Introdução a Orientação a Objetos no JavaScript

Desenvolvimento Front-end III

DFEIII

Gléderson L. dos Santos

gledersonsantos@ifsul.edu.br

Campus Charqueadas
Tecnologia em Sistemas para Internet

Introdução a disciplina

- No segundo semestre aprendemos a utilizar Javascript segundo um modelo de programação imperativo (procedural), um modelo de programação baseado na descrição e posterior chamada de procedimentos.
  - Criamos funções que descrevem um conjunto de passos computacionais a serem executados.
  - Um procedimento pode ser chamado a qualquer hora durante a execução de um programa, inclusive por outros procedimentos ou por si mesmo.

Introdução a disciplina

- No terceiro semestre aprendemos a utilizar programação orientada a objetos, utilizando para isso a linguagem Java.
  - Nesse paradigma objetos são elementos que possuem dados próprios (também chamados de atributos) e propriedades, também chamadas de métodos.

- Vamos revisar rapidamente do que se trata a programação orientada a objetos.

O que é POO?

- Paradigma (padrão, modelo) de programação cujo termo foi criado por Alan Kay, autor da linguagem de programação Smalltalk.
- Mesmo antes do Smalltalk, algumas das ideias já eram aplicadas (a primeira linguagem a utilizar essas ideias foi o simula 67 – 1967).
- Vocês viram POO utilizando Java, mas esses conceitos, com algumas particularidades, são previstos também no JavaScript.

Mas o que é POO?

- Da wikipedia:
  - Programação orientada a objetos (POO) é um paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de campos, também conhecidos como atributos, e códigos, na forma de procedimentos, também conhecidos como métodos.
  - POO é um modelo de análise, projeto e programação de software baseado na composição e interação entre diversas unidades chamadas de 'objetos'.

Continuando

- A POO aproxima o mundo real do mundo virtual: a idéia fundamental é tentar simular o mundo real dentro do computador.
- Essa representação é baseada em objetos

O que são objetos?

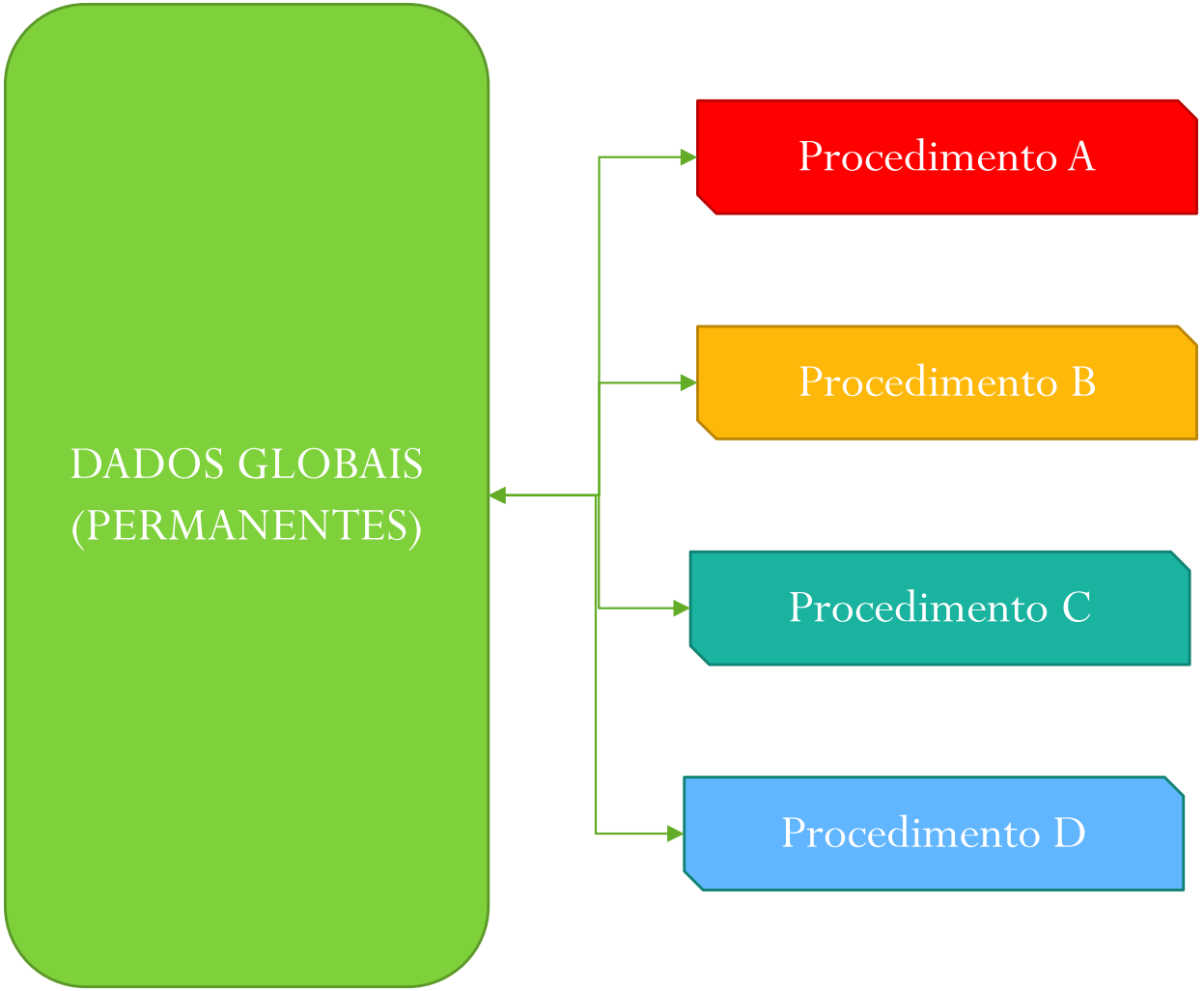
- Representações de entidades relacionadas com o problema em questão.

