# Desenvolvimento WEB II
# Prof. Ernani Gottardo

# Javascript String, Data Funções

# Javascript objeto Math

- O objeto JavaScript **Math** permite que você execute tarefas matemáticas em números.

| Método | Descrição |
|---|---|
| abs(x) | Retorna o valor absoluto de x |
| ceil(x) | Retorna x, arredondado para cima para o inteiro mais próximo |
| floor(x) | Retorna x, arredondado para baixo para o inteiro mais próximo |
| log(x) | Retorna o logaritmo natural (base E) de x |
| max(x, y, z, ..., n) | Retorna o número com o maior valor |
| min(x, y, z, ..., n) | Retorna o número com o menor valor |
| pow(x, y) | Retorna o valor de x à potência de y |
| random() | Retorna um número aleatório entre 0 e 1 |
| round(x) | Arredonda x para o inteiro mais próximo |
| sqrt(x) | Retorna a raiz quadrada de x |
| trunc(x) | Retorna a parte inteira de um número (x) |

# Javascript objeto Math

- Exemplos

```
const pi = Math.PI;
Math.abs(-4.7);
Math.min(0, 150, 30, 20, -8, -200);
Math.random();
let x = Math.floor(1.6);
let x = Math.random() * 100; // double entre 0 e 99
let x = Math.floor((Math.random() * 10) + 1); //inteiro
entre 1 e 10
```

# Javascript strings

- Uma string JavaScript armazena uma série de caracteres como "IFRS".

| Nome | Descrição |
|---|---|
| charAt() | Retorna o caractere em um índice especificado (posição) |
| concat() | Retorna duas ou mais strings unidas |
| indexOf() | Retorna o índice (posição) da primeira ocorrência de um valor em uma string |
| lastIndexOf() | Retorna o índice (posição) da última ocorrência de um valor em uma string |
| length | Retorna o comprimento de uma string |
| search() | Pesquisa uma string por um valor ou expressão regular e retorna o índice (posição) da correspondência |
| slice() | Extrai uma parte de uma string e retorna uma nova string |
| substr() | Extrai um número de caracteres de uma string, **de um índice inicial** (posição) um número especificado de caracteres |
| substring() | Método extrai caracteres entre dois índices/posições: início, fim **(exclusivo).** |
| toLowerCase() | Retorna uma string convertida em letras minúsculas |
| toUpperCase() | Retorna uma string convertida em letras maiúsculas |
| trim() | Retorna uma string com espaços em branco removidos |

# Javascript string

- Exemplos

```
let text = "HELLO WORLD";
let char = text.charAt(1);//resultado 'E'
let str = "Apple, Banana, Kiwi, Grape";
let part = str.substring(7, 13);//resultado 'Banana'
let part2 = str.substr(7, 13);//resultado 'Banana, Kiwi,'
let tam = str.length; // resultado 26
let pos = str.indexOf(','); //resultado 5
let pos = str.lastIndexOf(','); //resultado 19
```

# Javascript Date

- O objeto **Date** é um dos objetos intrínsecos do Javascript, utilizado para o gerenciamento de datas
- JS armazena datas como o número de milisegundos desde 1º de janeiro de 1970, 00:00:00 (método Date.now())
- Objetos de data são estáticos. O "relógio" não está "correndo".
- Exemplos:

```
const d = new Date();
const d1 = new Date("2024-03-25");
```

# Javascript funções de data

| Nome | Descrição |
|---|---|
| new Date() | Cria um novo objeto Date |
| getDate() | Retorna o dia do mês (de 1 a 31) |
| getDay() | Retorna o dia da semana (de 0 a 6) |
| getFullYear() | Retorna o ano |
| getHours() | Retorna a hora (de 0 a 23) |
| getMilliseconds() | Retorna os milisegundos (de 0 a 999) |
| getMinutes() | Retorna os minutos (de 0 a 59) |
| getMonth() | Retorna o mês (de 0 a 11) |
| getSeconds() | Retorna os segundos (de 0 a 59) |
| getTime() | Retorna o número de milissegundos desde a meia-noite de 1º de janeiro de 1970 e uma data especificada |
| now() | Retorna o número de milissegundos desde a meia-noite de 1º de janeiro de 1970 |

# Javascript funções de data

| Nome | Descrição |
| --- | --- |
| setDate() | Define o dia do mês de um objeto de data |
| setFullYear() | Define o ano de um objeto de data |
| setHours() | Define a hora de um objeto de data |
| setMilliseconds() | Define os milisegundos de um objeto de data |
| setMinutes() | Define os minutos de um objeto de data |
| setMonth() | Define o mês de um objeto de data |
| setSeconds() | Define os segundos de um objeto de data |

