

JavaScript



Introdução

Esta imersão é baseada no ótimo
livro “JavaScript: The Good Parts”
de Douglas Crockford



História e por quê ...

- É a linguagem do *browser*
- A linguagem mais popular do mundo
- Feita em 1995 para o Netscape 2.0 sob o nome de Mocha depois LiveScript e por fim JavaScript (ECMA Script)
- A Microsoft adotou JavaScript no IE 3.0 em 1996



Brendan Eich

O porquê da má fama

- Sempre foi “a outra”
- Nome terrível !
- Control + C e Control + V
- APIs diferentes entre browsers
- Algumas características bem ruins (*bad parts*)



The bad parts

- Variáveis globais
- Escopo
- Inserção de ponto-e-vírgula
- Palavras reservadas
- parseInt
- +
- Valores falsos
- ===



... vamos nos focar nas partes boas !

The good parts

- Funções como valores
- Objetos dinâmicos
- Tipos fracos
- JSON





... e mais importante

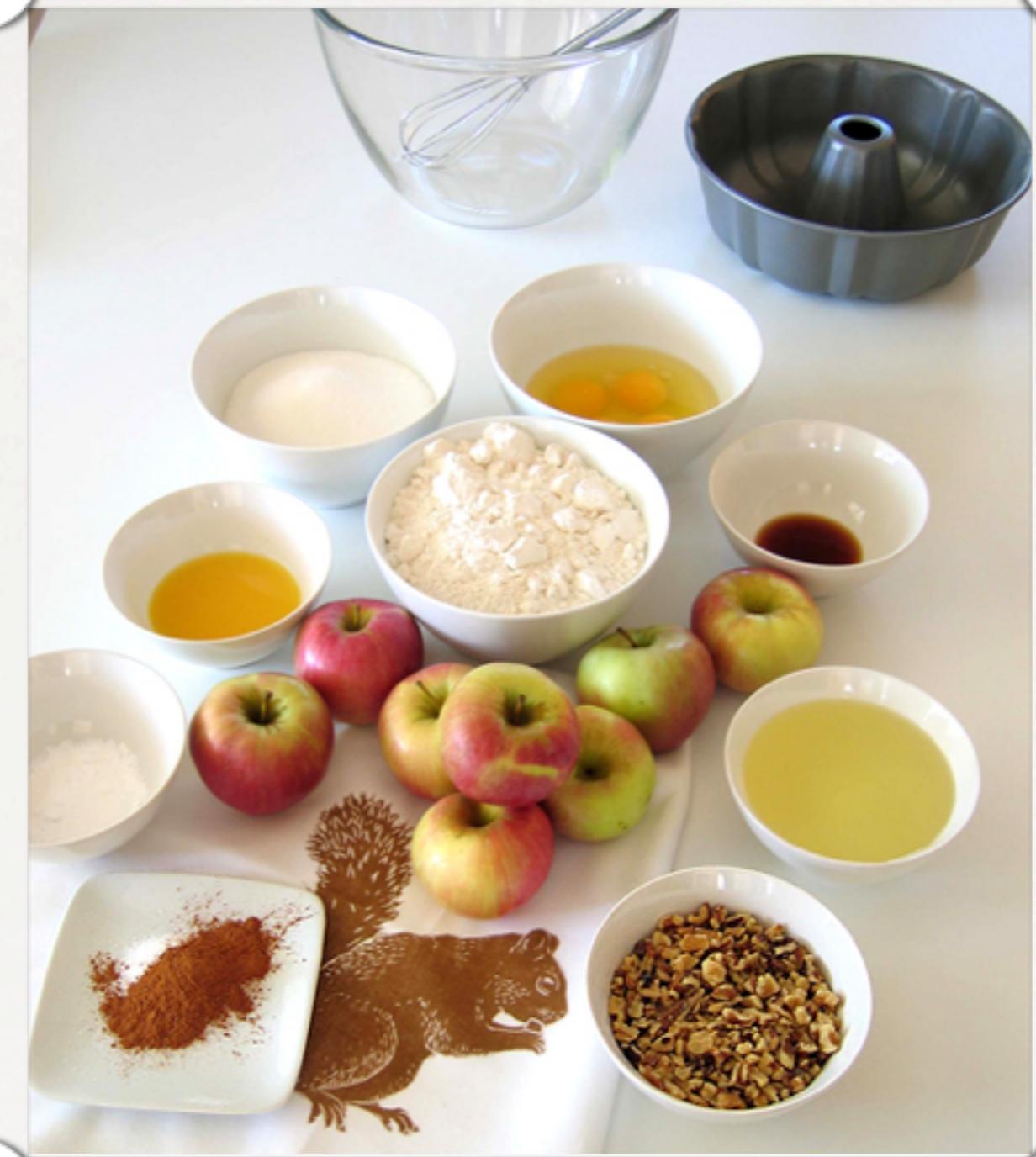


não dá para fugir



Ambiente de testes

- Você vai precisar:
 - Editor de texto
 - Browser
 - Conexão com a Internet



Ambiente de testes

- Editor de texto (sugestões):

