DIO
CEZAR
GO

JavaScript

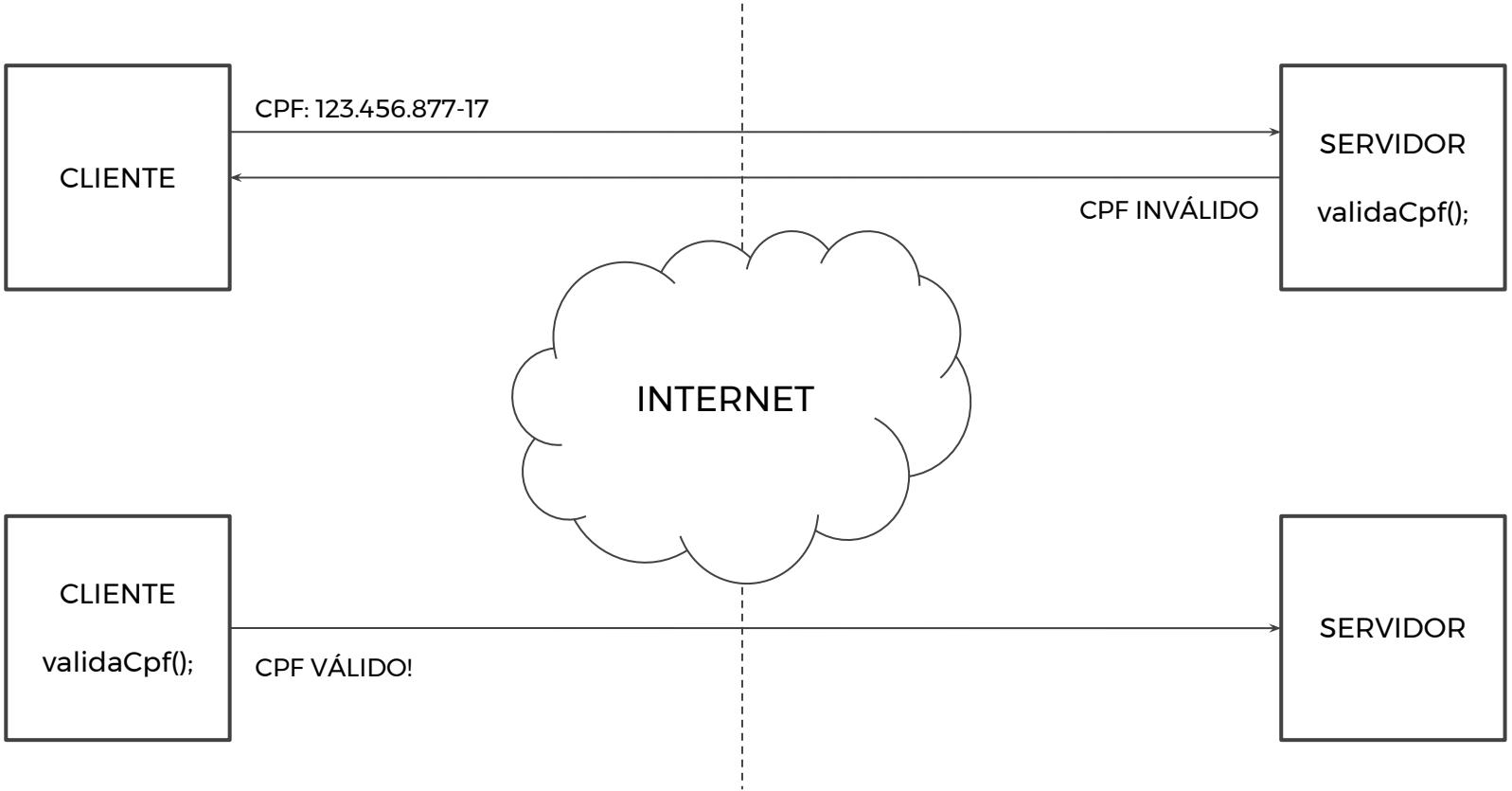
AGENDA

- o que é?
- exemplo de funcionamento;
- destaques da linguagem;
- incluindo um arquivo externo;
- sintaxe e estrutura;
- hierarquia dos objetos;
- capturando um elemento;
- alterando um estilo;
- alterando o conteúdo de uma div;
- objeto literal;
- construtores;

O QUE É?

- é uma linguagem de programação interpretada com características de orientação a objetos;
  - desenvolvida pela Netscape a fim de estender as capacidades de seu *browser*;
  - permite que um conteúdo executável seja incluído em páginas *web*;
  - sintaticamente semelhante a C e C++;

EXEMPLO DE FUNCIONAMENTO



DESTAQUES DA LINGUAGEM

- escrever código em um documento enquanto ele está sendo interpretado pelo *browser*;
- adicionar ou remover elementos *DOM* em tempo de execução;
- controlar o navegador → abrir novas janelas, obter suas dimensões, etc...;
- manipular formulários *HTML5*;

INCLUINDO UM ARQUIVO EXTERNO

como se faz para incluir um arquivo js em uma página html?

