**Introdução ao Desenvolvimento Web com JavaScript**

**Aula Introdução**

Site dribbble de design para inspiração da API poke deks.

**Aula Configuração do ambiente**

Npm (node packeged maneger) é o gerenciador de pacotes do nodejs.

Instalar a biblioteca npm install -g http-server.

Quando estamos trabalhando com paginas web nós trabalhamos abaixo de um servidor http.

Para abrir o servidor http: http-server ./

**Aula protocolo cliente servidor**

Protocolo orientado cliente servidor que um desses protocolos é o http que usamos na internet. Como funciona o protocolo cliente servidor: O cliente (browser) faz uma solicitação, o servidor vai processar essa solicitação e vai devolver uma resposta para o cliente. https://, o s significa que que o site está com uma camada de segurança (criptografia).

Quando falamos de comunicação entre computadores nós temos o padrão ip/porta onde ip é o endereço do computador dentro da rede e a porta é uma das aberturas do computador.

O http server ele identifica qual o arquivo que você está solicitando e devolve o arquivo. Ele mapeia a pasta onde esta o arquivo no computador e devolve caso seja solicitado.

**Aula introdução ao HTML**

O html vai nos dar a parte textual e a semântica do conteúdo.

**Aula introdução ao CSS**

O css vai dar os estilos da página.

**Aula introdução ao javascript voltando para web**

O javascript vai dar o dinamismo da página.

O javascript é o que da comportamento para a página.

2 objetos que vão ajudar a manipular o site: **window** (objeto que representa o brawser) e o **document** (documento corrente no caso o próprio arquivo html).

**document.getElementById(“btn”)** é para buscar o elemento (tags html) através do id da tag.

F12 para abrir o inspecionar.

**variável.addEventListener(“click, function () {}”)** que é para chamar um evento que no caso é o click. Quando acontecer o click execute a função.

##### Aula Criando um Projeto com HTML/CSS para Listagem de Pokémon

**Adicionando normalize css no projeto**

Normalize é um css que ele padroniza todo o estilo inicial para vários navegadores.

Normalize.css <https://necolas.github.io/normalize.css/>. Cdn normalize css é um servidor global. <https://cdnjs.com/libraries/normalize> copiar <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/normalize/8.0.1/normalize.min.css" integrity="sha512-NhSC1YmyruXifcj/KFRWoC561YpHpc5Jtzgvbuzx5VozKpWvQ+4nXhPdFgmx8xqexRcpAglTj9sIBWINXa8x5w==" crossorigin="anonymous" referrerpolicy="no-referrer" /> e colar abaixo da tag title no html.

**Aula organizando o projeto**

**Aula adicionando a font roboto no projeto**

<https://fonts.google.com/specimen/Roboto?query=roboto>

**Aula estilização da listagem de pokémon**

**Aula estilização inicial**

Técnica mobile first como uma boa prática que é fazer o css mobile primeiro e para as telas de outros tipos de dispositivos fazer depois.

**Aula estilização de listagem de pokémon**

**Aula adicionando a imagem do pokémon**

**Aula adicionando os tipos de pokémon e finalizando o layout da imagem**

**Aula tornando nosso layout responsivo**

A mídia query é um conjunto de css que vai ser aplicado de acordo com o tamanho da tela.

Centralização por margin vai ser colocado o máximo possível tanto para a esquerda quanto para a direita.

**Aula separando os estilos**

**Dominando o protocolo HTTP e integrando com a PokeAPI**

**Aula Introdução a APIs**

Uma api ela é um servidor, web service para servir dados para o usuário. E api do pokeapi roda em cima do protocolo http.

Tudo que está na internet acaba passando pelo protocolo http.

**Aula como funciona as requisições**

Quando o cliente (browser) faz uma requisição para o servidor, o servidor processa essa requisição e retorna como resposta um json contendo as informações da requisição. E o navegador (browser) faz a formatação desse json para mostrar para o usuário.

**Aula URL path e method**

Para uma requisição http ser montada temos primeiro a URL que é o path, ip e porta do servidor e o endereço do recurso que queremos. A URL é composta pelo IP/path.

Path significa caminho de identificação do recurso.

O servidor de dns faz a conversão de um domínio em ip para fazer a comunicação.

O que vem depois do ip é a identificação do recurso. Ex: https://...identificação do recurso > /api/v2/pokémon

Quando digitamos uma URL no browser e damos um enter o tipo de solicitação que ele sempre faz é o GET.

Request method ela tem um tipo de requisição que poder ser um GET, POST, PUT, DELETE, PATCH e REQUEST. E para cada tipo significa uma coisa para o servidor.

Para a requisição:

GET: Um get na url o servidor entendi que eu quero buscar o recurso.

POST: Um post na url o servidor entendi que estou querendo inserir.

PUT: Eu digo para o servidor que eu quero atualizar.

