Oficina de **JavaScript**

Criando uma aplicação em JS

Leonardo Trindade
Leonardo Steil
Luis Henrique Medeiros
Marinara Rübenich
Matheus Dalmolin

Sumário

1. Um pouco da história do JavaScript
2. Porquê utilizar JavaScript?
3. Introdução a linguagem JavaScript
    1. Variáveis
    2. Operadores
    3. Loops
    4. Funções
    5. Ex.: Imperativo e Funcional
    6. Estruturado
    7. Ex.: Orientado a Objetos
4. Versões: ES5 vs ES6

Um pouco da história do JavaScript

1.

Porquê foi criado?

- Partiu da necessidade de explorar a Web que estava surgindo
- Pelos passos largos da concorrência
- Pela falta de dinamismo no acesso do cliente
- Pela lentidão nos envios e respostas de tarefas simples ao servidor
- Pela incompatibilidade entre os navegadores da época

Um pouco da história do JS

- Criada por Brendan Eich, a serviço da empresa Netscape (em apenas 10 dias)
- 1995 – 1ª versão para o Netscape Navigator
- Nomes:
  - Mocha
  - LiveScript
  - JavaScript
  - ECMAScript
- Os trabalhos em cima da normativa ECMA-262 se iniciaram em 1996

Um pouco da história do JS

- Versões:
  - ES1 – 1997
  - ES2 – 1998
  - ES3 – 1999
  - ES4 – 2008 (abandonada, início da versão Harmony)
  - ES5 – 2009
  - ES5.1 – 2011
  - ES6 – 2015 (1ª versão Harmony)
  - ES7 – 2016 (versão final Harmony)
  - ES8 – 2017
  - ES9 – 2018

Mas o que é de fato JavaScript?

2.

JavaScript?

- Padronizada e funciona em todos os navegadores
- Leve e amigável tanto para usuários quanto para desenvolvedores
- Está em constantes melhorias e atualizações
  - Pelo menos uma vez ao ano
- É interpretada pelos navegadores
  - Cada navegador possui seu próprio Enguine (Motor), eles definem a velocidade com que o navegador vai interpretar o JS

Motores JS


V8 (Chrome e Opera)

SpiderMonkey (Firefox)


Chakra (IE e Edge)

SquirrelFish/Nitro (Safari)

Multiparadigmas

- É uma linguagem de programação **multi-paradigmas**
  - **Paradigma:** padrão de raciocínio para resolução dos problemas baseado nas funcionalidades que cada linguagem de programação nos fornece. JS suporta vários paradigmas, os principais são:
    - Funcional
    - Imperativo
    - Orientado a Objetos

Paradigmas Funcional e Imperativo

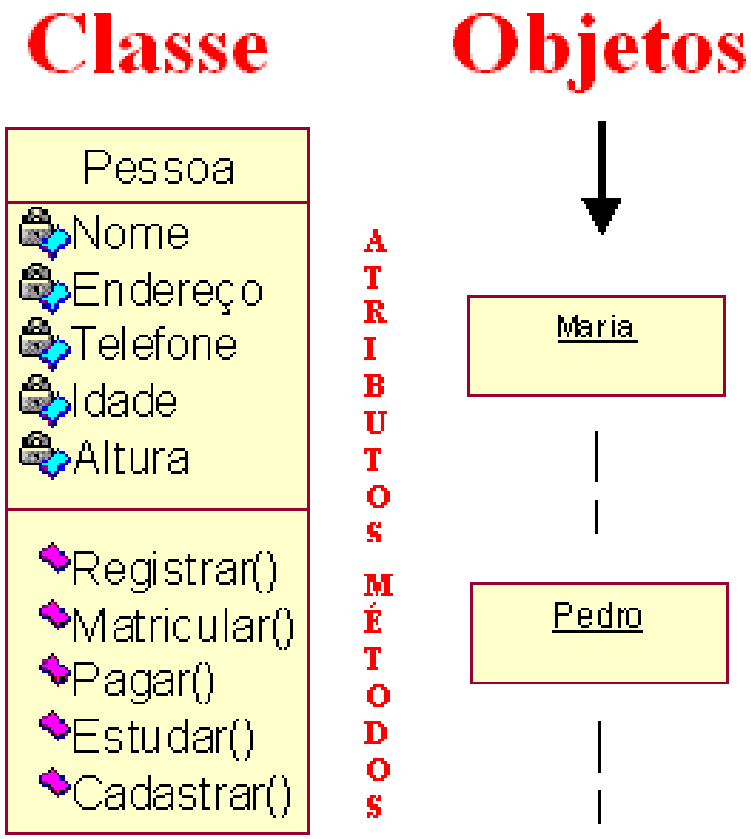
- Imperativo:
  - Programar o código de dizendo passo a passo dizendo COMO o computador deve executá-lo
  - Ocorrem diversas interações até chegarmos a um valor final

