JAVASCRIPT THE RIGHT WAY

GUILHERME CARREIRO

"A language that doesn't affect the way you think about programming is not worth knowing."

- Alan Perlis -

A linguagem

A linguagem

- *"Tudo" é um objeto*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*

```
a = ["Javascript", "Ruby", "Java", "Python", "Haskell"];

a.first();
// => TypeError: Object Javascript,Ruby,Java,Python,Haskell has no method 'first'

Array.prototype.first = function() {
  return this[0];
}

a.first();
// => "Javascript"
```

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*
- *JSLint*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*
- *JSLint*
- *var*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*

```
var js = 'JS';

function teste() {
  var ruby = 'Ruby';
  console.log(ruby);
  console.log(js);
  var js = 'Javascript';
}

teste();

// => "Ruby"
// => undefined
```

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*

```
var js = 'JS';

function teste() {
  var js, ruby = 'Ruby';
  console.log(ruby);
  console.log(js);
  js = 'Javascript';
}

teste();

// => "Ruby"
// => undefined
```

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*
- *JSLint*
- *var*
- *Funções*

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
  - *Prototype*
  - *JSLint*
  - *var*

```
(function () {
  console.log('I <3 Javascript');
}());
```

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*

```
var Person = function (name) {
  this.name = name;
  this.say = function () {
    return 'I am ' + this.name;
  }
}

var p = new Person('Guilherme');
p.say();

// => "I am Guilherme"
```

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*

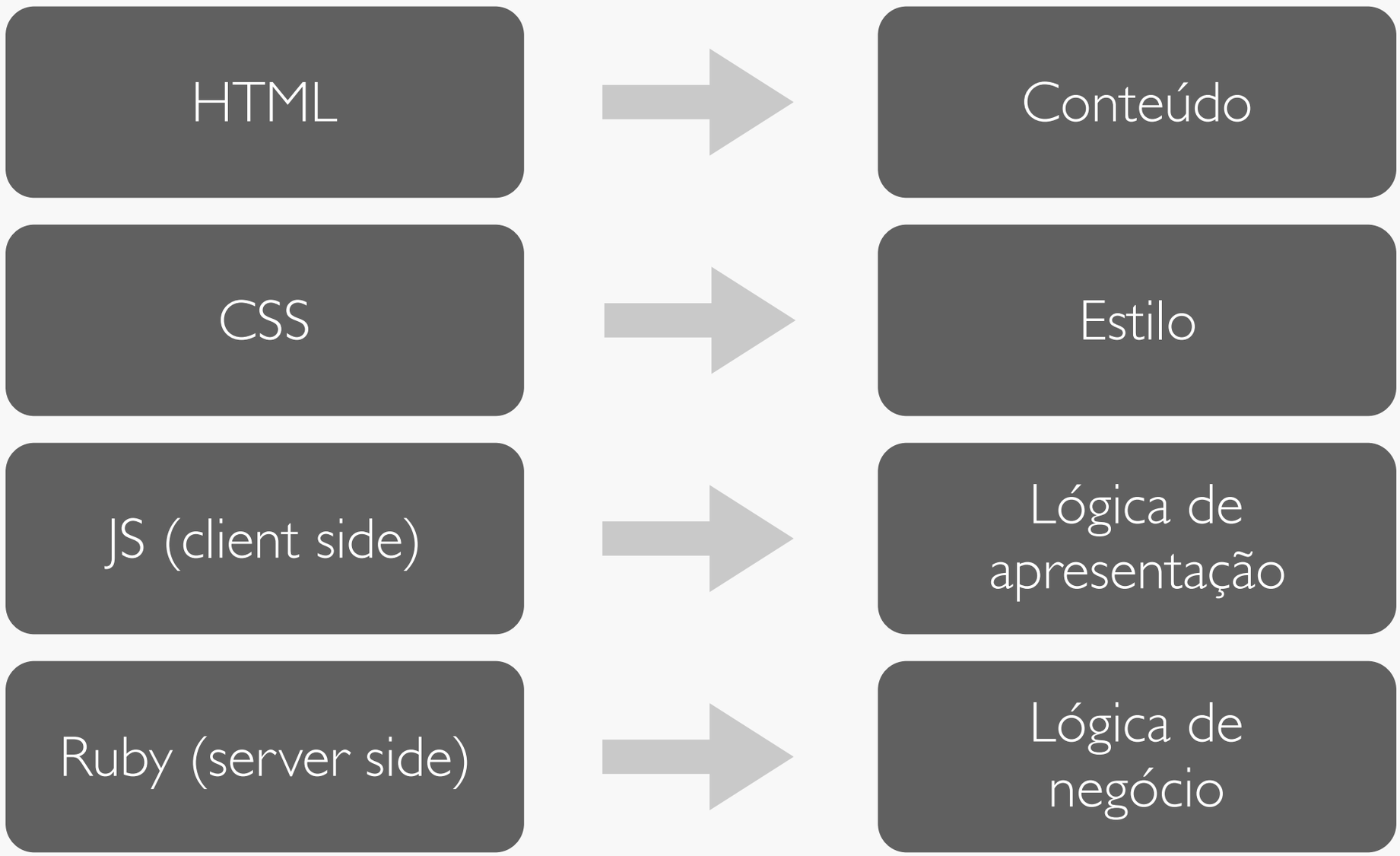
```
function atacar(intensidade, golpe) {
  console.log('Ataque com ' + intensidade + ' pontos de intensidade!');
  golpe(intensidade);
}

function espadada(i) {
  console.log('Espadada com intensidade de ' + i + ' pontos!');
}

atacar(10, espadada);
// Ataque com 10 pontos de intensidade!
// Espadada com intensidade de 10 pontos!
```

