MATERIAL EXTRA

Introdução ao DOM

growdev

JAVASCRIPT

Todas as páginas e aplicações web são construídas com base na linguagem HTML que, em conjunto com o CSS, consegue estruturar componentes e criar todo o layout do documento.

Porém, uma página apenas HTML + CSS é **estática**, ou seja, não possui um comportamento dinâmico. Páginas estáticas não permitem que o seu conteúdo seja alterado no documento sem que ocorra um reload (recarregamento). Desta forma, a **interatividade da página** com o usuário é comprometida.

Para adicionarmos um **comportamento** às páginas web, precisamos inserir scripts usando a linguagem **Javascript** (JS), completando a tríade básica do front-end: HTML + CSS + JS.

(Disponível em: 6 steps to become a Front-end Developer - Tolustar)

O JS é, **originalmente**, uma linguagem **client-side**. Ou seja, ela nasceu para ser executada no lado do cliente e não no lado do servidor[1]. Em páginas HTML, os scripts JS são executados diretamente no navegador.
O grande poder do Javascript está na possibilidade de criarmos **programação para o front-end**, fazendo que as páginas web antes apenas HTML e CSS sejam, então, dinâmicas. A capacidade de manipulação das páginas é obtida pelo JS a partir do **DOM**.

DOM

Toda a interação do Javascript com o documento HTML é feita através de uma interface chamada DOM (**Document Object Model**). O DOM representa como documentos escritos em HTML[2] são processados pelo navegador. Assim, enquanto lê a estrutura da página, o navegador cria um **objeto estruturado** que é fiel ao documento.

Historicamente, o **DOM** e o **Javascript** sempre estiveram fortemente ligados. Hoje ambos são considerados entidades diferentes. O Javascript é uma linguagem de programação e, por outro lado, o DOM é uma interface de programação para o mapeamento de HTML. Ainda assim, o DOM é a base para o uso da linguagem Javascript na programação client-side.

A interface que o DOM provê permite **manipular elementos** do HTML, além de alterar as propriedades CSS e seus valores, bem como adicionar rotinas que executam conforme a interação do usuário com a página. Todos esses recursos e modificações no documento podem ser feitos sem a necessidade de recarregamento da página.

---

[1] Hoje em dia, porém, o Javascript é bastante usado no lado servidor.
[2] O DOM também faz mapeamentos de outras linguagens de marcação, como é o caso do XML ou do SVG. Neste documento, porém, trataremos apenas do HTML por ser a linguagem usada para estruturar documentos na programação web.

Ou seja, o conteúdo da página é carregada em uma interface no navegador que pode ser acessada e manipulada pelo Javascript. Com isso, o Javascript pode ser programado para que a página seja alterada em **tempo real**. Cada modificação no DOM faz com o que o navegador modifique a renderização da página.

O DOM também define como a estrutura do documento pode ser percorrida, **como os elementos podem ser acessados** e, também, descreve funções para que os componentes da página sejam modificados.

Com o DOM, é possível modificar em tempo real as informações de uma página. Vejamos alguns exemplos.

Exemplo 1: Considere uma rede social como o **Twitter**. Enquanto estamos com o Twitter aberto no navegador, não precisamos recarregar a página sempre que o feed é atualizado. Quem atualiza a lista de tweets é o Javascript a partir do DOM.

Exemplo 2: uma loja virtual (e-commerce) que possui um carrinho com um contador de produtos. Ao selecionarmos um produto para a compra, a aplicação o adiciona no carrinho e imediatamente atualiza o contador na página. Usando o DOM + JS, o contador pode ser atualizado na página.

