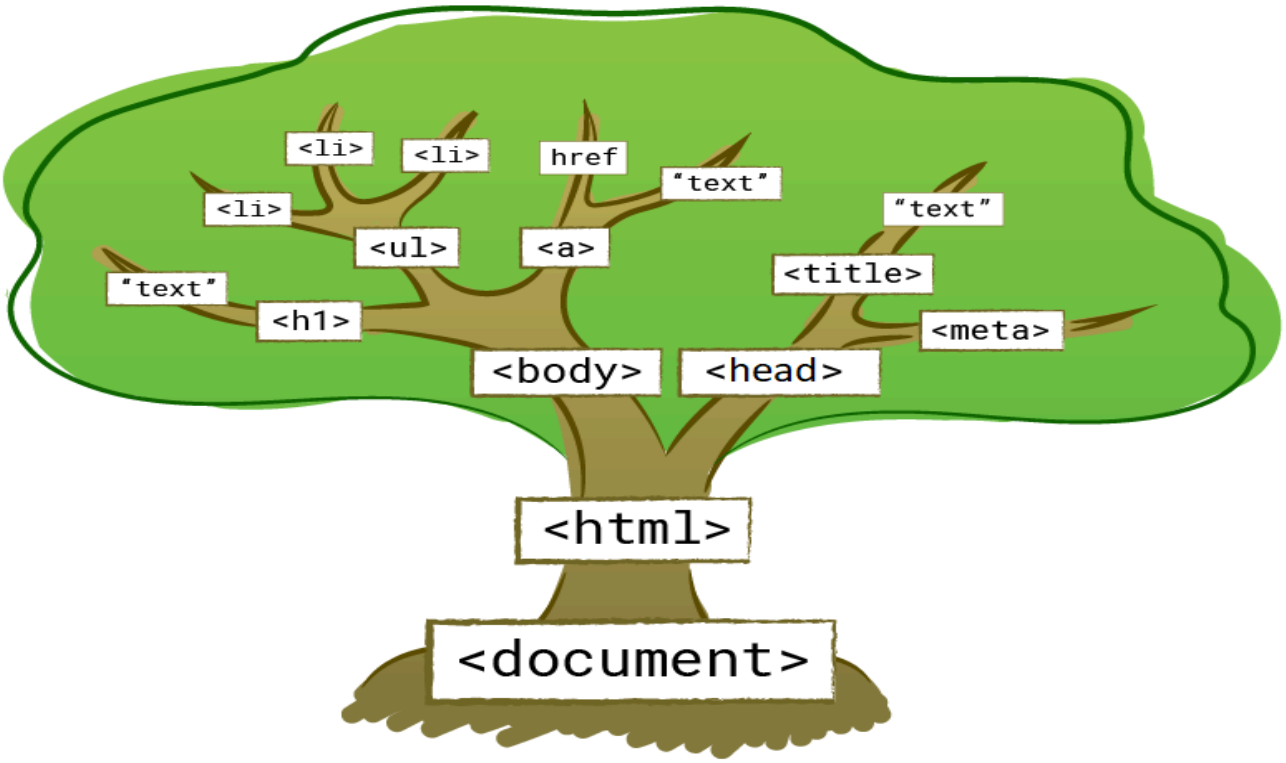


DOM

O Document Object Model ou simplesmente DOM é utilizado pelo navegador Web para representar a sua página Web. Quando altera-se esse modelo com o uso do **Javascript** altera-se também a página Web. É muito mais fácil trabalhar com DOM do que diretamente com código **HTML** ou **CSS**.

Um dos grandes responsáveis por isso tudo é o objeto *document* que é responsável por conceder ao código Javascript todo o acesso a **árvore DOM** do navegador Web.

Portanto, qualquer coisa criada pelo navegador Web no modelo da página Web poderá ser acessado através do objeto Javascript *document*.

Usa-se o DOM principalmente para atualizar uma página Web ou quando se quer construir uma interface de usuário avançada.

