor de #000 é a cor preta e #FFF a cor branca;

Mais sobre o HEX

- Se um valor for repetido 6 vezes, podemos escrever a cor de forma mais simples;
- No caso de #FFFFFF podemos reescrever com #FFF;
- A mesma coisa vale para #112233, essa cor pode ser escrita como: #123;
- Esta é uma técnica muito utilizada;
- Vamos ver na prática!

RGB

- RGB significa Red, Green e Blue;
- Nós precisamos inserir a intensidade de cada com, com valores de 0 a 255;
- 0 é o mais escuro e 255 o mais claro;
- Aplicamos RGB com a seguinte sintaxe: rgb(0-255, 0-255, 0-255)
- O primeiro valor representa o vermelho, depois verde e por fim azul;
- Para criar a cor verde inserimos: rgb(0, 255, 0);
- Vamos ver na prática!

RGBA

- Podemos criar cores também com o RGBA, A vem de alpha;
- A alteração dele muda a opacidade da cor;
- Os valores possíveis são de 0 a 1;
- Sendo 0 transparente e 1 totalmente visível;
- A sintaxe é quase a mesma: rgba(0-255, 0-255, 0-255, 0-1)
- Vamos ver na prática!

HSL

- HSL é um acrônimo para hue, saturation e lightness;
- Esta abordagem também não é muito utilizada, o ranking de uso é este: HEX > RGB > HSL > Nomes de cor;
- Podemos definir uma cor com: hsl(0-255, 0-100%, 0-100%)
- Vamos ver na prática!

Background color

- Quase todo elemento tem um background, e podemos mudar a cor dele;
- Todas as regras que vimos sobre cores podem ser aplicadas em cores de background;
- A regra é: background-color: “cor”;
- As regras de cor de fundo e cor de fonte podem ser utilizadas juntas;
- Vamos ver na prática!

Background opacity

- Podemos alterar a opacidade de uma cor de fundo com CSS;
- A regra é a opacity;
- Os valores vão de 0 a 1;
- Sendo 1 totalmente visível e 0 remove a cor;
- Com esta regra mudamos também a opacidade dos elementos dentro do elemento que alteramos a opacidade, veremos uma solução depois;
- Vamos ver na prática!

Resolvendo o problema da opacidade

- Se você não quer aplicar a opacidade para os elementos filhos, precisa utilizar o RGBA em vez de opacity;
- Alterando o valor de alpha temos a opacidade colocada apenas na cor de fundo;
- Então preservamos o conteúdo e alteramos o background;
- Vamos ver na prática!

Background images

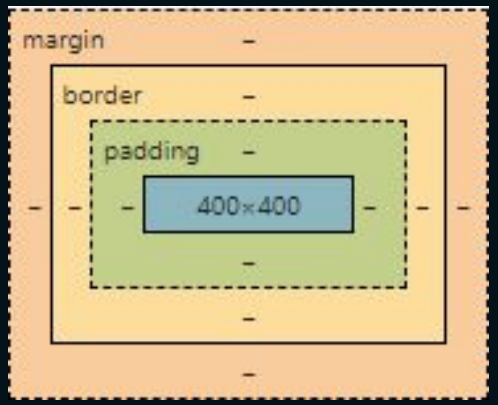
- Podemos inserir imagens no background dos elementos;
- A regra é: background-image: url(“pasta/imagem.jpg”)
- Geralmente a imagem fica em outra pasta, então temos que voltar um diretório;
- Isso pode ser feito com o símbolo ..
- Vamos ver na prática!

Centralizando a imagem de background

- Às vezes a imagem é muito mais que o elemento que estamos inserindo
- Então precisamos melhorar a visualização, centralizando a imagem;
- Isso pode ser feito com duas regras: background-position com o valor de center;
- E background-size com o valor de cover;
- Vamos ver na prática!

