🎖️

# Acessando Propriedades

> Quando precisamos realizar determinada tarefa, como acessar os valores de elementos por exemplo, temos que acessar as propriedades existentes no mesmo. Para isso, veremos os tópicos mais usados e comuns no desenvolvimento web.

---

### .value →

Dentro do HTML é comum termos inputs de texto e outros tipos também, e para fazer sentido a inserção desses dados pelo usuário precisamos pegar o que o usuário digitou para manipularmos de alguma forma.

Sua sintaxe é:

```
// Capturar valor:
var valor = elemento.value;

// Inserir valor:
elemento.value = 'novo valor';
```

O .value é responsável por capturar ou determinar o valor contido dentro de determinada tag, por exemplo:

```
<!-- HTML: -->

<input type="text" value="Clóvis">
```

```
// Javascript:

let valor = document.querySelector('input').value;
console.log(valor);
// Retorna: Clóvis
```

Entretanto, não faz muito sentido pegarmos um valor pré-definido de um input como o caso acima. Quando precisamos capturar valores temos que pegar os valores inseridos pelo usuário de sua aplicação, com uma função ou algo do tipo.

Abaixo temos um exemplo de função que pega o valor inserido no input e o coloca numa 'div', utilizando o evento 'input':

```
<!-- HTML: -->
<input type="text">
<div></div>
```

```
// Javascript:

document.querySelector('input').addEventListener('input', function () {
  const valor = document.querySelector('input').value;
  const div = document.querySelector('div');

  div.innerHTML = `<p>${valor}</p>`;
});
```

1 → Selecionamos o input diretamente na DOM e adicionamos o evento 'input' ao mesmo.

2 → Nesse evento chamamaos a função.

3→ A função é responsável por capturar o valor do input (perceba o '.value' no final da linha) e inseri-lo na tag div.

Da mesma forma que capturamos um valor, podemos inserir:

```
<!-- HTML: -->

<input type="text">
```

```
// Javascript:

document.querySelector('input').value = 'Clóvis';
```

Basta selecionar o elemento e usar o '.value' seguido do valor desejado.

---

## innerHTML →

Segundo a documentação MDN Web Docs, a propriedade Element.innerHTML define ou obtém a sintaxe HTML ou XML descrevendo os elementos descendentes.

Sua sintaxe é:

```
// Inserir um HTML:
var element = document.querySelector('body');
var content = element.innerHTML;

// Obter um HTML:
elemento.innerHTML;
```

Nesse caso, a variável 'content' contém todo o código HTML de um elemento qualquer que tenhamos selecionado.

Como exemplo de como obter a sintaxe HTML de um código temos:

```
<!-- HTML: -->

<div>
  <p>Esse é um parágrafo</p>
  <p>Esse também é um parágrafo</p>
</div>
```

```
// Javascript:

const div = document.querySelector('div');
console.log(div.innerHTML);

/* Retorna:

  <p>Esse é um parágrafo</p>
  <p>Esse também é um parágrafo</p>
*/
```

Agora um exemplo de como definir o HTML:

```
<!-- HTML: -->

<div></div>
```

```
// Javascript:

const div = document.querySelector('div');
div.innerHTML = '<p> Adicionei esse parágrafo </p>'
```

Selecionamos a div do HTML e adicionamos o HTML desejado. Perceba que para inserirmos qualquer conteúdo devemos escrever exatamente igual como tivéssemos num arquivo HTML, com tags completas e detalhes do tipo.

Um caso muito comum de uso é quando queremos limpar determinado elemento antes de inserir outro conteúdo. Por exemplo, numa lista de tarefas teremos que limpar o conteúdo a ser inserido antes da inserção para não duplicarmos o mesmo. Como:

```
elementoQualquer.innerHTML = '';
```

## innerText →

Segundo a documentação MDN Web Docs, innerText é uma propriedade que representa o conteúdo textual "renderizado" de um nó e seus descendentes.

Parecido com o innerHTML, ele também é responsável por obter e inserir algo nos elementos. Ou seja, ele irá selecionar o texto de um elemento e/ou inserir um texto.

