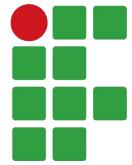


INSTITUTO FEDERAL
Paraná
Campus Paranaguá

JavaScript

Desenvolvimento Web

Prof. Diego Stiehl



JavaScript (JS)

- Linguagem de programação
- Criada para rodar em browsers
 - Lado cliente (front-end)
 - Atualmente tem mais usos
- Orientada a Objetos
- Multiplataforma
- Tipagem Dinâmica
- Sabe operar em conjunto com HTML e CSS
 - Incrementa as funcionalidades de um site

HTML



CONTEÚDO

SUBSTANTIVOS

`<p></p>`

Significa “parágrafo”

CSS



APRESENTAÇÃO

ADJETIVOS

`p { color: red; }`

Significa “o texto do parágrafo é vermelho”

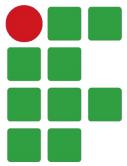
JS

EFEITOS DINÂMICOS
PROGRAMAÇÃO

VERBOS

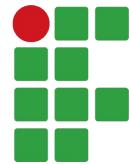
`p.hide();`

Significa “esconde o parágrafo”



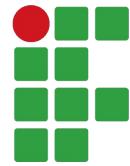
Java?

- JavaScript não tem relação alguma com Java
 - Semelhanças:
 - Nome
 - Alguns termos e palavras-chave
- As duas linguagens são estruturalmente diferentes
 - Apresentam estruturas muito distintas
 - Diferentes arquiteturas
 - Motivos de uso diferentes
 - Ambientes de execução diferentes
 - ...



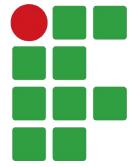
Breve Histórico

- Criado em 1995 com o nome LiveScript
- Em 1996 mudou de nome para JavaScript
 - Para atrair programadores Java
- Em 1997 a Ecma International cria um padrão para especificar a linguagem
 - Denominado ECMAScript 1
 - Portanto:
 - ECMAScript: padrão (documentos, especificações, ...)
 - JavaScript: a linguagem na prática



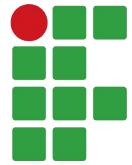
Breve Histórico

- Em 2009 é lançado o ECMAScript 5 (ES5)
 - Adição de muitos recursos
 - É o padrão mais bem estabelecido entre os navegadores atuais: suporte total (principais)
- 2015: ECMAScript 2015 (ES6 / ES2015)
 - Maior atualização na história da especificação
 - Nem todos navegadores atuais implementaram todos os recursos (verificar atualizações)
- Atualizações “menores” anuais
 - ES2016, ES2017, ES2018, ES2019 (ESNext) ...



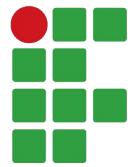
Detalhes sobre suporte

- ECMAScript 6 compatibility table
 - <https://kangax.github.io/compat-table/es6>
- Can I use... Support tables for HTML5, CSS3, etc
 - <https://caniuse.com>



JavaScript e HTML

- JS pode atuar em conjunto com HTML e CSS
 - É fundamental ter conhecimentos sobre ambos
- Scripts têm a **capacidade de acessar e alterar informações** contidas nas páginas HTML
 - Esta manipulação ocorre direto no browser
- Um trecho de script pode ser acionado por um evento do usuário em uma página HTML



JavaScript no HTML

- Para vincular o JS ao HTML, existem 3 formas
 - Script externo ← Forma mais indicada

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

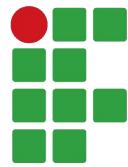
- Script junto ao arquivo HTML → tag <script>

```
<script type="text/javascript">  
    function mensagem() {  
        alert('mensagem');  
    }  
</script>
```

- Inline → trecho JS junto a um evento no HTML

```

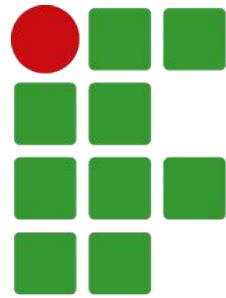
```



Chrome DevTools

- O Google Chrome possui uma ferramenta muito boa para “debugar” páginas web
 - Chrome DevTools (vocês já conhecem)
- Para abrir: F12
- A aba Console permite visualização/execução de trechos em JavaScript
- Podemos escrever nele do JS usando:

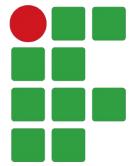
```
console.log('Qualquer coisa');
```
- Outros browsers têm equivalentes