- Sublime Text 2
- TextMate
- gedit
- Notepad++



Ambiente de testes

programa.html

```
<html>
  <body>
    <pre>
      <script src="programa.js"></script>
    </pre>
  </body>
</html>
```

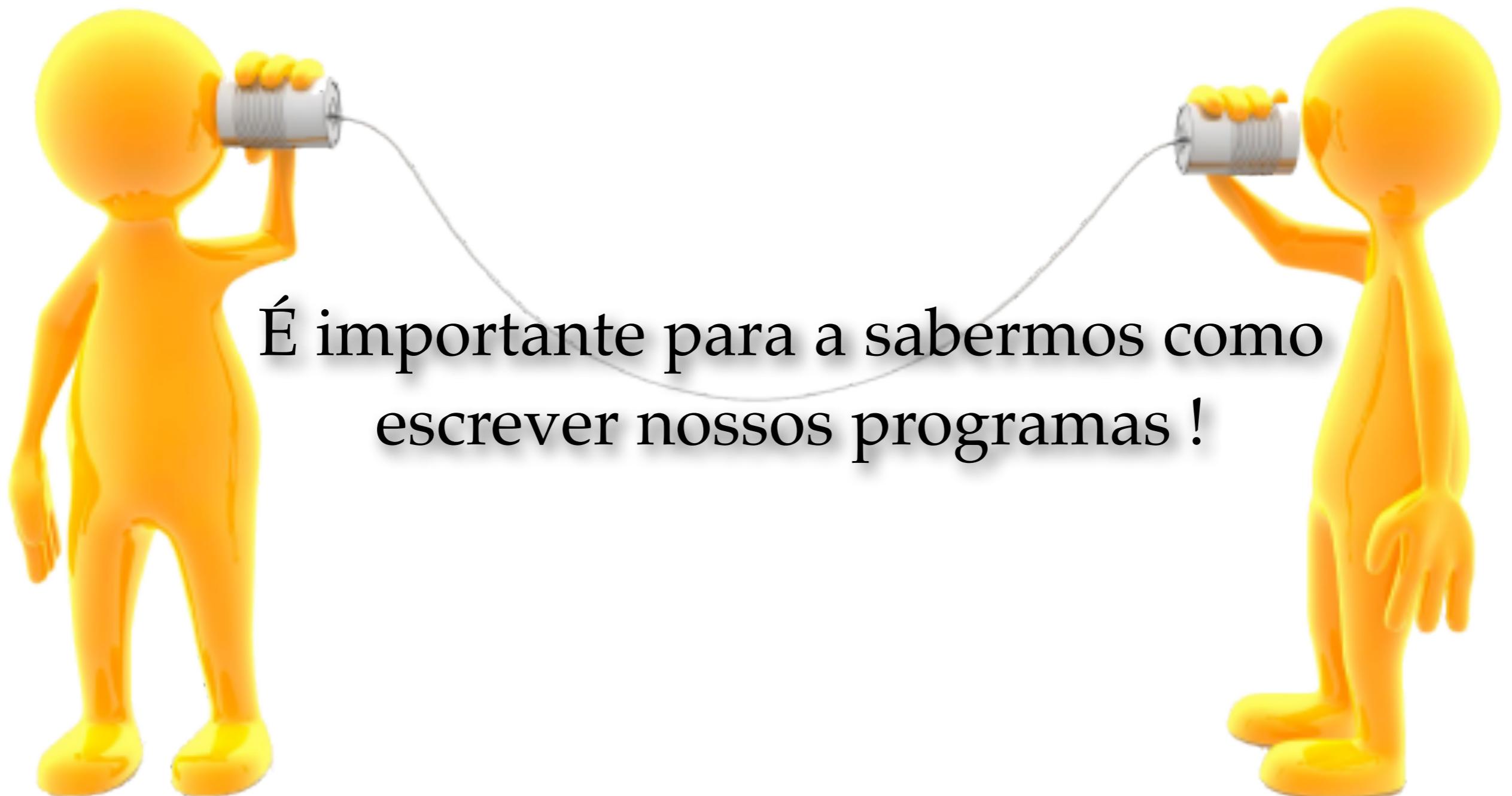
programa.js

```
document.writeln('123 Testando');
```

Resultado



Um pouco de sintaxe



É importante para a sabermos como
escrever nossos programas !

