Digital Innovation One: HTML Web Developer

Aprenda a programar páginas de internet e desenvolver websites utilizando HTML, CSS e JavaScript para iniciar sua trajetória profissional em desenvolvimento web front-end.

1 Bem vindo à DIO

Seja bem vindo à Digital Inoovation ne, o maior ecossistema open education em desenvolvimento de software da América Latina. VOcê vai comear agora uma jornada para criar o seu currículo com portfólio de projetos inovadores e acelerar a sua carreira para conquistar grandes oportunidades.

1.1 O que é a Digital Innovation One?

1.2 Por que a carreira em desenvolvimento de software tem um futuro muito promissor?

Digital transformation. Porque para ser competitiva as empresas precisam gerar valor contínuo para seus clientes, parceiros e sociedade. A tecnologia habilita isso.

1.3 Alinhando a sua mentalidade a dos desenvolvedores de software de alto desempenho

Soft skill (atitude, protagonismo, disciplina, criatividade, colaboração e curiosidade) e **hard skill**.

1.4 Conhecendo a plataforma da Digital Innovation One

1.5 Como tirar dúvidas técnicas sobre aulas, desafios e projetos?

1.6 10 dicas para ser contratado por uma empresa inovadora de tecnologia

Características dos aprovados:

- Repositório de projetos alinhado ao currículo;

- Artigos publicados e construção de conhecimento público na internet; Integração colaborativa a uma comunidade de desenvolvimento de software;
- Sonho grande e ambição para crescer profissionalmente;
- Utilização das principais comunidades e sites internacionais para suas pesquisas (stackoverflow);
- Graduação na área de exatas;
- Evidências de contatos com as novas tecnologias: Cloud, BigData, IA, reconhecimento de voz, mobilidade etc;
- Conhecimento em metodologias ágeis, devops e ux;
- Contas em provedores cloud e ferramentas online para devs;
- Evidências de aprendizagem contínua através da internet.

Características dos não aprovados:

- Não tem projetos e portfólio para apresentar;
- Não conhece e nunca praticou as novas tecnologias;
- Não está conectado a nenhuma comunidade de aprendizagem seja físico ou digital;
- Não apresenta elementos de colaboração e trabalho em equipe;
- Muito foco no currículo e não na jornada;
- Conhece codificação mas não conhece a arquitetura moderna e com pouca visão holística de desenvolvimento e operação de software;
- Não apresenta características de curiosidade, autodidata e protagonismo;
- Apresenta poucas evidências de disciplina para traçar metas de aprendizagem e aprender continuamente online;
- Não tem *fit* coma cultura e propósito da empresa;
- Dificuldade com algoritmos complexos e pensamento lógico.

1.7 O DIO PRO acelera a sua carreira em desenvolvimento de software

Live coding (projetos em lives e certificado) e **Labs** (projetos gravados).

1.8 A nossa equipe está torcendo por você

2 Lógica de programação essencial

Lógica de programação é a forma como o desenvolvedor entende a comunicação a fim de programar uma função de um programa. Faz uso de algoritmos, que são sequências de passos bem estabelecidos, como por exemplo, uma receita de bolo.

2.1 Introdução à lógica e à programação

2.1.1 Entendendo o que é lógica

Programar é resolver problemas, não se resume a digitar códigos.

Lógica é coerências de raciocínio, de ideias. Ou, sequência coerente, regular e necessária de acontecimentos, coisas.

Lógica de programação significa apenas contextualizar a lógica na programação de computadores, buscando a melhor sequência de ações para solucionar um problema. Em informática, essa sequência de ações é chamada **algoritmo**.

Metacognição: Pensar como você pensa.

Abstração é a abilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais (Remover informações e ações desnecessárias diminuindo o mapa mental).

Exercício final: Crie um mapa mental para resolver um determinado problema, por exemplo, calcular a média aritmética de 4 notas, sabendo que as notas são as seguintes: Nota 1: 5 Nota 2: 7 Nota 3: 10 Nota 4: 3

Resolução: Somar as notas e dividir pela quantidade de notas: (2+7+10+3)/4= 5,5.

2.1.2 O que são algoritmos e pseudocódigo

Algoritmo é uma sequência de passos que resolve um problema. O Algoritmo 1 é um exemplo de algoritmo.

O **Code.org** (https://studio.code.org/s/mc/stage/1/puzzle/1) é um site voltado para ensino de programação para iniciantes.

Algorithm 1: Exemplo de algoritmo

```
1 Acordei
2 Levantei da cama
3 Troquei de roupa
4 Escovei os dentes
5 Fui a padaria
6 Tomei café
7 Escovei os dentes
8 Fui ao trabalho
9 ...
```

Pseudocódigo é uma forma genérica de escrever um algoritmo, utilizando uma linguagem simples (nativa, ou seja, em português a quem o escreve, de forma a ser entendida por qualquer pessoa).

O jogo **olf-sheep-and-cabbage** é um jogo de lógica encontrado no site (https://www.proprofs.com/games/wolf-sheep-and-cabbage/) e o Algoritmo 2 mostra o pseudocódigo para sua resolução.

Algorithm 2: Psedocódigo para resolver o jogo Wolf, Sheep and Cabbage.

```
1 Embarca a ovelha
2 Move o barco através do rio
3 Desembarca a ovelha
4 Move o barco através do rio
5 Embarca o lobo
6 Move o barco através do rio
7 Desembarca o lobo
8 Embarca a ovelha
9 Move o barco através do rio
10 Desembarca a ovelha
11 Embarca o repolho
12 Move o barco através do rio
13 Desembarca o repolho
14 Move o barco através do rio
15 Embarca a ovelha
16 Move o barco através do rio
17 Desembarca a ovelha
```

Exercício final: 1- Crie um algoritmo do seu dia. 2 Abra o site Code.org

e tente resolver todos os problemas em menos de uma hora.

2.1.3 Aprendendo fluxograma, variáveis e constantes

Fluxograma é uma ferramenta utilizada para representar graficamente o algoritmo, isto é, a sequência lógica e coerente do fluxo de dados. **Fluxograma** é um tipo de diagrama e pode ser entendido como uma representação esquemática de um processo. Podemos entendê-lo, na prática, como a documentação dos passos necessários para a execução de um processo qualquer.

Diagrama de blocos é utilizado para representar o método do fluxograma. Exemplos de blocos (processos) são: processo, decisão, terminal, documento, vários documentos, entrada manual, preparação, dados, base de dados, display. Entretanto, não há um padrão universal de blocos.

Na programação, uma **variável** é um objeto (uma posição, frequentemente localizada na memória) capaz de reter e representar um valor ou expressão. Uma **variável** é um espaço na memória do computador destinado a um dado que é alterado durante a execução do algoritmo. A **declaração de variáveis** é feita no início do programa (Algoritmo 3), mas isso não significa que recebeu valores, apenas separou um espaço na memória.

Algorithm 3: Pseudogódigo de visualização da média de notas.

```
1  DECLARA nota 1: número
2  DECLARA nota 2: número
3  DECLARA nota 3: número
4  DECLARA nota 4: número
5  DECLARA media: número
6  LEIA(nota1)
7  LEIA(nota2)
8  LEIA(nota3)
9  LEIA(nota4)
10 media=(nota1+nota2+nota3+nota4)/4
11 IMPRIMIR(media)
```

Tipos de variáveis: As variáveis e as constantes podem ser classificadas basicamente de quatro tipos: numéricas, caracteres, alfanuméricas ou lógicas.