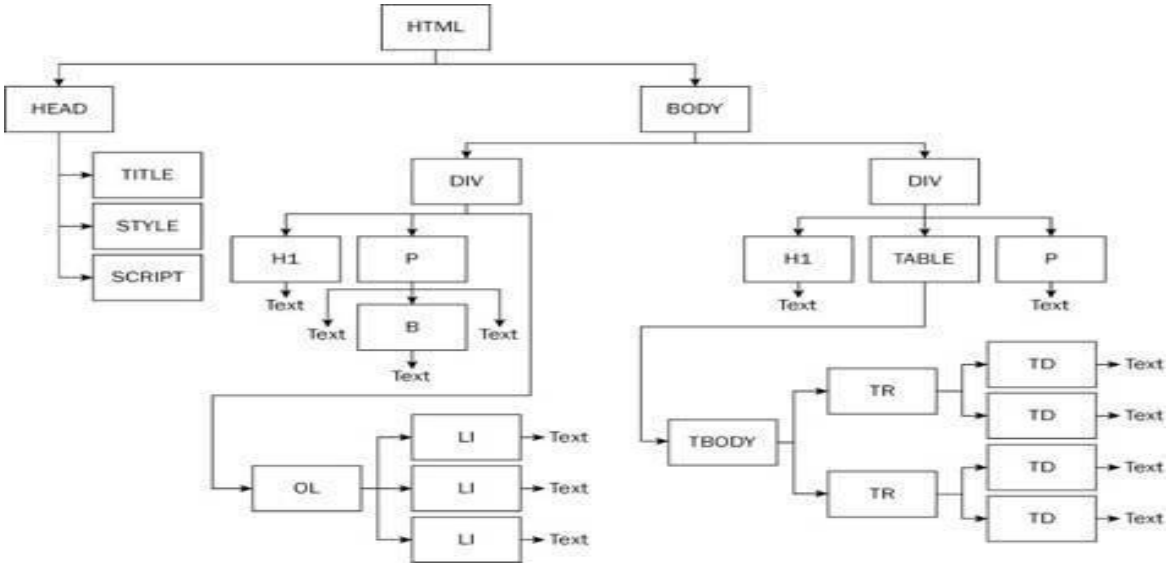
Com o DOM pode-se mover itens dentro de uma página ou criar efeitos CSS bastante interessantes sem precisar nem mesmo recarregar a página.

Objeto Document

Através do objeto *document* pode-se ter acesso a um grande número de propriedades. Segue abaixo algumas propriedades que podem ser utilizadas com o objeto *document*:

Propriedade	Descrição
documentElement	Captura o elemento raiz <html> de um documento HTML.
getElementById	Busca um elemento da página Web com o uso do atributo id do elemento.
createElement	Cria um nodo elemento na página.
createAttribute	Cria um nodo atributo na página.
createTextNode	Cria um nodo texto na página.
getElementsByTagName	Retorna um array dos elementos com o mesmo nome.
appendChild	Insere um novo elemento filho.
removeChild	Remove um elemento filho.
parentNode	Retorna o nodo pai de um nodo.

Segue abaixo um exemplo de uma árvore DOM de uma página Web. Pode-se notar que todos os elementos da página Web estão disponíveis para serem manipulados.



Exemplo de uma árvore DOM de uma página Web

Todas as páginas Web são de alguma forma uma árvore. Isso se deve ao fato de podermos ver uma página Web como uma árvore, com uma raiz como o elemento HTML e os seus filhos como o HEAD e o BODY que por sua vez também possuem elementos filhos e assim sucessivamente.

Os elementos que não possuem filhos são chamados de nós folhas, como por exemplo os elementos TITLE, STYLE, SCRIPT, LI, H1, P, TD demonstrados acima.

Note que Text é um texto que está dentro de um elemento. O nó <TD> por exemplo também é considerado um nó, mas um nó de tipo diferente (tipo texto).

Essa estrutura de árvore é a forma que o navegador organiza as marcações do HTML, é dessa forma que o navegador Web enxerga um documento HTML. A leitura da árvore se dá sempre da esquerda para a direita, assim teremos a página Web original.

