Instituto Federal de São Paulo

Análise e Desenvolvimento de Sistema

EDUCAÇÃO PÚBLICA 100% GRATUITA

Desenvolvimento Web II

Professor Johnata Santicioli

Sumários

- ✗ Introdução
- ✗ Sintaxe
- ✗ Operadores
- ✗ Variáveis
- ✗ Tipos de Dados
- ✗ Objetos
- ✗ Funções
- ✗ Eventos

Introdução

Estrutura | Estilo | Comportamento

Introdução

- JavaScript é uma linguagem de programação *interpretada*. Foi originalmente implementada como parte dos navegadores web para que scripts pudessem ser executados do *lado do cliente* e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicação assíncrona e alterando o conteúdo do documento exibido.

Flanagan, David; Ferguson, Paula (2002). JavaScript: The Definitive Guide 4th ed. O'Reilly & Associates [S.l.]

Introdução

- JavaScript:
- Interpretada pelo navegador
- Modelo de execução controlado por eventos
- Multiparadigma
- Baseado em objetos
- Tipagem dinâmica
- Case-sensitive

Introdução

- O que pode-se fazer com JavaScript:
- Animações
- Verificar formulários
- Manipular elementos do documento HTML
- Manipular arquivos XML e JSON
- Server-side com node.js e deno.js

Introdução

- Pode-se integrar o JavaScript no HTML das seguintes formas:

@ Integração interna

- No corpo da página (`<body>...</body>`)
- No cabeçalho (`<head>...</head>`)
- Em uma tag HTML

@ Integração externa

- Em um arquivo(.js) separado

Introdução

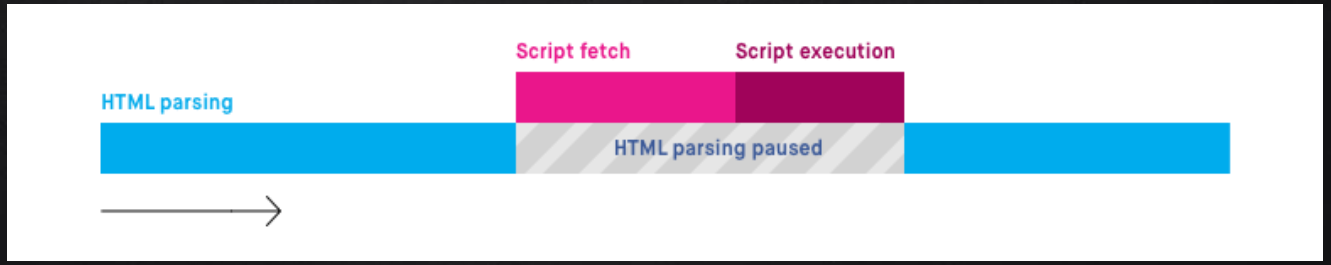
- Pode-se integrar o JavaScript no HTML das seguintes formas:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <script> alert("1-Olá Mundo!") </script> //no cabeçalho
  <script type="text/javascript" src="script.js"></script> //arquivo externa
</head>
<body>
  <input type="button" onclick="alert('3-Olá Mundo!')"> //na tag html
  <script> alert("2-Olá Mundo!") </script> //no corpo
  <script type="text/javascript" src="jquery.js"></script> //arquivo externa
</body>
</html>
```

Introdução

- ✗ Execução normal
- ✗ Por que normalmente coloca-se os scripts no fim da página?

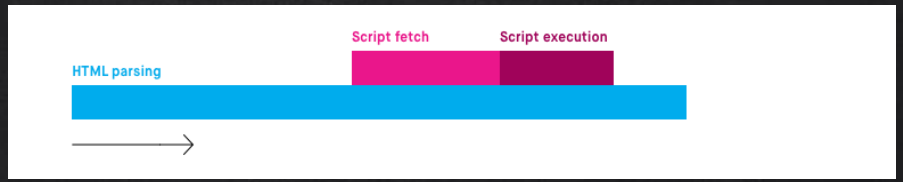
```
    ...
    <script type="text/javascript" src="script.js"></script>
</body>
```



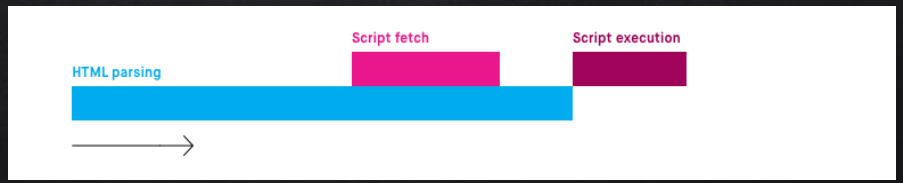
Introdução

- Atributos async e defer