Constantes são valores imutáveis e não são alterados durante a vida útil do programa. O Algoritmo 4 exemplifica a declaração de constantes.

O download do programa **Flowgorithm** pode ser feita no site (http://www.flowgorithm.org)

Obs.: O nome variável nunca começa com um número, sempre com uma letra. E deve-se evitar o uso de acentuação no nome da variável.

Algorithm 4: Pseudogódigo de visualização da média de notas.

```
1 DECLARA pi=3,14          // Declaração de uma constante
2 DECLARA raio: número     // Declaração de uma variável
```

Exercício final: Exibir a soma das notas complementando o fluxograma do cálculo da média.

2.1.4 Toadas de decisões e expressões

Expressões aritméticas são expressões que utilizam operadores aritméticos e funções aritméticas envolvendo constantes e variáveis. Exemplo: 50+50, total+50. **Operadores aritméticos**: soma (+), subtração (−), multiplicação (∗), divisão (/), potenciação (^) e porcentagem (%).

Expressões literais são expressões com constantes e/ou variáveis que tem como resultado valores literais na atribuição de valor para uma variável ou constante. Exemplos: nome="José da Silva", nome←"José da Silva", media = (nota1+nota2+nota3+nota4)/4

A Tabela 1 exemplifica as expressões aritméticas e literais.

Comando de atribuição/operação	Procedimento
$A = 2$	Armazenar o valor 2 na variável A
$B = A + 3$	Somar o valor de A (2) com 3 e armazenar em B (5)
$C = A + B$	Somar o valor de A (2) e o valor de B (3) e armazenar em C (7)

Table 1: Exemplos de expressões aritméticas e literais.

Expressões relacionais são expressões compostas por outras expressões ou variáveis numéricas com operadores relacionais. As expressões relacionais retornam valores lógicos (verdadeiro/falso). A Tabela 2 mostra os operadores relacionais.

Tomadas de decisão: Quando escrevemos programas, geralmente ocorre a necessidade de decidir o que fazer dependendo de alguma condição encontrada durante a execução.

Obs.: O Texto é atribuído entre aspas.

Símbolo	Nome do operador
>	Maior que
>=	Maior ou igual a
<	Menor que
<=	Menor ou igual a
==	Igual a
!= ou <>	Diferente de

Table 2: Operadores relacionais.

2.1.5 Como utilizar a concatenação

Concatenação é um termo usado em computação para designar a operação de unir o conteúdo de duas strings[1]. Outra definição é de agrupamento de duas ou mais células que, incluindo fórmulas, textos ou outras informações contida no seu interior, dá origem a um único resultado. Pode se utilizar o **&**, + ou a **vírgula (,)** para fazer a concatenação.

2.2 Introdução ao Portugol

2.2.1 Aprenda como utilizar uma estrutura de repetição

Dentro da lógica de programação, a **estrutura de repetição** é uma estrutura que permite executar mais de uma vez o mesmo comando ou conjunto de comandos, de acordo com uma condição ou com um **contador**.

2.2.2 O que são linguagens de programação?

Linguagens de programação é uma linguagem escrita e formal que especifica um conjunto de instruções e regras para gerar programas (software). Um software pode ser desenvolvido para rodar em um computador, dispositivo móvel ou em qualquer equipamento que permita sua execução. A função das linguagens de programação é servir de um meio de comunicação entre computadores e humanos.

Tipos de linguagem de programação:

Alto nível Essas são aquelas cuja sintaxe se aproxima mais da nossa linguagem e se distanciam mais da linguagem de máquina. Exemplos: C, PHP, JavaScript, C#, C++, Python etc.

[1]String é uma sequência de caracteres.

Baixo nível É aquela que se aproxima mais da linguagem de máquina. Essas são as que você precisa ter o conhecimento direto da arquitetura do computador para fazer alguma coisa. Exemplo: Assembly.

Outra classificação para linguagens de programação é:

Compiladas É uma linguagem de programação em que o código fonte é executado diretamente pelo sistema operacional ou pelo processador, após ser traduzio por meio de um processo chamado compilação. Exemplo: C#, Visaul Basic, Delphi, C++.

Interpretadas É uma linguagem de programaão em que o código fonte é executado por um programa de computador chamado interpretador, que em seguida é executado pelo sistema operacional ou processador. Exemplo: JavaScript, PHP, Python.

Portugol é uma pseudolinguagem que permite ao leitor desenvolver algoritmos estruturados em português de forma simples e intuitiva, independentemente de linguagem de programação. **Portugol** é uma pseudolinguagem que permite ao programador pensar no problema em si e não no equipamento que irá executar o algoritmo.

Link para baixar o **Portugol Studio** (IDE para o portugol):
https://github.com/UNIVALI-LITE/Portugol-Studio/releases/

2.2.3 Aprenda a utilizar desvios condicionais (estruturas de decisão - se) e boas práticas em programação (comentários)

Para o **desvio condicional** são utilizados a palavra reservada **se**, a condição a ser testada entre parênteses e as instruções devem ser executadas entre chaves caso o desvio seja verdadeiro. Se a condição for falsa um outro conjunto de comandos deve ser executado (**senao**).

No portugol é usado o // para comentar em uma linha.

O comando **caso** é similar aos comandos **se** e **senao**, e reduz a complexidade na escolha de diversas opções. Apesar de suas similaridades com o **se**, ele possui algumas diferenças. Neste comando não é possível o suo de operadores lógicos, ele apenas trabalha com valores definidos.

2.2.4 Trabalhando com laços de repetição em Portugol

Dentro da lógica de programação, os **laços de repetição (faca...enquanto)** são uma estrutura que permite executar mais de uma vez o mesmo comando ou conjunto de comandos, de acordo com uma condição ou com um **contador**.

2.2.5 Aplicação prática com matrizes e vetores

Uma **matriz** é uma coleção de variáveis de mesmo tipo acessíveis com um único nome e armazenados contiguamente na memória.

A individualização de cada variável de um vetor é feita através do uso de **índices**.

Os **vetores** são matrizes de uma só dimensão.

3 Introdução ao Git e ao GitHub

Nesse curso vamos aprender um pouco da história do Git e como ele se tornou essencial para otimizar projetos dos desenvolvedores. também vamos conhecer seus principais comandos, como funciona a plataforma e como ela pode simplificar o seu trabalho.

3.1 Introdução ao Git

3.1.1 Entendendo o que é Git e sua importância

Git foi criado em 2005 por Linus Torvalds e é um sistema de controle de versões de código distribuído, usado principalmente no desenvolvimento de software, mas pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo.

GitHub é uma plataforma da Microsoft de hospedagem de código-fonte e arquivos com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo.

GitHub é diferente do Git, mas complementar a esta. A outros programas equivalentes ao Git e ao GitHub, mas estes são os mais usados.

Benefícios de aprender Git e GitHub:

- Controle de versão
- Armazenamento m nuvem
- Trabalho em equipe
- Melhorar seu código
- Reconhecimento

3.2 Navegação via command line interface e instalação

3.2.1 Comandos básicos para um bom desempenho no terminal

O Git é um **CLI** (*Command-Line Interface*) e não um **GUI** (*Graphical User Interface*), ou seja, ele não tem uma interface gráfica.

