2

JavaScript

JavaScript

- Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web.
- JavaScript permite páginas da Web interativas e, portanto, é uma parte essencial dos aplicativos da web.
- Os principais navegadores têm um mecanismo JavaScript dedicado para executá-lo.
- Os website e aplicações web usam o JavaScript.
- É uma das linguagens de programação mais populares do mundo.

Porque estudar JavaScript?

- JavaScript é uma das três linguagens que todos os desenvolvedores Web precisam aprender:
  - HTML para definir o conteúdo de páginas da web
  - CSS para especificar o layout das páginas da web
  - JavaScript para programar o comportamento de páginas da web
- A **informação** fica com o **HTML**, a **formatação**, é responsabilidade do **CSS** e o comportamento, fica sob a **responsabilidade** do **JavaScript**.

Histórico

- A Netscape Communications percebeu que a Web precisava se tornar mais dinâmica.
- "O HTML precisava de uma *'linguagem de cola'* que fosse fácil de usar para montar componentes como imagens e plugins, onde o código poderia ser escrito diretamente na Web."
- A linguagem foi oficialmente chamada de LiveScript quando foi lançada em versões beta do Netscape Navigator 2.0 em setembro de 1995, mas foi renomeada para **JavaScript**

Histórico

- JavaScript tem uma similaridade sintática com o Java, mas do que com o Java tem com o C. Mas não é uma sub-linguagem do Java, assim como o Java não é uma sub-linguagem do C.
- A escolha final do nome causou confusão, dando a impressão de que a linguagem era uma derivação da linguagem de programação Java, e a escolha foi caracterizada como uma manobra de marketing da Netscape para dar ao JavaScript o status da linguagem da moda, o **Java**.

Histórico

- A Netscape decidiu criar uma linguagem de script que complementaria o Java e que deveria ter uma sintaxe semelhante.
- É atualmente a principal linguagem para programação client-side em navegadores web.
- Também tem sido utilizada do lado do servidor através do node.js.
- Como uma linguagem multi-paradigma, o JavaScript suporta estilos de programação orientados a eventos, funcionais e imperativos (incluindo orientado a objetos e prototype-based).

Histórico

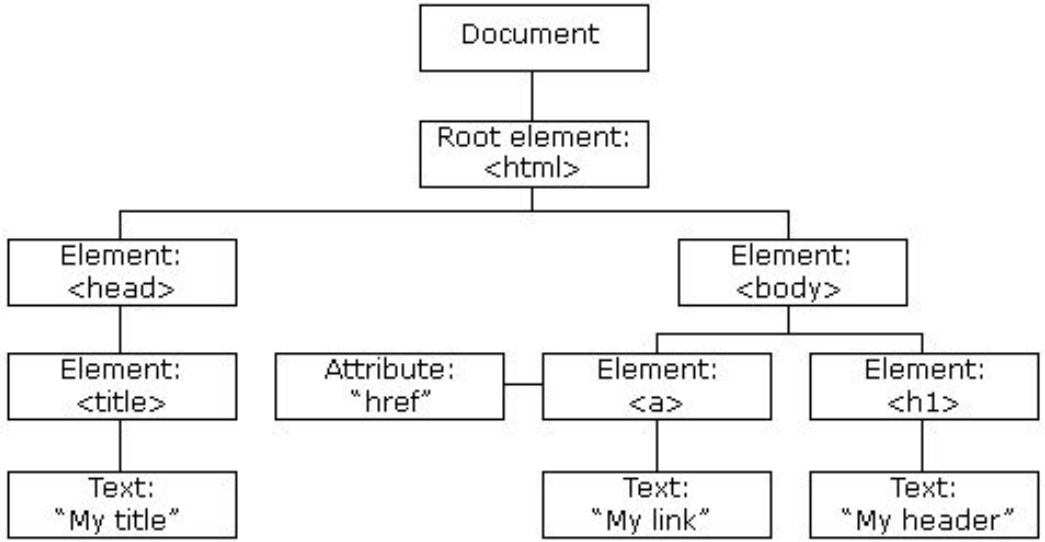
- As primeiras versões do JavaScript eram muito fracas e careciam de muitas funcionalidades. Faltavam manipuladores de exceção, funções privadas, e herança, etc.
- o JavaScript não é mantido pelo W3C, ele é uma linguagem criada e mantida pela pela ECMA (European Computer Manufacturers) e aprovada pela ISO.
- Em novembro de 1996, a Netscape submeteu o JavaScript à ECMA International para criar uma especificação padrão, que outros fornecedores de navegador poderiam implementar com base no trabalho feito na Netscape.