```
<script async src="script.js"></script> //só para arquivo externa
```




```
<script defer src="script.js"></script>
```



Saiba mais em: https://www.braziljs.org/p/diferencas-entre-async-e-defer

Tipo de Dados

@ Tipos primitivos

- Boolean (*lógico*)
- Null
- Undefined
- Number (*numérico*)
- BigInt (*número extensos e strings*)
- String (*caracteres*)
- Symbol (*símbolo único e imutável para chave de propriedade*)
- Function

@ Tipos Objetos

- Date, RegExp, Error, ... , objeto global e prototype
- Arrays

Tipo de Dados

Tipo	valor	Descrição	Exemplo
Boolean	`true`	Valor lógico verdadeiro	`var b = true;`
	`false`	Valor lógico falso	`var b = false;`

Number	`NaN`	**Not a Number** é o resultado de uma expressão com um operando que não pode ser convertido em valor numérico	`var a = NaN;`
			`var a = 10 / "x";`
			// a assume valor `NaN`

	`Infinity`	Representação de um valor infinito	`var a = Infinity;`
			`var a = 10/ 0;`
			// a assume valor `Infinity`

	`undefined`	conteúdo de variáveis não iniciadas	`var a;` // a assume valor `undefined`
	`null`	representa o **não valor** , ou seja a inexistência de valor associado a uma variável	`var a = null;` // x tem um não valor, ou seja `null`

Variáveis

- JavaScript é uma linguagem de tipagem dinâmica e fraca:
  - Não é necessário declarar o tipo de uma variável;
  - Todas as variáveis são objetos (referência);
  - A variável irá "alterar" o seu tipo de dado conforme os valores forem atribuídos:
    - Tipo de dado dinâmico:
      - var x; // x é indefinido
      - x = 5; // x é um número
      - x = "Johnata"; // x é uma string
      - x = true; // x é um valor lógico
      - x = null; // x é nulo

```
//template string
console.log(`Nome: ${x}`)
```

Palavras reservadas

```
abstract
boolean break byte
case catch char class const continue
debugger default delete do double
else enum export extends
false final finally float for function
goto
if implements import in instanceof int interface
long
native new null
package private protected public
return
short static super switch synchronized
this throw throws transient true try typeof
var volatile void
while with
```

Tipo de Dados

- Função typeof (operador) – retorna o tipo da variável ou constante

```
> a =9.5

> typeof(a)
'number'
```

Object	'object'
Array	'object'
Function	'function'
String	'string'
Number	'number'
Boolean	'boolean'
null	'object'
undefined	'undefined'

Atenção !

Atenção !

Escopo de Variáveis e Constantes

- Declarando variáveis e constantes:
  - var x = 5 – usada para declarar tanto variáveis locais em funções como variáveis globais
  - let x = 5 – usada para declarar uma variável local de escopo de bloco
  - const x = 5 – usada para declarar uma constante local de escopo de bloco

Operadores

- Operadores no JavaScript:

  - aritméticos: + , - , * , ** , / , %
  - atribuição: = , += , -= , *= , **= , /= , %= , ++ , --
  - relacionais: == , === , != , !== , < , <= , > , >=
  - lógicos: && , || , !
  - ternário: condição ? caso verdadeiro : caso falso

```
var salario = 1000
var bonus = salario * (salario > 1000 ? 0.10 : 0.15)
```

Operadores Lógicos

O operador === e !== avalia os operandos e então os compara, sem realizar conversão de tipo.

```
> 1 === 1
  true
> 1 === '1'
  false
> 1 !== 1
  false
> 1 !== '1'
  true
>
```

Os operadores == e != são menos estritos. Se os valores dos operandos não forem do mesmo tipo, ele tenta algumas conversões de tipo e realiza a comparação novamente.

```
> 1 == 1
  true
> 1 == '1'
  true
> 1 != '1'
  false
> 1 != 1
  false
>
```

Operadores e Prioridades

Operador	Descrição
`. [] ()`	Acesso a propriedades, indexação, chamadas a funções e sub-expressões
`++ — - ~ ! new delete typeof`	Operadores unários e criação de objectos
`* / %`	Multiplicação, divisão, divisão módulo
`+ -`	Adição, subtracção, concatenação de *strings*
`<< >> >>>`	Deslocação de Bit
`< <= > >= instanceof`	Menor, menor ou igual, maior, maior ou igual, `instanceof`
`== != === !==`	Igualdade, desigualdade, igualdade estrita, e desigualdade estrita
`&`	AND bit a bit
`^`	XOR bit a bit
`|`	OR bit a bit
`&&`	AND lógico
`||`	OR lógico
`? :`	Operador condicional (ternário)

Referências

EDUCAÇÃO PÚBLICA 100% GRATUITA

Você poderá estudar mais sobre o tema nas seguintes páginas:

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Grammar_and_types

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

https://www.braziljs.org/p/diferencas-entre-async-e-defer

https://www.alura.com.br/artigos/entenda-diferenca-entre-var-let-e-const-no-javascript

Objetos do JavaScript

- Abstração de parte do mundo real
- Coleção não ordenada de propriedades e métodos, constituída por pares nome e valor
- Pseudolinguagem orientada a objetos
- Tudo em JavaScript é objeto, exceto os tipos primitivos

Função Construtora

- A função construtora nativa do JS é denominada por:
  - `Object()`
- Exemplos de criação de objeto:
  - `var livro = new Object()`
  - `var carro = {}`

Objetos Globais

- Objetos Globais são objetos nativos, disponibilizados por padrão, na linguagem JavaScript.

- Podem ser categorizados por:
  - propriedades de valor – *Infinity, NaN*
  - propriedades de função – *eval(), parseFloat()*
  - objetos fundamentais – *Objetc, Function*
  - números e datas – *Number, Math, Date*
  - processamento de textos – *String, RegExp*
  - coleções (indexadas – *Array*, chaveadas – *Set, Map*)
  - dados estruturados – *JSON*
  - ...

- Exemplo:
  - objetos.js

Referências

EDUCAÇÃO PÚBLICA 100% GRATUITA

Você poderá estudar mais sobre o tema nas seguintes páginas:

https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Objects/Basics

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects

Funções do JavaScript

- Funções são subestruturas de código que podem receber parâmetros e retornar valores.
- Uma função é um objeto que tem código executável associado.
- Não é preciso declarar a visibilidade (todas são publicas) nem o tipo de retorno.
- Uma função pode ser chamada para executar o código dessa e retornar um valor calculado ou undefined

Funções do JavaScript

- Existem outras funções globais (nativas), pré-definidas, por exemplo:
  - A função eval avalia código JavaScript representado como uma string.
  - A função parseInt analisa um argumento do tipo string e retorna um inteiro da base especificada (base do sistema numérico).
  - A função parseFloat analisa um argumento do tipo string e retorna um número de ponto flutuante.
  - A função escape retorna uma nova string com certos caracteres substituídos por sua sequência hexadecimal.
  - A função alert mostrar ao usuário uma mensagem e um botão de confirmação de que o usuário tenha visto a mensagem

Definindo uma Função