A Tabela 3 mostra os comandos básicos de navegação no terminal.

Windows	Unix	Descrição
cd	cd	Muda de pasta. *cd /* vai para a base do sistema operacional. *cd ..* retrocede em um nível.
dir	ls	Lista os diretórios dentro da pasta em que estamos situados.
mkdir	mkdir	Cria pastas ou arquivos
del/rmdir *diretório* /S /Q	rm -rf (r- recursive; f - force)	Deleta pastas ou arquivos. (Obs.: o comando del no Windows deleta apenas arquivos, mas não as pastas)
cls (*clear screen*)	clear (ou Ctrl+l)	Limpa o terminal.
Tab	Tab	Autocompleta o texto
echo	echo	"Printa" no terminal a frase que eu escrever

Table 3: Comandos no terminal Windows e Unix.

Todos esses comandos tem variâncias através de flags que os complementam.

Se der tudo certo em um comando, o terminal não retorna nada (*silence on the sucess*).

O maior que (>) é um redirecionador de fluxo, ou seja, ele pega a saída de uma função e coloca em um arquivo. Por exemplo, *echo hello > hello.txt*, o comando *echo* "printaria" *hello* no terminal, mas o > vai salvar essa saída no arquivo *hello.txt*.

Com a seta para cima do teclado ↑ é possível navegar pelo histórico de comandos no terminal.

Dentro do Git Bash pode usar qualquer um dos comandos da Tabela 3 independentemente do sistema operacional do seu computador.

3.2.2 Ressaltando as principais diferenças entre os sistemas operacionais para instalação

3.3 Entendendo como o Git funciona por baixo dos panos

3.3.1 Tópicos funamentais para entender o funcionamento do Git

Ir na área de trabalho, clicar com o botão direito do mouse e ir em Git Bash Here

SHA1 A sigla SHA significa *Secure Hash Algorithm* (Algoritmo de Hash Seguro) e é um conjunto de funções hash criptográficas projetadas pela NSA (Agência de Segurança Nacional dos EUA). A encriptação gera conjunto de caracteres identificador de 40 dígitos único. É uma forma curta de representar um arquivo. *openssl sha1 texto.txt*

Objetos fundamentais Veja a subsubseção seguinte.

Sistema distribuído O Git é um sistema que possui várias cópias de si mesmo em diferentes locais.

Segurança O SHA1 consegue garantir segurança tanto contra mudanças acidentais quanto maliciosas e assegura que o histórico de alterações seja completamente rastreável. Além disso, nada dentro do Git pode ser alterado e nada pode ser perdido sem que ele perceba.

3.3.2 Objetos internos do Git

Blobs (bolhas) - arquivos Contém metadados do git (tipo, tamanho, \0 e conteúdo). Tem o SHA1 do arquivo

Trees (árvores) - pastas Apontam para os blobs (armazena o nome do arquivo, o que não é feito pelos blobs) e para outras árvores. Uma tree tem um SHA1 da própria árvore.

Commits Aponta para uma árvore, para um parente (último commit realizado antes dele), para um autor e para uma mensagem. Tem um timestamp (dia, horário, em que foi criado) e um SHA1.

3.4 Primeiros comandos com Git

3.4.1 Iniciando o Git e criando um commit

A Tabela 4 mostra os comandos do Git.

O passo a passo para criar um repositório é o seguinte:

Comando no Git	Descrição
git init	Inicia o Git (cria um repositório - .git)
git add	Inicia o versionamento
git commit	Cria um commit

Table 4: Comandos do Git.

1. Crie a pasta workspace em /C.
2. Git Bash here
3. cd workspace/
4. mkdir livro-receitas
5. cd livro-receitas/
6. git init
7. ls -a (a flag *-a* mostra arquivos ocultos como o .git, que é uma pasta gerencial do git)
8. cd .git/
9. ls
10. cd ..
11. git config –global user.email "aldrimagalhaes@gmail.com"
12. git config –global user.name Rilad

O passo a passo para adicionar um arquivo ao repositório é o seguinte:

1. *git add ** ou *git add .* - Pega tudo que estiver no diretório de trabalho e adiciona para a *staging area*
2. git commit -m "commit inicial"

Alguns exemplos de tipos de arquivos são:

Markdown Forma simplificada de escrever HTML. Typora é um dos seus editores, mas pode ser usado qualquer editor de texto e sua extensão é .md. Markdown reference em Ajuda para ler mais sobre as funcionalidades do Markdown no Typora.

Navegador

HTML

3.5 Ciclo de vida dos arquivos no Git

3.5.1 Passo a passo no ciclo de vida

A Tabela 5 mostra o ciclo de vida dos arquivos no Git, além das definições refentes a esse tema.

<table>
<tr><td rowspan="2">Untracked</td><td colspan="3">Tracked</td></tr>
<tr><td colspan="3">Arquivos que são rasteados pelo Git</td></tr>
<tr><td></td><td>Unmodified</td><td>Modified</td><td>Staged</td></tr>
<tr><td>Arquivos que o Git ainda não tem ciência</td><td>Arquivo que ainda não foi modificado</td><td>Arquivo que sofreu modifi-cação</td><td>Arquivos que estão aguardando para fazer parte de um commit</td></tr>
<tr><td>(git add) Adiciona o arquivo →</td><td></td><td></td><td></td></tr>
<tr><td></td><td>(Git com-para o SHA1 dos arquivos) Edita o arquivo →</td><td></td><td></td></tr>
<tr><td></td><td></td><td>(git add) "Stage" o arquivo →</td><td></td></tr>
<tr><td></td><td>← Remove o arquivo</td><td></td><td></td></tr>
<tr><td></td><td></td><td colspan="2">← (git commit) Commit</td></tr>
</table>

Table 5: Ciclo de vida dos arquivos no Git.

Ambiente de desenvolvimento representa tudo que está na nossa máquina, que consiste em diretório de trabalho, *staging area* e repositório local. O **servidor** consiste de um repositório remoto, por exemplo, o GitHub. As alterações feitas nos arquivos na sua máquina não reflete imediatamente no repositório remoto, são necessárias determinadas ações para mover do repositório local para o remoto. Quando o commit é feito os arquivos da *staging area* são movidos para o repositório local.

Passos da aula:

1. *git status* - mostra os status dos arquivos;
2. mkdir receitas;
3. mv strogonoff.md ./receitas/ → comando para mover o arquivo *strogonoff.md* para a pasta de receitas;
4. *git status*
5. *git add strogonoff.md receitas/*
6. *git status*
7. *git commit -m "cria pasta receitas, move arquivo para receitas"*
8. *git status*
9. *echo > README.md*
10. *git status*
11. Edita README.md;
12. git status;
13. git add * → Pega tudo que estiver no diretório de trabalho e adiciona para a *staging area*;
14. *git status*
15. *git commit -m "adiciona index"*

Working directory → *git add* → ***Staging area*** → *git commit -m* → ***Local repository***

3.6 Introdução ao GitHub

3.6.1 Trabalhando com o GitHub

É recomendado colocar o email e o username no GitHub equivalente ao email e nickname do Git. Para conferir a lista de todas as configurações no seu Git, usa-se o comando *git config –list*. Para desfazer a configuração do email no Git, usa-se o comando *git config –global –unset user.email*. E, no caso do *nickname*, usa-se *git config –global –unset user.nickname*. Não há como alterar o autor e email de commits feitos previamente.

