

JavaScript (Parte 3)

Esdras Lins Bispo Jr.
bispojr@ufg.br

Física para Ciência da Computação
Bacharelado em Ciência da Computação

25 de outubro de 2016

Plano de Aula

- 1 Revisão
- 2 Herança, Construtores e Protótipos
 - Herança
 - Construtores
 - Protótipo
- 3 Conceitos Básicos em JavaScript

Bônus (0,5 pt)

Desafio

(Halliday 3.23) O oásis B está 25 m a leste do oásis A. Partindo do oásis A, um camelo percorre 24 m em uma direção 15° ao sul do leste e 8,0 m para o norte. A que distância o camelo está do oásis B?

Informações úteis

- Candidaturas (25 de outubro, 17h20);
- Resposta escrita e apresentação (27 de outubro, 19h00).

Sumário

- 1 Revisão
- 2 Herança, Construtores e Protótipos
 - Herança
 - Construtores
 - Protótipo
- 3 Conceitos Básicos em JavaScript

Funções e Métodos

Atribuindo uma função como propriedade de um objeto...

```
1 nomeDoObjeto.nomeDaPropriedade = nomeDaFuncao;
```

Funções e Métodos

Atribuindo uma função como propriedade de um objeto...

```
1 nomeDoObjeto.nomeDaPropriedade = nomeDaFuncao;
```

Exemplo

```
1 obj.multiplicar = multiplicar;
```

Sumário

- 1 Revisão
- 2 Herança, Construtores e Protótipos
 - Herança
 - Construtores
 - Protótipo
- 3 Conceitos Básicos em JavaScript

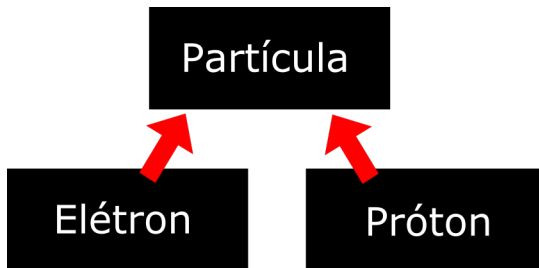
Herança

Partícula

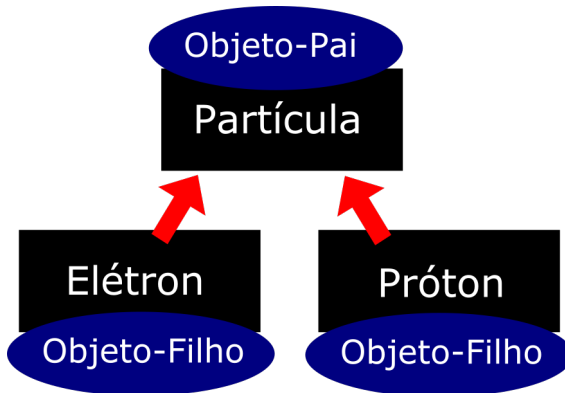
Elétron

Próton

Herança



Herança



Construtor

Objeto a partir de uma função

É possível construir um objeto a partir de uma “função-modelo”.

```
1 function Particula(pnome){  
2   this.nome = pnome;  
3   this.mover = function(){  
4     console.log(this.nome + " em movimento!");  
5   };  
6 }
```

Construtor

Objeto a partir de uma função

```
1 partícula1 = new Particula("eletron");  
2 console.log(partícula1.nome); //exibe "eletron"  
3 partícula1.mover();  
4 //exibe "eletron em movimento"
```

Construtor

Objeto a partir de uma função

```
1 partícula1 = new Particula("eletron");  
2 console.log(partícula1.nome); //exibe "eletron"  
3 partícula1.mover();  
4 //exibe "eletron em movimento"
```

Construtor...

Se uma função é utilizada intencionalmente para criar novos objetos, a chamamos de **construtor**.

Protótipo

Adicionando propriedades...

Após declarado, é possível adicionar propriedades a um construtor.

```
1 Particula.prototype.massa = 1;  
2 Particula.prototype.parar =  
3   function(){console.log("Eu parei.");};
```

Protótipo

Adicionando propriedades...

Após declarado, é possível adicionar propriedades a um construtor.

```
1 Particula.prototype.massa = 1;  
2 Particula.prototype.parar =  
3   function(){console.log("Eu parei.");};
```

Para novos objetos...

```
1 particula2 = new Particula("proton");  
2 console.log(particula2.massa); //exibe 1
```

Protótipo

Entretanto...

```
1 partícula3 = new Particula("neutron");  
2 partícula3.massa = 2; //propriedade com valor 2  
3 console.log(Particula.prototype.massa); //exibe 1
```

Necessário ter em mente...

A propriedade prototype modifica apenas o objeto-pai.

Protótipo

Algo interessante...

É possível adicionar propriedades apenas para o objeto-filho.

```
1 partícula4 = new Partícula("neutrino");  
2 partícula4.spin = 0; //exibe 0
```

Exemplo...

Objeto Bola

```
1 function Bola (raio , cor) {  
2   this.raio = raio;  
3   this.cor = cor;  
4   this.x = 0;  
5   this.y = 0;  
6   this.vx = 0;  
7   this.vy = 0;  
8 }
```

Exemplo...

