Construção de Software para Web

Java Script

Prof. Guilherme Zucatelli
2º Período - 2022

Programação Web

Relembrando…

- **HTML (*Hypertext Markup Language*):** Linguagem principal, responsável pela estruturação dos elementos existentes nas páginas Web.
- **CSS (*Cascading Style Sheets*):** Ferramenta de design, estilo, adotada para descrever/modificar a apresentação dos elementos da página.
- **JS (*JavaScript*):** Linguagem de script para páginas Web.

Introdução ao JS

JS (*Java Script*):

O JS é uma das linguagens de programação mais adotadas para o desenvolvimento de aplicações Web.

Trata-se de uma linguagem completamente inserida em todos os navegadores, sem necessidade de APIs e pode ser adotada tanto por aplicações *front-end* quanto aplicações *backend* (por meio do node.js).

Existem três formas principais de se programar em/interagir com JS:

- Por meio do console do próprio navegador.
- Utilizando a TAG em html **<script></script>**
- Adotando um arquivo dedicado para definição e inserção de scripts (scripts.js)

Introdução ao JS

Mensagens em Console:

Duas maneiras muito comuns de interação entre os elementos JS e a página Web ocorre por meio do **console do navegador e elementos de alerta**.

Como exemplo de funções para essas mensagens tem-se:

- console.log("mensagem")
- alert("mensagem")

Introdução ao JS

Variáveis:

Variáveis são formas de armazenar e manipular informações pelos programas de computador e podem assumir valores **locais e globais**.

Em JS existem três tipos de variáveis: primárias, compostas e objetos.

- **Variáveis Primárias:** São variáveis que armazenam informações numéricas (number), textuais (string), lógicas (boolean), nulas (null) ou não definidas (undefined).
  (**Ex.:** var nome="Arnaldo", idade=25, altura=1.75, casado=true;)

Introdução ao JS

Variáveis:

Variáveis são formas de armazenar e manipular informações pelos programas de computador e podem assumir valores **locais e globais**.

Em JS existem três tipos de variáveis: primárias, compostas e objetos.

- **Variáveis Primárias:** São variáveis que armazenam informações numéricas (number), textuais (string), lógicas (boolean), nulas (null) ou não definidas (undefined).
  (**Ex.:** var nome="Arnaldo", idade=25, altura=1.75, casado=true;)

- **Variáveis Compostas:** são definidas por vetores (*arrays*) e armazenam diversos elementos.
  (**Ex.:** var atributos = ["Bernaldo", 33, 1.81, false];)

  Atributos/Métodos: .push(value), .pop(), .shift(), .unshift(value), .indexof(value), .length

Introdução ao JS

Variáveis:

Variáveis são formas de armazenar e manipular informações pelos programas de computador e podem assumir valores **locais e globais**.

Em JS existem três tipos de variáveis: primárias, compostas e objetos.

- **Variáveis Primárias:** São variáveis que armazenam informações numéricas (number), textuais (string), lógicas (boolean), nulas (null) ou não definidas (undefined).
  (**Ex.:** var nome="Arnaldo", idade=25, altura=1.75, casado=true;)

- **Variáveis Compostas:** são definidas por vetores (*arrays*) e armazenam diversos elementos.
  (**Ex.:** var atributos = ["Bernaldo", 33, 1.81, false];)

  Atributos/Métodos: .push(value), .pop(), .shift(), .unshift(value), .indexof(value), .length

- **Objetos:** são elementos que armazenam atributos (variáveis) e métodos (funções)
  (**Ex.:** var pessoa = new Object(); var pessoa = {}; pessoa.nome="Cernaldo";
  var pessoa = {nome: "Cernaldo",
  idade: 18,
  altura: 1.65,
  casado: false})

Introdução ao JS

Operadores:

Operadores são funções que podem ser executadas para relacionar um ou mais elementos e variáveis, eles podem ser aritméticos ou comparativos/lógicos.

- Operadores Aritméticos: **+, -, *, /, %, +=, -=, *=, /=, ++, --, +string, -value**
- Operadores Comparativos e Lógicos: **==, !=, >, <, >=, ===, && (*and*), || (*or*), !**

Atividade

Introdução ao JS

Controle de Fluxo Condicional:

Elementos que executam certos blocos de código de acordo com a validade de alguma condição lógica.

- **Se/então (if/else)**:

```
if (condição lógica){
    … // Executa código se true
    } else {
        … // Executa código se false
        }

if (condição lógica){
    … // Executa código se true
    } else if (segunda condição lógica){
        … // Executa código se false e true
        } else {
            … // Executa código se false e false
            }
```

Introdução ao JS

Controle de Fluxo Condicional:

Elementos que executam certos blocos de código de acordo com a validade de alguma condição lógica.

