Aula 6

Fundamentos da Programação *Web*

Profª Margarete Klamas

1

Conversa Inicial

2

Temas

- **Tipos de dados**
- **Funções**
- **BOM (*Browser Object Model*)**
- **DOM (*Document Object Model*)**
- **Eventos**

3

JavaScript – tipos de dados

4

O que são objetos?

- **Objetos são as estruturas de dados conhecidas como *registro*. Um registro é uma coleção não ordenada de campos nomeados. Cada campo tem seu próprio nome (ou chave) e um valor atribuído. No caso de objetos JavaScript, esses campos geralmente são chamados de propriedades**

5

Exemplo de objetos

```
let exeObj = {
    nr: 500,
    str: "palavra"
};
```

6

Como criar um objeto

- **Sintaxe:**
  - **nomeObjeto =new construtor(argumento);**
- **Exemplo:**

```
data=new Data();
let livro = new Object();
```

7

Exemplo de um objeto

```
> var livro = new Object();
  livro.titulo="A Bela e a Adormecida";
  livro.autor="Neil Gaiman";
  livro.editora="Rocco Jovens Leitores";
  livro.anoPublicacao=2015;
  livro.edicao="1ª";
  livro.paginas=72;
  livro.preco="R$ 30,00";
  livro.frete= function(ceporigem, cepdestino,peso){
     var valorFrete=" ";
     //script do calculo frete
     return valorFrete;
  }
  livro.capitulo1="Era o reino mais próximo";
  livro.capitulo2="A rainha acordou cedo";
  livro.capitulo3="Os três anões emergiram";
  livro.capitulo4="-Dormindo? - perguntou a rainha";
  livro.capitulo5="Ela cavalgou um dia inteiro";
  livro.capitulo6="O castelo na Floresta de Acaire";
< 'O castelo na Floresta de Acaire'
```

8

Funções construtoras

```
function Carro(marca, preco)
{
  this.marca = marca;
  this.preco = preco;
}

const honda = new
Carro('Honda', 244000);
const nissan = new
Carro('Nissan', 120000);
nissan.marca = 'Nissan';
nissan.preco = 120000;
```

```
function Carro() {
  this.marca = 'Marca';
  this.preco = 0;
}

const honda = new Carro();
honda.marca = 'Honda';
honda.preco = 244000;
const fiat = new Carro();
fiat.marca = 'Nissan';
fiat.preco = 120000;
```

9

Como iterar por um objeto

- **O laço** **lades** **enume** **original** **de inse** **o para** **cada p**
- **Sintaxe**
  - **for (v**
  - **}**

```
let carro={
portas: 4,
rodas:4,
marca: "Um carro",
Avenda:true
};
for (let detalhes in carro){
console.log(detalhes+":"+
carro[detalhes]);
}
```

10

Funções

11

Conceito – função

- **Uma função é apenas um pedaço separado de código que você nomeia, da mesma forma que você nomeia uma variável. Se você quiser usá-lo em algum lugar, ele é simplesmente referenciado por esse nome (dizemos chamar a função)**

12

Funções – sintaxe

```
function nome([param[, param[, ... param]]]) {
   instruções
}
```

- **nome: é o nome da função**
- **param: o nome de um argumento a ser passado para a função**

13

Exemplo

```
function olaMundo() {
      console.log("Ola");
      console.log("Mundo");
 }
olaMundo();
```

14

Funções – *return*

- **A função termina onde está a palavra *return* e permite armazenar numa variável o resultado da função**

15

Return – exemplos

```
 console.log("mensagem A");
    return;
    console.log("mensagem B");
}
mostrarMsg();
```

16

Parâmetros da função

```
function somar(num1, num2) {
return num1 + num2;
}
somar(35,2);
```

17

Função *arrow*

- **A função de seta (*arrow*) é uma forma mais curta de uma expressão de função**

```
let somar=(n1,n2) =>n1+n2;
console.log(somar(2,3));
```

18

BOM (*Browser Object Model*)

19

***Browser Object Model* – BOM**

- **O *Browser Object Model* (BOM) é um conjunto de objetos que permitem a interação entre JavaScript e o navegador. Ele fornece acesso às características do navegador, como janelas, histórico, *cookies* e muito mais**

20

Objeto *window*

- **O objeto de nível mais elevado do BOM é o *window*. Representa uma janela aberta do navegador**

