Navegação interna de páginas

Utilizando a tag `<a>`, podemos criar links para páginas internas.

Por exemplo, criando outro arquivo html, e referenciando ele no atributo `href`.

```
<a href="pagina2.html">Ir para a página 2</a>
```

Ao clicar no link, o navegador irá carregar a página `pagina2.html` e todo seu conteúdo.

JavaScript

O JavaScript é como se fosse o cérebro de uma página web, ele é responsável por adicionar interatividade e dinamismo a uma página.

É a linguagem de programação mais utilizada na web, sendo suportada por todos os navegadores modernos.

Através do JavaScript é possível manipular o conteúdo da página, adicionar ou remover elementos, alterar estilos, criar animações, etc.

Para adicionar JavaScript a uma página, basta adicionar a tag `<script>` no cabeçalho ou no corpo da página.

Normalmente é utilizado no final do corpo da página, para que o JavaScript seja carregado após o conteúdo da página, pois carregar o JavaScript no cabeçalho pode bloquear o carregamento do site e impactar o desempenho.

É possível adicionar o código diretamente na tag `<script>`, ou referenciar um arquivo externo. Assim como o CSS.

MDN Script element

Utilizando com a tag script:

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript</title>
    <script>
      alert('Olá, mundo!')
    </script>
  </head>
  <body>
    <h1>JavaScript</h1>
  </body>
</html>
```

Referencia a um arquivo externo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript</title>
    <script src="script.js"></script>
  </head>
  <body>
    <h1>JavaScript</h1>
  </body>
</html>
```

Carregando scripts externamente é uma boa prática, pois podemos reutilizar código, separar a lógica do design, facilita a manutenção e também podem ser cacheados pelo navegador. Um script externo também pode referenciar urls, como, por exemplo, a url de uma biblioteca, ex: jQuery.

Com o JavaScript conseguimos manipular o DOM, que é a estrutura de uma página web, e também podemos manipular estilos, eventos, etc.

Através do JavaScript temos acesso aos atributos globais do navegador, como o `window`, `document`, `navigator`, `screen`, `location`, `history`, `localStorage`, `sessionStorage`, etc.

Para acessar um elemento do DOM, podemos utilizar o método `document.querySelector()`, que retorna o primeiro elemento que corresponde ao seletor especificado.

```
// Seleciona o primeiro elemento <h1> da página
const h1 = document.querySelector('h1')

// Altera a cor do texto do h1 para vermelho
h1.style.color = 'red'
```

Também conseguimos acessar elementos através de classes e ids, utilizando `document.querySelector('.classe')` e `document.querySelector('#id')`.

Através da api `document` podemos criar elementos, adicionar ou remover elementos, alterar estilos, etc. Instance methods

```
// Cria um novo elemento <p>
const p = document.createElement('p')

// Adiciona um texto ao elemento <p>
p.textContent = 'Texto do parágrafo'

// Adiciona o elemento <p> ao final do body
document.body.appendChild(p)

// Remove o elemento <h1> da página
document.querySelector('h1').remove()

// Adiciona o p a uma div
document.querySelector('div').appendChild(p)

// Adiciona uma classe ao elemento <p>
p.classList.add('paragrafo')

// Adiciona um estilo ao elemento <p>
p.style.color = 'blue'
```

Com o dom também é possível adicionar eventos a elementos, por exemplo, um evento de clique, um evento de mouse sobre o elemento, um evento de teclado, etc.

Usando a DOM

```
// Adiciona um evento de clique ao elemento <p>
p.addEventListener('click', () => {
  alert('Clicou no parágrafo')
})

// Adiciona um evento ao passar o mouse sobre o elemento <p>
p.addEventListener('mouseover', () => {
  alert('Passou o mouse sobre o parágrafo')
})

// Adiciona um evento ao pressionar uma tecla
document.addEventListener('keydown', (event) => {
  console.log(event.key)
})
```

Para criação de lógica ou eventos que envolvam a manipulação do DOM, é importante que o JavaScript seja carregado após o conteúdo da página, para garantir que os elementos já estejam disponíveis.

Para isso, é comum adicionar o script no final do corpo da página, ou utilizar o evento `DOMContentLoaded`.

```
document.addEventListener('DOMContentLoaded', () => {
  // Código JavaScript
})
```

Também podemos acrescentar eventos js diretamente no html, utilizando atributos de funções que serão chamadas quando o evento ocorrer, que referenciam funções js ou que são códigos js.

```
function myFunction() {
  alert('Clicou no botão')
}
```

```
<button onclick="myFunction()">Clique aqui</button>

<button onclick="this.innerHTML = Date().toLocaleDateString()">
  Qual a data atual?
</button>
```

Tutorial JavaScript W3Schools

APIS:

Window, Document, Navigator, Location, History, Local storage, Session storage