- **Se/então (if/else)**:

```
if (numero % 2 == 0){
        console.log("Número é par")
        } else {
                console.log("Número é ímpar")
                }

if (numero < 10){
        console.log("Número menor que 10")
        } else if (numero < 100){
                console.log("Número menor que 100")
                } else {
                        console.log("Número é maior que 100")
                        }
```

Atividade

Introdução ao JS

Estruturas de Repetição (*Loops*):

Em JS existem duas estruturas de repetição essenciais: ***for*** e ***while***. Essa estruturas servem para repetir algum tipo de operação ou função, de acordo com alguma condição de parada.

- Loop **FOR**:

```
for(var x=1; x<10; x++){
    … // Executa código
    }
```

- Loop **WHILE**:

```
while(x<10){
    … // Executa código
    }
```

Introdução ao JS

Funções:

Funções são maneiras de condensar blocos de códigos em estruturas que podem ser chamadas em diferentes momentos da execução.

```
function printFunction( ){
                    … // Executa código
                    }

function addNumbers(aux1, axu2){
                    return aux1 + aux2;
                    }
```

As funções podem ser invocadas de acordo com a interação dos usuários, por exemplo pela adoção de botões **<buttom></buttom>** e atributo **onclick="nome_funcao()"**

Introdução ao JS

Atividades de Fixação:

1. Crie os diferentes tipos de variáveis apresentados em sala e utilize a função console.log() para visualizá-las no navegador.

2. Faça operações aritméticas com variáveis numéricas.

3. Crie uma função que execute a soma de dois valores.

4. Desenvolva uma função em JS que calcule o Índice de Massa Corpórea (IMC) de uma pessoa.

5. Crie um vetor (*array*) com os quatro nomes das Tartarugas Ninjas e faça um loop para imprimi-los no console do navegador.

6. Desenvolva uma função que calcule a potência de um número.

7. Desenvolva um objeto base que armazene as informações de alunos da UVV e mostre as informações no log do console.

Introdução ao JS

Atividades de Fixação:

8. Crie uma função que verifica a existência de um elemento em um vetor de strings retornando "Esse elemento existe aqui!" ou "Elemento não encontrado".

9. Crie um array com os nomes Arnoldo, Bernoldo e Cernoldo. Troque a posição do primeiro e do último elemento e imprima as informações de antes e depois no console do navegador.

10. Faça um loop while para calcular o fatorial (N!) de um número.

11. Desenvolva uma função que calcule o n-ésimo termo da sequência de Fibonacci.

Iteratividade em JS

Seletores:

Uma das principais formas de interagir com os elementos de uma página Web é por meio de funções seletoras (métodos seletores).

Seletores são muito utilizados com o objeto ***document*** e ***window,*** que representam os elementos HTML da página Web e a própria janela aberta, respectivamente.

Essa funcionalidade é comumente adotada para **instanciar variáveis e objetos** que apontam para elementos delimitados em nossa aplicação.

A seguir, alguns exemplos de métodos para essa finalidade:

Métodos	Descrição
.getElementByTagName()	Seleciona elementos que correspondem a uma TAG HTML específica.
.getElementByClassName()	Reúne os elementos que possuem uma determinada classe.
.getElementById()	Seleciona elementos com base em seus Ids.

Iteratividade em JS

Seletores:

A interatividade do JS pode agora ser combinada com os elementos de estilo (CSS) que definimos anteriormente.

Ao ter acesso ao DOM pelo objeto ***document*** e utilizando-se de seus métodos podemos alterar o comportamento de estilo de todos os elementos HMTL.

Observe o exemplo a seguir:

```
lista_div = document.getElementsByTagName("div")
```

A variável **lista_div** agora é uma lista que compreende todos os elementos ** ** presentes em um HTML. Dessa maneira, ao acessar um índice **lista_div[0],** temos também acesso as regras de estilo disponíveis a esse elemento fazendo **lista_div[0].style .**

Iteratividade em JS

Atividade Seletores:

Crie uma página WEB sobre seus hobbies preferidos, que contenha pelo menos os seguintes elementos estruturais e suas quantidades:

- **1x <h1>**
- **2x <h2>**
- **3x <p>**
- **1x <table>**
- **1x **
- **3x <imgs>**
- **2x e 1x **

A partir dos elementos criados, utilize JS para **criar funções** que adotem o método **.getElementByTagName()** e:

1. Altere a cor do primeiro parágrafo para cinza escuro.
2. Mude o tamanho, bordas e estilo das três imagens de forma que fiquem completamente diferentes umas das outras.
3. Altere o “display” dos itens da lista para que apareçam uns ao lado dos outros e também que desapareçam.
4. Altere a transparência dos elementos , de maneira que um fique mais próximo de completamente opaco e o outro de completamente transparente.
5. Desapareça e reapareça com o elemento .
6. Refaça os exercícios anteriores utilizando outros métodos do **objeto document**.

