Pós-graduação em Desenvolvimento Web e Aplicativos Móveis

Javascript

Fábio Rodrigues Jorge
fabinhojorgenet@gmail.com

https://github.com/fabinhojorge/aula-javascript

Agenda

- Closure
  - Module Pattern
- Escopo e Hoisting
- Funções - Avançado
- Orientação a Objeto
  - Prototype

Closures

Closures

Closures são **funções** que referenciam as **variáveis** que foram criadas no mesmo escopo que ela.

Obs: Toda função em JS é uma closure

Closures

1º: Criar um contador.
Problema: Variável exposta no escopo global e compartilhada

2º: Trazer a variável para dentro do escopo da func.
Problema: Cada vez que executa ela recebe 0 (zero)

```
//1ª tentativa
var c=0;
function add(){
    return c++;
}

//2ª tentativa
function add(){
    var c=0;
    return c++;
}
```

closure.js

Closures

Solução:

- Criar um escopo por meio de uma IIFE
- Instanciar **var c**
- Retornar a função referenciando a variável que foi criada no mesmo escopo

```
add = function(){
    var c = 0;

    return function(){
        return c++;
    };

}();
```

closure.js

Closures

Existe um Design Patter chamado Module Pattern (ou Singleton) que usa Closures para criar métodos e atributos privados

Atributos e métodos **Privados**

Atributos e métodos **Públicos**

```
var Pessoa = (function (nome, peso, altura) {
    var nome = nome || "";
    var peso = peso || 0;
    var altura = altura || 0;

    var calculaIMC = function() {
        return peso/(altura * altura );
    };

    return {
      meuNomePublico: "Meu nome é "+nome+"!",
      falarIMC: function() {
          alert(calculaIMC());
      }
    };
})("Luiz", 85, 1.60);

Pessoa.falarIMC();
```

closure3_module_pattern.js

Hoisting

Hoisting

Hoisting = “levantamento” ou “suspender”

Hoisting é o processo que o interpretador do JS utiliza para Identificar variáveis e funções.

- Funções tem a declaração elevada para o começo

- Variáveis tem a declaração elevada, mas não a atribuição de valores

Hoisting

Exemplo:

```
function doExample() {
    var x = 10;
    console.log( x ); // undefined
    console.log( y ); // ERROR
}
```

Hoisting

Exemplo2:

```
function doSomething() {
    var x = 10;
    console.log( y ); // undefined
    console.log( z ); // undefined

    if ( true ) {
        var y = 5;
    }

    var z = x + 5;
    console.log( y ); // 5
    console.log( z ); // 15

}
```

Funções - Avançado

Funções

4 modos de chamar uma função:

- Função: `obj.sayHello();`
- Método: `sayHello();`
- Construtor: `new sayHello();`
- Apply e Call: `sayHello.apply( obj, [ p1, p2 ] );`

  `sayHello.call( obj,  p1,  p2 );`

Chamada de Função

```
function soma (n1, n2){
    return n1 + n2;
}

//Se não tiver return o valor retornado
//é undefined

//Função Anônima
var multiplica = function(n1, n2) {
    return n1 * n2;
}

soma(2, 8);       // 10
multiplica(2,3); // 6
```

Chamada de Método

```
var pessoa = {
    name:'Lucas',
    sayHello:function(){
        alert("Olá, eu sou o "+this.name);
    }
}

pessoa.sayHello();
```

chamada_metodo.js

Chamada de Construtor

```
function Pessoa(nome){
    this.name = nome;
    this.sayHello = function(){
        alert("Olá, eu sou o "+this.name);
    }
}

var p = new Pessoa("oi");
console.debug(p);
p.sayHello();
```

chamada_construtor.js

Apply e Call

Apply e Call são funções usadas para passar parâmetros a outras funções.