- Funcional:
  - Resolver qualquer problema através da execução de funções em Objetos
  - De mais difícil entendimento, não explícita e por isso mais suscetível a erros

Paradigma Orientado a Objetos

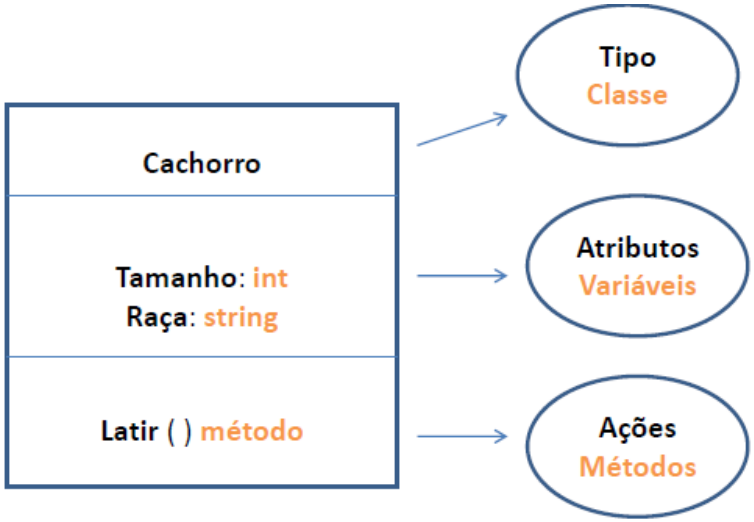
- Orientado a Objetos:
  - Tem o objetivo de aproximar o mundo real do mundo virtual
  - Através da “objetificação”, afinal nosso mundo é composto por objetos
  - Nos dá: maior reutilização do código, menos linhas de código, organização
  - Porém é bem mais complexo de entender

Orientação a Objetos



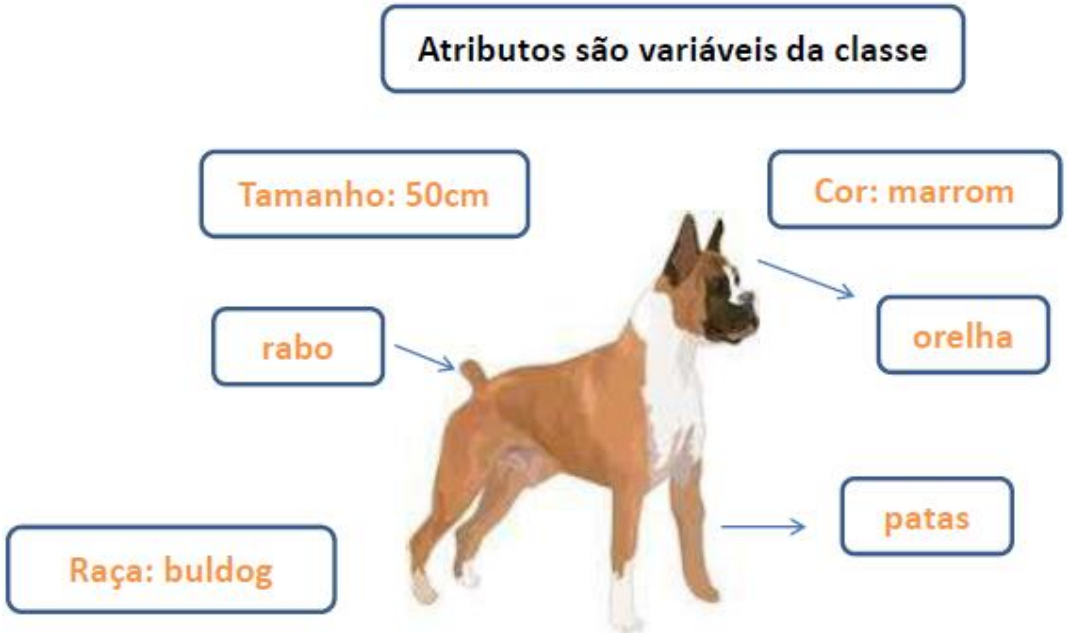
OO - Classe

- Classe:
  - Representa um conjunto de Objetos com características semelhantes



OO - Objeto

- Instância de uma Classe
- Existem um, ou vários deles dentro de cada classe e todos possuem os mesmos atributos da classe, inclusive também podem ter seus próprios atributos