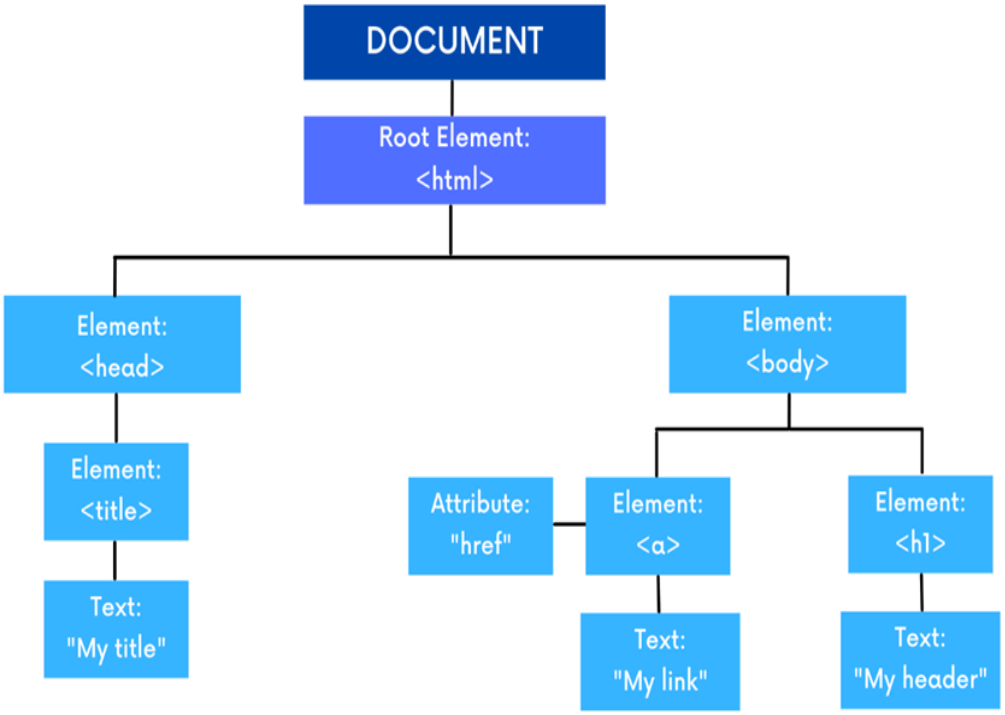
Uma boa prática para evitar que os navegadores apresentem a pagina Web de formas diferentes é sempre escrever HTML padrão, se possível sempre validando a página HTML. Fechar os elementos corretamente, utilizar tags atuais evitando as tags desatualizadas ajudam os navegadores a exibirem uma página Web de maneira correta.

As especificações do Object Document Model são publicadas pela World Wide Web Consortium (W3C). Portanto, o DOM é um padrão de fato.

Estrutura

O DOM é como uma árvore genealógica, porém, de forma invertida. O elemento que antecede o *document* é o *window*, que nada mais é que a janela do navegador. Em sua estrutura, o *document* está no topo como objeto global e tem como elemento raiz a *tag* **html** e todas as outras descendem dela através das suas ramificações (*branchs*).

A *tag* **html**, objeto pai, apresenta dois objetos filhos: o **head** e o **body** (o cabeçalho e o corpo). Os objetos que seguem nas ramificações de baixo são denominados como *child*, e os de cima, *parent*. A *tag* **head** é *parent* da *tag* **title**, e a **body** é *parent* das *tags* **a** e **h1**, e assim sucessivamente, de acordo com a hierarquia. Das *tags*, derivam os atributos, e destes, seus valores.



No código, ele ficaria traduzido dessa maneira:

```
<!DOCTYPE html>
```

```
<html>
<head>
 <title>My title</title>
</head>
<body>
 <a href="My link"></a>
 <h1>My header</h1>
</body>
</html>
```

A árvore de elementos do HTML.

Sigla para Document Object Model, o DOM é a interface entre a linguagem Javascript e os objetos do HTML.