JavaScript

O JS permite adicionar recursos e lógica de programação em geral nas páginas HTML:

- Validação de entrada de dados
- Manipulação de strings
- Desenvolvimento de algoritmos
- Atualização dos elementos da página, baseada em eventos: clique, mouse, tempo, ...

DOM (Document Object Model)



DOM (Document Object Model)

- Com o DOM, o JavaScript pode criar HTML dinâmico:
  - JS pode mudar todos os elementos HTML na página
  - JS pode alterar todos os atributos HTML na página
  - JS pode mudar todos os estilos CSS na página
  - JS pode remover elementos e atributos HTML existentes
  - JS pode adicionar novos elementos e atributos HTML
  - JS pode reagir a todos os eventos HTML existentes na página
  - JS pode criar novos eventos HTML na página

JavaScript

- É uma linguagem interpretada baseada em objetos e sem declaração de tipos;

- O código JavaScript pode ser definido no <head> da página

```
<head>
      <script>
            function helloWorld() {
              alert('Olá mundo')
            }
      </script>
</head>
```

JavaScript - exemplo

```
<html>
      <head>
            <script type=”text/javascript”>
                  function helloWorld() {
                    alert(‘Olá mundo’)
                  }
            </script>
      </head>
      <body>
            <button onclick="helloWorld()">Exibir mensagem.</button>
      </body>
</html>
```

JavaScript

- Código JS pode ser definido no <body> do HTML

```
<body>
      <script>
            function helloWorld() {
              alert('Olá mundo')
            }
      </script>
</body>
```

- Os trechos de código dentro do documento são executados na ordem, sequência, em que aparecem na página (documento HTML)

JavaScript - exemplo

```
<html>
      <body>
            <script type=”text/javascript”>
                  function helloWorld() {
                    alert(‘Olá mundo’)
                  }
            </script>
            <h1>E-commerce</h1>
            <script type=”text/javascript”>
                  alert(‘Seja bem-vindo(a)!’)
            </script>
      </body>
</html>
```

Atividade

1. Teste o código JS no body do seu documento HTML e descreva a ordem que os textos apareceram.

```
<html>
      <body>
            <script>
                  alert('Seja bem-vindo(a)!')
            </script>
            <h1>E-commerce</h1>
            <script type="text/javascript">
                  alert('Realize seu pedido aqui!')
            </script>
      </body>
</html>
```

JavaScript

- Código JS pode definido em um arquivo separado

```
<head>
      <script type=”text/javascript” src=”script.js” />
</head>
```

- Vantagens:
  - Separa o HTML do JS
  - Permite o armazenamento em cache do JS

JavaScript

- Os comandos JS são sensitive case, ou seja, diferencia entre letras minúsculas e maiúsculas
- Se houver erro de sintaxe, o JS interpretará o comando como um nome de variável
  - Cuidado! Ele tentará executar o seu código
- Formas de comunicação com o usuário:
  - Apenas texto:
    - `alert(mensagem);`
  - Caixa de diálogo que retorna verdadeiro (OK) e falso (cancel)
    - `if (confirm("Aceita os termos de uso?")){ alert("Concordo!")} else {alert("NÃO concordo!")}`

JavaScript

- Para receber inputs dos usuários pode-se usar o **prompt**
  - `valor = prompt(“Minha mensagem”, “meu texto”);`
  - **valor** é a variável que receberá o texto digitado pelo usuário
  - **prompt** é a função nativa do JS
  - “**Minha mensagem**” é a mensagem que aparecerá como label na caixa de input
  - **“Meu texto”** é um texto, opcional, que aparecerá na linha de digitação do usuário

```
<head>
      <script>
            cpf = prompt("Informe seu CPF")
            alert("Seu CPF é: " + cpf)
      </script>
</head>
```

Atividade

2. Escreva um código JS usando as caixas de diálogo adequadas. Você deve perguntar ao usuário se ele deseja realizar um pagamento, se ele desejar solicite o CPF e informe pagamento realizado com sucesso, se não exiba uma mensagem se agradecendo.