Sua sintaxe é:

```
// Inserir:
elemento.innerText = 'texto';

// Obter:
elemento.innerText;
```

Um exemplo de inserção de texto em um elemento qualquer pode ser:

```
<!-- HTML: -->

<div>
  <p></p>
</div>
```

```
// Javascript:

const paragrafo = document.querySelector('p');
paragrafo.innerText = 'Inseri esse texto';
```

Para obter algum texto de determinado elemento também é muito simples:

```
<!-- HTML: -->

<div>
  <p>Vamos obter esse texto</p>
</div>
```

```
// Javascript:

let valor = document.querySelector('p').innerText;
console.log(valor);
```

Podemos também definir textos para algumas condições, por exemplo, definimos um texto para o usuário ficar em alerta caso passe do valor máximo definido:

```
<!-- HTML: -->

<input type="text" placeholder="Limite de caracteres: 10">
<p></p>
```

```
// Javascript:

let paragrafo = document.querySelector('p');

document.querySelector('input').addEventListener('input', function(){
  const txt = document.querySelector('input').value;
  if(txt.length > 10){
    paragrafo.innerText = 'Você passou do limite estipulado'
  } else {
    paragrafo.innerText = 'Você está dentro do limite'
  }
})
```

1 → Selecionamos o parágrafo;

2 → Selecionamos o input e inserimos um evento de 'input' nele;

3 → Ele será responsável por chamar a função definida;

4 → A função receberá o valor do input em si e depois verificará: se, comprimento do texto maior que 10 caracteres, insira 'Você passou do limite estipulado' no parágrafo. Caso não, insira 'Você está dentro do limite'.

---

## Como saberei quando usar cada um desses? →

Bom, basicamente, você deve utilizar `.value` quando tratar de campos de inserção de valores pelo usuário, enquanto `.innerHTML` e `.innerText` deve ser utilizado quando queremos acessar o conteúdo HTML ou textual dentro de um elemento HTML.

- .value: Acessar valores de elementos <input>, <select> entre outros campos de formulário;
- .innerHTML: Acessar elemento HTML descendente ao elemento atual;
- .innerText: Obter conteúdo textual dentro de tags <p>, , <pre> entre outras.

---

## O que é um atributo? →

Antes de ver algumas funcionalidades com atributos, precisamos saber o que são eles.

Os atributos são propriedades que um elemento possui, seja de estilo, formatação, conteúdo, ou quaisquer outras propriedades que um elemento possa ter. Por exemplo, o ID de uma tag é um atributo, e ao mesmo tempo que isso ocorre, a cor do texto que faz parte deste elemento também é um atributo.

---

## .setAttribute →

É responsável por conceder um atributo a um elemento qualquer. Se o atributo existir, o mesmo será atualizado.

Sua sintaxe é:

```
elemento.setAttribute('nomeDoAtributo', 'valorDoAtributo');
```

Como exemplo, iremos definir um atributo a um parágrafo qualquer. Esse atributo será uma classe, a mesma será estilizada pelo css:

```
<!-- HTML: -->

<div>
  <p>Esse texto será estilizado</p>
  <button type="button">Estilizar</button>
</div>
```

```
/* CSS: */
.paragrafos {
  color: red;
  font-size: 2em;
  text-align: center;
}
```

```
// Javascript:

document.querySelector('button').addEventListener('click', function() {
  const paragrafo = document.querySelector('p');

  paragrafo.setAttribute('class', 'paragrafos')
})
```

1 → Selecionamos o botão e adicionamos o evento de 'click';

2 → A função é chamada no clique do botão;

3 → Ela é responsável por conceder a classe 'paragrafos' ao elemento 'p';

Outro exemplo que podemos ter é adicionar uma imagem no clique de um botão a partir da url inserida pelo usuário:

```
<!-- HTML: -->

<div>
  <input type="text" placeholder="Insira a url da imagem">
  <button type="button">Mostrar imagem</button>
</div>
```

```
/* CSS: */

.imagem {
  width: 400px;
  border-radius: 16px;
  margin: 10vh;
}
```

```
// Javascript:

document.querySelector('button').addEventListener('click', function () {
  const url = document.querySelector('input').value;
  const img = document.createElement('img');

  img.setAttribute('src', url);
  img.setAttribute('class', 'imagem');

  document.querySelector('div').appendChild(img);
})
```

1 → Selecionamos o botão e adicionamos o evento de clique para chamar a função;

2 → A função é responsável por buscar o valor inserido no input, criar uma imagem e conceder a url ao atributo 'src'. Por último, adiciona a imagem gerada à 'div'.