Quando o GitHub encontra um arquivo Markdown, ele mostra de forma pareceida com o Typora, por esse motivo é convencional ter arquivos README.md, que descrevem o repositório.

Você precisa criar uma chave SSH no seu computador e configurar a sua chave SSH no GITHUB. Assim não corre o risco do navegador está pedindo para logar no GITHUB em um popup minúsculo que as vezes não é exibido. Segue abaixo os comandos que utilizei para criar a chave SSH e adicionar ao GITHUB, os comandos abaixo foram executados em S.O Windows:

1. Instale o Git para windows como exibido na aula;
2. Clique com o botão direito e escolha "git Bash here"
3. digite o comando em uma única linha: *ssh-keygen -t rsa -b 4096 -C aldrimgalhaes@gmail.com*
4. Perceba que será informado o diretório onde as chaves pública e privada serão geradas, você pode alterar o diretório, mas não recomendo. Apenas aperte Enter.
5. Será exibido: *Enter passphrase (empty for no passphrase):*
6. Digite a senha para sua chave e dê Enter.
7. Será exibido: *Enter same passphrase again:*
8. Digite a mesma senha e dê Enter.
9. Você ainda pode precisar adicionar esta chave gerada ao seu "chaveiro", então por garantia digite no bash: *eval ‘ssh-agent‘*
10. Digite *ssh-add /c/User/usuario/.ssh/id_ rsa* (Obs.: ssh-add com ocaminho que foi salva a sua chave)
11. Abra a pasta onde salvou as chaves, seja pelo windows ou com o comando CD /c/Users/usuario ou diretorio escolhido.
12. Abra com o bloco de notas o arquivo .pub que é a sua chave PÚBLICA, copie todo o conteúdo deste arquivo.
13. Acesse a URL: https://github.com/settings/ssh/new
14. Digite um nome para esta chave em Title e cole a chave no campo Key, em seguida clique no botão Add SSH Key

Sua chave SSH foi adicionada ao GITHUB neste momento.

Para empurrar a versão atual do repositório local para o GitHub são executados os seguintes comandos:

1. *git status*
2. *git remote add origin git@github.com:Rilad/livro-receitas.git*[2] - adiciona a origem do repositório remoto para onde serão enviados os arquivos do repositório local
3. *git remote -v* - mostra a lista de repositórios remotos cadastrados
4. *git branch -M main* (para ficar igual ao GitHub)
5. *git push -u –verbose origin main*
6. Digite sua chave

Para remover o origin usa-se: *git remote remove origin*
Obs.: name → nickname

3.7 Resolvendo conflitos

3.7.1 Como os conflitos acontecem no GitHub e como resolvê-los

Passos da aula:

1. Modificar README.md
2. *git status*
3. *git add **
4. *git status*
5. *git commit -m "Adiciona receita pave"*
6. *git push origin main*
7. Caso tenha alguma versão nova no repositório remoto que conflite com a versão no meu repositório remoto, faz *git pull origin main* - puxa o que está no repositório remoto para o repositório local e o Git tentará juntar essas versões dos arquivos

[2]origin é um alias usado por convenção para que não seja necessário digitar o URL do repositório remoto nos próximos passos

8. Corrige o conflito manualmente e realiza os primeiros passos novamente

Um conflito é quando duas pessoas editam a mesma linha de código.
Para clonar um repositório remoto para minha máquina:

1. Code → Copia link
2. Git bash no workspace
3. git clone

3.8 Seção auxiliar - Lista de comandos complementares do Git ou GitHub

A Tabela 6 lista comandos auxiliares do Git.

Comando no Git	Descrição
git status	Mostra os status dos arquivos
mv	Move um arquivo para outra pasta
git config –list	mostra a lista de todas as configurações no seu Git
git config –global –unset user.email	Desfaz a configuração do email no Git
git config –global –unset user.nickname	Desfaz a configuração do nickname no Git

Table 6: Comandos complementares.

4 Introdução a criação de websites com HTML5 e CSS3

Nesse curso o especialista vai contar um pouco sobre a história do HTML5 e do CSS3, explicar como funciona a estrutura básica dessas tecnologias, sua semântica, principais elementos e comandos.

4.1 Introdução ao curso de HTML

4.1.1 Estrutura básica

Requisitos: editor de texto (VS code - uma outra alternativa é o notepad++) e navegador de internet (Chrome)

Tudo dentro de um arquivo HTML é um elemento HTML. A Tabela 7 mostra o Elemento HTML.

Abrindo tag		Conteúdo	Fechando tag
Tag de abertura (tipo de elemento)	Atributo		Tag de fechamento
<h1	class="titulo">	Título	</h1>

Table 7: Elemento HTML.

Estrutura básica do código em HTML5:

```
1  <!DOCTYPE HTML> <!-- It is not a HTML element. The
       doctype is found at the top of all documents and
       ensures that the browser makes an effort to try
       to follow the relevant specifications, rather
       than using a different rendering mode that is
       incompatible with some specifications.  -->
2  <html> <!-- The HTML <html> element (or HTML root
       element) represents the root of an HTML or XHTML
       document. All other elements must be descendants
       of that element. -->
3    <head> <!-- The HTML <head> element provides
         general information (metadata) about a document
         , including its title and links to scripts and
         style sheets. -->
4      <meta> <!-- Sets any metadata information that
           cannot be set by other HTML elements -->
5      <title></title> <!-- Put the title in the
           browser tab -->
6    </head>
7    <body> <!-- The HTML <body> element represents the
         content of an HTML document. Only one <body>
         is allowed per document. -->
8    </body>
9  </html>
```

Exercício: Criaremos a estrutura básica para o nosso site.

4.1.2 Material de apoio - HTML5

Definição e estrutura básica

Em 1991 Tim Berners-Lee criou essa linguagem de marcação para melhorar a comunicação entre ele e seus colegas de trabalho no CERN, desde então já surgiram 5 versões e o HTML se tornou a base da web.

Com o HTML definimos o significado e a estrutura do conteúdo da web e, além de texto, nossas páginas precisam de imagens, vídeos e vários outros formatos e para isso temos elementos HTML.

Um elemento HTML é formado pela tag de abertura e seus atributos, o conteúdo e uma tag de fechamento. E mais a frente veremos que existem elementos que não tem tag de fechamento.

Com esses elementos podemos agrupar tipos de conteúdo, alterar tamanho e forma de fontes e adicionar diferentes mídias ao nossa página web.

E agora podemos ver como é a estrutura básica de um arquivo HTML.

A primeira linha do documento deve ser o **<!DOCTYPE html>**, apesar de aparecer um elemento HTML ela apenas diz ao navegador que ele está lidando com um arquivo do tipo HTML5. Os elementos HTML vem logo abaixo.

<html> A tag html é a raiz do seu documento, todos os elementos HTML devem estar dentro dela. E nela nós informamos ao navegador qual é o idioma desse nosso documento, através do atributo lang, para o português brasileiro usamos pt-BR.