Objeto Bola

```
1 Bola.prototype.desenhar = function (contexto) {  
2     contexto.fillStyle = this.cor;  
3     contexto.beginPath();  
4     contexto.arc(this.x, this.y,  
5                 this.raio, 0, 2*Math.PI, true);  
6     contexto.closePath();  
7     contexto.fill();  
8 };
```

Exemplo...

HTML Canvas arc

A propriedade `arc()` cria um arco
(utilizado para criar círculos ou parte de círculos).

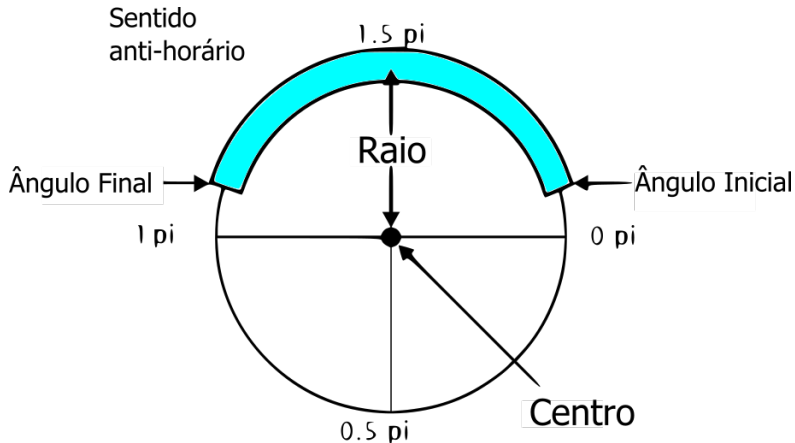
Sintaxe

```
arc(x, y, raio, angIni, angFin, antiHor);
```

Argumentos

- `x` e `y`: coordenadas do arco;
- `raio`: raio do arco;
- `angIni` e `angFin`: ângulos inicial e final do arco (em radianos);
- `antiHor`: valor booleano para o sentido anti-horário.

Exemplo...



Exemplo...

Objeto Bola

```
1 var canvas = document.getElementById('canvas');  
2 var contexto = canvas.getContext('2d');  
3 var bola = new Bola(50, '#0000ff');  
4 bola.x = 100;  
5 bola.y = 100;  
6 bola.desenhar(contexto);
```

Sumário

- 1 Revisão
- 2 Herança, Construtores e Protótipos
 - Herança
 - Construtores
 - Protótipo
- 3 Conceitos Básicos em JavaScript

Conceitos Básicos em JavaScript

Variáveis

Uma **variável** funciona como uma caixa que armazena valores de dados.

Conceitos Básicos em JavaScript

Variáveis

Uma **variável** funciona como uma caixa que armazena valores de dados.

Exemplo

```
1 var x ;  
2 x = 2 ;  
3 var y = 3 ;  
4 z = 2*x + y ; //z - variavel global  
5 x = x + 1 ;
```

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;
- i.e., podem armazenar tipos de dados diferentes em momentos diferentes.

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;
- i.e., podem armazenar tipos de dados diferentes em momentos diferentes.

Descrição

- Number: número de ponto flutuante com dupla precisão de 64 bits.

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;
- i.e., podem armazenar tipos de dados diferentes em momentos diferentes.

Descrição

- Number: número de ponto flutuante com dupla precisão de 64 bits.
- String: uma sequência de caracteres de 16 bits.

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;
- i.e., podem armazenar tipos de dados diferentes em momentos diferentes.

Descrição

- Number: número de ponto flutuante com dupla precisão de 64 bits.
- String: uma sequência de caracteres de 16 bits.
- Boolean: tem dois valores possíveis: true e false, ou 1 e 0.

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;
- i.e., podem armazenar tipos de dados diferentes em momentos diferentes.

Descrição

- Number: número de ponto flutuante com dupla precisão de 64 bits.
- String: uma sequência de caracteres de 16 bits.
- Boolean: tem dois valores possíveis: true e false, ou 1 e 0.
- Undefined: é retornado para uma propriedade de objeto não-existente ou uma variável sem um valor.

Conceitos Básicos em JavaScript

Tipos de Dados

- Variáveis em JavaScript têm tipos de dados dinâmicos;
- i.e., podem armazenar tipos de dados diferentes em momentos diferentes.

Descrição

- Number: número de ponto flutuante com dupla precisão de 64 bits.
- String: uma sequência de caracteres de 16 bits.
- Boolean: tem dois valores possíveis: true e false, ou 1 e 0.
- Undefined: é retornado para uma propriedade de objeto não-existente ou uma variável sem um valor.
- Null: tem apenas um valor → null

Conceitos Básicos em JavaScript

Descrição

- Object: armazena uma coleção de propriedades e métodos.

Conceitos Básicos em JavaScript

Descrição

- Object: armazena uma coleção de propriedades e métodos.
- Array: um objeto consistindo de uma lista de dados de algum tipo.

Conceitos Básicos em JavaScript

Descrição

- Object: armazena uma coleção de propriedades e métodos.
- Array: um objeto consistindo de uma lista de dados de algum tipo.
- Function: um objeto passível de ser chamado, que executa um bloco de código.

JavaScript (Parte 3)

Esdras Lins Bispo Jr.
bispojr@ufg.br

Física para Ciência da Computação
Bacharelado em Ciência da Computação

25 de outubro de 2016