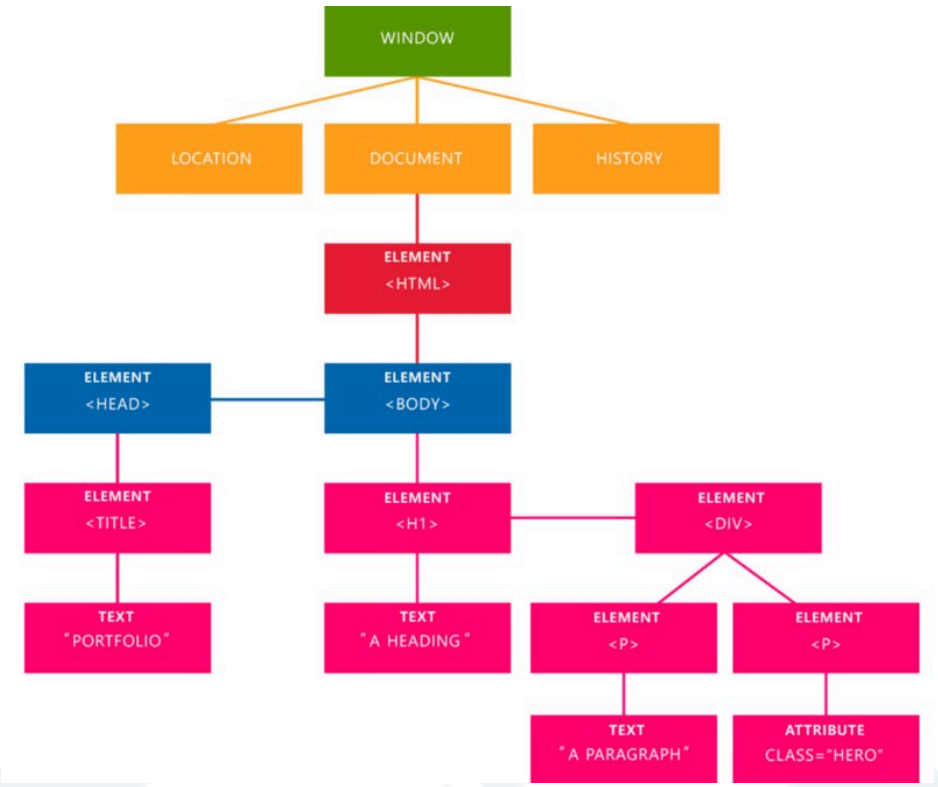
Além de ser possível mudar o texto ou o conteúdo de elementos HTML, o DOM permite que a **estilização da página** seja modificada. Considerando o exemplo do e-commerce já citado, um script JS poderia ser programado para trocar a cor de fundo do carrinho conforme o número de itens adicionados.

Essas características auxiliam na percepção do usuário em estar utilizando uma **aplicação web interativa**. Com os avanços nas tecnologias de comunicação e da Internet, o número de aplicações web cresceu de forma significativa nos últimos anos, facilitando o acesso dos usuários a esse tipo de software. Esse avanço foi potencializado com o crescimento do Javascript junto ao DOM.

Representação visual

O DOM é montado em uma **estrutura de árvore**, onde um nó pode ter N filhos de acordo com uma hierarquia de elementos. O fato de ser uma árvore aproxima o DOM do próprio HTML que segue uma estrutura similar. Assim o DOM consegue criar um mapeamento completo de uma página web.

A representação básica do DOM segue a estrutura da imagem abaixo. O objeto criado consiste nos seguintes pilares principais: **window**, **location**, **history**, **document** e **element**. O elemento raiz é o window, sendo que location, document e history são seus filhos diretos. Já o **element** é um componente que está ligado ao **document**.



(Disponível em: O que é DOM e por que essa interface é essencial para a web?)

O elemento raiz do DOM é o **window** que armazena informações sobre a página em uma **visão da janela** usada. Isto é, o window é a representação do DOM no navegador onde a página web é renderizada. Quando o navegador possui mais de uma aba aberta, cada uma das abas tem seu próprio DOM. Consequentemente, cada página tem seu próprio componente **window**.

Os filhos diretos de window são: document, location e history. O **document** é responsável por **representar o documento HTML** da página. Já o objetivo do **location** é gerenciar a **localização da página** (URL). Por fim, o **history** mantém o **histórico de navegação** do navegador.

O item **document** representa a página web carregada no navegador e que foi mapeada na estrutura de árvore do DOM. Seu principal objetivo é

fornecer uma **interface** entre a página web e o código Javascript, possibilitando alterações na página a partir de lógica de programação.

Document é filho de window e representa, então, a raiz de uma página HTML. Cada componente na estrutura HTML é transformado em um item **element** do DOM. A partir de então, a árvore DOM é montada como uma árvore de itens element conforme estrutura definida na página.