Nomes

- Um nome é uma letra seguido, opcionalmente, por outras letras, números e underlines
- Ex:
 - a
 - x1_
 - nomeCompleto
 - data_de_nascimento

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

Números

- Exemplos:
 - Inteiros: 0 12 -39 42
 - Frações: 3.14
 - Exponenciais: 3.14e+30

Operadores Aritméticos

+ soma	--
- subtração	+=
/ divisão	--=
* multiplicação	*=
% resto da divisão	/=
++	%=

Funções

- **toFixed(casasDecimais)**
 - Arredonda para o número de casas decimais
- **isNaN(número)**
 - Diz se o número é NaN (not a number - valor indeterminado)

Strings

- Aspas:
 - “Fulano de Tal”
- Apóstrofos :
 - ‘123 Testando’

Strings

- Propriedades:
 - **length** - Quantidade de caracteres
- Métodos:
 - **indexOf(str)**
 - **replace(a, b)**
 - **split(delimitador)**
 - **substring(inicio, fim)**
 - **toLowerCase()**
 - **toUpperCase()**

Strings

- Conversões:
 - **parseInt(str)** - String para inteiro
 - **parseFloat(str)** - String para float

Booleanos

true e false

- ▶ Valores falsos:
 - ▶ false
 - ▶ null
 - ▶ undefined
 - ▶ String vazio “ ”
 - ▶ 0
 - ▶ NaN



Arrays (*literais*)

- Vazio:
 - []
 - new Array()
- Com elementos:
 - ['banana', 'laranja', 'pitomba']

Arrays

- Exemplo:

```
var carros = new Array( );
carros[0] = "Volvo";
carros[1] = "BMW";
carros[2] = "Mercedes";
```

- Tamanho do array:

```
document.writeln(carros.length);
```

- Acessando valores:

```
document.writeln(carros[1]);
```

A black and white photograph of a man with glasses and a mustache, wearing a dark suit and tie. He is shouting into a large megaphone held up to his mouth. The background is plain white.

Statements

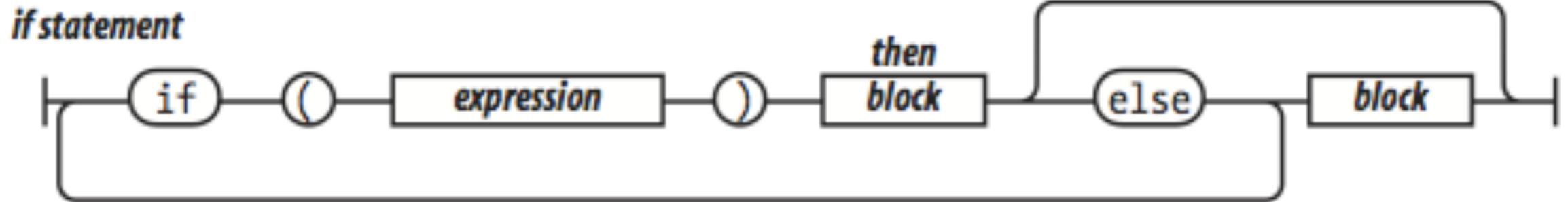
COMENTÁRIOS

- De linha
 - // Comentário
- De bloco
 - /* Comentário de 1 ou n linhas */

VARIÁVEIS

- var nome = expressão;
- Exemplo:
 - var idade = 20;
 - var pais = 'Brasil';