O DOM foi criado pela W3C com o objetivo de desenvolver um padrão para linguagens de script para os navegadores já que antigamente cada navegador tinha seu próprio modo de manipular os objetos, o que gerava muita incompatibilidade e obrigava os desenvolvedores a escrever uma versão de script para cada navegador.

Quando uma página web é carregada o navegador cria o DOM, a árvore de elementos do HTML.

Entendendo a árvore

- ***Document***

Quando um documento HTML é carregado no navegador da Web, torna-se um objeto de documento. O objeto de documento é o nó raiz do documento HTML e o "dono" de todos os outros nós.

- ***Element***

O objeto de elemento representa todas as tags que estão em arquivos HTML. Os objetos de elemento pode ter nós filhos de nós de texto, além de atributos.

- *Text*

Texto que vai entre os elementos, o conteúdo das tags (este é um texto).

- *Attribute*

O objeto atributo representa um atributo que pertence sempre a um elemento HTML.

Através da estrutura criada, é possível, adicionar, alterar e remover elementos e atributos da árvore DOM utilizando JavaScript.

Onde ele está inserido?

E a pergunta que surge é: mas o DOM faz parte do HTML ou do JavaScript? Na verdade, de nenhum, ele é gerado pelo *browser*. Ao carregar a página, o navegador cria o documento, a interface, e o Javascript usa o DOM para se conectar ao HTML.

Para realizar a comunicação entre eles é necessário inserir a *tag* **script** no arquivo HTML, e como boa prática, ela deve estar antes do fechamento da *tag* **body** para que os *scripts* sejam carregados após o código base.

É possível realizar de duas formas: escrevendo o código em JavaScript dentro da própria *tag* script, ou inserindo o caminho relativo do arquivo externo. Também como boa prática, a segunda opção é a mais recomendada para a separação de responsabilidades e melhor manutenção do código.

Exemplo:

```
<script src="script.js"> </script>
```

ou

```
<script>    alert("Olá, Mundo!")</script>
```

Maneiras de manipulá-lo

São várias as formas de navegação dentro do DOM, no JavaScript utilizamos o objeto *document* e através do ponto(.) acessamos as propriedades e métodos, possibilitando selecionar, alterar, deletar, e criar elementos aos componentes do website, de acordo com a padronização criada pela W3Schools.

Para realizar essas ações temos alguns métodos, tais como:

- ✔ document.getElementById();
- ✔ document.getElementsByClass();
- ✔ document.getElementsByTagName();
- ✔ document.querySelector();
- ✔ document.querySelectorAll();
- ✔ document.createElement();
- ✔ element.addEventListener();

Com o document.querySelector, por exemplo, dentre as opções que ele oferece, podemos alterar o texto no documento HTML:

```
<body>
 <h1> Olá! </h1>
 <script src="script.js"></script>
</body>
```

Ou

```
document.querySelector("h1").innerText = "Olá, Mundo!"
```

A janela do navegador é representada pelo objeto *window*.

Utilizando outros exemplos, quando escrevemos a função *alert* ou o método *write*, é como se escrevêssemos dessa forma:

<script> window.alert("Olá, Mundo!") window.document.write("Olá, Mundo!")</script>

O *window* pode ser omitido, ele não é obrigatório no JS, assim como acontece com o ponto-e-vírgula (;) ao final de um comando.

<script> alert("Olá, Mundo!") document.write("Olá, Mundo!")</script>

O DOM é um conjunto de objetos e sua estrutura de dados é representada através de um diagrama, ou de forma figurativa, uma árvore de objetos.

Quando se fala do desenvolvimento *web* básico, temos a tríade inseparável: HTML, CSS e JavaScript, cada um com o seu papel. Conhecer a base da programação e o que eles representam é essencial para poder avançar com *frameworks* e bibliotecas como Angular, Vue.js e React.

A tecnologia é uma área dinâmica, e assim como em diversos aspectos da nossa vida, não é necessário ter um dom para realizar algo, mas sim, estudo contínuo e muita prática.

Saiba mais:

https://www.w3schools.com/js/js_htmldom.asp

https://www.youtube.com/watch?v=HOv9CqqAZk0