Box Model

- Box model é uma entidade que é criada em todo elemento do HTML;
- Ela consiste em quatro partes: altura e largura, padding, border e margin;
- Todas elas podem ser alteradas por CSS;
- Alguns elementos do HTML já vem com valor nestas regras;
- Este conceito é muito importante, veremos em detalhes nas próximas aulas;



Height and width

- A altura e a largura são o core do box model;
- Estas propriedades consistem no conteúdo do elemento;
- Podemos alterar as duas e mudar o tamanho do elemento na tela;
- Alguns elementos do HTML, os block elements, já vem com 100% de largura;
- Preenchendo a tela toda na horizontal;
- Vamos ver na prática!

Padding

- Padding é o espaço entre o conteúdo e a borda do elemento, também conhecimento como espaçamento interno;
- Este recurso é utilizado para criar uma distância entre o conteúdo (texto) e a extremidade do elemento;
- Vamos ver na prática!

Lados individuais

- Podemos adicionar o padding aos lados individuais de um elemento;
- A regra é: padding-*
- Onde * pode ser: top, left, right ou bottom;
- Assim teremos valores customizados em cada um dos lados do elemento;
- Vamos ver na prática!

Shorthand properties

- As propriedades de shorthand nos permitem adicionar padding a todas as direções com uma regra;
- Apenas precisamos utilizar a regra padding, e configurar top, right, bottom e left nesta ordem;
- Exemplo: padding: 10px 5px 12px 20px;
- Esta regra de shorthand pode ser aplicada para outras propriedades, como a margin;

Padding e width

- A padding é adicionada a largura do elemento, e isso pode ser um problema;
- Por exemplo: se precisamos seguir um layout perfeitamente;
- Um elemento com 200px de width e 25px de padding tem um tamanho total de 250px na horizontal;
- Podemos diminuir a largura do elemento, mas isso dificulta o cálculo também;
- Isso pode ser resolvido com a regra box-sizing e o valor de border-box, isso faz o elemento respeitar o tamanho que está em width;

Border

- A borda é o elemento central, fica entre padding e margin;
- Padding é o espaçamento interno e margin o externo;
- Geralmente esta regra é utilizado com propósito decorativo;
- A regra de border é definida em algumas partes: tamanho, aspecto e cor da borda;
- Vamos ver na prática!

Lados individuais da border

- Podemos adicionar borda aos lados específicos de um elemento também;
- Podemos utilizar border-*, onde * pode ser top, right, bottom e left;
- Isso é utilizado frequentemente, especialmente com a border-bottom e left;
- Vamos ver na prática!

Borda arredondada

- Com a propriedade border-radius podemos arredondar os cantos de um elemento;
- Podemos aplicá-la assim: border-radius: 5px;
- Isso faz com que o canto seja arredondado em 5px;
- Importante: Podemos arredondar elementos que não a regra de borda aplicada;
- Vamos ver na prática!

Margin

- A propriedade de margin é responsável pelo espaçamento externo do elemento;
- Podemos aplicar o recurso como aplicamos padding;
- Ou seja: lados individuais e também o shorthand;
- Vamos ver na prática!

Elementos do box model juntos

- Em alguns elementos vamos utilizar todos os recursos do box model;
- Ou seja, vamos definir o tamanho (width e height);
- Um espaçamento interno (padding);
- Decorar o elemento com bordas (border);
- Afastar o elemento de outros (margin);
- Vamos ver na prática!

Desafio 3

1. Crie um elemento que tenha todas as regras do box model;
2. Tente inserir diferentes cores, para destacar as áreas;
3. Insira um texto no elemento com a tag h2;
4. Mudar a cor de fundo do texto;
5. O elemento do box model deve ter um id, o texto uma classe;
6. Adicionar uma regra chamada outline, com o mesmo valor da propriedade border, apenas com a cor diferente;

Alinhamento de texto

- Nossos textos podem ser alinhados em várias direções;
- Por padrão ele é alinhado a esquerda;
- Porém com a regra text-align configuramos center (centro) ou right (direita), para alterar o valor default;
- Usamos muito o valor de center;
- Vamos ver na prática!

Text decoration

- Com a decoration podemos adicionar efeitos ao texto;
- É possível colocar um underline ou até mesmo uma linha que corta o texto;
- Esta regra é utilizada em casos específicos;
- A tag a tem um underline por padrão, podemos remover isso com a regra de text-decoration e o valor de none;
- Vamos ver na prática!