# Javascript  date format

- Três tipos de formatos de entrada de data são usados em JavaScript:

| Tipo | Exemplo |
| --- | --- |
| ISO Date (ano-mês-dia) | "2018-03-25" (The International Standard)<br>(Atenção !! ver time zone) |
| Short Date (mês/dia/ano) | "03/25/2018" |
| Long Date (mês dia ano) | "Mar 25 2018" or "25 Mar 2018" |

# Javascript  checar se data é valida

- Entrada de dados

```
var data_tela = prompt("Digite uma data no formato DD/MM/YYYY ");
var ano = data_tela.substring(6, 10);
var mes = data_tela.substring(3, 5);
var dia = data_tela.substring(0, 2);
var data = new Date(mes + '/' + dia + "/" + ano);
document.write("Data Digitada " + data.toLocaleDateString());
if (isNaN(data) || parseInt(dia) != parseInt(data.getDate())) {
  alert("Cuidado, Data Inválida!!");
}
```

# Javascript  cálculos com data

- Somar dias

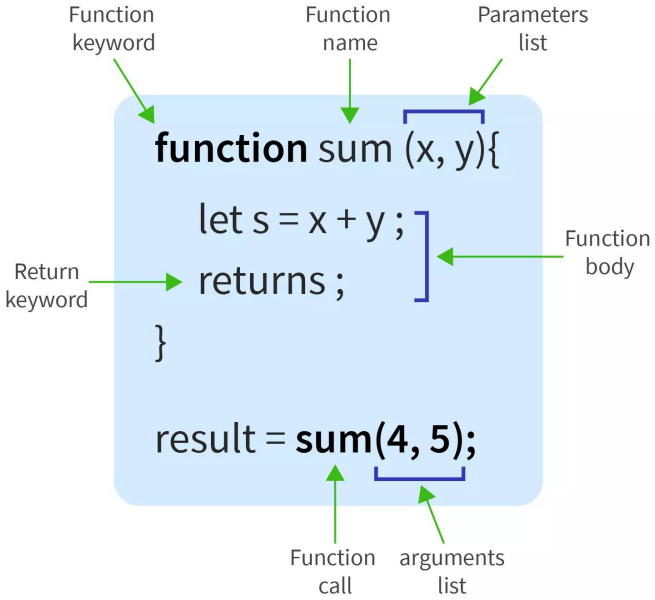
```
var dataatual = new Date();
dataatual.setDate(dataatual.getDate() + 7);
document.write('Data + 7 dias ' +
dataatual.toLocaleDateString());
```

- Diferença entre duas datas

```
var dataatual = new Date();
var dtvencto = new Date('2024-03-01');
document.write('Dias vencidos: ' + Math.floor((dataatual -
dtvencto) / (1000 * 60 * 60 * 24)));
//(1000 * 60 * 60 * 24) -> calcula o número de milisegundos em um
dia porque a diferença entre duas datas é dada em milisegundos
```

# Javascript  funções

- Funções são consideradas um dos componentes fundamentais em Javascript.
- Uma função JavaScript é um **bloco de código** projetado para executar uma tarefa específica.
- Uma função JavaScript é executada quando "algo" a **invoca** (a chama).



# Javascript  funções anonimas

- Uma função JavaScript também pode ser definida usando uma **expressão**.
- Uma expressão de função pode ser armazenada em uma **variável**
- Exemplo

```
const x = function (a, b) {
  return a * b
};
let z = x(4, 3);
```

# Javascript  funções seta (arrow)

- As funções de seta foram introduzidas no ES6.
- As funções de seta nos permitem escrever uma sintaxe de função mais curta
- Exemplo

```
let myFunction = (a, b) => a * b;
```

# Javascript escopo de variáveis

- Variáveis declaradas dentro de uma função JavaScript são **LOCAIS** para a função
- Exemplo

```
// código aqui NÃO pode usar carName

function myFunction() {
  let carName = "Volvo";
  //código aqui PODE usar carName
}

// código aqui NÃO pode usar carName
```

# Javascript escopo de variáveis: let vs var

- Variáveis declaradas por let estão disponíveis somente **dentro do bloco** onde são definidas.
- Variáveis declaradas por var estão disponíveis **em toda a função** na qual são declaradas.
- Exemplo

```
function testa(x) {
  if (x == 0) {
    var y = 1;
    let w = 1;
  }
  console.log(y); //ok
  console.log(w); //erro, w só é visível dentro do bloco{}
}
testa(0);
```

# End