21

Closed

- **A propriedade *closed* retorna o valor booleano *true* se a janela estiver fechada e *false* se estiver aberta**

22

History

- **O objeto *history* contém as URLs visitadas pelo usuário (na janela do navegador)**

MÉTODO	DESCRIÇÃO
back()	Carrega URL anteriormente visitada
forward()	Carrega URL visitada depois da URL atual
go(n)	Admite número como parâmetro
length()	Quantidade

```
<button onclick="history.back()">Voltar</button>
```

23

Open

- **O método open() abre uma nova janela. Abrir janelas sem o consentimento do usuário deve ser evitado**

Sintaxe:

`window.open(URL, name, specs, replace)`.Os parâmetros são opcionais. Caso não informados, abre uma nova janela em branco.

24

Close

- **O método close() fecha uma janela**

25

Onload

- **O evento onload( ) ocorre quando uma página termina de ser carregada**

26

Document

- **A propriedade *document* é uma referência ao objeto *document* que representa a marcação HTML de um documento apresentado na janela do navegador. É possível acessar todos os elementos da marcação via *script***

27

DOM (*Document Object Model*)

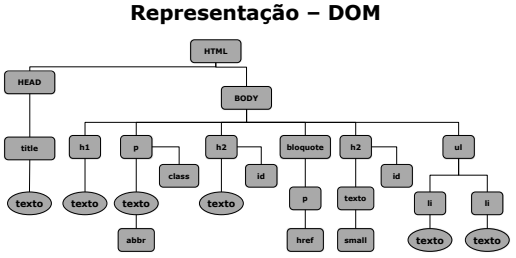
28

Conceito

- **A definição do DOM segundo o W3C:**
  "O DOM – Document Object Model é uma interface independente de plataforma e linguagem que permite aos programas e scripts acessar e atualizar dinamicamente a estrutura, o conteúdo e a estilização de documentos" (World Wide Web Consortium)

29

Representação – DOM



DOM: diagrama representativo de um documento HTML. Fonte: SILVA, 2010

30

Objeto document

- **O objeto *document* representa um documento aberto no navegador. Assim, quando abrimos um documento HTML em um navegador, é criado um objeto documento. Esse objeto é um objeto *window* e pode ser acessado com a sintaxe window.document. Desta maneira, podemos acessar todos os elementos HTML de uma página HTML**

31

getElement*

- **O método getElementById() acessa o elemento DOM identificando o id. ID é identificador único, exclusivo no documento**

```
var Um =
document.getElementById("def");
```

32

O getElemenstByClassName()

- **Retorna uma lista com todos os elementos que possuem o nome da classe dada**

```
Exemplo:
let lista =document.getElementsByClassName("item")
```

33

getElementsByTagName

- **O getElementsByTagName retorna uma lista com todos os elementos da *tag* informada**

```
Exemplo:
let lista = document.getElementsByTagName("p")
```

34

element.style

- **A propriedade *style* dos elementos do DOM permite que se definam regras de estilo. Sintaxe: el.style.propriedade="valor da propriedade"**

35

Eventos

36

Eventos

- **Os eventos podem ocorrer advindos da interação do usuário, como o clique num *link*, ou passar o *mouse* em cima, ou advindos do navegador, como carregar uma página. Evento é qualquer interação com um elemento HTML, clicar, mover, soltar etc.**

37

Eventos

EVENTO	DESCRIÇÃO
click	Ocorre quando o usuário pressiona o botão esquerdo do *mouse*
dclick	Ocorre quando o usuário pressiona duas vezes o botão esquerdo do *mouse*
mousedown	Ocorre quando o usuário pressiona qualquer um dos botões do *mouse*
mouseover	Ocorre quando o usuário solta qualquer um dos botões do *mouse*
mouseout	Ocorre quando o usuário desloca o ponteiro do *mouse* de onde estava

FONTE: Klamas,2023

38

Eventos

EVENTO	DESCRIÇÃO
mousemove	Ocorre quando o usuário move o ponteiro do *mouse*.
load	Ocorre quando há o carregamento da página
unload	Ocorre quando há o fechamento de uma janela
scroll	Ocorre quando o elemento possui barra de rolagem
focus	Ocorre quando o elemento recebe foco

FONTE: Klamas,2023

39

eventListener

- **O evento eventListener possui os métodos addEventListener e removeEventListener, que nos permitem anexar um manipulador de eventos a um elemento sem sobrescrever os eventos anteriores**

40