JS

▼ DOM

O DOM (Document Object Model) e o JavaScript, juntos, possuem grande poder de modificar dinamicamente a estrutura de um documento HTML. Sendo possível, por exemplo:

- Adicionar/modificar/remover tags, textos, imagens e qualquer elemento no HTML.
- Alterar estilos CSS da página.
- Criar novos eventos HTML.

Vamos conferir formas de realizar os itens listados no tópico acima.

Métodos para selecionar elementos no HTML

- `document.getElementByID(id)` - Selecionar um elemento pelo ID.
- `document.getElementsByTagName(name)` - Selecionar um elemento pelo nome.
- `document.getElementsByClassName(name)` - Selecionar um elemento pelo nome da classe.

Propriedades e métodos para alterar elementos no HTML

- `element.innerHTML` - Esta propriedade obtém ou altera qualquer elemento no HTML, inclusive tags.
- `element.innerText` - Esta propriedade permite inserir textos no HTML.
- `element.attribute` - Esta propriedade altera o valor de um elemento HTML
- `element.setAttribute(atributo, valor)` - Este método altera o valor de um atributo de um elemento HTML.

Adicionando e excluindo elementos

- `document.write()` - Escreve no fluxo de saída do HTML.
- `document.appendChild()` - Adiciona um elemento HTML.
- `document.removeChild()` - Remove um elemento HTML.
- `document.replaceChild()` - Substitui um elemento HTML.
- `document.createElement()` - Cria um elemento HTML.

Mais sobre data-attribute

`Data-attributes` são utilizados para guardar valores em elementos HTML. Esses valores podem ser manipulados através do JavaScript. Também é possível estilizar elementos HTML com CSS referenciando o seu `data-attribute`. Essa funcionalidade é bem recente no mundo do desenvolvimento, sendo lançada na última versão do HTML(HTML5).

`Data-attributes` não devem ser utilizados para dados visíveis, pois tecnologias de acessibilidade podem não identificar seus valores.

Sua estrutura é dividida em duas partes:

- A primeira parte é o identificador deste dado, que consiste em `data-` + o nome de sua escolha que melhor identifica o tipo de dado a ser atribuído.
- A segunda parte é o valor atribuído à este `data-attribute`, este valor deve sempre estar entre aspas “”, e dessa maneira é lido como uma `string`.

Segue exemplo abaixo:

HTML

```
  <h1>Lista de tintas:</h1>
  <ul id="lista">
    <li data-cor="laranja" data-tipo="tinta-exterior" onclick="mudaCores(this)"
class="item">Tinta laranja</li>
    <li data-cor="vermelho" data-tipo="tinta-interior"  onclick="mudaCores(thi
s)" class="item">Tinta vermelha</li>
    <li data-cor="branco" data-tipo="tinta-interior"  onclick="mudaCores(this)"
class="item">Tinta branca</li>
    <li data-cor="amarelo" data-tipo="tinta-exterior"  onclick="mudaCores(thi
s)" class="item">Tinta amarelo</li>
    <li data-cor="rosa" data-tipo="tinta-exterior"  onclick="mudaCores(this)" c
lass="item">Tinta rosa</li>
  </ul>COPIAR CÓDIGO
```

No JavaScript, podemos criar uma função que recebe esses atributos, permitindo manipulá-los:

JS

```
functionmudaCores(elementos){
var cores = elementos.getAttribute("data-cor");
var tipoElemento = elementos.getAttribute("data-tipo");
  }COPIAR CÓDIGO
```

O prefixo `data-` não é obrigatório para definir um `data-attribute` personalizado, podemos alterar o código para a seguinte forma:

HTML

```
  <h1>Lista de tintas:</h1>
  <ul id="lista">
    <li cor="laranja" tipo="tinta-exterior" onclick="mudaCores(this)" class="it
em">Tinta laranja</li>
    <li cor="vermelho" tipo="tinta-interior"  onclick="mudaCores(this)" class
="item">Tinta vermelha</li>
    <li cor="branco" tipo="tinta-interior"  onclick="mudaCores(this)" class="it
em">Tinta branca</li>
    <li cor="amarelo" tipo="tinta-exterior"  onclick="mudaCores(this)" class="i
tem">Tinta amarelo</li>
    <li cor="rosa" tipo="tinta-exterior"  onclick="mudaCores(this)" class="ite
m">Tinta rosa</li>
  </ul>COPIAR CÓDIGO
```

JS

```
functionmudaCores(elementos){
var cores = elementos.getAttribute("cor");
var tipoElemento = elementos.getAttribute("tipo");
  }
```

O que são objetos em JavaScript?

Na linguagem JavaScript, objetos podem ser definidos como uma coleção de dados. Assim como na vida real, um livro é um objeto. Todo livro possui um título, um autor, um gênero, uma editora, quantidade de páginas, entre outros atributos.

Para criar um objeto JavaScript, vamos utilizar as chaves `{ }`, como ilustra o código abaixo:

```
var livro = {}COPIAR CÓDIGO
```

Criamos um objeto livro, porém ele não possui nenhuma propriedade ou valor. Os objetos em JavaScript têm pares de valor de propriedade, assim como um livro físico por exemplo, `título`: `Senhor dos anéis`. A maioria dos livros tem as mesmas `propriedades`, mas os **valores** das propriedades diferem de um livro para outro. Veja o exemplo abaixo:

```
var livro = {
titulo: "Cangaceiro JavaScript",
autor: "Flávio Almeida",
genero: "Front-end",
editora: "Casa do código",
preco:31,92
}COPIAR CÓDIGO
```

As propriedades do objeto livro são: título, autor, gênero e editora. Já os valores de cada propriedade são indicados após o sinal de dois pontos (:).