Iteratividade em JS

Eventos:

Uma das principais formas de adicionar iteração às páginas Web é por meio da **observação de eventos** (*event listeners*).

Eventos são atividades desempenhadas pelo **usuário** (atividades mouse/teclado) ou pela própria **página** (início/fim de alguma ação).

Alguns exemplos de eventos: **"click"**, **"mouseenter"**, **"keyup"**, **"scroll"**...

Eventos são adicionados/removidos com a utilização de métodos:

Métodos	Descrição
.addEventListener(**"nome_evento"**, function(event){...})	Adiciona um *event listener* no objeto.
.removeEventListener(**"nome_evento"**, nome_function)	Remove um *event listener* do objeto.

Iteratividade em JS

Exemplos Práticos - Adicionando *Event Listeners*:

Na página Nimbus2000, adicione os seguintes *listeners* e veja o comportamento no console.

1.
```
var tit = document.getElementById("chapterNumber")
tit.addEventListener("mouseover", function(event){
    event.preventDefault();
    console.log("Passou!");
    })
```

Iteratividade em JS

Exemplos Práticos - Adicionando *Event Listeners*:

Na página Nimbus2000, adicione os seguintes *listeners* e veja o comportamento no console.

1.
```
var tit = document.getElementById("chapterNumber")
tit.addEventListener("mouseover", function(event){
        event.preventDefault();
        console.log("Passou!");
        })
```

2.
```
document.addEventListener("keyup", function(){
        console.log("teste")
        })
```

Iteratividade em JS

Exemplos Práticos - Monitorando posição da Tela e Aparecimento Botão:

```
var body = document.body;
var docElem = document.documentElement;
var limite=100;

var tamPagina = Math.max(body.scrollHeight, body.offsetHeight, docElem.clientHeight, docElem.scrollHeight,
docElem.offsetHeight)

var botao = document.getElementById("back-to-top");

if(tamPagina != undefined){
  limite = tamPagina/4;
}
```

Iteratividade em JS

Exemplos Práticos - Monitorando posição da Tela e Aparecimento Botão:

```
var body = document.body;
var docElem = document.documentElement;
var limite=100;

var tamPagina = Math.max(body.scrollHeight, body.offsetHeight, docElem.clientHeight, docElem.scrollHeight,
docElem.offsetHeight)

var botao = document.getElementById("back-to-top");

if(tamPagina != undefined){
  limite = tamPagina/4;
}

window.addEventListener("scroll", function(event){
  if(docElem.scrollTop > limite){
    botao.style.opacity = 1;
  }else{
    botao.style.opacity = 0;
  }})
```

Iteratividade em JS

Atividade Prática - Botão "Voltar ao Topo":

Na página Nimbus2000, desenvolva uma funcionalidade de um botão que retorna ao topo da página a partir de um quarto de rolagem.

- Utilize os objetos **body** (document.body) e **docElem (**document.documentElement) para generalizar os atributos de altura da página independente do navegador adotado (body.scrollHeight, body.offsetHeight, docElem.clientHeight, docElem.scrollHeight, docElem.offsetHeight)

- Adicione um *event listener* de scroll na janela da página. O botão só deve ser visível a partir de um quarto da página (body.scrollTop || docElem.scrollTop > offset).

- Adicione um *event listener* de clique ao botão. Ao ser acionado, o botão deve zerar os valores de scrollTop e retornar ao início da página, além disso previna seu comportamento defaut.

Iteratividade em JS

Atividade Prática - Capas Iterativas:

Na página Nimbus2000, desenvolva a funcionalidade de seleção e movimentação das capas dos livros de acordo com a vontade do usuário. Inicie na capa 1.

- Crie uma função clearSelection() que "limpa" a edição de estilo dos elementos de **Id** "bt"+"ii" e "book"+ii.
- Desenvolva uma função setCover(idx), que chame clearSelection() e defina a capa do livro escolhido de acordo com a variável idx. Essa função deve ser chamada na primeira execução do script como setCover(1);
- Adicione um *event listener* de clique no objeto *document*. Adote **event.target.id** para identificar o ID do elemento clicado com a função **indexOf("bt") ou indexOf("move-")**.
- Selecione o identificador com a divisão de strings **split()** e altere as configurações de estilo de acordo com a seleção dos botões.

Prof. Guilherme Zucatelli

PhD on Defense Engineering

e-mail: guilherme.zucatelli@uvv.br