- Tais entidades podem ser por exemplo entes físicos (casa, carro, pessoa, aluno, professor), entes conceituais (conta corrente, apólice de seguros) ou de software (a conexão com o banco de dados).

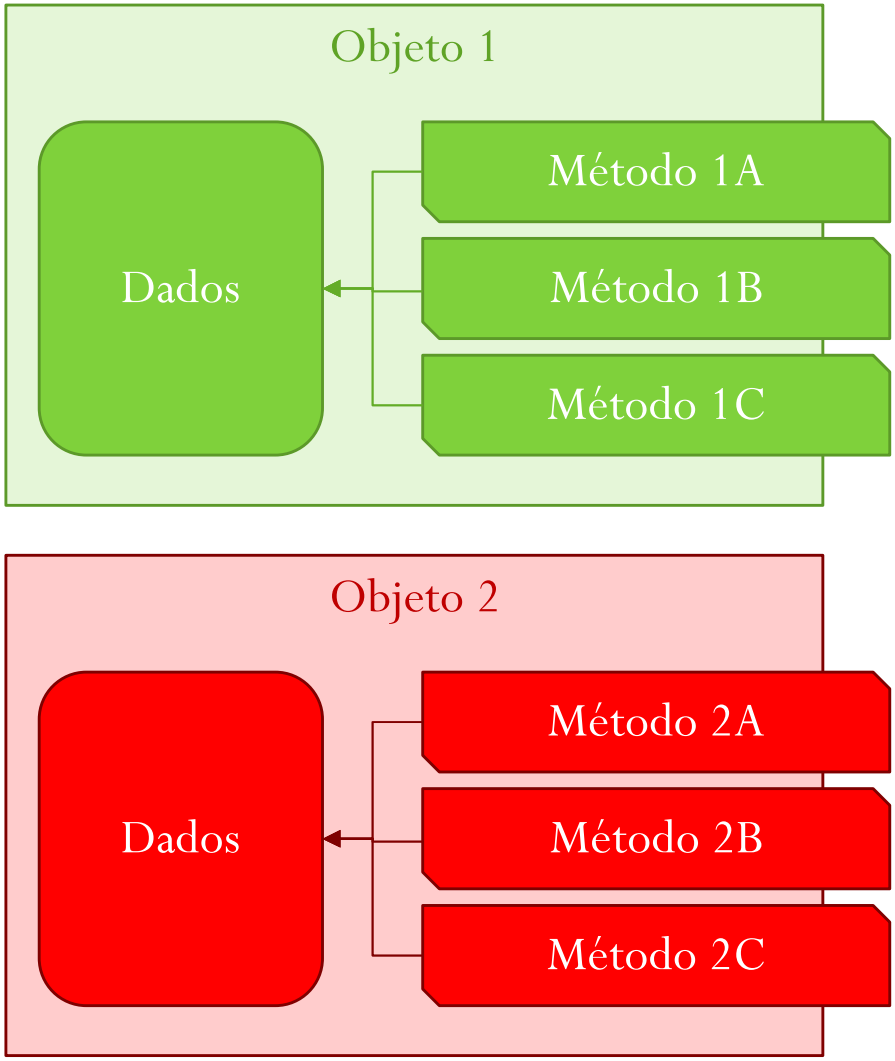
E o que faz o programador nesse contexto?

