

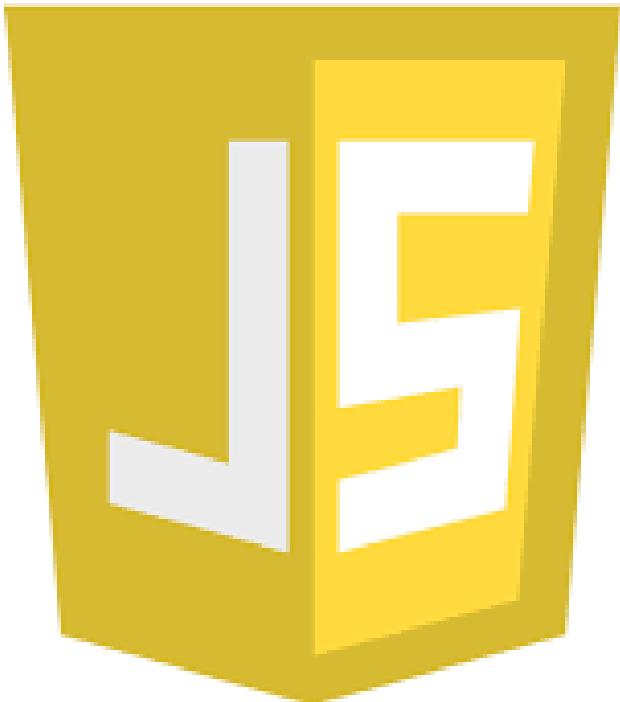
```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Laboratório de Programação

Aula 8

Introdução à JavaScript

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```



JavaScript

```
.ui-helper-hidden-accessible
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

O que é JavaScript?

- A especificação ECMAScript descreve:
"Uma linguagem vibrante e ela não para de evoluir, avanços técnicos significativos continuaram aparecendo"
- linguagens de script são usadas para manipular, customizar e automatizar os recursos de um dado sistema

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

O que é JavaScript?

- JavaScript é orientado a objeto. Recursos do hospedeiro são providenciados por objetos e um programa é um cluster de objetos comunicantes
- Objetos são uma coleção de propriedades cujo tipos podem ser outros objetos, tipos primitivos ou funções

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

História

- 1994 – Lançamento do (Mosaic) Netscape
 - Era necessária uma linguagem que auxiliasse desenvolvedores
- Nome original: Mocha
 - Criada em 10 dias (!) por Brendan Eich, em Maio/95
 - Escolhido por Marc Andressen, fundador da NetScape
- Setembro/95
 - Livescript
- Dez/95

• JavaScript (jogada de marketing!)

• Lançada junto com o Netscape 2.0

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

História

- 1996
 - Jscript (Microsoft) – versão do Javascript para IE
 - Problemas de compatibilidade
- 96-97
 - ECMA (European Computer Manufacturers Association)
 - Padrão ECMAScript
- 98 – ECMA2
- 99 – ECMA3
- ECMA4 – Projeto Abandonado

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

História

- 2000 – JSON Douglas Crockford
- 2005 – Ajax Jesse James
- 2006 – Jquery
- 2009 – ECMA5, strict
- 2009 – Node.js
- 2010 – NPM

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

História

- 2010 – Underscore
- 2011 – ECMA5.1
- 2012 – Bluebird
- 2014 – Babel
- 2015 – ECMA6

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

Processo de Especificação TC39

- Comitê para especificação das novas versões para o EcmaScript
- Pós ES6, para manter um ritmo crescente de aprimoramento
- Realizado de forma comunitária, cada proposta passa por 5 níveis de maturidade: Rascunho, Proposta, Preliminar, Candidato e Estável