<head> A tag head contém elementos que serão lidos pelo navegador, como os metadados - um exemplo é o charset, que é a codificação de caracteres e a mais comum é a UTF-8, o JavaScript com a tag script, o CSS através das tags style e link - veremos a diferença quando falarmos sobre CSS - e o título da página com a tag title.

<body> E dentro da tag body colocamos todo o conteúdo visível ao usuário: textos, imagens, vídeos.

Prática

Como exercício para esse curso iremos construir um site pessoal, e precisamos começar com a estrutura básica que acabamos de ver.

Vamos criar um arquivo index.html e adicionar o doctype e os elementos html, head e body.

Depois adicionaremos os elementos meta e title, no primeiro adicionamos o atributo charset com o valor UTF-8 para dizer ao navegador qual é a codificação dos caracteres e no segundo podemos colocar nosso nome.

E por último escreveremos nosso nome dentro do elemento body apenas para enxergarmos isso no navegador.

Semântica

Durante muitos anos o elemento padrão no HTML era a div, construíamos nosso conteúdo todo baseado nela, e assim nascia a sopa de divs.

Mas em 2014 saiu a quinta versão do HTML, e com ela vieram vários mudanças importantes, como performance e acessibilidade, mas nesse curso introdutório vamos focar na semântica.

A semântica nos permite descrever mais precisamente o nosso conteúdo, agora um bloco de texto não é apenas uma div, agora é um article e tem mais significado assim. E temos vários elementos para ressignificar as divs:

section Representa uma seção genérica de conteúdo quando não houver um elemento mais específico para isso.

header É o cabeçalho da página ou de uma seção da página e normalmente contém logotipos, menus, campos de busca.

article Representa um conteúdo independente e de maior relevância dentro de uma página, como um post de blog, uma notícia em uma barra lateral ou um bloco de comentários. Um article pode conter outros elementos, como header, cabeçalhos, parágrafos e imagens.

aside É uma seção que engloba conteúdos relacionados ao conteúdo principal, como artigos relacionados, biografia do autor e publicidade. Normalmente são representadas como barras laterais.

footer Esse elemento representa o rodapé do conteúdo ou de parte dele, pois ele é aceito dentro de vários elementos, como article e section e até do body. Exemplos de conteúdo de um <footer> são informações de autor e links relacionados.

<h1>-<h6> Eles não foram criados na versão 5 do HTML e nem são específicos para semântica, mas servem para esse propósito. São utilizados para marcar a importância dos títulos, sendo <h1> o mais importante e <h6> o menos. Uma dica: use apenas um <h1> por página, pois ele representa o objetivo da sua página.

Prática

Dando continuidade ao nosso site iremos montar sua estrutura. Pensei em adicionarmos um cabeçalho com nosso nome, uma lista de posts (como um blog) e um rodapé para nossos contatos.

Vamos abrir nosso arquivo index.html e começar pelo cabeçalho: criamos um <header> logo abaixo do <body> e colocamos o título da nossa página dentro de um <h1>.

Depois criaremos a lista de postagens: abrimos um elemento section e dentro dele adicionamos outro <header> contendo um <h2>. Notem que eu posso ter mais de um <header> na página.

Para criar nossa postagem adicionamos um <article> com um <header> e um <h3>.

O último passo desta etapa é criar um rodapé para nossas informações de contato: crie um elemento footer antes de fechar o </body>.

Não se preocupe com o layout e com conteúdo da página, nós vamos tratar isso mais a frente.

Textos e links

A criação do HTML foi motivada pela necessidade de compartilhar textos e documentos, e mesmo depois de quase 30 anos, com toda a evolução da web, isso ainda representa uma boa parte do conteúdo da web.

Já falamos anteriormente sobre os elementos h1-h6 e, eles são essenciais para nos indicar visualmente a importância e localização de seções de texto na página, mas para textos maiores e mais densos usamos o elemento p.

O <p> representa um parágrafo, mas ele não suporta apenas texto, podemos adicionar imagens, código, vídeos e vários outros tipos de conteúdo dentro dele.

Um outro elemento interessante e extremamente necessário na web é o <a> - que significa anchor/âncora, ele representa um hyperlink, é ele que interliga vários conteúdos e páginas na web.

O elemento a tem vários atributos, mas vamos focar em dois, o href e o target.

O href representa o hyperlink para onde sua âncora aponta, pode ser uma página do seu ou de outro site, um e-mail e até mesmo um telefone, os dois últimos precisam dos prefixos mailto: e tel:, respectivamente.

O target neste momento vai servir para nos ajudar a abrir nossos links em outra aba do navegador usando o valor _blank.

Prática

Vamos adicionar um texto fictício a nossa postagem: logo após o fechamento do </header> vamos adicionar um elemento p e inserir um texto que vamos retirar do site lipsum.com

E em alguma parte deste texto vamos adicionar um hiperlink para outra página e um para nosso e-mail.

Criarei um hyperlink para meu perfil no LinkedIn: adicione o hyperlink no atributo href e o valor _blank no atributo target, assim o link será aberto em outra aba. E em algum outro lugar do texto adicionarei meu e-mail e um link para ele, desta forma: lucas@vilaboim.com

Imagens

A web também é feita de imagens e para representá-las temos o elemento <img>, ele é um daqueles elementos sem tag de fechamento.

O elemento img é bem simples, contendo apenas 2 atributos próprios, o src e o alt.

O src é obrigatório e guarda o caminho para a imagem que você quer mostrar na página.

O alt não é obrigatório mas é altamente recomendado por melhorar a acessibilidade, ele mostra a descrição da imagem caso ela não carregue e leitores de tela usam esse atributo para descrever a imagem para o usuário saber o que ela significa.

Prática

Vamos adicionar uma imagem ao cabeçalho da página e uma imagem a postagem.

Primeiro vamos colocar as imagens na pasta do nosso projeto. Para a imagem do cabeçalho eu escolhi uma foto minha com 100 pixels de largura e 100 pixels de altura e para a imagem da postagem eu procurei por html code no site Unsplash, escolhi uma das imagens e deixei ela com 960 pixels de largura por 322 pixels de altura.

Dentro do primeiro <header> da página e antes do <h1> iremos adicionar um elemento img e no atributo src colocamos o caminho para a nossa foto, /lucas-vilaboim.jpg, e o atributo alt deve conter um significado para a imagem, como no meu caso é uma ilustração, colocarei Ilustração do rosto de Lucas Vilaboim.

E dentro do <header> do <article> vamos fazer a mesma coisa, mas agora depois do <h3>, e no atributo alt colocaremos Editor de texto mostrando códigos HTML.

Listas

Os últimos elementos que veremos neste módulo são os relacionados a listas: <ul>, <ol> e <li>.

Listas servem para agrupar uma coleção de itens, como uma lista de ingredientes ou, como será no nosso caso, uma lista com contatos.

O elemento ul cria uma lista não ordenada, onde a ordem dos elementos não é importante, e é representada com pontos, círculos ou quadrados.

O <ol> serve para criar lista ordenadas, nessas a ordem importa, portanto elas são representadas com números, algarismos romanos ou letras.

E o elemento li é um item dentro de uma dessas listas. Um <li> pode conter vários tipos de conteúdos, como parágrafos, imagens e até outras listas.

Prática

Adicionaremos uma lista de contatos ao rodapé da nossa página, e para isso usaremos também o elemento a que vimos anteriormente.