Transformação de texto

- Com a regra text-transform podemos alterar com o texto é exibido;
- É possível alterar para uppercase ou lowercase (letras maiúsculas ou minúsculas);
- Não há muitos valores para esta regra;
- Cuidado: O CSS deve ser aplicado quando queremos texto em uppercase, nunca escreva o texto com capslock no HTML;
- Vamos ver na prática!

Espaçamento de letras

- Com a regra letter-spacing podemos alterar o espaçamento entre letras de um texto;
- Isso é interessante em situações que o layout pede esta mudança;
- A regra é aplicada da seguinte maneira: letter-spacing: 5px;
- Vamos ver na prática!

Fontes

- Com CSS podemos alterar o tipo da fonte, com a regra font-family;
- As regras disponíveis são: Serif, Sans-serif, Monospace, Cursive e Fantasy;
- Todos os navegadores tem várias fontes que podemos utilizar;
- E ainda podemos adicionar fontes externas, por exemplo com o Google Fonts;
- Vamos ver na prática!

Estilo de fonte

- Nós podemos utilizar a propriedade font-style para mudar o aspecto das letras;
- Os valores são: normal, italic e oblique;
- Normal é o valor default, e os outros as variantes;
- Oblique é como o tipo itálico, com pequenas diferenças;
- Vamos ver na prática!

Font weight

- A propriedade font-weight pode deixar nossa fonte de texto mais fina ou grossa;
- Os valores vão de 100 a 900, sendo 100 o mais fino;
- Algumas palavras também funcionam como valores, por exemplo: bold (600);
- Existem fontes que não tem todas as variações, devemos tomar cuidado com isso;
- Vamos ver na prática!

Font size

- A regra de font-size é responsável por deixar nossa fonte grande ou pequena;
- Quanto menor o valor, menor a fonte e vice-versa;
- Esta regra é configurada com unidades de medida, como o px;
- Em CSS temos diversas unidades, veremos ao longo do curso;
- Vamos ver na prática!

Display

- Em HTML e CSS temos alguns elementos que são considerados de bloco e outros inline;
- A tag div é um exemplo de block e span um exemplo de inline;
- Com a regra display podemos mudar este comportamento, ou seja, deixar uma div como inline;
- Vamos ver na prática!

Escondendo elementos

- Existem situações que precisamos ocultar elementos;
- Basta adicionar a regra display com o valor de none;
- Então o elemento não é mais exibido, porém ainda consta no HTML;
- Vamos ver na prática!

Sobre as posições dos elementos

- A regra position e seus valores são responsáveis por posicionar o elemento na tela;
- Temos algumas possibilidades: relative, fixed, absolute, sticky e mais;
- O valor padrão é static, todo elemento começa com esta posição;
- Esta regra é essencial quando precisamos mudar onde o elemento deve ficar no projeto;
- Vamos ver em detalhes nas próximas aulas;

Position static

- O valor de static na position não faz nada com o elemento;
- Porque este é o valor padrão, ou seja, todo elemento já tem este tipo de position;
- Outros valores são afetados pelas regras top, left, right e bottom, porém static não;
- Static apenas segue o fluxo do HTML;
- Vamos ver na prática!

Position relative

- Com a position configurada como relative temos mais possibilidades que static;
- Agora as regras top, left, right e bottom, movem o elemento pela tela;
- O elemento ainda segue o fluxo do HTML;
- Atenção: normalmente não utilizamos estas regras de posição com relative;
- Vamos ver na prática!

Position absolute

- Com o valor de absolute em position, o elemento pode ser movido pela tela toda;
- Ou seja, quebramos o fluxo do HTML;
- Esta regra também é afetada por top, left, right e bottom;
- Posicionar com absolute pode ser uma solução ou um problema, dependendo do ponto de vista;
- Vamos ver na prática!

Relative com absolute

- Podemos resolver o problema de absolute com relative;
- Um elemento com position absolute é ligado ao elemento mais próximo com posição relativa, se não ele é ligado ao body;
- Isso foi o que aconteceu na aula passada;
- Então com um container com posição relative, podemos controlar melhor a área de ação dos elementos com absolute;
- Vamos ver na prática!