IF

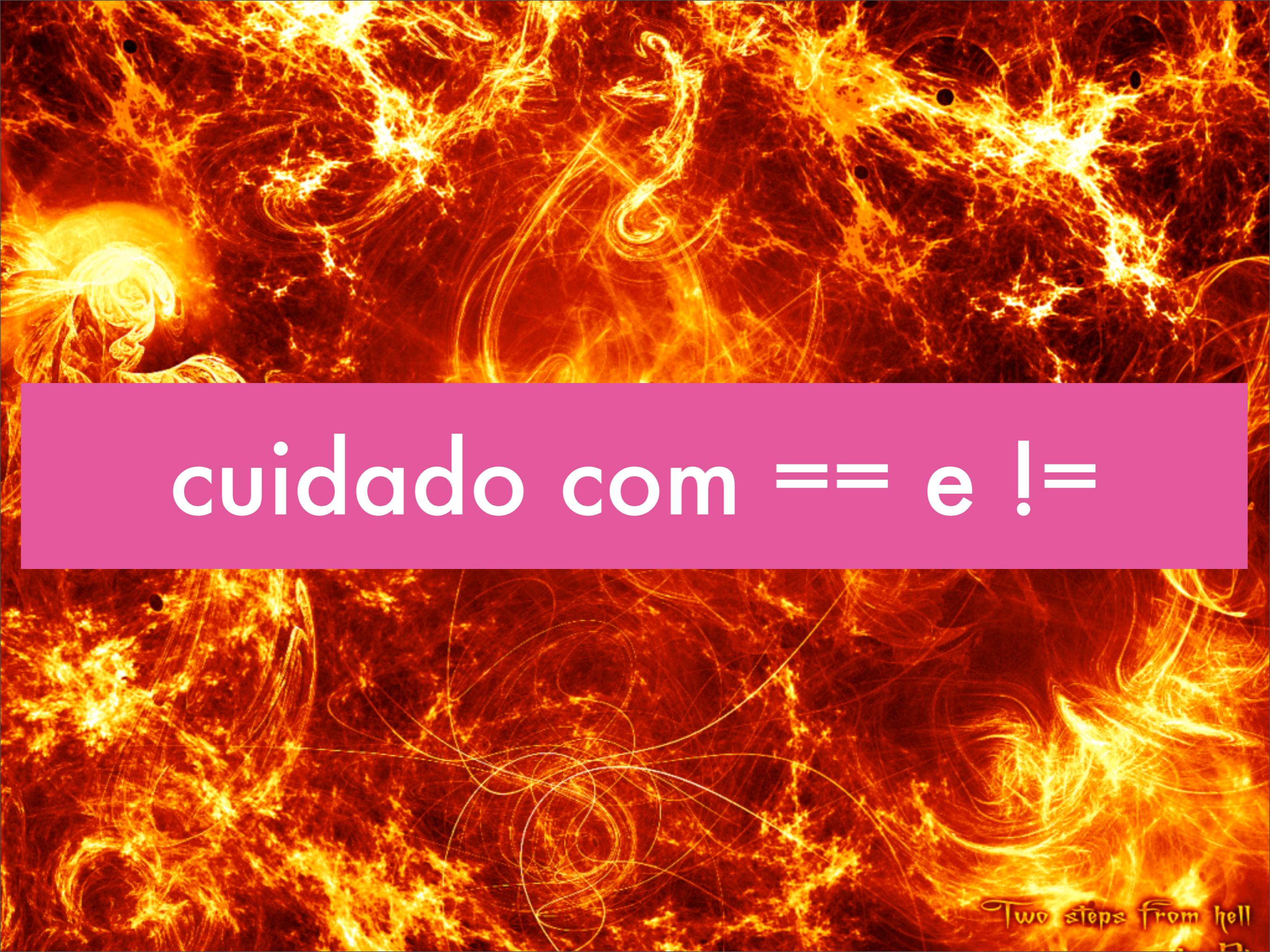


- Exemplo:

```
var saldo = 10.0;  
// Verificando se o saldo é suficiente  
if (saldo < 0) {  
    document.writeln ("Saldo insuficiente");  
} else {  
    document.writeln ("Operação realizada");  
}
```

Operadores Relacionais

- `>` Maior
- `>=` Maior ou igual
- `<` Menor
- `<=` Menor ou igual
- `==` Igual
- `!=` Diferente
- `====` Igual
- `!==` Diferente



cuidado com == e !=



```
' ' == '0'          // false
0 == ''            // true
0 == '0'           // true
false == 'false'   // false
false == '0'        // true
false == undefined // false
false == null       // false
null == undefined  // true
' \t\r\n ' == 0     // true
```



SEMPRE use === e !=

====

```
' ' == '0'          // false
0 == ''             // false
0 == '0'            // false
false == 'false'    // false
false == '0'         // false
false == undefined // false
false == null       // false
null == undefined  // false
' \t\r\n ' == 0     // false
```

Operadores Lógicos

- `&&` e lógico (AND)
- `||` ou lógico (OR)



CURTO CIRCUITO



Curto Circuito

- Os operadores lógicos têm curto circuito
- Assim que for possível inferir o valor da expressão, sua avaliação é interrompida
- **&& :**
 - Assim que um valor **false** for encontrado toda a expressão será **false**
- **|| :**
 - Assim que um valor **true** for encontrado toda a expressão será **true**