É possível notar que os elementos possuem características diferentes conforme o **componente** a que se referem. Um elemento pode representar uma tag `<h1>`, uma tag `<div>`, ou mesmo as tags `<body>`, `<header>` e `<html>` da página. Além disso, o elemento armazena informações como class, id, valores e demais atributos do componente.

DOM no Javascript

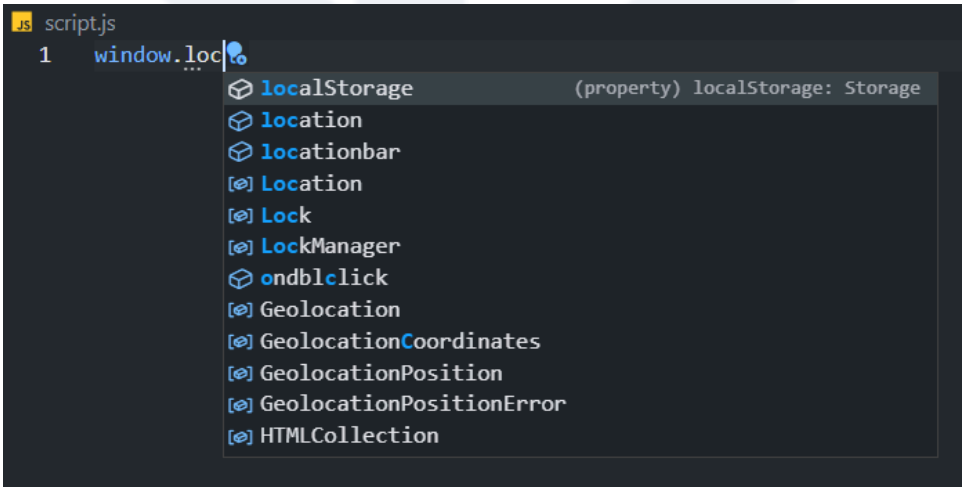
A interface do DOM pode ser acessada facilmente via Javascript, já que a linguagem proporciona variáveis globais que realizam acesso direto a todos os métodos e propriedades disponíveis. Considere o código abaixo:

```
// Acesso ao window
console.log(window)

// Acesso ao document
console.log(document)

// Acesso a propriedade location de window
console.log(window.location)
```

Neste código, o conteúdo dos objetos `window`, `document` e `location` são programados para serem exibidos no **console**[3] do navegador. O acesso a esses itens é direto, uma vez que o Javascript possui as variáveis definidas por padrão.

Perceba que o item location é acessado como um filho do objeto window, bem como a estrutura do DOM define. Porém, `document` também é filho de `window` e mesmo assim pode ser acessado por uma variável independente. Isto acontece pois `document` é uma das propriedades mais usadas no JS/DOM, e por isso, possui uma variável para acesso direto.

A integração do DOM com o Javascript também pode ser facilitada a depender da IDE usada no desenvolvimento. No Visual Studio Code, o mecanismo de sugestão inteligente de código (IntelliSense) mostra todas as propriedades e métodos disponíveis na interface DOM conforme os termos são digitados.



[3] O console de um navegador geralmente está dentro da seção "dev tools", onde existem ferramentas para programadores web. No Google Chrome, o atalho para abrir o console é o F12.

REFERÊNCIAS

1. Sites estáticos e sites dinâmicos: quais as diferenças?
2. 6 steps to become a Front-end Developer - Tolustar
3. Qual é a diferença entre website e aplicação web? - Stack Overflow em Português
4. Dynamic HTML - Wikipedia
5. O Que é DOM (Document Object Model) e Quais Suas Vantagens
6. Entendendo o DOM (Document Object Model) - Tableless - Website com artigos e textos sobre Padrões Web, Design, Back-end e Front-end tudo em um só lugar.
7. O que é DOM (Document Object Model)? Trabalhando com DOM em JavaScript
8. Introdução ao DOM - APIs da Web | MDN
9. JavaScript DOM Tutorial
10. O que é DOM do JavaScript? (Document Object Model) | by Matheus Budkewicz | horaDeCodar | Medium

BONS ESTUDOS

growdev