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

No Navegador

- Os scripts JavaScript são inseridos nos documentos HTML utilizando a tag `<script>`
- Um script JS pode ser definido diretamente entre a tag, ex:
`<script> alert('Olá'); </script>`
- Ou ser oriundo de um arquivo externo, usualmente com a extensão .js, ex:

```
.ui-helper-hidden-accessible {
border: 0; <script src="meu-script.js"></script>
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        elem.  
        elem.length,  
        callbackInvert = !invert;  
    if ( typeof invert === 'function' ) {  
        callback = invert;  
        invert = undefined;  
    }  
    if ( !callback ) {  
        callback = matchCallback;  
    }  
    if ( !elem ) {  
        elem = document.documentElement;  
    }  
    if ( !matches.length ) {  
        matches = elem.querySelectorAll( selector );  
    }  
    if ( invert ) {  
        matches = matches.filter( invert );  
    }  
    if ( callback ) {  
        callback( matches );  
    }  
}
```

Run To Complete, Hoisting

- A ordem que os scripts são carregados importa. Se em um script anterior se fizer uma consulta à uma variável que só foi declarada e iniciada num segundo script, o valor não será encontrado
- As declarações de variáveis podem ser escritas em qualquer lugar no corpo de código, mas serão virtualmente interpretadas como se estivessem no topo

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Variáveis

- Variáveis
 - Variáveis não possuem tipo:
`var a = 123;`
`a = "agora sou uma string";`
 - O valor atribuído a elas possuem tipo. Mas as variáveis podem ser ocupadas por valores de qualquer tipo

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Variáveis

Var foi a declaração de variável, ela tem seu comportamento alterado se estiver num ambiente de execução 'use strict'. Além dela existem outra 2:

- Let – A variável não pode ser redecladara
- Const – O valor da variável não pode ser alterado. Se o valor for uma referência a um objeto, este pode ser alterado sem se modificar referência para ele.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Escopo

- O escopo é léxico, ou estático. As variáveis não encontradas no bloco de código atual são buscadas na região adjacentes especificadas no código fonte
- A variável *this* é mantida como referência ao contexto de execução, similar a um escopo dinâmico

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Objetos

Uma forma de declarar objetos é:

```
var obj = {  
    nome : "João",  
    idade : "21",  
    curso : "SI"  
};
```

Os atributos de um objeto pode ser acessados utilizando:

- `obj.nome`, `obj.idade`, `obj.curso`
- `obj['nome']`, `obj['idade']`, `obj['curso']`

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Objetos

```
var aluno = {
  nome : "João",
  idade : "21",
  curso : "SI",
  correr : function () {
    if (this.idade <= 30) {
      console.log(this.nome+" Corre muito!");
    }
    else {
      console.log(this.nome+" Corre pouco!");
    }
  }
};
```

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Objetos

Cada atributo possui os seguintes descritores:

- **Value** – o valor do atributo
- **Writable** – o valor pode ser sobreescrito
- **Get/Set** – funções para modificar o valor
- **Configurable** – os descritores podem ser modificados
- **Enumerable** – o atributo é listável

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Herança

Os atributos não encontrados num objeto são buscados no próximo da cadeia prototípica.

Todo objeto possui um vínculo com outro objeto chamado prototype.

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Herança

```
var mae = {
  'nome': 'joana',
  'olhos': 2
}
```

```
var filha = Object.create(mae)
filha.nome = 'baderna'
```

```
.ui-helper-hidden-accessible
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
background-color: transparent;
border: none;
clip-path: inset(50%);">
  console.log(filha.olhos)
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Objetos especializados

- Array
- Date
- RegExp
- Null
- Number, String, Boolean
- Promise

- Map, Set

.ui-helper-hidden-accessible {
border: 0px solid black;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0px; position: absolute;
width: 1px;}

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Arrays

Usualmente, declaramos arrays em JS de duas formas:

```
var arr = ["a", "b", "c", "d"];
```

Os valores do array podem ser acessados utilizando: `arr[0]`, `arr[1]`

Todo array possui um atributo chamado `length`, que é não enumerável, e possui valor igual ao número de elementos deste vetor

O protótipo do array possui métodos como:

- `push()`, `pop()`, `sort()`, `slice()`, `join()`

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

JSON

É um formato de texto que possui sintaxe comum a várias linguagens da família do C. São objetos, vetores e tipos primitivos.

Podem ser transformados em objetos e viceversa com `JSON.parse` e `JSON.stringify`, respectivamente.

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

Navegadores

- Browsers possuem uma série de tarefas e recursos além do javascript, eles são responsáveis pela renderização da página html e css, além de toda forma de comunicação e gerenciamento de memória.
- Fornecem um ambiente de execução isolado e restrito para o javascript, sem acesso direto a outros processos ou subsistemas do usuário.

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Navegadores

- Window é o objeto global que oferece diversos recursos para serem manuseados pela linguagem.
 - Como o objeto navigator que possui dados sobre a geolocalização, velocidade de conexão e número de threads.
 - Ou o objeto indexeddb, que é um banco de dados não relacional utilizado para armazenar informações localmente no usuário

Navegadores

- Document Object Model é um objeto global que possui uma representação da estrutura do html e métodos convenientes para sua manipulação.
- É possível resgatar informações colocadas em campos input de formulário e até mesmo criar todo a página html em tempo de execução

Navegadores

- DOM possui eventos que podem disparar a execução de trechos específicos de código.
- Existem uma ampla variedade de eventos, de interação com usuários a notificações automáticas da renderização.
- O navegador executa a função correspondente a cada evento despertando o código de seu estado latente.

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Eventos

- Alguns exemplos de eventos:
 - ***onclick***: Disparado quando há um **click do mouse**
 - ***onchange***: Dispara quando há **alteração no valor**.
 - ***onload***: Disparado quando a **página é carregada**
 - ***onkeydown***: Disparado quando uma **tecla** do teclado é **pressionada**.
 - ***onmouseover***: Disparado quando o **ponteiro do mouse passa sobre** um objeto.
 - ***onfocus***: Disparado quando um **campo** de formulário **recebe o foco** (por exemplo o cursor é colocado em um campo de texto)

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Funções

- Funções podem receber parâmetros e fornecer um retorno.
Usa-se a palavra reservada `function`:

```
function soma(a,b) {  
    var c = a + b  
  
    return c  
}
```

- Podemos colocar uma função "dentro de uma variável", depois executar ou passar como parâmetro:

```
var soma = function (a,b) {  
    var c = a + b  
  
    return c
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Funções

```
var soma = function (a,b) {  
    var c = a + b  
    return c  
}  
  
var executaF = function (f, b) {  
    var a = 3  
    return f(a, b)  
}
```

```
var a = -Infinity  
executaF(soma, 0)
```

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Funções Setas

arrow function possuem a seguint sintaxe:

```
var soma = (a, b) => {  
    return a + b  
}
```

- Podem ser omitidas algumas partes

```
var doble = x => 2 * x
```

- *This* é local

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Argumento padrão

Argumentos das funções podem receber parâmetros padrões:

```
var soma = (a = 0, b = 0) => a + b
```

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Condicional

```
if (2 > 3) {
  console.log('2 > 3')
} else if (2 < 3) {
  console.log('2 < 3')
} else {
  console.log('2 === 3')
```

```
.ui-helper-hidden-accessible {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;}
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Condicional

O operador ternário é uma expressão similar ao if-else, mas ele retorna um valor

```
var res = 2 < 3 ? 'menor' : 'não menor'
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Laços

```
while (false) {
    console.log('nunca vai ser impresso')
}
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Laços

```
for (var i = 0; i < 10; i++) {
    console.log(`O valor de i é ${i}`)
}
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;
```

**Instituto de Computação
UFMT 2018**

**Programação em Ambiente Web I
Prof. Jivago Medeiros**

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Laços

```
for (const key in window) {
  console.dir(window[key])
}
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

Instituto de Computação
UFMT 2018

Programação em Ambiente Web I
Prof. Jivago Medeiros

Iteradores

- São objetos que implementam um protocolo de comunicação com valores e comportamento específicos
- Vetores são iteráveis.
- Geradoers são iteráveis.
- O laço for-of e o operador ... (spread) consomem iteráveis

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

Laços for-of

```
for (const el of ['a','e','i','o','u']) {
  console.log(el)
}
```

```
.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}
```

