



Tutorial: `getAttribute()` e `setAttribute()`

Agora a gente vai dominar dois métodos **essenciais** pra manipular atributos de elementos no DOM:

👉 `getAttribute()` e `setAttribute()`.

Este é um **tutorial completo**, do básico ao avançado, com **explicações, sintaxe, exemplos e exercícios práticos**.



1 O que são atributos no HTML?

Atributos são **informações extras** dentro das tags HTML — eles modificam o comportamento ou estilo do elemento.

Exemplo:

```

```

Aqui:

- `src`, `alt`, e `width` são **atributos** do elemento ``.
-



2 Manipulando atributos com JavaScript

O JavaScript permite **acessar, alterar, adicionar ou remover** atributos dinamicamente — sem precisar editar o HTML.

Pra isso usamos:

- `getAttribute()` → **lê** um atributo.
 - `setAttribute()` → **cria ou altera** um atributo.
-

3 Método `getAttribute()`

Função:

Obtém o **valor atual** de um atributo em um elemento HTML.

Sintaxe:

```
elemento.getAttribute(nomeDoAtributo);
```

Parâmetro:


Nome	Tipo	Descrição
nomeDoAtributo	string	O nome do atributo que você quer consultar.

Exemplo:

```

```

```
const imagem = document.querySelector("#foto");  
const endereco = imagem.getAttribute("src");  
console.log(endereco); // → "perfil.jpg"
```

 O `getAttribute()` é útil pra ler valores dinâmicos, como o link de uma imagem, o destino de um link ou o ID de um elemento.

4 Método `setAttribute()`

Função:

Adiciona ou altera um atributo em um elemento HTML.

Sintaxe:

```
elemento.setAttribute(nomeDoAtributo, valor);
```

Parâmetros:

Nome	Tipo	Descrição
nomeDoAtributo	string	Nome do atributo que será criado/modificado
valor	string	Novo valor a ser definido para o atributo

Exemplo:

```
const imagem = document.querySelector("#foto");

// muda a imagem
imagem.setAttribute("src", "nova-foto.jpg");

// adiciona um atributo novo
imagem.setAttribute("title", "Foto atualizada");
```

 Agora o HTML ficou assim:

```

```

5 Usando os dois juntos

Você pode ler e alterar dinamicamente um atributo com base no valor atual:

```
const link = document.querySelector("a");

const destinoAtual = link.getAttribute("href");
console.log("Antes:", destinoAtual);

link.setAttribute("href", "https://www.google.com");
console.log("Depois:", link.getAttribute("href"));
```

6 Manipulando classes, estilos e atributos booleanos

🧠 Alterando estilo via atributo

```
const caixa = document.querySelector(".caixa");  
caixa.setAttribute("style", "background: blue; color: white;");
```

✂️ Trabalhando com classes

```
const titulo = document.querySelector("h1");
```

```
// lê a classe atual  
console.log(titulo.getAttribute("class"));
```

```
// adiciona uma nova classe  
titulo.setAttribute("class", "destaque");
```

✅ Atributos booleanos (ex: disabled, checked)

```
const botao = document.querySelector("button");  
botao.setAttribute("disabled", "true"); // desativa o botão
```

```
// para reativar:  
botao.removeAttribute("disabled");
```

🧠 7 Diferença entre `setAttribute()` e acesso direto

🆚 Muita gente confunde isso:

```
imagem.src = "nova.jpg";           // forma direta  
imagem.setAttribute("src", "nova.jpg"); // forma genérica
```

As duas funcionam, mas:

- `setAttribute()` é **mais universal** (serve pra qualquer atributo);
- o acesso direto (`elemento.propriedade`) é **mais rápido** pra propriedades específicas do DOM.