Podemos inserir uma lista dentro de um objeto JavaScript, como mostra o exemplo abaixo:

```
var livros = [
    {titulo: "Cangaceiro JavaScript",
    autor: "Flávio Almeida",
    genero: "Front-end",
    editora: "Casa do código",
    preco:31.92
},
    {titulo: "Cangaceiro JavaScript",
    autor: "Flávio Almeida",
    genero: "Front-end",
    editora: "Casa do código",
    preco:21.89},

    {titulo: "Cangaceiro JavaScript",
    autor: "Flávio Almeida",
    genero: "Front-end",
    editora: "Casa do código",
    preco:31.92},
]COPIAR CÓDIGO
```

Veja que é possível inserir dados do tipo `string`, `number`, `array`. Agora vamos descobrir formas de trabalhar com objetos Javascript.

Acessando propriedades

É possível acessar propriedades de objetos utilizando ponto `.` ou array `[]`:

Acessando com ponto

Podemos acessar os valores atribuídos à `preco`, que se refere ao preço do objeto `livro` da seguinte forma:

```
livro.precoCOPIAR CÓDIGO
```

Nossa saída será:

```
31,92COPIAR CÓDIGO
```

> observe que neste exemplo usamos o objeto livro (no singular).

Acessando com array

Também é possível acessar utilizando colchetes `[]`.

```
var livros = [

    {titulo: "Cangaceiro JavaScript",
    autor: "Flávio Almeida",
    genero: "Front-end",
    editora: "Casa do código",
    preco: 31.92},

    {titulo: "Introdução e boas práticas em UX Design",
    autor: "Fabricio Teixeira",
    genero: "UX & UI",
    editora: "Casa do código",
    preco: 31.92},

    {titulo: "Scrum",
    autor: "Rafael Sabbagh",
    genero: "Métodos Ágeis",
    editora: "Casa do código",
    preco: 31.92},
]

console.log(livros[0]["autor"])
console.log(livros[0].autor)COPIAR CÓDIGO
```

> Não esqueça as aspas quando utilizar os colchetes.

As duas formas acima retornam os valores: `Flávio Almeida`.

Editando objetos

Vimos como podemos consultar valores em objetos e, agora, vamos aprender como podemos alterar esses valores, criar novos ou deletá-los.

Alterando valor de um objeto

Para alterar o valor da propriedade `genero` do livro “Introdução e boas práticas em UX Design”, por exemplo, é possível fazer da seguinte forma:

```
livros[1].genero = "UX e Design"COPIAR CÓDIGO
```

> Passamos o parâmetro que o livro se encontra.

Podemos conferir no console digitando `livros[1].genero`, e será retornado: `"UX e Design"`.

Inserindo novos valores em objetos

Podemos inserir novos valores em objetos utilizando ponto `.` ou array `[]`:

Agora vamos conferir como adicionar uma nova propriedade chamada `estoque`, junto do seu valor, no primeiro livro da nossa lista:

```
livros[0].estoque = "11 unidades"COPIAR CÓDIGO
```

É possível conferir no console as informações adicionadas digitando `console.log(livros[0])`. E o console irá retornar:

```
autor: "Flávio Almeida"
editora: "Casa do código"
estoque: "11 unidades"
genero: "Front-end"
titulo: "Cangaceiro JavaScript"
preco: 31.92COPIAR CÓDIGO
```

Deletando valores de um objeto

Podemos deletar valores de algum atributo, digitando `delete` + o índice do seu valor da seguinte forma:

```
delete livros[1].autorCOPIAR CÓDIGO
```

> Assim, foi deletado o valor da propriedade autor do livro de índice 1.

Também podemos deletar o livro do índice 1 e seus valores por completo:

```
delete livros[1]COPIAR CÓDIGO
```

E ao consultar digitando `livros[1]` será retornado: `undefined`.

Utilizando `this`

`this` e `window`

Em JavaScript, é possível utilizar o `this` para acessar o objeto `window`.

```
console.log(this === window); // trueCOPIAR CÓDIGO
```

> O window representa uma janela que contém o elemento DOM da página acessada. No objeto window é definida todas variáveis globais e funções que são executadas em um navegador.

Podemos conferir uma série de propriedades que vêm por padrão do objeto `window` escrevendo o seguinte código:

```
console.log(this)COPIAR CÓDIGO
```

Acessando objetos com `this`

Com `this` é possível também acessar um objeto e suas propriedades:

```
var livros = {
    titulo: "Cangaceiro JavaScript",
    autor: "Flávio Almeida",
    genero: "Front-end",
    editora: "Casa do código",
    preco: 31.92,
    chamaLivro:function() {
```

```
return this.autor;
    }
}
console.log('O autor do livro se chama ' + livros.chamaLivro())COPIAR CÓDIGO
```

> No exemplo acima, o this é uma representação do objeto livros. E será retornado no console:

```
O autordo livro se chama Flávio AlmeidaCOPIAR CÓDIGO
```

Alterando propriedades de um objeto com `this`

```
var livros = {
    titulo: "Cangaceiro JavaScript",
    autor: "Flávio Almeida",
    genero: "Front-end",
    editora: "Casa do código",
    preco: 31.92,
    alteraAutor:function() {
return this.autor = "Pedro Marins";
    },
}

console.log('É possível alterar o nome do autor para ' + livros.alteraAutor())C
OPIAR CÓDIGO
```

> No console será retornado “É possível alterar o nome do autor para Pedro Marins”.

Atenção ao uso do `this`

> Quando utilizado no escopo global, o this tem valor do objeto window.