Crie um elemento ul e dentro dele adicione um <li> com um elemento a, no atributo href adicione o link de alguma rede social que você mantenha e, no conteúdo da âncora coloque o nome dessa rede.

4.2 Entendendo o que é semântica

4.2.1 Semântica - Parte 1

Em versões anteriores do HTML, o elemento padrão do HTML era o ** ** e o significado do elemento era definido através de classes. Com a nova versão, foram criados novos elementos como[3]:

<section> Representa uma seção genérica contida em um documento HTML, geralmente com um título, quando não existir um elemento semântico mais específico para representá-lo. Por exemplo, um menu de navegação deve estar dentro um elemento <nav>, mas uma lista de resultados de pesquisa ou a exibição de um mapa e seus controles não possuem elementos específicos, e podem ser colocados dentro de uma <section>.

[3] As informações desses elementos e de outros do HTML foram complementadas através do site: https://developer.mozilla.org/pt-BR/

<header> Representa um grupo de suporte introdutório ou navegacional. Pode conter alguns elementos de cabeçalho mas também outros elementos como um logo, seções de cabeçalho, formulário de pesquisa, e outros.

<article> Representa uma composição independente em um documento, página, aplicação, ou site, ou que é destinado a ser distribuido de forma independente ou reutilizável, por exemplo, em sindicação. Este poderia ser o post de um fórum, um artigo de revista ou jornal, um post de um blog, um comentário enviado por um usuário, um gadget ou widget interativos, ou qualquer outra forma de conteúdo independente.

<aside> Representa uma seção de uma página que consiste de conteúdo que é tangencialmente relacionado ao conteúdo do seu entorno, que poderia ser considerado separado do conteúdo. Essas seções são, muitas vezes, representadas como barras laterais. Elas muitas vezes contem explicações laterais, como a definição de um glossário; conteúdo vagamente relacionado, como avisos; a biografia do autor; ou, em aplicações web, informações de perfil ou links de blogs relacionados.

<footer> Representa um rodapé para o seu sectioning content (conteúdo de seção) mais próximo ou sectioning root elemento (ou seja, seu parente mais próximo <article>, <aside>, <nav>, <section>, <blockquote>, <body>, <details>, <fieldset>, <figure>, <td> (en-US)). Normalmente um rodapé contém informações sobre o autor da seção de dados, direitos autorais ou links para documentos relacionados.

<h1-<h6> Representam seis níveis de título de seção. <h1> é o nível de seção mais alto e <h6> é o mais baixo, e só pode haver um por página.

4.2.2 Semântica - Parte 2

Exercício Daremos continuidade à estrutura do nosso site.

4.3 Como usar textos e links em HTML

4.3.1 Tags para textos

A criação do HTML foi motivada para compartilhar textos e documentos. Para textos maiores e mais densos, em vex de usar <h1-<h6>, usa-se o elemento ** **, que representa um parágrafo, mas ele não suporta apenas texto, pode adicionar imagens, vídeos, códigos, entre outros conteúdos. Exemplo: Conteúdo do artigo.

4.3.2 Tags para links

O elemento <a> é uma âncora, interliga vários conteúdos na web. Dois de seus atributos são:

href É o hiperlink para onde a sua âncora está apontando, pode ser uma página do seu site, um site externo, e-mails ou telefones, com estes dois últimos usando os prefixos *mailto* e *tel*. Exemplo: <a href ="linkedin.com/in/vilaboim">Lin ou E-mail

terget Indica como o link vai ser aberto, como o _ *blank* que indica que o link vai ser aberto em uma nova aba. Exemplo:Link

4.3.3 Exercício prático

Exercício: Vamos adicionar um texto a nossa postagem e um link dentro desse texto.

4.4 Como inserir imagens em seu site

4.4.1 Tag img

O elemento **<img>** representa imagens e não possui tag de fechamento. Tem dois atributos próprios:

src É obrigatório, guarda o caminho da imagem e pode ser uma imagem dentro do seu próprio site ou de outro lugar. Exemplo: <img alt="img/avatar.jpg">

alt É altamente recomendável para melhorar a acessibilidade, ele mostra a .descrição da foto quando ele não é carregada. Exemplo: <img alt="Foto de Lucas Vilaboim">

4.4.2 Exercício prático

Exercício: Vamos adicionar uma imagem ao cabeçalho da página e uma imagem a postagem[4].

4.5 Como organizar listas com HTML

4.5.1 Tags lo, ul e ol

Listas servem para agrupar uma coleção de itens. O elementos são:

[4]Site para banco de imagens: https://unsplash.com/. E site para otimizar imagens: tinypng.com

<**ul**> Representa uma lista em que a ordem dos itens não é importante. Exemplo:

Item 1

Item 2

<**ol**> Representa uma lista em que a ordem é importante, usando números, letras ou algarismos romanos. Exemplo:

1. Item 1
2. Item 2

<**li**> É um item dessa lista.

4.5.2 Exercício prático

Exercício: Adicionaremos uma lista de contatos ao rodapé.

4.6 Introdução e conceitos básicos do CSS3

4.6.1 Introdução ao CSS3

O CSS é uma linguagem de estilo que surgiu da necessidade de formatar páginas. Você cria regras de estilos para elemntos ou grupos de elementos. Uma regra CSS, mostrada na Figura 1, é formada por um seletor ou um grupo de seletores (um seletor representa apenas um elemento HTML) e dentro de chaves há as declarações (formada por uma propriedade e um valor). Este exemplo cria uma regra CSS que altera as propriedades do tipo de elemento HTML em toda a página.

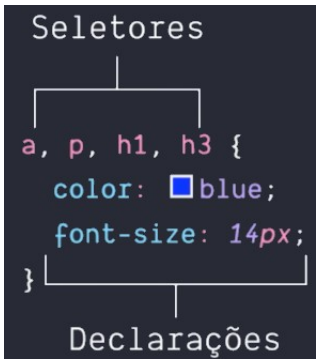

Figure 1: O que são seletores (neste exemplo, o seletor é um seletor de tipo).

ID ou classe podem ser criadas para alterar propriedades de alguns elementos, mesmo que não sejam do mesmo tipo. Dessa forma, pode-se alterar especificamente um elemnto sem modificar os outros elementos do mesmo tipo dentro da página.

ID x Classe: podem representar quaisquer tipos de elementos. No HTML, você declara o seu Id com *id* e sua classe com *class*. No CSS, uma classe é precedida por um ponto e um ID, por um *hash*, como mostrado na Figura 2. Um Id só pode ser usado uma vez na página. Exemplo: *<header id="header" class="header"></header>*

<header id="header" class="header"></header>

Figure 2: ID x CLasse.

Exercício: Vamos adicionar algumas classes no nosso site e alterar alguns elementos.

4.6.2 Conceitos básicos

Box model, mostrado na Figura 3.

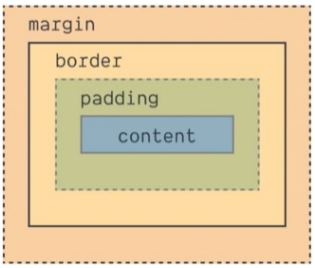

Figure 3: Box-model.