```
function nomeFunção (parametro){
   corpofunção
}
```

```
//Função com parâmetros e com retorno
//num recebe 1 caso null
function quadrada(num=1){
   return num*num
}
```

```
//Expressão de função
var q = function quadrada2(num){
   return num*num
}
```

```
//Função sem parâmetros
function hello() {
   alert("Olá")
}
//invocando a função
hello()
```

Definindo uma Função

```
//Expressão de função anônima
var q2 = function (num) {
    return num*num
}
```

- ✗ Exemplo (Extensão Node Exec):
  - ○ funcao.js

```
//arrow(seta) function
(parâmetros) => {corpo}

var q3 = (num)  =>  { num * num }

var q3 = num => num * num

let soma = (num1, num2) => {
    let sum = num1 + num2
    return sum
}

let arrow = () => console.log("Olá!")
```

Referências

EDUCAÇÃO PÚBLICA 100% GRATUITA

Você poderá estudar mais sobre o tema nas seguintes páginas:

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Functions

https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

Eventos do JavaScript

- A interatividade entre o usuário e uma página web depende fortemente dos eventos que podem ser reconhecidos pelo navegador
- Há uma quantidade bastante grande de eventos
- Cada ação do usuário corresponde à um evento que pode ser capturado e processado
  - Exemplos: ao clicar o mouse, ao mover o mouse, ao pressionar uma tecla, ao modificar o conteúdo de um campo de formulário, etc.

Eventos do JavaScript

- Normalmente são associados à uma função que será executado quando o evento acontecer
  - – A associação de uma função ao evento de um elemento pode ser feito via HTML

```
<p id="p" onclick="mudarcor()">Texto</p>
```

ou via JavaScript

```
var p = document.getElementById("p")
        p.onclick = function () { ... }
```

Eventos do JavaScript

- Cada evento é representado por um objeto que é baseado na interface Event, e pode ter campos customizados adicionados e/ou funções usadas para obter informações adicionais sobre o que aconteceu.
- Categorias comuns de eventos: teclado, mouse, recursos, rede, foco, histórico de sessão, animações e transição CSS, formulários, impressão, composição de texto, tela etc.

- Exemplo
  - lampada

Eventos do JavaScript

- Onload – Evento chamado quando o carregamento de um elemento é finalizado.
  - Pode ser utilizado nas tags: body, img, script, link
- Onunload – Ocorre antes que um elemento seja removido da página
- Onfocus – Quando um elemento ganha foco
  - Pode ser nas tags: input, select e textarea.
- Onblur – Quando um elemento perde o foco
- Onselect – Quando um texto em um formulário é selecionado

Eventos do JavaScript

- Onclick – Evento chamado quando o elemento recebe o clique do mouse.
- Ondblclick – Quando um duplo-clique é executado sobre um elemento
- Onmouseenter – quando o ponteiro do mouse entra na área de um elemento
- Onmouseleave – quando o ponteiro do mouse sai da área de um elemento
- Onmousemove – quando o ponteiro do mouse se movimenta sobre um elemento (a cada pixel movimentado)

Referências

EDUCAÇÃO PÚBLICA 100% GRATUITA

Você poderá estudar mais sobre o tema nas seguintes páginas:

https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/Building_blocks/Events

https://developer.mozilla.org/pt-BR/docs/Web/Events

https://developer.mozilla.org/pt-BR/docs/Web/API/Event

THANKS!

Até a próxima aula!!!

www.ifsp.edu.br
johnata.santicioli@ifsp.edu.br