DELETE: Eu digo para o servidor que eu quero deletar.

Toda requisição para um recurso o método diz o que quero fazer sobre esse recurso principalmente em apis que seguem padrões de projetos REST.

**Aula path params e query string**

Eu consigo transferir dados através do path.

Query string é uma discrição de busca. A sintaxe query string começa após o ponto de interrogação (?) e a query string sempre vai ser a estrutura chave/valor.

Ex da sintaxe:

?type=grass&name=i

Type = grass

Name = i

**Aula headers**

Request headers e response headers são área de dados que podemos transmitir coisas para o servidor e receber coisas do servidor. Esses headers normalmente eles são meta dados da nossa requisição, são coisas para descrever a nossa requisição ou complementar a nossa requisição. Os headers são uma espécie de configuração da nossa requisição.

Response headers é a configuração da resposta e quem faz essa configuração é o servidor.

Request headers é a configuração da requisição e quem faz essa configuração é o cliente (browser).

E o response headers e o request headers é a configuração da nossa API e configuração da nossa requisição.

**Aula body e status code**

O body é o corpo da nossa requisição que é o dado que eu quero trafegar na requisição.

O body da resposta para a requisição é um json.

Uma resposta do servidor ele vai ser um status code que significa o que aconteceu com a nossa requisição se ela foi processada ou não foi processada. O status code é a resposta através de um número que volta e ele vai indicar o que aconteceu com a nossa requisição.

Exemplos: A familia 200 a 299 quer dizer sucesso. A família 400 erro por parte do cliente. A família 500 é erro interno do servidor.

**Resumidamente a requisição HTTP:**

URL: <http://pokeapi.co/api/v2/pokemon>

Request Method: GET

Request Headers: contente-types: application/json

Request Body: {“i”: “teste”}

Status Code: 200 OK

Response Headers

Response Body

E essa é a ilustração de toda a composição de uma requisição HTTP.

**Integrando com a pokeAPI**

**Aula introdução ao consumo da pokeAPI**

Sem conteúdo

**Aula Primeira requisição**

O objetivo é chamar o endpoint via http.

A 1ª coisa que precisamos para chamar o endpoint é a url.

Eu inicio o fetch() passando a url como parâmetro.

**O fetch()** ele vai retornar uma promisse e essa promisse é importante para poder lidar com assincronismo no js.

**Promisse** é uma interface que usamos em js muito importante e sempre que vamos usar assincronismo resolvemos através de promisse.

**Endpoint** <https://pokeapi.co/api/v2/pokemon/ditto> e essa endpoint me retorna um pokémon em especifico (ditto).

**E**ssa **endpoint** me dá a lista com os primeiros pokémons <https://pokeapi.co/api/v2/pokemon?offset=0&limit=10>

Chamar o endpoint via http usando o fetchAPI que é uma biblioteca que já é integrada com o javascript do browser. <https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API>

<https://developer.mozilla.org/en-US/docs/Web/API/response/json>

**Aula manipulando o resultado de uma requisição através de uma promisse**

Método .then() é faça alguma coisa.

Método .catch() para tratar o error.

Método .finally() vai chamar algo independente de erro ou sucesso na requisição e o promisse sempre vai passar pelo finally.

**A interface de uma promisse** é os métodos .then(), .catch(), .finally() ou eu posso usar o try{}, catch{} e finally{}.

**O fetch** usa o método get por padrão.

Eu posso fazer um encadeamento do método .then() para receber o retorno do 1º .then() para evitar o tratamento com muito call-back.

**Aron function (=>)** é uma sintaxe reduzida em definição de uma função. Eu posso usar essa função passando direto o return sem a palavra return diminuindo o código.

No encadeamento do .then() sempre o que vai para o segundo .thne() é o retorno do primeiro e o que vai para o primeiro .then() é o retorno da promisse.

**Aula transformando nossa lista de pokémon em uma lista de HTML**

**Aula separando o consumo da API da manipulação da HTML**

Separar os códigos de consumo de api e códigos de manipulação de html em arquivos separados.

**O map** é uma função auxiliar que nos ajuda a manipular listas. Essa função vai transformar um elemento em outro elemento.

**Aula utilizando a função map para diminuir a verbosidade do nosso código**

**A função .join()** vai juntar todos os elementos da minha lista em uma string com o separador que eu passar dentro dos parênteses do join.

**Aula revisando o nosso código e entendendo os próximos passos**

**Aula manipulando múltiplas requisições em paralelo**

**Aula convertendo o modelo do pokeAPI para nosso modelo**

**Aula adicionando os tipos de pokémons dinamicamente**

**Aula adicionando o botão de paginação**

Criação do botão de paginação

**Aula criando mecanismo de paginação**

**Aula criando mecanismo para limitar na primeira geração de pokemons**

Criando o mecanismo de limitação de pokemons