Noções básicas de objetos – Objetos Literais

Desenvolvimento Front-end III

DFEIII

Gléderson L. dos Santos

gledersonsantos@ifsul.edu.br

Campus Charqueadas
Tecnologia em Sistemas para Internet

Introdução

- Na última aula revisitamos alguns conceitos de JavaScript e de orientação a objetos.

- Hoje iremos começar a trabalhar com os conceitos de orientação a objetos quando aplicados a linguagem de programação Javascript, a partir da forma de criação mais fundamental de um objeto nessa linguagem, os chamados objetos literais.

Introdução

- Javascript pode inicializar objetos a partir de 3 estruturas fundamentais:
  - Operador new
  - Notação literal
  - Object.create()

Introdução

- Para criar um objeto a partir do operador new, podemos usar o construtor de Object de forma muito simples:

```
let obj=new Object();
```

- Note que o objeto assim criado é bastante genérico, ou seja, não possui quaisquer propriedades criadas pelo desenvolvedor (é dito ser um objeto “vazio”).

Introdução

- Assim como ocorrem com arrays e o uso de [], podemos criar objetos a partir de {}, sem fazer uso direto do operador new

```
let obj={};
```

Para que serve um objeto vazio?

- Note que a grande diferença entre o JavaScript e outras linguagens tipicamente orientadas a objetos consiste na **mutabilidade*** dos objetos

  **Um objeto pode perder ou ganhar propriedades no decorrer da execução do código*

- Assim sendo, podemos criar propriedades preenchendo objetos vazios, a partir de simples atribuições.

Formas de se adicionar/acessar propriedades

obj.name=“João”;

ou

obj[“name”] = “João”;

- Embora semanticamente equivalentes, a segunda forma tem a vantagem de que o valor da chave é passado como uma string que pode ser calculada em tempo de execução
- O segundo método também permite usar uma palavra reservada como nome de propriedade
- Entretanto, esse método de criação dificulta o uso da minificação de código

Exercício

- Crie um objeto que representa um livro, o qual possui as propriedades: título, subtítulo, ISBN, autor, editora, ano, valor, com os dados obtidos em https://www.amazon.com.br/JavaScript-Guia-Definitivo-David-Flanagan/dp/856583719X/ref=tmm_pap_swatch_0?_encoding=UTF8&qid=1666644637&sr=8-2

- Crie um código que acessa estas propriedades e apresenta as mesmas na tela.

Notação literal

- Em muitos casos, é vantajoso criar objetos com todas as propriedades inicializadas (ou um conjunto de propriedades iniciais deste objeto).

- Para isso podemos utilizar a notação literal

Criando Objetos com notação literal

- A sintaxe do objeto literal pode ser usada para inicializar completamente um objeto

```
var obj = {
        name: “Banana”,
        “for”: “Jorge”,
        details:{
                color: “Amarela”,
                quantity: 12
        }
}
```

Acessando propriedades

- O acesso a atributos pode também ser encadeado

```
var obj = {
        name: “Banana”,
        “for”: “Jorge”,
        details:{
                color: “Amarela”,
                quantity: 12
        }
   }
alert(obj.details.color);
alert(obj.details.quantity);
```

O operador in

- O operador in espera que o operando a esquerda seja uma string
- A partir dessa string, o operador in busca por uma propriedade (método ou atributo) no operando da esquerda, que é um objeto
- Finalmente, o operador in retorna se essa string é uma propriedade do objeto (true ou false)

O operador in

- Exemplo

```
var point = { x:1, y:1 };
alert("x" in point);
alert("z" in point);
```

- O operador in é utilizado tanto para verificar a existência de uma propriedade quanto para compor um arranjo que percorre todas as propriedades de um objeto.

Revisão: Vetores

- Vetores funcionam de forma similar à objetos regulares, possuindo a sintaxe []

- Criando vetores:

```
var a = new Array(); //ou var a = [];
a[0] = “laranja”;
a[1]=“morango”;

var a = [“laranja”, “morango”];
```

Percorrendo vetores

- Podemos percorrer naturalmente um vetor a partir de um simples laço for
- No exemplo anterior:

```
var len=a.length;
for (var i=0;i<len;i++){
  console.log(a[i]);
  console.log(i);
}
```

Percorrendo vetores

- Como vocês podem ver, esse método de varredura de vetores não é muito eficiente
- Como alternativa podemos usar o for.. in
- Ainda no exemplo anterior:

```
for (var i in a){
   console.log(a[i]);
   console.log(i);
}
```

E os métodos de um objeto?