Position fixed

- Com fixed o elemento pode ser fixado na tela;
- Mesmo após o scroll na página, o elemento permanece na mesma posição, estando sempre presente;
- O recurso é frequentemente utilizado para criar barra de navegação fixa;
- Vamos ver na prática!

Position sticky

- Sticky também faz o elemento ficar fixo na tela;
- Mas tem um outro comportamento também, quando o elemento volta para a sua posição original ele se comporta como relative;
- A posição do elemento é onde ele foi inserido no HTML;
- Vamos ver na prática!

z-index

- Se temos dois elementos com as mesmas posições ou se eles colidem na página, podemos escolher qual será exibido;
- Utilizamos o z-index para isso;
- O elemento com maior valor prevalece;
- Vamos ver na prática!

Fundamentos do CSS

Conclusão

Formulários com HTML

Introdução

O que é um formulário?

- Forms tem o papel de receber dados do usuário e enviar para um servidor;
- Podemos validar dados;
- As tags mais utilizadas são: form, label e input;
- A tag form cria o formulário e o delimita;
- Label descreve os inputs;
- E o input é a tag que inserimos dados, temos vários tipos: number, email, text e etc;

Criando nosso formulário

- Para criar um formulário vamos precisar da tag form, que encapsula todos os elementos do formulário;
- Dentro dela temos labels e inputs, mas podemos ter outras tags como divs;
- A tag de input tem um atributo chamado type, que é onde definimos o propósito do input;
- Um input do tipo text, recebe dados de texto;
- Vamos ver na prática!

Atributos da tag form

- A tag form tem dois atributos geralmente, que são:
- action: o arquivo/página que os dados serão enviados;
- method: GET ou POST, receber dado ou enviar dado;
- Não os exploraremos em detalhes pois fazem parte do back-end, ou seja, estão fora do escopo deste curso;
- Vamos ver na prática!

Atributo name

- Utilizamos o atributo name para configurar os nossos inputs;
- O valor é ligado ao propósito do input, a sua categoria;
- Exemplo: Um input que recebe a idade de um usuário, pode ter um name com o valor de age/idade;
- Este atributo é utilizado para pegar o valor quando o form é enviado para o servidor;
- Vamos ver na prática!

Atributo da label

- A tag label tem um atributo;
- E nós o utilizamos para linkar com um input, o nome do atributo é for;
- O valor deve ser o mesmo que o atributo name do atributo que corresponde aquela label;
- Utilizamos for por propósitos semânticos;
- Isso ajuda o nosso site a ser melhor rankeado no Google;
- Vamos ver na prática!

Enviando dados de formulário

- Podemos enviar os dados do form para o servidor através de um botão de submit;
- O botão também é um input, porém mudamos o type para submit;
- Quando o usuário clicar no botão o processamento do form acontece;
- Os dados serão enviados ao servidor através de uma requisição HTTP;
- Aqui é onde precisamos de uma integração com o back-end, para aproveitar os dados do formulário;

Elemento select

- A tag select tem as opções representadas por tags de option;
- Select também tem um atributo name;
- O value estará em cada uma das options, e é isso que receberemos no lado do servidor;
- Então temos duas tags para criar um elemento de seleção: select e option;
- Vamos ver na prática!

Atributo selected

- Podemos iniciar o nosso input de select com uma opção selecionada;
- Para isso esta option precisa ter o atributo selected;
- Abordagem interessante para quando temos uma opção muito provável;
- Vamos ver na prática!

Múltiplas seleções

- Podemos criar um select que nos permite mais de uma option selecionada;
- Interessante quando queremos aceitar um ou mais dados;
- Por exemplo: opcionais de um carro;
- Precisamos apenas inserir o atributo multiple na tag select;
- Vamos ver na prática!

Elemento de textarea

- A tag textarea é semelhante ao input text;
- Podemos utilizamos para textos maiores;
- Por exemplo: A bio do Instagram;
- Isso nos permite uma área maior para digitar e verificar o texto que digitamos;
- Vamos ver na prática!

Fieldset e legend

- Fieldset é uma tag para agrupar inputs;
- E legend é como uma label, que descreve os inputs agrupados;
- Utilizamos esta tag para conectar dois ou mais inputs que tenham o mesmo sentido;
- Por exemplo: nome e sobrenome;
- Vamos ver na prática!