Variáveis

- Recebem valores numéricos ou texto
- Podem ser criadas pela simples atribuição de valores
  - nome = “Ana”
- Podem ser criadas com o uso do **var**
  - **var** nome =”Ana”
- Podem ser criadas com o uso **let**
  - **let** nome =”Ana”
- O uso depende do escopo das variáveis

Variáveis

- **Variáveis locais a blocos:** criadas e visíveis somente dentro de um bloco (ex. if, for, ...)
  - let x=100
- **Variáveis locais a funções:** criadas e visíveis somente dentro de uma função
  - var x=100
- **Variáveis globais**: criadas em qualquer lugar e visíveis em todo o documento.
  - x=5

Variáveis - exemplo

```
a=5
if(a==5) {
        let b =;
        c= a*b
}
console.log(a);//5
console.log(c);//20
console.log(b);//Não definido
```

Atividade

3. Escreva um código JS que realize os seguintes passos:
   - Guarde o valor da gasolina em uma variável global;
   - Leia a distância percorrida e o consumo de um carro, guardando os valores em variáveis globais;
   - Se ambos os valores forem maiores do que zero, calcule o valor gasto com combustível em uma variável de bloco, exibindo esse valor em uma caixa de mensagem.

JavaScript

- Funções são blocos de construção fundamentais em JavaScript.
- Uma função é um procedimento de JavaScript
- Uma função é uma sequência de instruções.

```
<script>
	function nomeDaFuncao(var1, var2, …, varX) {
		//var, if, for, …
		//return X
	}
</script>
```

JavaScript

- Função para identificar se o aluno foi aprovado, reprovado ou precisa fazer VS

```
<script>
	function verificaAprovacao(nota) {
		if(nota > 7) { alert("Aprovado");}
		else if(nota <= 4) { alert("Reprovado");}
		else{
			alert("Precisa realizar VS!");
		}
	}
</script>
```

Atividade

4. Escreva um código JS da função anterior, de verificar se o aluno foi aprovado e:

- Simule exemplos para as três condições (reprovado, aprovado e fazer VS)
- Use uma caixa de diálogo para ler a nota do aluno e verificar sua situação
- Modifique sua função para retornar o valor da função (dica: use o return “MENSAGEM” e exiba a mensagem retornada pela função

JavaScript

```
function exemploEscopo(){
        var a=5
        var b=10
        ///=== compara tipo e valor == compara o valor
        if(a === 5) {
                let a = 4; // escopo de bloco
                b=15; // escopo da função
                c= 20; // escopo global
                console.log(a);//4
                console.log(b);//15
        }
        console.log(a);//5
        console.log(b);//15
}
```

Vetores

- Armazena uma sequência de valores;
  - var numeros = [1,10,100,1000]
    - console.log(numeros) // [1,10,100,1000]
    - console.log(numeros[0]) // 1
  - var objetos = [3, ’nome’, numeros]
    - console.log(objetos[2][1]) // 10

Vetores

- **+=** é abreviação para atribuição com adição
  - x += y é o mesmo que x = x + y
- Da mesma forma -=, *=, /= ou %= (% é o resto da divisão)
- x++ equivale a acrescentar 1 a x
  - x = x+1
- x++ equivale a subtrair 1 a x
  - x = x-1

Vetores

- Operadores a serem utilizados em comandos condicionais (If, for e While)
  - == igual sem considerar o tipo
  - === igual considerando o tipo
  - != diferente sem considerar o tipo
  - !== diferente considerando o tipo
  - > Maior
  - >= Maior ou igual
  - < Menor
  - <= Menor ou igual
  - && E
  - || Ou

Estruturas de controle

- If

```
if (condição1)
   instrução1
else if (condição2)
   instrução2
...
Else
  instruçãoN
```

```
<script>
      function verificaAprovacao(nota) {
            if(nota > 7) { alert("Aprovado");}
            else if(nota <= 4) {
                  alert("Reprovado");
            }
            else{
                  alert("Precisa realizar VS!");
            }
      }
</script>
```

Estruturas de controle

- **For**
  - for ([inicialização]; [condição]; [expressão final]) declaração

```
for (var i = 0; i < 9; i++) {
console.log(i);
}
```

Estruturas de controle

- **While**
  - while (condição) { rotina }
  - Exemplo:

```
var n = 0;
var x = 0;

while (n < 3) {
  n++;
  x += n;
}
```

Atividade

5. Escreva uma função JS que recebe um vetor de números e retorna à posição do maior deles:

- Dicas:
  - Uma variável vetor contém propriedades que retorna a quantidade de seus elementos
    - Exemplo: vetor.length

Dicas

- Learning
  - w3schools (principalmente JS Functions e JS HTML DOM
- Exercícios e prática
  - https://www.w3schools.com/js/exercise_js.asp
  - https://www.learn-js.org/
- Teste online
  - JSFiddle