- Conforme dito anteriormente, os métodos de um objeto, no contexto do JavaScript, podem ser entendidos como propriedades que armazenam funções

- Assim, para o entendimento de como funcionam os métodos dos objetos JavaScript, vamos analisar a criação de funções no JavaScript

Revisão: Declaração de funções

```
function add(x,y){
   var total = x+y;
   return total;
}
alert(add(5,2));
```

Revisão: Expressão de função

```
var  add = function(x,y){
   var total = x+y;
   return total;
}
alert(add(5,2));
```

Métodos em objetos

- Simplesmente aplicamos o conceito de expressões de função a uma propriedade do objeto.

```
var meuobjeto = {
        meumetodo: function(){
        }
}
```

Métodos em objetos

```
var  pessoa = {
        nome:"Glederson",    saudacao:
        function(turno){
                if(turno==1){
                        return "bom dia!";
                }else if(turno=2){
                        return "boa tarde!";
                }else{
                        return "boa noite!";
                }
        }
}
```

Operador this

- Assim como outras linguagens orientadas a objetos, o JavaScript faz uso do operador **this**

- A partir dele conseguimos acessar o objeto e, portanto, suas propriedades

Exercício

Altere o método do objeto pessoa criado anteriormente, para que o objeto retorne em saudacao o nome de pessoa, na forma:

“<bom dia,||boa tarde||boa noite>, meu nome é <atributo nome do objeto>”

Exercício

- Crie o objeto literal Pessoa. Tal objeto será composto pelas propriedades:
  - primeiroNome
  - ultimoNome
  - altura
  - peso
  - sexo (‘M’ ou ‘F’)

- Crie o objeto com os valores que julgar interessante

Exercício – Parte 2

- Sabendo que um método é uma propriedade associada a uma função você seria capaz de criar os métodos nomeCompleto() que retorna o nome da pessoa e o método imc() que retorna a condição da pessoa conforme tabela abaixo.

Condição	IMC em Mulheres	IMC em Homens
abaixo do peso	< 19,1	< 20,7
no peso normal	19,1 - 25,8	20,7 - 26,4
marginalmente acima do peso	25,8 - 27,3	26,4 - 27,8
acima do peso ideal	27,3 - 32,3	27,8 - 31,1
obeso	> 32,3	> 31,1

  - O imc é dado pelo peso da pessoa dividido pelo quadrado da altura

Operador instanceof

- Operador que testa se um objeto é instância de um determinado tipo (se ele foi criado a partir de um tipo de objeto)

- Embora já saibamos como criar objetos, ainda não criamos funções construtoras de classes. Isso será visto mais adiante

Operador instanceof

```
var d = new Date();
d instanceof Date; //true
d instanceof Object; //true
d instanceof Number; //false
var a = [1, 2, 3];
a instanceof Array; //true
a instanceof Object; //true
a instanceof RegExp; //false
```

Note que aqui fica exemplificado que todos objetos são instâncias de Object.

operador typeof

- Operador que testa se um objeto é instância de um determinado tipo

```
var f = function(){return 5;};
typeof f
var x=5;
typeof x
```

operador delete

- operador que remove as variáveis
- Normalmente uma variável é removida a partir de um garbage colector do JS
- O delete indica que tal variável não será mais utilizada e, portanto pode ser desalocada, liberando espaço da memória
- Da mesma forma, o delete pode ser usado para remover propriedades de um objeto

```
delete d;
delete personagem.nome;
```

operador delete

- Variáveis globais não são deletáveis

- A exceção são variáveis criadas sem o operador var, que são deletáveis
  - Esse comportamento se dá por essas variáveis serem tratadas como propriedades do objeto global.

Testando o operador delete

- Delete um método e uma propriedade de Pessoa.