Instituto de Computação
UFMT 2018

Programação em Ambiente Web I
Prof. Jivago Medeiros

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        elem; if( invert ) {  
            elem = elems.length - 1;  
            callbackInverse = !invert;  
        } else {  
            elem = 0;  
            callbackInverse = invert;  
        }  
        for( ; elem < elems.length; elem++ ) {  
            if( callbackInverse ) {  
                if( !callback( elems[ elem ] ) )  
                    matches.push( elem );  
            } else {  
                if( callback( elems[ elem ] ) )  
                    matches.push( elem );  
            }  
        }  
        return matches;  
}
```

Laços *Map*, *Filter* e *reduce*

Vetores possuem métodos que iteram sob todos seus elementos

```
[0, 1, 2].map(x => x + 1)
```

```
[1, 2, 3].filter(x => x % 2)
```

```
[1, 3].reduce((a, b) => a + b, 0)
```

Laços Map, Filter e reduce

Ou aproveitando que o retorno do map e filter são outros vetores

```
[0, 1, 2].map(  
  x => x + 1  
).filter(  
  x => x % 2  
).reduce((a, b) => a + b, 0)
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Destruindo

É possível utilizar expressões para desempacotar valores de objetos.

```
var a, b, rest
```

```
[a, b, ...rest] = [1, 3, 5, 7]
```

```
var pessoa, rest
```

```
{ nome: pessoa, ...rest } =
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;}
```

```
{ nome: 'joão', idade: 120 }
```

Módulos ES6

Durante bom tempo não existia suporte nativo a módulos no javascript, haviam bibliotecas que implementavam essa funcionalidade

Atualmente é feito com import/export e podem ser dinâmicos

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Módulos ES6

Para importar o módulo:

```
<script type="module">  
    import xyz from './xyz.mjs'  
</script>
```

O módulo xyz.mjs:

```
export default function () {
```

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Promessas

São objetos que eventualmente podem possuir algum valor, eles possuem atributos que recebem funções que seriam executadas quando este valor for encontrado ou retornar algum erro

A função fetch retorna uma promessa

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

Instituto de Computação
UFMT 2018

```
grep: function( elems, callback, invert ) {  
    var callbackInverse,  
        matches = [],  
        i = 0,  
        length = elems.length,  
        callbackExpect = !invert;
```

Promessas

Promessa imediatamente resolvida:

```
var p1 = Promise.resolve('valor da  
promessa')  
  
p1.then(console.log)
```

```
.ui-helper-hidden-accessible {  
border: 0;  
clip: rect(0 0 0 0);  
height: 1px;  
margin: -1px;  
overflow: hidden;  
padding: 0;  
position: absolute;  
width: 1px;}
```

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Promessas

```
var p1 = new Promise(function (resolve, reject) {
    setTimeout(function () {
        if (Math.random() > 0.5) {
            resolve("OK")
        } else {
            reject("ops...")
        }
    }, 1000)
})

p1.then(console.log)
p1.catch(console.error)
```

.ui-helper-hidden
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
Instituto de Computação
UFMT 2018
position: absolute;
width: 1px;

```
grep: function( elems, callback, invert ) {
  var callbackInverse,
      matches = [],
      i = 0,
      length = elems.length,
      callbackExpect = !invert;
```

NodeJS



```
.ui-helper-hidden-accessible {
  border: 0;
  clip: rect(0 0 0 0);
  height: 1px;
  margin: -1px;
  overflow: hidden;
  padding: 0;
  position: absolute;
  width: 1px;}
```

**Instituto de Computação
UFMT 2018**

```
grep: function( elems, callback, invert ) {
    var callbackInverse,
        matches = [],
        i = 0,
        length = elems.length,
        callbackExpect = !invert;
```

Nodejs



V8 + LibUV

- Engine javascript do google chrome
- Biblioteca de eventos, usados para pacotes de rede e demais processos bloqueantes

.ui-helper-hidden-accessible {
border: 0;
clip: rect(0 0 0 0);
height: 1px;
margin: -1px;
overflow: hidden;
padding: 0;
position: absolute;
width: 1px;}

Instituto de Computação
UFMT 2018