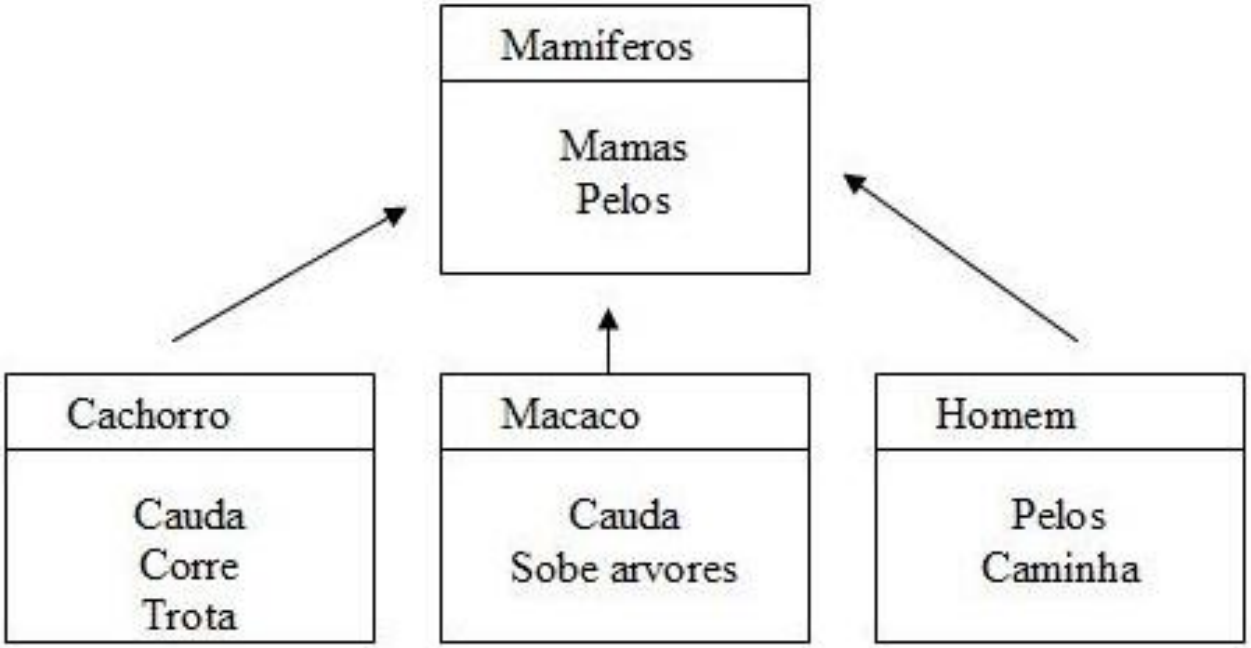
OO - Método

- Ações que a classe dispõe

OO – Herança e Polimorfismo

- Herança:
  - Quando uma outra classe (subclasse) herda os comportamentos de uma superclasse
  - Cada subclasse também poderá ter seus próprios atributos
- Polimorfismo:
  - É a capacidade que os objetos de diferentes classes tem para agir de formas distintas, mesmo possuindo métodos herdados da superclasse

OO – Herança e Polimorfismo



OO - Encapsulamento

- Encapsulamento:
  - Serve para proteger as propriedades de um objeto, pois evita o acesso direto aos mesmos
  - Disponibiliza métodos que os acessem

Introdução a linguagem JavaScript

3.

Variáveis

- Pode assumir qualquer valor
- Fracamente tipada:
  - minhaVar = “OficinaJS” // a variável é uma String
  - minhaVar = 123 // agora um número inteiro (int)
  - minhaVar = 123.1 /* um número real (float) */
  - minhaVar = [1, 2, 3] /* agora um array */
  - minhaVar = true // um booleano (true/false)

Operadores de Atribuição

- Atribuem valores a minha variável

Operador	Atribuição	Resulta
=	x = y	x = 3
+=	x = x + y	x = 12
-=	x = x - y	x = 6
*=	x = x * y	x = 27
/=	x = x / y	x = 3
%=	x = x % y	x = 0

- X = 9;
- Y = 3;

Operadores de Comparação

Operador	Descrição	Resulta
==	x == 5	TRUE
	x == "5"	TRUE
===	x === 5	TRUE
	x === "5"	FALSE
!=	x != 8	TRUE
!==	x !== 5	FALSE
	x !== "5"	TRUE
>	x > 8	FALSE
<	x < 8	TRUE
>=	x >= 8	FALSE
<=	x <= 8	TRUE