INSTITUTO FEDERAL
Paraná
Campus Paranaguá

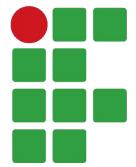
JavaScript

Básicos da Linguagem



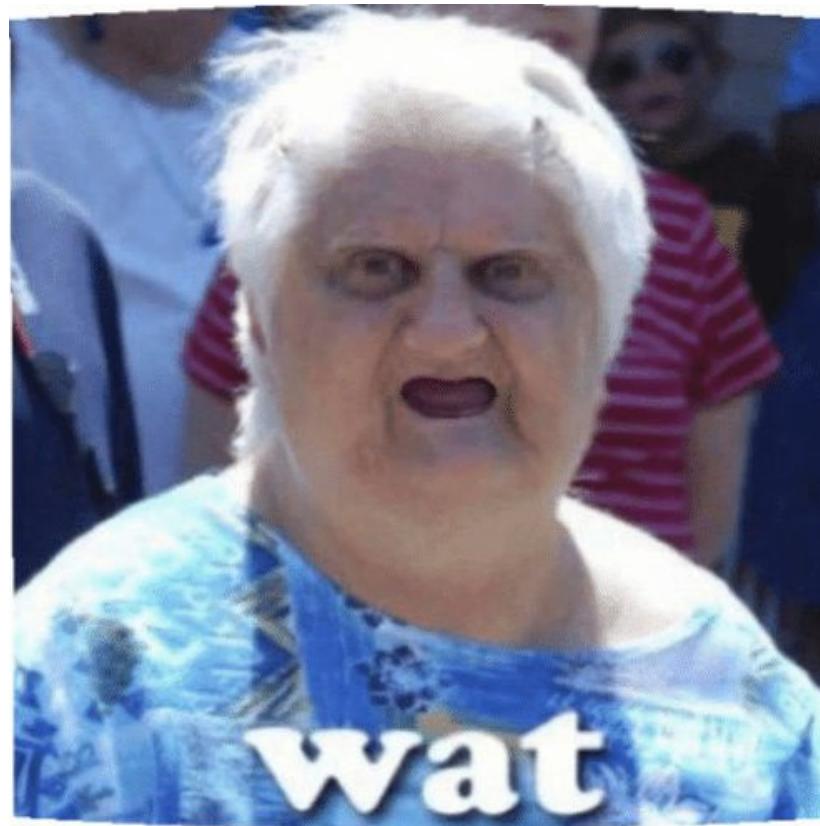
Tipagem Dinâmica

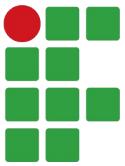
- JavaScript é uma linguagem dinamicamente tipada
 - Não precisamos informar o tipo de uma variável
- O tipo são inferidos com base nas atribuições de valores
- O tipo de uma variável pode mudar com o passar do tempo



Tipagem Dinâmica

- Tipagem Dinâmica? Ähn?

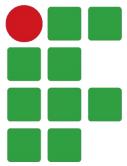




Variáveis

- A definição de variáveis em JavaScript pode ser feita com três palavras reservadas
 - `var` ← Qualquer variável (forma antiga <= ES5)
 - `let` ← Variável mutável (ES6)
 - `const` ← Variável imutável (ES6)
- Declaramos com a palavra mais o nome:

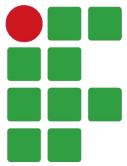
```
var idade;  
  
let peso;  
  
const nome = "Diego";  
  
let altura = 1.92;
```



Case Sensitive

- JavaScript é case sensitive
 - Nome é diferente de nome
- Convencionamos utilizar CamelCase na nomeação de variáveis, objetos e outros elementos

```
let euSouUmaVariavel;
```



Popups

- Mensagem

```
alert("Seja bem vindo!");
```

- Entrada de dados

```
const nome = prompt("Informe seu nome:");
```

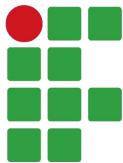
- Confirmação

```
const aceito = confirm("Aceita os termos?");
```

- Observação:

- Usar somente para testes

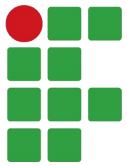
- Experiência péssima para os usuários na web



Tipos

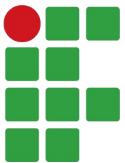
- Como a tipagem das variáveis em JavaScript é dinâmica, não especificamos o tipo na declaração
 - O tipo é descoberto durante uma atribuição de valor

```
let idade;      // Tipo ainda não definido
idade = 15;    // O tipo agora é "number"
idade = "20";  // O tipo agora é "string"
```



Tipos de Dados

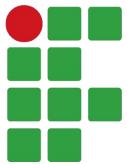
- JavaScript possui 5 tipos primitivos
 - **number** → Floats, decimais e inteiros
 - **string** → Sequências de caracteres
 - **boolean** → true / false
 - **undefined** → Tipo de dados de uma variável que ainda não teve seu valor definido
 - **null** → Não existente
 - Usado com objetos



number

- Utilização

```
let i = 3;  
let peso = 80.25;  
let negativo = -55;  
let realNegativo = -854.557;  
let expressao = 2 + (4 * 2 + 20 / 4) - 3;
```



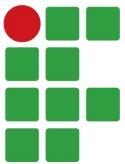
string

- Aspas simples ou duplas

```
let nome = "Diego";  
let endereco = "Rua " + "João Paulo II";  
nome = nome + ' Stiehl';  
endereco = endereco + ', ' + 987;
```