4.6.3 Material de apoio - CSS3

Definição e seletores

Após a criação do HTML a necessidade de formatar as páginas ficou evidente, assim, em 1996, foi criada a linguagem de estilo que conhecemos por CSS.

A sintaxe é bem simples e pode ser explicada com a frase "você cria regras de estilo para elementos ou grupos de elementos".

Vamos usar um elemento HTML que vimos anteriormente, a âncora <a>, para exemplificar.

Uma regra CSS é representada por um seletor ou um grupo de seletores, no nosso caso é o <a>, então dentro de um par de chaves adicionamos as declarações, no exemplo acima estamos alterando cor e tamanho da fonte dessa âncora, as declarações são formadas por uma propriedade e um valor.

Percebam que podemos colocar vários seletores em uma regra separando-os por vírgula.

E há um último detalhe nesse exemplo: a pseudo-classe. Elementos HTML sofrem alterações causadas pela interação do usuário, como mover o mouse por cima ou clicar nesse elemento.

O a:hover do exemplo significa que a âncora também terá essa aparência quando o usuário passar o mouse por cima de um hyperlink.

ID x Classe

No exemplo anterior criamos uma regra que altera um elemento HTML diretamente, mas isso significa que todos os elementos <a> ficarão com aquela aparência, e normalmente temos sites mais complexos que precisam de várias regras diferentes para elementos iguais.

Para ficar mais tangível vamos relembrar um pouco o site que começamos a fazer no módulo passado, ele tinha vários elementos header, mas não vamos querer que o header principal tenha a mesma formatação que o header de uma postagem, é aí que entram os IDs e Classes.

O seletor que vimos no primeiro exemplo é um seletor de tipo, pois ele representa um elemento HTML, e com IDs e Classes podemos representar qualquer tipo de elemento mas há algumas diferenças entre eles:

ID é representado pelo símbolo # (hash) seguido de um nome para esse ID.

Classe a classe é representada de forma parecida do ID, mas é precedida por um ponto em vez do hash.

E a diferença mais importante entre eles é a forma como devem ser usados: o ID só pode ser usado uma vez em uma página HTML enquanto a classe não tem restrições.

Exercício

Vamos adicionar algumas classes no nosso site e alterar alguns elementos, mas antes precisamos adicionar um arquivo CSS a nossa página.

No módulo de HTML descobrimos que podemos adicionar CSS de duas formas, com o elemento style, e assim suas regras ficarão no arquivo HTML, ou podemos criar um arquivo CSS e adicioná-lo na página através do elemento link, e é essa forma que usaremos.

Crie um elemento link dentro do head do seu arquivo e adicione os atributos rel="stylesheet" e href="style.css", o rel denota o tipo de arquivo que estamos incluindo na página e o href é o caminho para o arquivo. E na mesma pasta do arquivo HTML crie um arquivo chamado style.css.

Agora sim vamos ao CSS, adicione um ID #title ao h1 da página, pois queremos que ele seja único, e depois adicione as classes .subtitle e .post_title ao h2 e h3, respectivamente.

No arquivo CSS vamos mudar a cor desses três títulos, e depois alterar o tamanho da fonte do título da postagem.

Box-model

Quando estamos criando o layout de um site o navegador representa cada elemento HTML como uma caixa retangular, isso é o box-model. E com CSS nós alteramos a aparência dessa caixa (largura, altura, cor de fundo, etc.). Essa caixa é composta por 4 áreas: o conteúdo, o padding, a borda e a margem.

margens (margin) são espaçamentos entre elementos;

bordas (border)

padding é um espaçamento entre as bordas e o conteúdo, a diferença para as margens é que declarações de imagem de fundo funcionam nele;

conteúdo (content) é o que o seu bloco representa, um texto, uma imagem, um vídeo;

Exercício

Para enxergamos o box-model vamos adicionar cores e bordas a alguns elementos.

Primeiro adicionaremos uma cor de fundo para a visualização ficar mais fácil, usaremos a propriedade background com o valor #fcfcfc no elemento body.

Depois vamos adicionar uma classe ao <article>, pode ser .post, e então vamos colocar a cor branca de fundo com a propriedade background e o valor #FFF. Agora conseguimos enxergar o content do box-model.

Vamos adicionar um padding de 10 pixels neste mesmo article. Perceberam o espaçamento que surgiu em volta do nosso conteúdo?

Agora adicionamos um borda mais escura a ele com a propriedade border. Vou falar mais detalhadamente sobre border mais a frente, mas por enquanto vamos deixar essa borda com 3 pixels de largura, o contorno sólido e a cor azul.

E por último vamos adicionar uma margem do lado de fora do post com a propriedade margin e o valor 10 pixels.

E agora inspecionando o nosso elemento conseguimos todas aquelas camadas citadas antes: o conteúdo em azul, o padding em verde, as bordas em marrom e as margens em laranja.

E já que começamos a falar sobre bordas e cor de fundo, no próximo vídeo vamos nos aprofundar nessas propriedades.

Estilizando elementos

Agora que entendemos o box-model podemos focar em deixar nosso site mais bonito, então vamos repassar pelas propriedades já citadas:

Padding e Margin Anteriormente usamos o padding e o margin da forma mais básica, com apenas um valor, mas eles são mais poderosos que isso. Se quisermos atribuir tamanhos diferentes para cada lado do box nós podemos, e vamos ver três formas de fazer isso.

A primeira é colocando um valor para as partes superior e inferior e depois para os lados esquerdo e direito.

O valor de 10 pixels se refere ao eixo Y, ou partes superior e inferior, e os 5 pixels se referem aos lados esquerdo e direito.

A segunda forma é dando valores para cada lado do box.

Então começamos pelo topo com 15 pixels, passamos o lado direito com 10 pixels, depois para a parte inferior com 5 pixels e por último o lado esquerdo com 0, e sempre nessa ordem.

Uma boa dica também é que quando o valor for 0 não precisamos não precisamos colocar a unidade.

A terceira forma é com as propriedades específicas para cada lado, até agora tínhamos visto atalhos para essas propriedades.

Essa opção é mais usada quando temos o mesmo valor para 3 lados, e o quarto precisa ter um valor diferente, então usamos o padding com apenas um valor e uma dessas opções para representar o lado diferente.

Background A propriedade background também é um atalho para várias propriedades, mas isso vocês podem absorver aos poucos, e uma boa opção de leitura é a documentação do MDN.

Por enquanto veremos apenas como mudar a cor de fundo.

E aqui temos 3 formas de colocar uma cor de fundo, e ainda existem outras.

A primeira é pelo nome da cor em inglês, a segunda é pelo código hexadecimal e a terceira é usando apenas o atalho background.

Border Vimos que a propriedade border pode ter 3 valores: a largura, a cor e o estilo, mas existem algumas particularidades nisso.

A largura pode ser usada com várias unidades, como px, em e mm. A cor pode ser atribuída pelo nome ou por um código hexadecimal, assim como fizemos com o background, e o estilo é representada por palavras-chave, vamos ver algumas delas:

solid mostra uma borda simples e reta;

dotted são bolinhas com um pequeno espaçamento entre elas;

dashed forma uma linha tracejada.

E aproveitando que mostrei esse código temos que falar sobre como separar a estilização dos lados de uma borda.

E se você não quiser usar a propriedade border existem as propriedade específicas para cada aspecto de uma borda, são elas border-width para a largura, border-color para a cor e border-style para o estilo.

