



JavaScript A-Z

# Apresentar o JavaScript como uma linguagem.

- Parar de ver JavaScript como uma ferramenta de auxílio.
- Criar seus próprios scripts do zero.
- Ter a capacidade de entender o funcionamento de scripts de terceiros.

### Características básicas

- Load and go delivery
- Tipagem fraca
- Objetos são containers
- Herança por protótipos
- Lambda

### 6 tipos básicos

- Numbers
- Strings
- Booleans
- Objects
- null
- undefined

# Numbers

- Somente Double (64-bit float)
- NaN
  - Não é  $>$  nem  $<$  que outro NaN
  - Não é igual a NaN
  - O erro propaga através das expressões
  - O tipo do valor ainda é Number
  - $5 * \text{"texto"} = \text{NaN}$
  - `Number("texto")`

# Numbers

- Infinity
  - Dividir por 0
  - $5 * \text{Infinity} = \text{Infinity}$
  - $\text{Infinity} + \text{Infinity} = \text{Infinity}$
  - $\text{Infinity} - \text{Infinity} = \text{NaN}$
  - $\text{Infinity} / \text{Infinity} = \text{NaN}$
- Math

## Strings

- Sequência de 0 ou mais caracteres de 16-bits
- Chars são strings de tamanho 1
- São imutáveis
- Strings semelhantes são iguais
- São permitidos os usos de " e "" para literals

## null e undefined

- Null é o valor para nada
- Undefined é o valor default para membros inexistentes de um objeto, parâmetros de função não informados e variáveis não inicializadas



## Booleans

- true ou false
- Valores “falsos”: false, null, undefined, “”, 0, NaN
- Valores “verdadeiros”: todo o resto (incluindo “0”, “false” e instâncias de objeto)

# Objects

- São containers de chave-valor (similar a um hashtable)
- `new Object()` cria um container vazio (= {})
- Membros podem ser acessados de duas maneiras:
  - `objeto.membro`
  - `objeto['membro']`
- Membros podem ser valores ou métodos
- Podem derivar de outros objetos: `Object.create`

# Objects

- Literals e Nested Objects
- Membros podem ser adicionados on the fly
- Prototype
  - hasOwnProperty
  - NÃO PROTOTYPE OBJECT

# Arrays

- Derivam de object
- Vários tipos entre os elementos
- 0-based
- Construtores
  - `new Array(e1, e2, e3, ..., eN)`
  - `[e1, e2, e3, ..., eN]`

# Operadores

- + - \* / %
- == != < > <= >=
- && || !
- & | ^ >> >>> <<
- ? :

# Operadores

- "==" e "==="
- Guard operators "&&" e "||"
- Booleanizador "!!"

- typeof
- delete
- eval
- EVAL IS EVIL
- Date
- Regex
- SÉRIO!!! NÃO USE EVAL!!!!

## Statements

- If, switch, while, do, for, break, continue, return, try/throw
- For in
- Breaks nomeados



### Escopo

- {bloco}
- Function
- var

- Objetos de primeira classe
- Podem ser transitadas igual aos outros valores
- Derivam de Object
- Statement básico
- Named Statement
- Lambda
- Lexical Scope
- Closure

- `this`
- Parametros
  - Objetos passados como referência
  - Não é necessário passar a mesma quantidade de argumentos requeridos
  - Não há validação do tipo do argumento
  - Array de parâmetros (arguments)

- “Maker/Builder” functions
- Constructor functions
  - Operador new
  - Retorna this por default
- Chamando funções
  - Method form
  - Call e Apply

- Global Objects
- Namespaces

NÃÃO USE EVAL!!!!!!!!!!!!

- Bootcamp **Desenvolvendo Aplicações WEB: Client-Side**
  - jQuery
  - KnockoutJS
  - Bootstrap
  - DataTables
- Coding DOJO **Desenvolvendo Aplicações WEB: Client-Side**



<https://github.com/ITLab-Academy>





<http://goo.gl/forms/hSs61zgkGT>