* **npm**: Node Package Manager
* Para instalar uma biblioteca do nodejs, vamos no terminal e digitamos: **npm install -g nome-da-biblioteca**
* Criar um arquivo a partir do terminal: **touch nome-do-arquivo.extensão**
* **Ctrl + p**: usado para pesquisar arquivos por nome
* Para inicializar um servidor http, digitamos no terminal: **http-server ./** (caso queiramos que seja inicializado na pasta atual). Isso trará um **endereço IP** no terminal, que abrirá nossa página ao clicarmos
* **Protocolo cliente-servidor** (exemplo é o HTTP**):** um cliente faz uma solicitação ao servidor que por sua vez processa essa solicitação, entende o que está sendo pedido, faz o que ele deve fazer e devolve uma resposta para o cliente

Exemplo de servidor: <http://192.168.0.101:8080/> (IP:porta), onde:

- IP (http://192.168.0.101): endereço da máquina dentro da rede

- Porta (8080): uma das aberturas da máquina

* O **HTML** é uma linguagem de marcação de texto (semelhante a um documento Word): utilizamos essa linguagem para marcar nosso texto dando significado para ele
* O **CSS** é o estilo do nosso texto: podemos utilizá-lo através de uma **tag style** dentro do HTML
* O JavaScript é toda a parte que atribui comportamento às nossas páginas
* Boa prática: colocar o CSS dentro da tag head e o JS no final do HTML, dentro da **tag body**, pois no navegador, ele sempre executa o que está no header primeiro, para depois interpretar o restante do código
* Podemos utilizar o JavaScript dentro do HTML através da **tag script**
* Comandos do JavaScript importantes para a manipulação da página:

1. **window** (manipulação do navegador);
2. **document** (documento atual, ou seja, o HTML em que o código está sendo escrito);
3. **getElementByID** (acessa qualquer objeto da página que criamos através do seu ID que declaramos);
4. **addEventListener** (cria uma funcionalidade, exemplo: clicar)

* **Normalize:** um CSS que padroniza todo o estilo inicial, ou seja, todo estilo padrão ficará igual/padronizado para todos os navegadores, já que existe diferença na interpretação do CSS entre eles **(usar Normalize.css)**
* Quando subimos uma página a partir do servidor local (nossa máquina), o navegador pode guardar **cache** do que fizemos. Então, o ideal é que façamos o seguinte:

- Na nossa página na web, apertar F12 para inspecionar

- Selecionar o menu superior **Network**

- Clicar em “**Disable cache**”

* Desabilitando o cache, temos a garantia de que estamos vendo na página exatamente o que estamos programando. Com o cache habilitado, pode acontecer de fazermos uma alteração e ela não ser refletida na página, nos fazendo acreditar que fizemos algo incorretamente
* Para fins de organização de projeto, devemos criar uma página **assets**, onde colocaremos tudo que for CSS, imagens e JavaScript
* Importar o **CSS** externo para o HTML: dentro da **tag head**, embaixo da última instrução digitamos **link** e damos tab para o autocomplete do vscode. Digitamos o caminho do arquivo css dentro da **tag** **href** (Exemplo: <link rel="stylesheet" **href="/assets/css/global.css"**>)
* Importar o **JavaScrit** externo para o HTML: dentro da **tag body**, como última instrução digitamos **script:src** (**src**: source, significa que está vindo de um arquivo) e damos tab para o autocomplete do vscode. Digitamos o caminho do arquivo css dentro da **tag** **href** (Exemplo: <link rel="stylesheet" **href="/assets/css/global.css"**>)
* Para fazer uso de diversas fontes no projeto, devemos acessar **Google Fonts**, selecionar a fonte pelo nome, escolher algumas variedades dessa fonte e copiar os **links HTML** dados pelo site
* Copiamos esse link e colocamos dentro da **tag head** do arquivo HTML, antes da **tag** que referencia o caminho do arquivo CSS
* Copiamos também o comando CSS dado pelo site do Google Fonts e colocamos dentro do nosso arquivo CSS
* Para aplicar uma formatação para todo o site, usamos o **asterisco \***

Exemplo:

**\*** {

font-family: 'Roboto', sans-serif;

}

* **Técnica mobile-first**: fazer primeiro o estilo de CSS para celulares e incrementar CSS para as telas posteriores (PC, tablet, etc.). Se começamos pelas telas grandes, quando fossemos fazer a responsividade para o mobile, teríamos que **reescrever** muito código CSS, diminuindo os atributos
* Uma **lista ordenada** no HTML é dada pela **tag ol** (enumerando os itens). Para remover essa numeração, devemos aplicar um **list-style: none;** no CSS
* Uma **lista por tópicos** no HTML é dada pela **tag ul**
* Cada item dessa lista ordenada deve ser colocado dentro de uma **tag li**
* Uma **tag section** indica que teremos um conteúdo naquele espaço
* Dentro das nossas tags HTML podemos criar uma **class**

Exemplo:

<section **class="content"**>

<h1>Pokedex</h1>

<ol>

<li>

Bulbasaur

</li>

<li>

Ivysaur

</li>

<li>

Venusaur

</li>

</ol>

</section>

* É a partir do nome que damos a essas classes (**contente**, no código acima) que atribuímos estilo para aquele tag no CSS (formato: **.nome-da classe**)

Exemplo:

**.content** {

padding: 1rem;

}

* Para transformar algo em **tabela** no CSS, utilizamos o comando **display: grid;**
* Para colocar uma imagem no nosso HTML, utilizamos a **tag img** e passamos o caminho da imagem dentro de **src**
* O comando **display: flex;** faz com que os elementos se organizem dentro de um espaço, sem um sobrepor o outro

- **flex-direction:column;:** um elemento fica acima e outro abaixo

- **flex-direction: row;:** um elemento fica ao lado do outro

* O comando **max-width: 100%;** faz com que o elemento nunca ultrapasse a delimitação em que ele está dentro, saindo para fora da margem

Media Query

* **Media query**: conjunto de CSS que vai ser aplicado de acordo com o tamanho da tela. Em outras palavras, é a responsividade da aplicação