WHILE

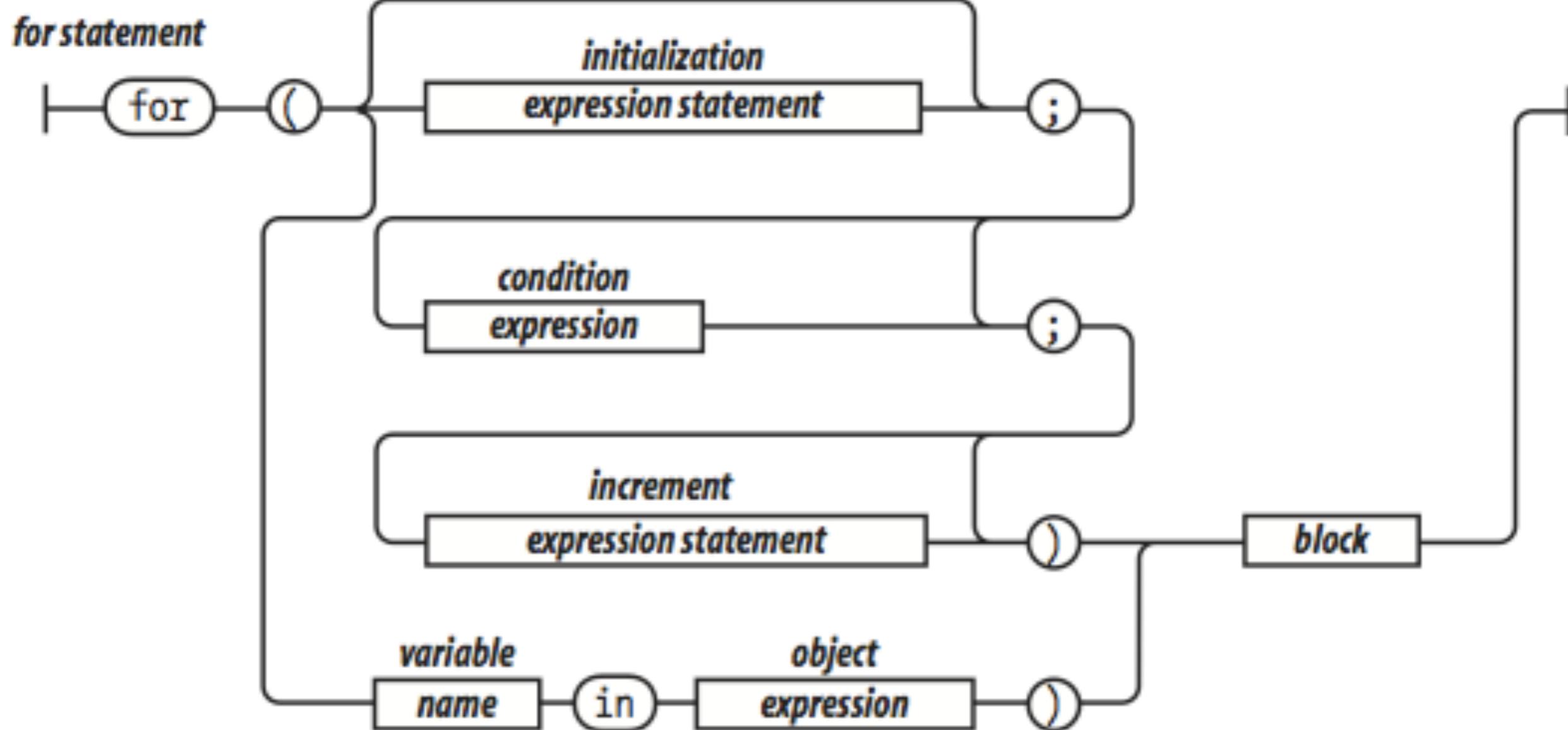
while statement



- Exemplo:

```
var contador = 10;  
// Contagem regressiva  
while (contador >= 0) {  
    document.writeln(contador);  
    contador--;  
}
```

FOR



FOR

- Exemplo:

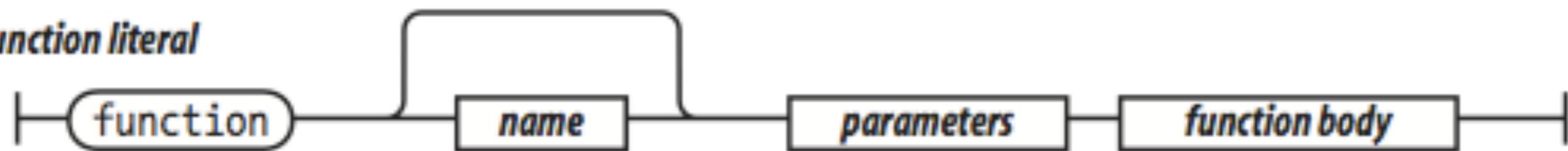
```
for (var i = 0; i < 100; i++) {  
    document.writeln(i);  
}
```

- Exemplo:

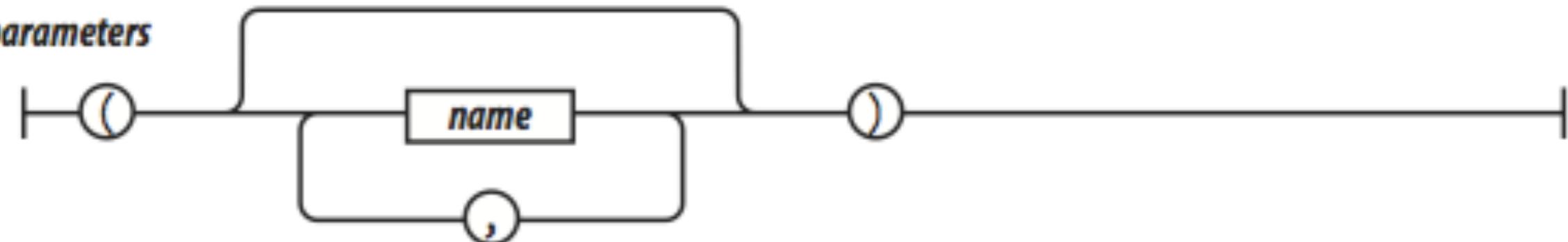
```
var nomes = [ 'fulano', 'cicrano', 'beltrano' ];  
  
for (var indice in nomes) {  
    document.writeln(nomes[indice]);  
}
```

FUNÇÕES

function literal



parameters



function body



FUNÇÕES

- Exemplo:

```
function soma(a, b) {  
    return a + b;  
}
```

- Exemplo:

Uma variável com uma função !

```
var maior = function(a, b) {  
    if (a > b) {  
        return a;  
    } else {  
        return b;  
    }  
}
```

OBJETOS

- Em JavaScript objetos guardam um conjunto de duplas chave-valor
- Por exemplo, um objeto vôo pode ter a seguinte chave: horário, com o valor: “18:30”

OBJETOS

- Objeto vazio:

```
var objeto = {}
```

- Objeto com propriedades:

```
var candidato = {  
    nome: "Manoel da Silva",  
    idade: 28,  
    terceiroGrau: true,  
    cargoPretendido: "Analista de Sistemas"  
}
```

OBJETOS

- Podem ser complexos:

```
var voo = {  
    companhia: "Oceanic",  
    numero: 815,  
    partida: {  
        aeroporto: "SYD",  
        horario: "22/09/2004 14:55",  
        cidade: "Sydney"  
    },  
    chegada: {  
        aeroporto: "LAX",  
        horario: "23/09/2004 10:42",  
        cidade: "Los Angeles"  
    }  
}
```

Acessando valores

- candidato[“nome”]
- candidato.nome
- voo.partida.horario

Atribuindo valores

- candidato.idade = 29
- candidato['aprovado'] = false
- voo.situacao = 'Desaparecido'

CONSTRutores

- Um construtor é uma função normal que cria um novo objeto utilizando a variável implícita `this`
- Um construtor deve ser sempre chamado utilizando o comando `new`

CONSTRutores

```
function Conta(titular) {  
    this.titular = titular;  
    this.saldo = 0;  
  
    this.depositar = function(valor) {  
        this.saldo += valor;  
    };  
    this.sacar = function(valor) {  
        this.saldo -= valor;  
    };  
    this.transferir = function(outraConta, valor) {  
        this.sacar(valor);  
        outraConta.depositar(valor);  
    }  
}
```

CONSTRutores

```
var c1 = new Conta("Pedro");
var c2 = new Conta("Maria");

c1.depositar(100);
c2.depositar(50);
c1.transferir(c2, 60);

document.writeln(c1.titular + " - R$ " + c1.saldo);
document.writeln(c2.titular + " - R$ " + c2.saldo);
```

PROTOTYPE

- Todos os objetos estão ligados a um prototype
- Os objetos heram suas propriedades deste prototype
- Alterando-se o prototype, altera-se todos os objetos ligados a ele
- Todos os prototypes herdam de Object.prototype

PROTOTYPE

```
Conta.prototype.toString = function() {  
    return this.titular + " - R$ " + this.saldo;  
};  
  
document.writeln(c1.toString());  
document.writeln(c2.toString());
```

PROTOTYPE

```
Number.prototype.dinheiro = function() {  
    var valor = "" + this.toFixed(2);  
    valor = valor.replace(".", ",");  
    return "R$ " + valor;  
}
```

```
var salario = 649.67;  
document.writeln(salario.dinheiro());
```

REFLEXÃO

- **typeof**
 - Retorna um String com o tipo de um valor
 - Exemplo:

```
document.writeln(typeof Conta);
document.writeln(typeof c1);
document.writeln(typeof c1.titular);
document.writeln(typeof c1.saldo);
document.writeln(typeof c1.sacar);
```

REFLEXÃO

- É possível utilizar um `for` para percorrer as propriedades de um objeto:

```
document.writeln("Analizando o objeto c1 ...");
for (propriedade in c1) {
  if (typeof c1[propriedade] === 'function') {
    document.writeln(" Metodo: " + propriedade);
  } else {
    document.writeln(" Atributo: " + propriedade);
  }
}
```

REFLEXÃO

- E se eu quiser criar um método em todos os objetos chamado inspect, que irá mostrar os atributos e métodos dos objetos ?