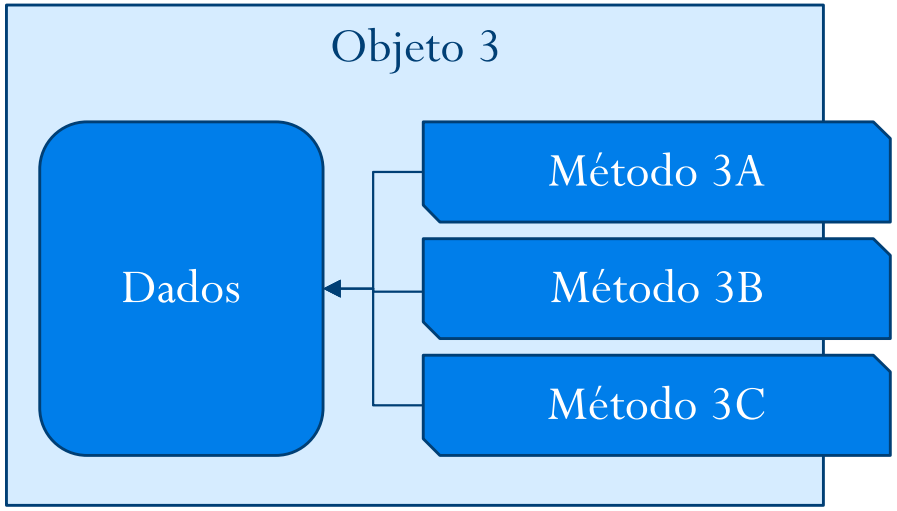
- Identifica os objetos que interagem no problema
- Identifica o que precisa ser representado sobre esse objeto
- Categoriza esses objetos
- Inter-relaciona essas categorias
- Identifica como esses objetos interagem entre si (ou, como trocam mensagens)

Programação Estruturada X POO



Programação Estruturada X POO





Introdução a disciplina

- Nesse semestre, nosso principal objetivo é verificar de que forma o JS aplica os princípios da orientação a objetos

Revisando conceitos básicos

- JavaScript trabalha com 2 categorias de tipos principais:
  - Tipos primitivos
    - number
    - string
    - boolean
  - Tipos de Objetos
    - arrays
    - function
    - Global
    - Date
    - RegExp
    - Math
    - ...

Revisando conceitos básicos

- Além disso JavaScript possui dois outros tipos bastante específicos, chamados valores primitivos
  - null
  - undefined

Objetos no JavaScript

- Objetos são elementos centrais para o desenvolvimento de JavaScript (Linguagem Baseada em Objetos)

- Como visto em outras disciplinas a Programação Orientada à Objetos é bastante importante no desenvolvimento de aplicativos em geral, assim faremos uma introdução a esse tópico e, posteriormente, apresentar as peculiaridades do JavaScript

Revisando Objetos

- Objetos podem ser inicialmente vistos como representações virtuais de entidades físicas
- Alguns livros de orientação a objetos relacionam objetos a substantivos
  - Ex.: cadeira, carro, pessoa, casa.
- Como representar um objeto, como um carro para um simulador de corrida?

Precisamos representar os aspectos relevantes para essa simulação.

O que são esses aspectos relevantes?

- Atributos ou propriedades do objeto (análogo a adjetivos)
  - Características inerentes a esses objetos que os diferenciam dos demais objetos de um mesmo tipo (classe).

- Exemplos:
  - cor
  - modelo
  - velocidade máxima
  - velocidade atual
  - aceleração

O que são esses aspectos relevantes?

- Métodos (análogo a verbos)
  - Funcionalidades com as quais os objetos interagem entre si e com o ambiente simulado.
  - Podemos associar métodos com o conceito de funções
  - Métodos permitem a interação com as propriedades de um objeto

- Exemplos:
  - ligar
  - acelerar
  - frear
  - buzinar

Eventos

- Por ter uma origem fortemente associada a interfaces de usuário (GUI) o JavaScript tem sua origem relacionada fortemente a programação dirigida a eventos (event-driven)

Eventos

- Assim, ao invés de seguir um fluxo de controle pré-definido, o programa é fortemente guiado por acontecimentos ou **Eventos**. Eventos são as formas de avisar alterações de estado dentro do programa.
  - pressionar a tecla A
  - um determinado intervalo de tempo se passar desde a abertura da página
  - uma consulta ao servidor retornar dados
  - a rolagem de uma página
  - um objeto carro sair da região asfalto em direção a região brita