COMO FAZER?

```
<!DOCTYPE html>
<html lang="pt-br">
     <head>
          <meta charset="utf-8">
          <title></title>
     </head>
<body>
     <script language=”JavaScript” src=”./js/myLib.js”></script>
</body>
</html>
```

SINTAXE E ESTRUTURA

- é uma linguagem case sensitive;
- ignora espaços em branco;
- ponto e vírgula é opcional;
- suporta comentários no estilo clássico /* */ e //;
- não é necessário realizar a declaração de variáveis;
- até a especificação **ES5** tudo é função!
  - consegue-se *emular* classes com prototype;
- tipos primitivos: string, number, boolean, undefined e null;
  - são implícitos;
- *objetos* e *arrays* → a grande sacada!

SINTAXE E ESTRUTURA

- uma função é definida com o uso da palavra *function*;
- eventos → é parte importante do JavaScript;
  - estão atreladas quase sempre a ações por partes dos usuários:
    - onclick;
    - onmouseenter;
    - onmouseleave;
    - onkeyup;
- mais sobre eventos:
  - https://goo.gl/nmMQ3k
  - https://goo.gl/XD1YXP

HIERARQUIA DE OBJETOS

- **window**: o objeto mais acima na hierarquia, contém propriedades que se aplicam a toda a janela.
  - **location**: contém as propriedades da URL atual;
  - **history**: contém as propriedades das URLs visitadas anteriormente
  - **document**: contém as propriedades do documento contido na janela, tais como o seu conteúdo, título, cores, etc;

CAPTURANDO UM ELEMENTO

pode-se fazê-lo de duas formas:

COMO FAZER?

```
<script>
     var nome = document.getElementById("nome");
     var nome = document.getElementByName("nome");
</script>
```

- a partir desse momento a variável **nome** assume as propriedades do elemento com id nome;
- se for um *input* terá o atributo **value**;
- se for um *div* terá o atributo **style**;

ALTERANDO UM ESTILO

pode-se alterar o estilo de um elemento por seu **id**:

COMO FAZER?

```
<script>
     var texto = document.getElementById('mudaCor');
     texto.style.color = 'red';
</script>
```

ALTERANDO O CONTEÚDO DE UMA DIV

pode-se alterar o estilo de um elemento por seu **id**:

COMO FAZER?

```
<script>
     var myDiv = document.getElementById(‘myDiv’);
     myDiv.innerHTML = 'O texto foi trocado';
</script>
```

OBJETO LITERAL

- **o que é? →** é um tipo básico de objetos em *JavaScript*;
- formato popularizado através do *JSON*;
- objeto é criado utilizando um par de chaves {};
- suas propriedades e métodos são públicos;
- todo objeto literal é único (como se fosse static);
- seu uso é recomendado em situações onde não podem existir mais de uma instância do objeto;
- **DICA →** utilize-o para representar uma página html e organizar sua estrutura JavaScript;

OBJETO LITERAL

EXEMPLO DE UM OBJETO LITERAL

```
<script>
     var home = {
          init: function () {
               console.log(“Olá Mundo”);
          }
     };
     var home2 = home;
</script>
```

no exemplo acima, caso você altere/adicione propriedades em qualquer uma das variáveis (home ou home2), as modificações valem para ambas.

CONSTRUTORES

- **o que é? →** nada mais é que uma função JavaScript;
- pode ser executada como uma função ou...
- ser utilizada para instanciar um objeto utilizando a palavra reservada *new*;
- se executada como uma função normal → this equivalerá ao window;

CONSTRUTORES

EXEMPLO DE UM CONSTRUTOR

```
<script>
     function Categoria(nome) {
          this.nome = nome;
     }
     var categoria = new Categoria('Livros');
</script>
```

pode-se criar várias instâncias de um objeto Categoria.

CONSTRUTORES

EXEMPLO DE UM CONSTRUTOR

```
<script>
     function Categoria(nome) {
          var totalProdutos = 0;
          var self = this;
          var atualizaTotalProdutos = function() {
               self.totalProdutos += 1;
          };
          this.nome = nome;
          atualizaTotalProdutos();
     }
     var categoria = new Categoria('Livros');
</script>
```

`self` guarda o contexto para ser utilizado pela função local `atualizatotalProdutos`

CONSTRUTORES

EXEMPLO DE UM CONSTRUTOR

```
<script>
     Categoria.prototype.exibeProdutos = function () {
          var html = '',
          i;
          for (i = 0; i < this.produtos.length; i++) {
               html += this.produtos[i].nome;
          }
          return html;
     };
</script>
```

podemos atualizar/inserir novas propriedades em objetos criados a partir de funções construtoras;

APRENDAM A DEBUGAR!

https://goo.gl/VREr8f

UM GUIA COMPLETO PARA APRENDER MÓDULOS

https://goo.gl/mw9AU1

EXEMPLO AO VIVO

- criar um objeto literal para manipular elementos de uma página html utilizando `addEventListener`;

MATERIAIS COMPLEMENTARES

- https://goo.gl/brv4Wc
- https://goo.gl/wMYWE9
- https://goo.gl/ubT1My
- https://goo.gl/3G6xo2
- https://goo.gl/sML2HD