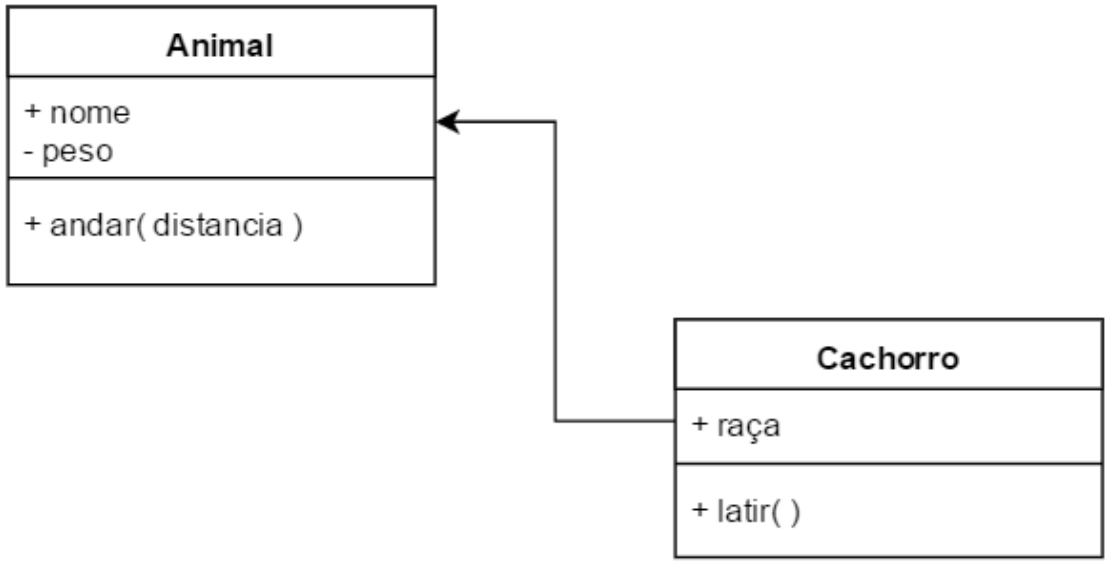
Elas alteram os valores do ***this***

```
function sayHello(outro){
    console.debug("Olá "+outro+", eu sou o "+
        this.name+" e tenho "+this.age+" anos");
}

name = "Lucas";
age = 15;

p1 = {name:"Paulo", age: 29};
p2 = {name:"Rodrigo", age: 33};

sayHello("Miguel");
sayHello.apply(p1, ["Miguel"]);
sayHello.call(p2, "Miguel");
```

chamada_apply_call.js

Orientação a Objetos

Orientação a Objetos



Orientação a Objetos

```
function Animal(nome, peso){
    this.nome = nome;
    this.peso = peso;
}

var a = new Animal("Rex", 20);

console.debug(a instanceof Object);
```

- E o método Andar?

orientacao_objetos.js

Orientação a Objetos

- Modificamos o protótipo e adicionamos **andar**.

```
Animal.prototype.andar = function(d){
    alert("Andando "+d+" metros!");
}

var a = new Animal("Rex", 20);

a.andar(150);
```

orientacao_objetos.js

Orientação a Objetos - Herança

- Herança é prototipal

```
//Herança
function Cachorro(raca){
    this.raca = raca;
}

Cachorro.prototype = new Animal();
Cachorro.prototype.latir = function(){
    alert("Au Au");
};

var c = new Cachorro("Dalmata");
c.andar(200);
c.latir();
```

orientacao_objetos.js

Dúvidas?

Referências

- BALDUINO, Plínio. “***Dominando JavaScript com jQuery***”. São Paulo, Casa do Código, 2012

- FLANAGAN, David. “***JavaScript: O Guia Definitivo***”. Editora Bookman, 6ª edição, 2012

- Introdução JS http://pt.slideshare.net/fernandosimeone/javascript-30043260

- Peculiariedades do JS http://leobetosouza.com.br/Palestra-Peculiaridades-do-JavaScript/#/

Links úteis

- Front-End: (http://willianjusten.com.br/como-se-tornar-um-desenvolvedor-front-end/)

- Front-End-Performance (https://github.com/davidsonfellipe/awesome-wpo)

- **Introdução (js4girls)** (https://github.com/Webschool-io/js4girls/blob/master/material-didatico/etapa-1/js-introduction.md)

- Performance Client Side (https://developer.yahoo.com/blogs/ydn/high-performance-sites-importance-front-end-performance-7160.html)

- **Linguagens que compilam para JS**
(https://github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js)

Links úteis

- IIFE – Chamada imediata de função (http://benalman.com/news/2010/11/immediately-invoked-function-expression/)

- **Escopo e Hoisting** (http://loopinfinito.com.br/2014/10/29/hoisting-e-escopo-em-javascript/)

- **26 técnicas de otimização de sites** (http://blog.caelum.com.br/por-uma-web-mais-rapida-26-tecnicas-de-otimizacao-de-sites/)

- Automatizar tarefa com GRUNT (https://zenorocha.com/automatizando-tarefas-js-com-grunt)

Links úteis

- Design Patterns para Javascript (https://scotch.io/bar-talk/4-javascript-design-patterns-you-should-know)

- Design Patterns (http://udgwebdev.com/design-patterns-para-javascript-parte-1)

- Eventos DOM (http://www.w3schools.com/jsref/dom_obj_event.asp)