- X = 5

Operadores Lógicos

Operador	Descrição	Resulta
&&	E (and)	(x < 10 && y > 1) é TRUE (verdade)
||	OU (or)	(x == 5 || y == 5) é FALSE (falso) (x == 9 || y == 5) é TRUE
!	NÃO (not)	!(x == y) é TRUE

- X = 9;
- Y = 3;

Loops

- Tipos:
  - Os loops mais utilizados são:
    - FOR - percorre um bloco de código quantas vezes for necessário até a condição ser alcançada
    - WHILE - percorre um bloco de código apenas se, ou enquanto uma condição especificada é VERDADEIRA
    - DO / WHILE - também percorre um bloco de código pelo menos uma vez, mesmo que uma condição especificada seja FALSA

Função

- É um bloco de código responsável por executar uma determinada tarefa
- Esse bloco tem uma certa função no programa
- Permite o reuso do código e organização

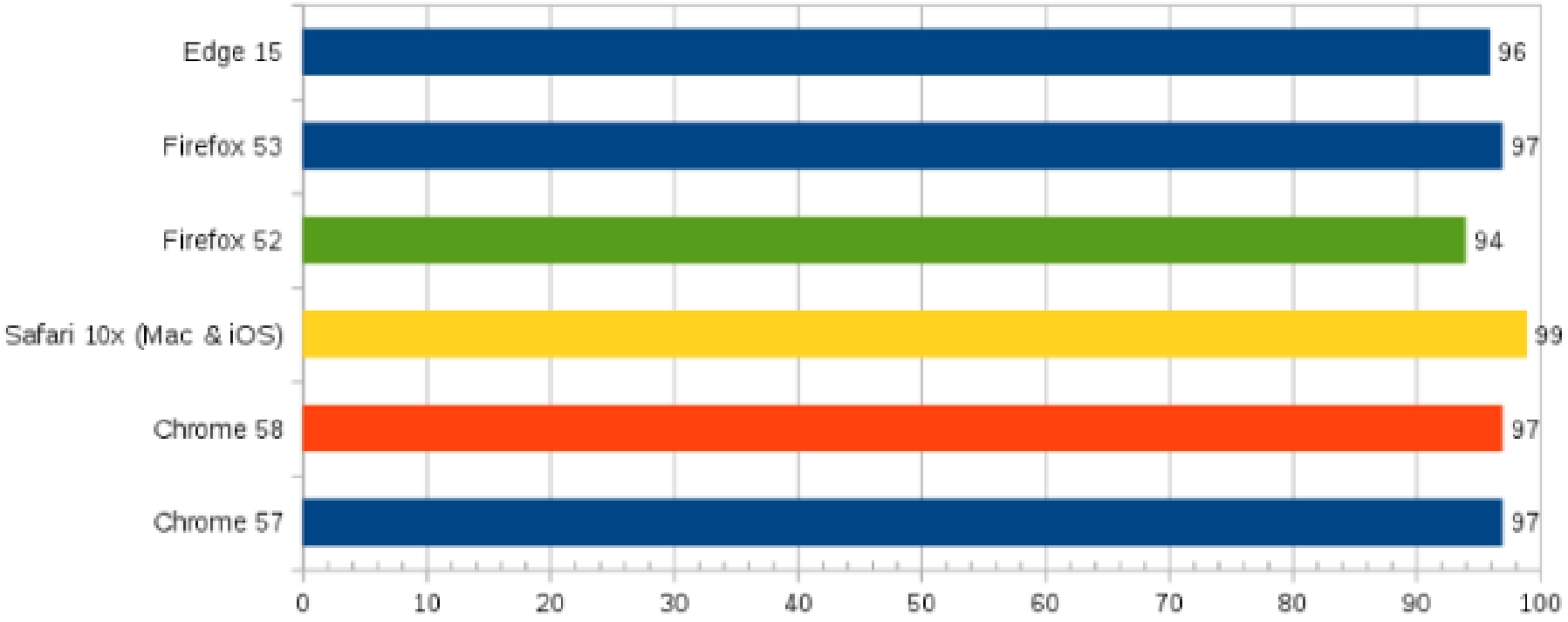
Versões:
ES vs ES6

4.

ES5 vs ES6

- Todos os navegadores interpretam a versão ES5
- Porém, nem todos suportam a versão ES6
  - Neste caso a solução é “transpilar” o código
    - Transpilar é uma mistura de Compilar e Traduzir
    - Transforma seu código escrito na versão ES6 em um código versão ES5

Compatibilidade dos Navegadores



Contatos

pet-si@inf.ufsm.br

www.fb.com/pet.si.ufsm

@petsiufsm

www.ufsm.br/pet-si