REFLEXÃO

```
// Criação do método inspect
Object.prototype.inspect = function() {
    var str = "";
    for (propriedade in this) {
        var tipo = typeof this[propriedade];
        if (tipo === 'function') {
            str += "Método: " + propriedade + "\n";
        } else {
            str += "Atributo: " + propriedade + " (" + tipo + ")" + "\n";
        }
    }
    return str;
};

// Testando o método inspect
document.writeln("c1:\n" + c1.inspect());
document.writeln("c2:\n" + c2.inspect());
```

EXCEÇÕES

- O tratamento de exceções permite a notificação (lançamento) e o tratamento de situações de falha

EXCEÇÕES

- Lançamento de exceções:
 - throw valor
 - Exemplo:
 - • •
 - ```
this.sacar = function(valor) {
 if (valor > this.saldo) {
 throw "Saldo Insuficiente";
 }
 this.saldo -= valor;
};
• • •
```

# EXCEÇÕES

- Tratamento:

```
var c1 = new Conta("Pedro");

try {
 c1.sacar(500);
} catch (erro) {
 document.writeln("Erro: " + erro);
}
```



# Uma Boa Prática

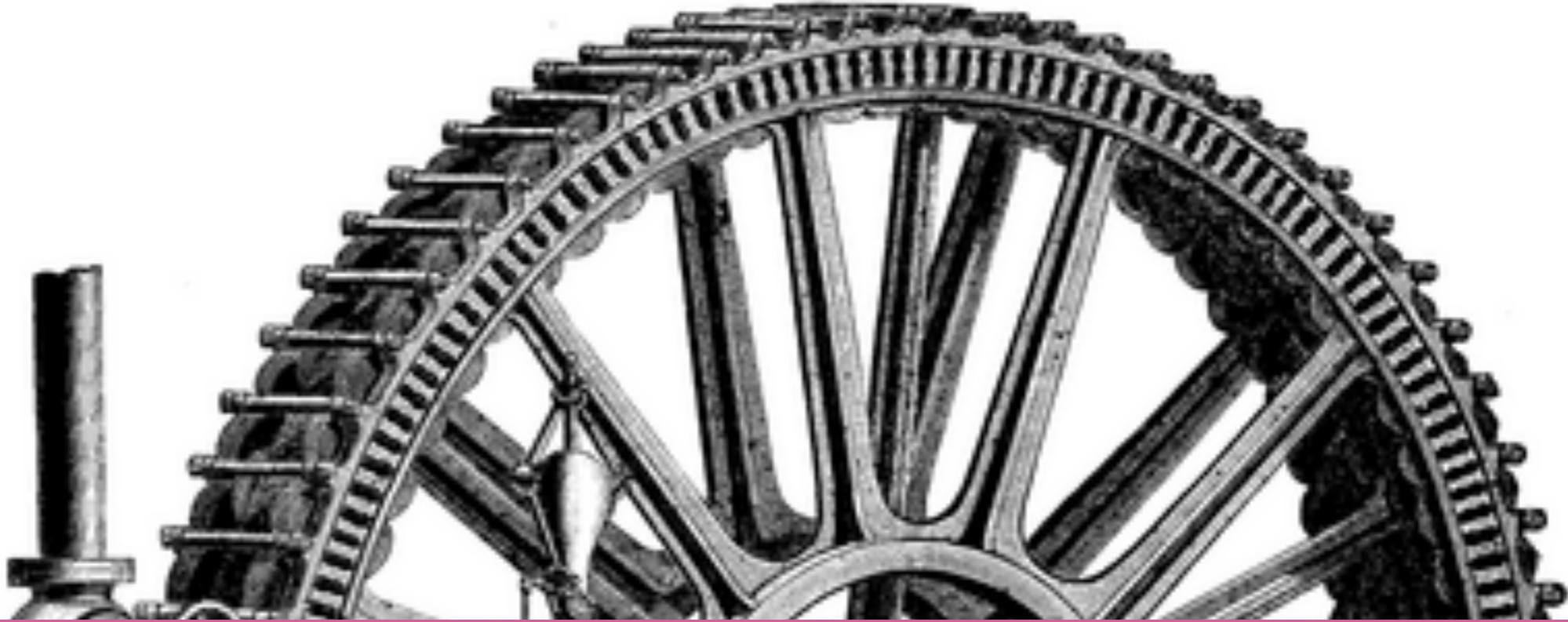


# Utilize Namespaces

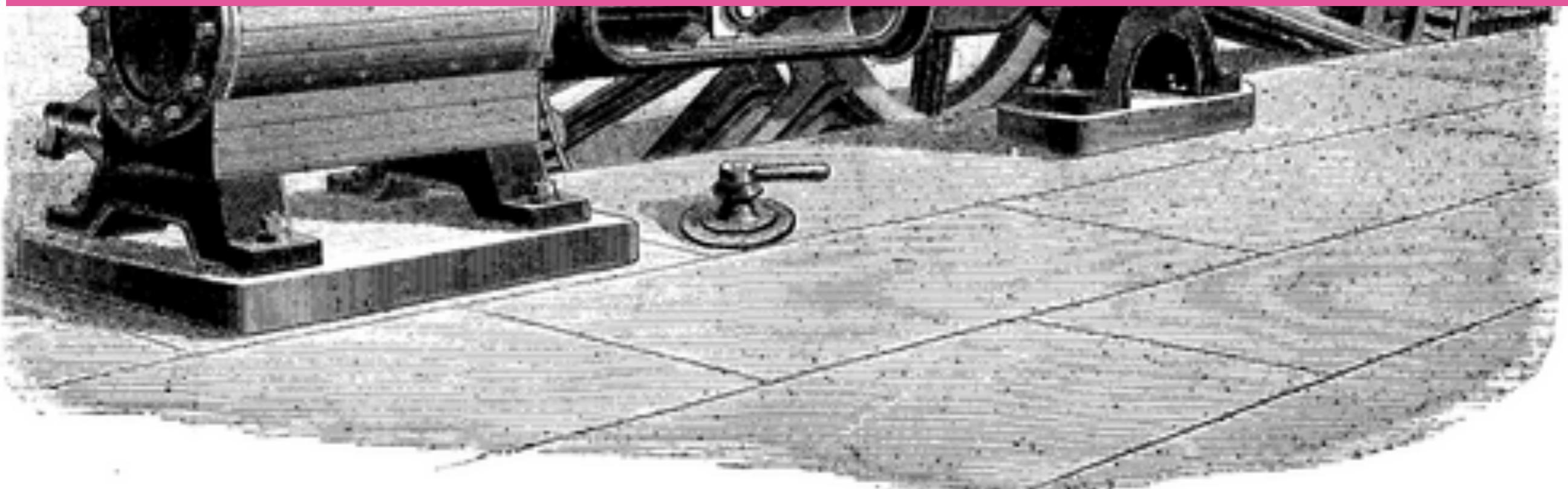
- Evite utilizar variáveis globais;
- Crie todas as suas variáveis, funções, objetos dentro de namespaces (containers);
- Exemplo:

```
var MINHA_APPLICACAO = {};

MINHA_APPLICACAO.versao = 2.0;
MINHA_APPLICACAO.Pessoa = function(nome, idade) {
 this.nome = nome;
 this.idade = idade;
};
MINHA_APPLICACAO.pessoas = new Array();
```



# Aplicações



# JQuery

jquery.com

- Biblioteca JavaScript mais utilizada
- Uma camada acima da API do browser
- Simplifica o acesso aos objetos das páginas



# Instalação

- Faça o download do jQuery e copie para seu projeto
- OU
- Utilize direto de uma CDN
  - *Ex: Google CDN*