---

## .getAttribute →

Retorna o atributo de um elemento em específico. Se o atributo não existir, é retornado "" ou null.

Sua sintaxe é:

```
let atributo = elemento.getAttribute('nomeDoAtributo');
```

Um exemplo de buscar e guardar um atributo é:

```
<div>
  <p id="paragrafo">Parágrafo qualquer</p>
</div>
```

```
const paragrafo = document.querySelector('p');
let id = paragrafo.getAttribute('id');

console.log(id);

// Retorna: idDoParagrafo
```

Nesse caso selecionamos o parágrafo e depois buscamos seu id e inserimos na variável chamada 'id'.

Possuímos possibilidades mais simples e rápidas, porém, um caso possível é pegarmos o atributo e colocar em algum outro elemento, por exemplo:

```
<!-- HTML: -->

<div>
  <p class="paragrafos">Parágrafo qualquer</p>
  <p>Outro parágrafo</p>
</div>
```

```
/* CSS: */

.paragrafos {
  color: red;
  font-size: 2em;
  text-align: center;
}
```

```
// Javascript:

document.querySelector('button').addEventListener('click', function() {
  const paragrafo = document.querySelector('#paragrafo1');
  let classe = paragrafo.getAttribute('class');

  let paragrafo2 = document.querySelector('#paragrafo2');

  paragrafo2.setAttribute('class', classe)
})
```

No exemplo acima guardamos a classe do parágrafo 1 numa variável e inserimos a mesmo no parágrafo 2.

---

## .removeAttribute →

É responsável por remover um atributo de um elemento específico.

Sua sintaxe é:

```
elemento.removeAttribute('nomeDoAtributo');
```

Podemos usar como exemplo a remoção do estilo de algum elemento:

```
<div>
  <p style="color: red; font-size: 2em; text-align: center;">
    Parágrafo qualquer
  </p>
  <button type="button">
    Remover estilo
  </button>
</div>
```

```
document.querySelector('button').addEventListener('click', function() {
  const paragrafo = document.querySelector('p');
  paragrafo.removeAttribute('style');
})
```

Nesse exemplo removemos o atributo 'style' do parágrafo.

Um caso que podemos testar essas fucionalidades é remover o atributo de um elemento e adicioná-lo num outro:

```
<div>
  <p class="paragrafos" id="paragrafo1">Parágrafo qualquer</p>
  <p id="paragrafo2">Outro parágrafo</p>
  <button type="button">Mudar</button>
</div>
```

```
.paragrafos {
  color: red;
  font-size: 2em;
  text-align: center;
}
```

```
document.querySelector('button').addEventListener('click', function () {
  const paragrafo1 = document.getElementById('paragrafo1');
  const paragrafo2 = document.getElementById('paragrafo2');

  const classe = paragrafo1.getAttribute('class');
```

```
  paragrafo1.removeAttribute('class');
  paragrafo2.setAttribute('class', classe);
})
```

1 → Adicionamos o evento de clique no botão para chamar uma função;

2 → A função é responsável por buscar os parágrafos e o atributo 'class', remover o mesmo do parágrafo 1 e adicioná-lo ao parágrafo 2.

---

## Adicionando estilo através do JavaScript? 🤔

Através do JavaScript podemos selecionar elementos e fazer manipulações, isso já sabemos, mas entre essas manipulações também é possível alterar o estilo de um elemento, sou seja, definir propriedades e valores CSS através do JavaScript, e isso é muito fácil, veja um exemplo:

```
const paragrafo = document.querySelector('p');

p.style.color = 'red';
```

Esse acima é um exemplo dos mais simples, mas você pode definir as propriedades e valores que quiser!

---


Bibliografias:

- https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML;
- https://developer.mozilla.org/pt-BR/docs/Web/API/HTMLElement/innerText;
- https://developer.mozilla.org/en-US/docs/Web/API/Element/removeAttribute;
- https://developer.mozilla.org/en-US/docs/Web/API/Element/getAttribute;
- https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/style;