Eventos típicos do HTML

- Mouse
  - click, mouse (down, up, over)
- Keyboard
  - key (down, up, press)
- HTML
  - load, unload, resize, scroll
- Form
  - select, change, submit, focus

Objetos Nativos

- Conforme citado anteriormente, o JavaScript possui uma série de objetos nativos (core Objects)
  - Math
  - Date
  - document (Objeto que representa o próprio documento HTML aberto no navegador)
  - Screen (Objeto que representa a tela do usuário)
  - String
  - Global

Criação de Objetos Nativos

- Cada um desses objetos nativos é implementado ou pode ser acessado de uma forma distinta conforme se segue:
  - Objetos Globais
    - Utilizamos uma instância global e única do objeto
  - Wrapper Objects
    - Objeto criado dinamicamente para acessar propriedades a partir de um tipo primitivo
  - Objetos típicos
    - Criados a partir do operador new, assim como vocês viram em Java

O que são objetos globais?

- De forma simplificada, podemos dizer que objetos globais são objetos que existem em todo o contexto do programa
- Assim, tais objetos não precisam ser inicializados (instanciados) podendo ser utilizados em todo o nosso código.
- O Objeto Math é um exemplo de objeto global, diferentemente de outros objetos nativos, como o Date.
- Assim precisamos instanciar uma variável Date para usá-la, mas não precisamos fazer esse processo com o Math

```
console.log(Math.PI);
var d = new Date();
console.log(d.getMonth());
```

Wrappers
String: objeto ou tipo primitivo?

- Como citado anteriormente, strings, booleans e numbers são tipos primitivos

- Entretanto, o JavaScript possui uma série de métodos e propriedades para auxiliar no uso desses tipos

  - Exemplo

```
var s = "hello world!";
var word = s.substring(s.indexOf(" ")+1, s.length);
```

- Para encapsular esses métodos foram criados os **objetos wrapper** Number, String e Boolean

Wrapper Objects

- Diferentemente de objetos como Date e Arrays, wrappers são criados automaticamente pelo JavaScript;

- Assim, no exemplo anterior, o JavaScript converte provisoriamente o valor da string s em um objeto String, chamando new String(s) e aplicando as propriedades/métodos necessários. Após, esse objeto String é descartado.

Não Confunda!!!!

- Possuímos no JavaScript o tipo de variável string (que armazena um texto) e o objeto String (que além de armazenar um texto possui uma série de métodos para tratar esse texto
- Isso vale para o Number também

```
var a = "laranja";
var b= new String("laranja");
console.log(a);
console.log(b);
```

console.log é um chamado usado para debug do sistema. Através desse comando conseguimos verificar o conteúdo de uma variável

Cuidado!!!

- Wrappers podem gerar alguns erros bastante inconvenientes:

```
var texto = "uva";
var texto2 = "uva";
alert(texto == texto2);
var str = new String("uva");
var str2 = new String("uva");
console.log(str == str2);
```

Declarar um objeto wrapper mais do que desnecessário, é desaconselhável

Objetos Típicos

- Assim como em outras linguagens, podemos criar objetos a partir do operador new para um determinado tipo de objeto
- Exemplos que utilizamos no passado

```
var d=new Date();
var arr=[1,2,3];
var p = document.createElement(“<p>”);
```

Atribuindo objetos para variáveis

- Variáveis armazenam o endereço de objetos, ao invés de uma cópia dos mesmos
- Isso faz com que uma atribuição crie uma nova referência a um mesmo objeto

```
var c=[1,2,3];
var d = c;
c[0]=0;
console.log(d);
```

Objeto Global

- O Objeto global é um objeto específico do JavaScript que habilita propriedades e métodos que serão utilizáveis em todo o escopo do programa.
- Exemplos de propriedades globais
  - undefined, Infinity, NaN...
- Exemplos de métodos globais
  - isNaN(), parseInt(), eval()...
  - Construtores de outros objetos como Date(), RegExp(), Object(), Array()...
  - Outros objetos globais como Math e JSON

undefined e null

- undefined é o valor primitivo usado para representar que uma variável, embora exista, não possui valor atribuído

- null é um valor primitivo que indica a ausência de um objeto

- Embora sútil, essa diferença pode trazer bugs difíceis de serem identificados.

undefined e null

- Lembre do seguinte
  - variáveis, funções, propriedades e métodos nativos em que se espera que o retorno seja um objeto, você irá encontrar null caso esse objeto não exista
    - Exemplo

```
console.log(document.getElementById("nome"));
```

  - Por outro lado, caso você possua uma variável que ainda não tem valor atribuído, essa variável retornará o valor undefined
    - Exemplo:

```
var vazio;
console.log(vazio);
```