Declarações em JavaScript

Prof. Romerito Campos

Eventos e Manipulação de elementos

Conteúdo

- Eventos
- Manipulação de elementos

Modelo de funcionamento

- Modelo de programação assíncrono orientado a eventos.
- O browser gera eventos quando alguma intenção acontece.

Definições básicas

- **Event Type:** Tipo de evento, como mover o mouse ou clicar.
- **Event Target:** Um botão pode ser um alvo (`target`) e um evento associado a ele.
- **Event Listener, Event Handler:** É uma maneira de responder aos eventos. O navegador invoca o handler quando ocorre um evento.
- **Event Object:** Mantém detalhes sobre o evento e é passado como argumento. É possível saber o `target` .

Definições básicas

- **Event Propagation:** Quando um evento é invocado no documento ou em algum elemento dele, há a propagação. Exemplo: em vez de adicionar um evento para cada input de um formulário, adiciona-se um `handler` para o formulário, já que um evento disparado no input se propaga para o formulário.

Categoria de Eventos

- **Eventos de Entrada Dependentes de Dispositivos:**
  - `mousedown`, `mousemove`, `touchstart`, `keydown`, `keyup`.
- **Eventos de Entrada Independente de Dispositivos:**
  - `click` pode ser via mouse ou teclado.
- **Eventos de Interface de Usuários:**
  - `"focus"`, `"change"`, `"submit"`.

Categoria de Eventos

- **Eventos de Mudança de Estado:**
  - Atividades da rede ou browser, navegação no histórico.
- **Eventos Específicos de API:**
  - `<video>`, `<audio>`: `"waiting"`, `"playing"`.

Registrar Manipuladores de Eventos

- Há duas maneiras:
  - Definir uma propriedade no `target`.
  - Passar o `handler` para o método `addEventListener`. Exemplos:

```
form.onSubmit = function (event) {}
form.addEventListener("submit", function() { /* ... */ })
```

- Com **`addEventListener`**, é possível adicionar múltiplos handlers para o mesmo evento.
- Também é possível remover com **`removeEventListener`**.

Scripting Documents

Scripting Documents

- Torna o HTML Dinâmico
- Cada Window tem um objeto document, que contém todo o conteúdo da página.
- `document` é um objeto de `Document`.
- É possível usar os seletores do CSS para obter um elemento HTML presente no DOM.

```
document.querySelector("#button");
document.querySelectorAll("h1");
```

Scripting Documents

- Há outros métodos para obter um elemento:
  - `getElementById("id")`
  - `getElementsByName("name")`: observa o atributo name
  - `getElementsByTagName("tag")`
  - `getElementsByClassName("class")`
  - Podem ser aplicados ao `document` ou a um elemento que tenha filhos.

Elementos pré-selecionados

- Alguns elementos podem ser selecionados diretamente: `document.forms.login`
- Semelhante a `document.getElementByName("login")`

Estrutura do Documento e Travessia

- Há uma API de travessia que permite navegar no DOM.
- Leva em conta pai, filho, contagem de elementos filhos, primeiro filho, último filho, próximo irmão, irmão anterior.

Atributos

- Os elementos HTML possuem atributos, que são pares de nome = valor.
- É possível acessar e manipular os atributos:
  - `getAttribute()`, `setAttribute()`, `hasAttribute()`, `removeAttribute()`
- Os atributos podem ser acessados como propriedades:

```
let image = document.querySelector("#image");
let src = image.src;
```

- Atributos não são case-sensitive, mas as propriedades JavaScript são.

Atributo `class`

- Há uma propriedade `classList` que permite manipular a lista de classes do elemento.

```
button.classList.add();
button.classList.remove();
```

Atributo `dataset`

- Qualquer atributo HTML prefixado com `data-` é válido.
- Não altera a apresentação do elemento.
- Maneira adicional de incluir dados.

```
<div id="title" data-phone="XXX">✗ 4h2></div>

document.querySelector("#title").dataset.phone;
```

Conteúdo do Elemento

- A propriedade `innerHTML` permite redefinir o conteúdo de um elemento, incluindo marcação.

```
document.body.innerHTML = "<h1>ah 17 oi</h1>";
```

- O novo conteúdo do corpo será apenas o `<h1>`.
- É possível acessar o conteúdo de um elemento como texto.

```
document.body.textContent; // texto concatenado
document.body.innerText; // texto com quebra
```

- `innerText` é menos usual e não compatível com muitos browsers

Criar, Inserir e Remover Nós

- É possível criar elementos usando `document.createElement('P')`.
- As funções `append` e `prepend` adicionam elementos HTML no final e no início da lista de filhos.
- Há também as funções `after` e `before` que podem ser usadas tanto para texto quanto para HTML.

Referências

FLANAGAN, D. **JavaScript - The Definitive Guide**. 7. ed. Sebastopol, CA, USA: O'Reilly Media, 2020.