Aula 5

Fundamentos da Programação Web

Profª Margarete Klamas Marzani

1

Conversa Inicial

2

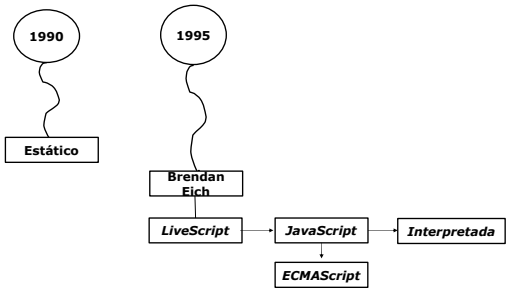
Temas

- **No tema 1 – Introdução**
- **No tema 2 – Variáveis, tipos de dados, comentários**
- **No tema 3 – Operadores e interação**
- **No tema 4 – Estruturas de controle condicionais**
- **No tema 5 – Loops**

3

JavaScript

4



5

6

Ambiente de desenvolvimento online

- CodePen: https://codepen.io/pen/
- JsBin: https://jsbin.com/?html,output
- JSFiddle: https://jsfiddle.net/

7

Ambiente de desenvolvimento local

- Visual Studio Code

8

Ambiente de #desenvolvimento

Interpretador

Depurador

9

Variáveis, Tipos de Dados e Operadores

10

Variáveis e constantes

Variáveis ⟹ Peso, Idade, Temperatura

Constantes ⟹ Valor do π

11

Variáveis

Variáveis ⟹ let var

Constantes ⟹ const

12

Variáveis locais e globais

- Se utilizarmos let e const fora de blocos de códigos, que são delimitados por { }, serão variáveis globais

13

Tipos de Dados - Dados literais primitivos

- Existem seis tipos de dados literais primitivos:
  - Boolean ⟹ true ou false
  - Number ⟹ números reais (decimais e inteiros)
  - BigInt ⟹ números inteiros de qualquer tamanho
  - String ⟹ representa uma sequência de caracteres
  - Undefined ⟹ valor padrão de uma variável, sem valor atribuído
  - Symbol

14

Interpolação de expressões

- let a=5;
- let b=10;
- console.log
  (`O resultado da operação é ${a+b} `);

15

Métodos

- length: propriedade que informa o número de caracteres de uma string
- charAt(index): informa a posição "index" da string (começam com zero)

16

- slice(beginIndex, [opcional] endIndex)
  - Método que retorna uma nova string que é criada a partir dos caracteres entre beginIndex (incluído) e endIndex (excluído)
  - Se endIndex for omitido, a nova string será de beginIndex até o final da string

17

- split(separator, [opcional] limit)
  - Divide a string em substrings sempre que um separador é encontrado nessa string e retorna um array dessas substrings (falaremos algumas palavras sobre arrays em um momento), enquanto um limite opcional limit limita o número de substrings adicionadas à lista

18

Operadores e Interação

19

Operadores

Grupo	Operador	Descrição
Atribuição	=	Atribuição simples
	+=	Atribuição de adição
	-=	Atribuição de subtração
	*=	Atribuição de multiplicação
	/=	Atribuição de divisão
	%=	Atribuição de resto
	**=	Atribuição de exponenciação

20

Grupo	Operador	Descrição
Relacional	==	Igual
	===	Exatamente igual (conteúdo e tipo de dado)
	!=	Diferente
	!==	Exatamente diferente (conteúdo e tipo de dado)
	<	Menor
	<=	Menor ou igual
	>	Maior
	>=	Maior ou igual

21

Grupo	Operador	Descrição
Aritméticos	+	Adição
	-	Subtração
	*	Multiplicação
	/	Divisão
	%	Resto da divisão
	**	Exponenciação
	++	Incremento
	--	Decremento

22

Grupo	Operador	Descrição
Lógicos	&&	E (AND)
	||	OU (OR)
	!	NÃO (NOT)

23

Interação com o usuário

- Caixa de alerta
- Caixa de confirmação
- Caixa de prompt

24

Estruturas de Controle de Fluxo

25

if

```
if(condição) {
 //bloco de código a ser executado se a condição for verdadeira
}
```

26

Estrutura if .... else

- **if (condição) {**
  condição - código verdadeiro
  } else {
  condição - código falso

27

Estrutura if .... else....if

```
if (condição_1) {
 codigo
} else if (condição_2) {
 codigo
} else if (condição_3) {
 codigo
} else {
 codigo
}
```

28

Estrutura swich ... case

```
switch (expressão) {
 case resultado 1:
  código 1
  break;
 case resultado 2:
  código 2
  break;
 default:
  código
}
```

29

Laços de Repetição

30

for

- **Utilizamos o for em situações em que sabemos quantas vezes executar o loop**

31

```
for (inicialização; condição; incremento) {
 bloco de código (faça isso)
}
```

32

while

- **O loop while é um dos loops que normalmente usamos quando não sabemos exatamente quantas vezes repetir algo, mas sabemos quando parar**

33

```
while(condição) {
 bloco de código
}
```

34

do .... while

- **O loop do ... while é muito semelhante ao loop while, a principal diferença é que em um loop while, a condição é verificada antes de cada iteração, e no loop do ... while, a condição é verificada após cada iteração**

35

```
do {
 bloco de código
} while(condição);
```

36