Datalist

- Datalist é como um select, porém com um autocomplete;
- Podemos pesquisar por possíveis valores para preencher o input;
- Ou selecionar alguma opção por meio de uma lista;
- As opções são linkadas por um atributo chamado list;
- Vamos ver na prática!

Input para senha

- Se nós estamos esperando uma senha do usuário, podemos utilizar a tag input;
- Porém no atributo type colocaremos password;
- E então o texto passa a ser exibido com *, para mascarar os dígitos;
- Vamos ver na prática!

Reiniciando o form

- Podemos reiniciar todos os campos do form;
- Isso é feito através de um input do tipo reset;
- Ele é um botão, que ao ser clicado limpa o form;
- Vamos ver na prática!

Input radio

- Este input é utilizado para selecionar apenas uma opção de várias possibilidades;
- Por exemplo: O modelo do carro que estamos comprando;
- Não podemos escolher dois, então há a necessidade da decisão entre uma das opções;
- Vamos ver na prática!

Input checkbox

- O checkbox é similar ao radio;
- Temos que selecionar uma ou mais opções, e também cancelar a seleção de uma opção;
- Por exemplo: os opcionais de um carro;
- Vamos ver na prática!

Input de data

- O input do tipo date é utilizado para selecionar uma data;
- Temos um calendário que nos auxilia para datas passadas ou futuras;
- Podemos também preencher o valor digitando;
- Vamos ver na prática!

Input para arquivos

- O input de arquivos pode ser criado com o type igual a file;
- Assim podemos enviar um arquivo ao servidor;
- Por exemplo: imagens ou pdfs;
- Vamos ver na prática!

Input para números

- Configurando o type para number temos um input que só aceita dígitos;
- Este input possui setas, que nos permite alterar o número através de cliques;
- Vamos ver na prática!

Input para e-mails

- O input de e-mail é similar ao de texto;
- Porém quando enviamos o formulário temos uma validação;
- Que checa se o texto tem o padrão de um e-mail, verificando a @, por exemplo;
- Vamos ver na prática!

O atributo value

- Com o value podemos definir um valor ao input alvo;
- Como se o usuário já tivesse preenchido algo;
- Muito utilizado quando temos um valor padrão;
- Vamos ver na prática!

Atributo disabled

- O atributo disabled é utilizado para bloquear um input;
- Então não podemos mais digitar neste input;
- Útil quando não queremos que o usuário preencha um determinado input;
- Vamos ver na prática!

Atributo placeholder

- Como atributo placeholder podemos adicionar dicas para os usuários do sistema;
- Ela será exibida no próprio input;
- Ao começarmos a preencher com algum valor, a dica some e o nosso valor que fica sendo exibido;
- Vamos ver na prática!

Atributo required

- O atributo required força o preenchimento de algum campo;
- Se tentarmos enviar o form sem um valor no campo com required, receberemos um alerta da página;
- Isto é um tipo de validação HTML;
- Vamos ver na prática!

Formulários com HTML

Conclusão

Responsividade

Introdução

O que é responsividade?

- Responsividade é a técnica de adaptar uma página web para vários dispositivos diferentes;
- Ou seja, temos mudanças de CSS baseada na resolução;
- Detectamos o que o usuário está utilizando, e adaptamos o nosso site a resolução;
- As regras de CSS são as mesmas, porém dentro de um recurso chamado media query;

Configurando a responsividade

- Primeiro vamos adicionar uma meta tag ao nosso head;
- Ela é:

  <meta name=”viewport” content=”width=device-width, initial-scale=1.0”>
- Isso faz com que o conteúdo se adapte com os dispositivos;
- Adicionamos também uma escala de 1:1;
- Vamos ver na prática!

Media query

- **Media query** é o recurso que utilizamos para criar os breakpoints;
- Os breakpoints adaptam o nosso projeto para diferentes tipos de tela;
- Configuramos uma largura, e depois as regras começam a serem alternadas dependendo do tamanho da tela;
- Vamos ver na prática!