Aqui temos o mesmo código anterior de duas formas diferentes, a primeira com o atalho border e a segunda com cada propriedade específica.

E depois disso podemos juntar os lados com os aspectos de uma borda e criar uma regra mais específica ainda.

Border-radius E a última propriedade é o border-radius, ele permite arredondar os cantos de um elemento. Podemos usar várias unidades, mas as mais comuns são os pixels e a porcentagem.

Colocando apenas um valor mudamos todos os cantos do elemento, mas seguindo aquela mesma ordem que vimos no padding e margin - topo, direita, inferior e esquerda - conseguimos alterar cada canto separadamente.

Exercício

Neste exercício vamos deixar o nosso site um pouco mais bonito usando as propriedades que acabamos de ver.

Vamos aumentar o padding para 15 pixels e colocar uma margem de também de 15 pixels só na parte de baixo do post.

Quando olhamos para os textos percebemos que os espaçamentos estão diferentes do restante do post, então vamos padronizar isso.

No título do post vamos retirar todas as margens para depois colocar apenas uma margem inferior de 15 pixels. E no corpo do post precisamos adicionar uma classe e remover todas as margens para depois adicionar uma margem superior de 15 pixels.

Podemos manter o background branco, mas vamos diminuir a largura das bordas para 2 pixels e mudar a cor para a mesma do texto - #505050 - e por último adicionaremos um border-radius, 5 pixels são suficientes. Podemos adicionar esse mesmo de valor de border-radius na imagem, para isso vamos acrescentar uma class a imagem antes.

Estilizando textos

Já sabemos que podemos mudar cor e tamanho de algumas fontes, e agora vamos nos aprofundar nisso.

font-family Com o font-family podemos alterar a fonte dos nossos textos, como uma fonte da internet ou uma que esteja instalada no nosso computador, mas vamos nos ater às fontes seguras, chamadas de web safe fonts.

Essas fontes são chamadas assim pois são encontradas em quases todos os sistemas e podem ser usadas sem preocupação.

font-size O font-size nos ajuda a mudar o tamanho do texto, existem algumas unidades de medida para ele mas por enquanto os pixels são suficientes para nós.

font-style Usamos o font-style para tornar um texto itálico, na maioria das vezes você usará apenas o valor italic para ele, mas se precisar tirar o itálico de um texto você pode usar o valor normal.

4.7 Estilizando elementos, textos e listas

4.7.1 Estilizando elementos

4.7.2 Estilizando textos

4.7.3 Estilizando listas

4.8 Dimensão e alinhamento

4.8.1 Propriedades de dimensões e alinhamento

5 Recriando a página inicial do Instagram

Nesse projeto você terá o desafio de reconstruir a página inicial de login do Instagram no qual será abordado o conceito sobre CSS utilizando Flexbox, uma metodologia de posicionamento de elementos em tela mais utilizada no mercado assim como conceitos de responsividade, além disso a expert disponibiliza todo o material necessário em seu GitHub para que você possa realizar o seu projeto.

5.1 Conteúdos

5.1.1 Como usar os desafios de projetos para cirar seu portifólio

5.1.2 Parte 1

5.1.3 Parte 2

5.1.4 Parte 3

5.1.5 Parte 4

5.1.6 Objetivo do projeto

5.2 Informações

Nesse projeto você terá o desafio de reconstruir a página inicial de login do Instagram, no qual será abordado o conceito sobre CSS utilizando Flexbox, uma metodologia de posicionamento de elementos em tela mais utilizada no mercado assim como conceitos de responsividade, além disso a expert disponibiliza todo o material necessário em seu GitHub para que você possa realizar o seu projeto.

6 Programação para internet com JavaScript

JavaScript é uma das mais importantes linguagens front-end, e nesse curso você entenderá o porquê disso e como trabalhar com ela.

6.1 Introdução ao JavaScript

6.1.1 Introdução ao JavaScript

6.1.2 Array e dicionário

6.1.3 Condicionais, laços de repetição e Date

6.2 Desenvolva páginas web com JavaScript

6.2.1 Desenvolva páginas web com JavaScript

6.2.2 Parte 2: Manipulando elementos da página

7 Introdução a Programação com JavaScript (Teste)

Nesse desafio de codificação você irá praticar através do desenvolvimento de algoritmos os conceitos de pensamento computacional apresentados nas aulas e exercícios anteriores.

8 Recriando a Interface do Netflix

Recrie a interface do principal site de streaming mundial utilizando tecnologias simples como HTML5, CSS3 e JavaScript. Nesse projeto você aprenderá: como estruturar o layout, técnicas de CSS3 com containers e variáveis, como posicionar os elementos com Flexbox e como utilizar plugins Jquery a favor da sua aplicação.

8.0.1 Como usar os desafios de projetos para cirar seu portifólio

8.0.2 Parte 1

8.0.3 Parte 2

8.0.4 Parte 3

8.0.5 Parte 4

8.0.6 Parte 5

8.0.7 Parte 6

8.0.8 Parte 7

8.0.9 Parte 8

8.0.10 Parte 9

8.1 Informações

Recrie a interface do principal site de streaming mundial utilizando tecnologias simples como HTML5, CSS3 e JavaScript. Nesse projeto você aprenderá: como estruturar um layout, técnicas de CSS3 com containers e variáveis, como posicionar os elementos com Flexbox e como utilizar plugins Jquery a favor da sua aplicação.

9 Construindo páginas para internet com Bootstrap

Aprenda a utilizar o framework Bootstrap e aprofunde mais o seu conhecimento em HTML5 e CSS3.

9.1 Introdução ao Bootstrap

9.1.1 Aprenda sobre o framework Bootstrap

9.2 Aprenda a utilizar o Bootstrap na sua página WEB

9.2.1 Aprenda a utilizar o Bootstrap na sua página WEB

9.3 Crie containers personalizados para o seu site

9.3.1 Crie containers personalizados para o seu site

9.4 Trabalhando com imagens dentro de containers

9.4.1 Trabalhando com imagens dentro de containers

9.5 Crie lista de navegação com imagens em seu projeto

9.5.1 Crie lista de navegação com imagens em seu projeto

9.6 Crie lista de navegação dentro de containers utilizando imagens

9.6.1 Crie lista de navegação dentro de containers utilizando imagens

10 Fundamentos Aritméticos em JavaScript (Teste)

Nesse desafio de codificação você irá praticar através do desenvolvimento de algoritmos os conceitos de pensamento computacional apresentados nas aulas e exercícios anteriores.

11 Recriando o jogo da cobrinha com JavaScript

Já pensou em criar seu próprio jogo do zero? Aprenda a desenvolver de forma simples o clássico jogo da cobrinha utilizando HTML, CSS e JavaScript.

11.1 Conteúdos

11.1.1 Como usar os desafios de projetos para cirar seu portifólio

11.1.2 Parte 1

11.1.3 Parte 2

11.1.4 Parte 3

11.1.5 Parte 4

11.1.6 Parte 5

11.1.7 Parte 6

11.1.8 Objetivo do projeto

11.2 Informações

Já pensou em criar seu próprio jogo do zero? Aprenda a desenvolver de forma simples o clássico jogo da cobrinha utilizando HTML, CSS e JavaScript.