- Template Strings → Usar `sinal de crase`

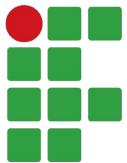
```
nome = `${nome} Stiehl`;  
endereco = `${endereco}, 987`;
```



boolean

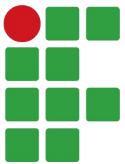
- Utilização

```
const identificado = true;  
let running = false;  
if (running) // ...  
let stopped = !running;  
const mensagem = stopped ? "Parado" : "OK";
```



undefined

- Como o JavaScript é uma linguagem de tipagem dinâmica, precisamos ter cuidado especial com a atribuição de valores
- Uma variável só terá um tipo “válido” assim que um valor for atribuído à ela
- Antes disso ela é do tipo undefined
 - Indefinida



undefined

- Utilização

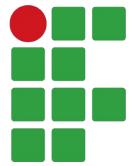
```
var nome = 'Diego' ;
```

```
var sobrenome;
```

```
console.log(nome);           // saída -> Diego
```

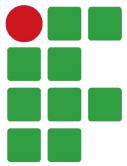
```
console.log(sobrenome); // saída -> undefined
```

```
console.log(idade);          // saída → ReferenceError:  
                            // idade is not defined
```



Coerção de Tipos

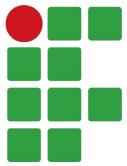
- A **coerção de tipos** acontece sempre que quisermos comparar, juntar ou relacionar variáveis de tipos distintos
- O JS vai considerar a “melhor forma” de tornar as variáveis compatíveis
 - string X number
 - number X boolean
 - string X boolean
 - ...



Coerção de Tipos

- Exemplos

```
const nome = "Diego";  
  
const idade = 31;  
  
const peso = 80.5;  
  
const confirmado = true;  
  
const teste1 = nome + idade;          // "Diego31"  
  
const teste2 = nome + idade + peso; // "Diego3180.5"  
  
const teste3 = idade + peso + nome; // "111.5Diego"  
  
const teste4 = confirmado + 10;      // 11  
  
const teste5 = false + nome;        // "falseDiego"  
  
if (idade + peso + "" == 111.5) {} // Avalia true  
if ("" + idade + peso == 3180.5) {} // Avalia true
```



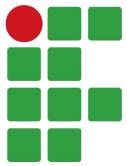
Conversão

- Em alguns momentos a coerção pode atrapalhar nosso algoritmo

"10" + "20" // "1020", mas eu queria 30 ou "30"

- Neste caso, usar as funções:
 - `parseInt()` ou `parseFloat()`

`parseInt(10) + parseInt(20) // 30 <-- number`



Operadores Básicos

- Matemáticos

+ - * / %

- Condicionais e Lógicos

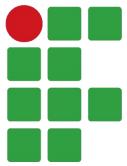
> >= < <= == != && || ! === !==

- Atribuição

= += -= *= /= %=

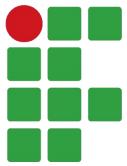
- Descoberta de tipo

typeof nomeVariavel



Operador ===

- Igualdade Estrita
- Usado quando queremos fazer uma **comparação sem realizar a coerção** de tipos
- Elementos precisam ser idênticos
 - Mesmo valor e tipo
- Útil devido à tipagem dinâmica
- Também existe o inverso
 - !=



Operador ===

- Exemplos com expressões

```
20 == 20      // true
```

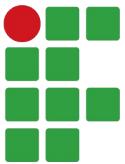
```
20 == "20"    // true
```

```
20 === 20     // true
```

```
20 === "20"   // false
```

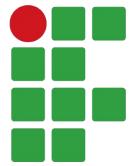
```
1 == true    // true
```

```
1 === true   // false
```



if

```
if (condicao) {  
    // VERDADEIRO  
} else if (outraCondicao) {  
    // TAMBÉM VERDADEIRO  
} else {  
    // FALSO  
}  
    if (fofoca) {  
        console.log("Nossa! Tô chocado!");  
    } else if (quemContouFoi0Elvis) {  
        console.log("Melhor averiguar!");  
    } else {  
        console.log("Para de mentir, piá!");  
    }
```



Truthy e Falsy

- Os seguintes valores são avaliados como false em expressões condicionais (if)

`false` // boolean

`null` // object

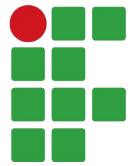
`undefined` // undefined

`0` // number

`NaN` // number

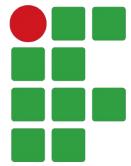
`''` // string

- Todos os demais são avaliados como true



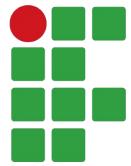
Prática 1 - IMC 1

- Coletar nome, peso e altura de duas pessoas
- Calcular o IMC delas
- Informar no console quem tem o maior IMC
 - Formato:
 - “O IMC de Diego (24.55) é maior que o de José (19.8)”



Prática 2 - IMC 2

- Colete o nome, altura e peso de uma pessoa
- Calcule seu IMC
- Considere a tabela → <http://www.calculoimc.com.br/tabela-de-imc>
- Dê uma mensagem e uma dica no formato:
 - “Diego! Você está acima do peso.
Dica: É hora de parar de comer Big Mac.”



Prática 3 - Login

- Capture login e senha do usuário
- Informe, em uma janela de alerta se houve sucesso ou não no procedimento de login
- Considere como válidos os logins:
 - Login: aluno | Senha: 123
 - Login: professor | Senha: 456
- Utilizar && e ||