Media query com min-width

- Se utilizarmos min-width em vez de max-width, a media query funciona ao contrário;
- Então podemos desenvolver projetos com uma técnica chamada mobile first;
- Os projetos que tem mais usuários mobile geralmente são feitos em mobile first;
- Vamos ver na prática!

Padrão de breakpoints

- Estes breakpoints são utilizados frequentemente:
- 600px e abaixo: celular;
- 768px até 600px: tablets;
- 992px até 768px: mini laptops;
- 992px e acima: laptops e desktop;
- É comum em projetos profissionais utilizarmos estes valores para desenvolvermos para diferentes dispositivos;

Adaptação para landscape

- Podemos alterar CSS apenas para usuários com visão em landscape;
- Para isso precisamos de um atributo chamado orientation com o valor de landscape, na media query;
- O código é ativado quando a orientação é alterada;
- Vamos ver na prática!

Responsividade

Conclusão

Flexbox

Introdução

O que é flexbox?

- Flexbox é um valor da propriedade display;
- Esta é a maneira mais utilizada para acondicionar elementos em um container;
- Temos diversas regras dentro do flex, veremos nas aulas seguintes;
- O flex deve ser considerado no elemento pai, e os elementos filhos serão os movimentados;
- Os elementos filhos também podem ter regras específicas;

Aplicando flex

- Para aplicar o flex vamos precisar de uma estrutura base;
- Consiste em um container e elementos filhos dentro do container;
- Colocamos a regra display com o valor de flex no container, e agora os elementos estão seguindo o fluxo do flex;
- Com flex todos os elementos se comportam parecido com os elementos inline;
- Vamos ver na prática!

Flex direction

- Podemos mudar o comportamento inline do flex (chamado de row);
- Alterando o flex-direction para column, teremos agora os itens se comportando como elementos de bloco;
- O valor padrão de direction é row;
- Vamos ver na prática!

Flex wrap

- O flex tenta colocar todos os elementos na mesma linha por padrão;
- Porém há situações que queremos x elementos por linha;
- Para isso acontecer devemos aplicar a regra flex-wrap com o valor de wrap;
- Agora as linhas respeitam a largura dos elementos;
- Vamos ver na prática!

Posicionando conteúdo

- Com justify-content é possível mudar como o conteúdo é posicionado no eixo horizontal;
- Temos alguns valores interessantes nesta propriedade;
- Por exemplo: centralizar os elementos na horizontal, basta colocar o valor de center;
- Vamos ver na prática!

Posicionamento na vertical

- Com justify-content modificamos os elementos na horizontal, já align-items nos permite mudá-los na vertical;
- A propriedade tem vários valores, como o center;
- Vamos ver na prática!

Gap

- O gap é uma regra que serve para colocar espaço entre elementos que estão no flex;
- Nós especificamos a medida em px, por exemplo;
- E este valor é adicionado entre cada um dos elementos;
- Vamos ver na prática!

Order

- Com a order podemos mudar a ordem dos elementos;
- Esta propriedade é utilizada nos elementos filhos;
- Agora começamos com as regras que são aplicadas aos elementos filhos, não ao elemento pai;
- Vamos ver na prática!

Grow

- Com a regra flex-grow podemos mudar a proporção de um ou mais elementos filhos;
- Nota: A width precisa estar sem valor, como automática;
- Exemplo: Se colocarmos grow como 2, o elemento vai crescer duas vezes mais que os outros quando estiver se adaptando no container;
- Vamos ver na prática!

Basis

- A regra flex-basis configura a largura base do elemento;
- Nós podemos trabalhar com basis e grow juntas!
- Grow vai preencher toda a largura que basis deixar vazia;
- Vamos ver na prática!

Shrink

- Shrink é o oposto de grow;
- Quando utilizamos precisamos manter o tamanho dos outros elementos, então o elemento do shrink diminui o seu tamanho para manter o dos outros;
- Nota: usamos esta regra em conjunto de basis e grow;
- Vamos ver na prática!

Flex shorthand

- Utilizando apenas flex conseguimos configurar grow, shrink e basis;
- Colocamos os valores nesta ordem também;
- Desta maneira: flex: 2 1 100px;
- Isso dá ao elemento: grow = 2, shrink = 1 e basis = 100px;
- Vamos ver na prática!