# Utilizando a Google CDN

programa.html

```
<html>
 <body>
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js">
 </script>
 <script src="programa.js"></script>
 </body>
</html>
```

programa.js

```
$(function() {
 alert("oi");
});
```

# JQuery

- Ponto de entrada
- Todo script JQuery começa com um ponto de entrada que será executado apenas quando a página estiver pronta

```
$ (function() {
 // Ponto de entrada
});
```

# JQuery

- Os elementos de uma página são acessados em JQuery através de seletores
- Seletores:
  - id: \$('#id')
  - classe: \$('.classe')
  - tag: \$('tag')

# Atributos dos Elementos

- Um elemento pode ser modificado e consultado através dos seguintes métodos:
  - attr
  - removeAttr
  - val
  - text
  - html

# Eventos

- blur
- change
- click
- dblclick
- focus
- keydown
- keypress

# Calculadora

programa.html

```
<html>
 <body>
 A <input type="text" id="a">
 B <input type="text" id="b">
 <button id="mais">+</button>
 <input type="text" id="resultado">
 <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.1/jquery.min.js">
 </script>
 <script src="programa.js">
 </script>
 </body>
</html>
```

# Calculadora

programa.js

```
$(function() {
 $('#mais').click(function() {
 var a = parseFloat($('#a').val());
 var b = parseFloat($('#b').val());
 var c = a + b;
 $('#resultado').val(c.toFixed(2));
 });
});
```

# Calculadora

programa.js

```
var CALCULADORA = {};

CALCULADORA.somar = function(a, b) {
 var c = a + b;
 return c.toFixed(2);
};

$(function() {
 $('#mais').click(function() {
 var a = parseFloat($('#a').val());
 var b = parseFloat($('#b').val());
 $('#resultado').val(CALCULADORA.somar(a, b));
 });
});
```

# O que não vimos

- Regex
- Outras bibliotecas
  - ExtJS
  - YUI
- NodeJS