Auto alinhamento

- A regra align-self alinha um elemento diferente dos demais;
- Podemos centralizar um elemento enquanto os outros seguem a regra de alinhamento do container;
- Vamos ver na prática!

Flexbox

Conclusão

HTML Semântico

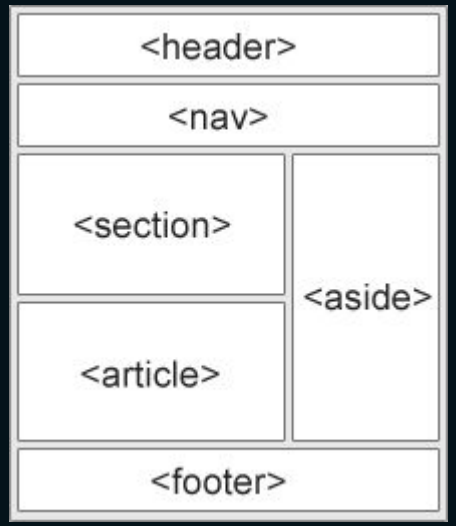
introdução da seção

O que é HTML semântico?

- Tags de HTML que dão significado ao elemento;
- Isso ajuda a estruturar melhor nosso site;
- Contribui para o SEO da nossa página;
- Exemplos de elementos não semânticos: **div**, **span**;
- Exemplos de elementos semânticos: **main**, **section**, **article**;

Estrutura da página com semântica

- Ao adicionar elementos semânticos podemos **estruturar nossa página**;
- Veja a imagem ao lado;
- **Cada seção do site** tem uma tag que faz sentido ao elemento;
- Isso até facilita a manutenção do nosso projeto;



Section

- A tag **section** define uma seção no documento;
- Geralmente um agrupamento por **categorias**, exemplo: seção dos produtos, seção de contato;
- Provavelmente um elemento que **será utilizado muitas vezes** em um projeto;
- Vamos ver na prática!

Article

- A tag **article** é utilizada para elementos informativos;
- Podemos aplicar em: post de blog, comentários, card de produtos;
- **O conteúdo é individual**, não fazendo relação com outro elemento ou article;
- Vamos ver na prática!

Header

- A tag de **header** é utilizada para um conteúdo introdução ou links de navegação;
- Geralmente tem elementos como: headings, logo, informação do autor do post/página;
- **Podemos ter mais de um header** por página;
- Vamos ver na prática!

Footer

- A tag **footer** é utilizada como rodapé de página, última seção do site;
- **Mas também pode ser utilizada em outras tags**, que precisem deste último elemento;
- Geralmente contém: informações sobre a empresa, copyright, dados de contato, sitemap;
- Vamos ver na prática!

Nav

- A tag nav é utilizada para demarcar blocos de navegação;
- Ou seja, é esperado que haja **links** dentro desta tag;
- E **nem todo link precisa estar numa nav**;
- Costumeiramente são blocos grandes de link, exemplo: barra de navegação principal;
- Vamos ver na prática!

Aside

- A tag **aside** é utilizada para criar conteúdos ao lado do conteúdo principal;
- O que fica dentro de aside **precisa estar indiretamente relacionado ao conteúdo principal**;
- Exemplo: Temos uma lista de produtos sendo exibida, em aside temos os filtros e ordenação;
- Vamos ver na prática!

Figure e Figcaption

- As tags **figure e figcaption** são utilizadas em conjunto;
- **Figure é o elemento pai**, e sua função é exibir alguma imagem na página;
- A imagem é inserida pela **tag img**;
- E figcaption serve de **legenda para a imagem**;

Main

- A **tag main** é a que contém o conteúdo principal da página;
- **Não podemos utilizar mais de uma tag main** por página;
- **O conteúdo de main deve ser único**, e não repetido ao longo da página;
- Vamos ver na prática!

Mark

- A tag **mark** deve conter um texto dentro;
- Ela define um **conteúdo que precisa estar em evidência**;
- Gerando uma importância maior para o mesmo;
- Vamos ver na prática!

HTML Semântico

Conclusão da seção