A linguagem

- *"Tudo" é um objeto*
- *Elegante, mas sem classe...*
- *Padrões*
- *Prototype*
- *JSLint*
- *var*
- *Funções*

Bad smells (front-end)

Separar responsabilidades



Fonte: http://www.slideshare.net/fgalassi/refactoring-to-unobtrusive-javascript

Código Javascript misturado com código HTML

Código Javascript misturado com código HTML

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <script type="text/javascript">
    doSomething();
  </script>
</body>
</html>
```

Código Javascript misturado com código HTML

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <input type="submit" onclick="validateAndSubmit();" />
</body>
</html>
```

CSS misturado com código Javascript

CSS misturado com código Javascript

```
var botao = document.getElementById('botao');

botao.onclick = function(e) {
  this.style.border = '2px solid red';
}
```

Lógica de negócio no Javascript

Lógica de negócio no Javascript

```
var botao = document.getElementById('botao'),
    saldo = <%= @saldo %>;

botao.onclick = function(e) {
  if(saldo > 0) {
    comprar();
  } else {
    return false;
  }
}
```

Código HTML no Javascript

```
var botao = document.getElementById('botao'),
    saldo = <%= @saldo %>;

botao.onclick = function(e) {
  var status = document.getElementById('status'),
      html = '<div>',
      foto = getUserPicture();
  if(saldo > 0) {
    html += '<img src="' +  foto + '" alt="Foto" />';
    html += '<h1>Saldo: ' + saldo + ' =)</h1>';
  } else {
    html += '<h2>Saldo insuficiente. =(</h2>';
  }
  html += '</div>';
  status.innerHTML = html;
}
```

Herança clássica

```
function Parent() {
  this.name = 'Joey';
  this.dance = function() {
    console.log('Dancing...');
  }
}
Parent.prototype.say = function() {
  console.log('I\'m ' + this.name);
}

function Child() {
  this.name = 'Dee Dee';
}

inherits(Child, Parent);
```

Padrão clássico I

```
function inherits(Child, Parent) {
  Child.prototype = new Parent();
}

var a = new Child();

a.dance(); // => Dancing...
a.say(); // => I'm Dee Dee

delete a.name

a.say(); // => I'm Joey
```

Padrão clássico II

```
function inherits(Child, Parent) {
  Child.prototype = Parent.prototype;
}

var a = new Child();

a.dance(); // => TypeError: Object #<Parent> has no method 'dance'
a.say(); // => I'm Dee Dee

delete a.name

a.say(); // => I'm undefined

Child.prototype.say = function() {
   console.log('Ohhh!');
}

var b = new Parent();
b.say(); // => Ohhh!
```

Padrão clássico III

```
function inherits(Child, Parent) {
  var F = function() {}
  F.prototype = Parent.prototype;
  Child.prototype = new F();
  Child.uber = Parent.prototype;
}

var a = new Child();

a.dance(); // => TypeError: Object #<Parent> has no method 'dance'
a.say(); // => I'm Dee Dee

delete a.name

a.say(); // => I'm undefined

Child.prototype.say = function() {
   console.log('Ohhh!');
}

var b = new Parent();
b.say(); // => I'm Joey
```

Padrão klass

```
var SuperMan = klass(Man, {
  __construct: function (what) {
    console.log('SuperMan\'s constructor!');
  },
  getName: function () {
    var name = SuperMan.uber.getName.call(this);
    return 'I am ' + name;
  }
});
```

Fonte: http://www.slideshare.net/fgalassi/refactoring-to-unobtrusive-javascript

Herança moderna

Herança prototípica

```
function object(o) {
  function F() {}
  F.prototype = o;
  return new F();
}
```

```
var parent = {
  name: 'Papa'
}

var child = object(parent);

console.log(child.name); // => Papa
```

Herança por cópia de propriedades

```
function extend(parent, child) {
  var i;
  child = child || {};
  for (i in parent) {
    if (parent.hasOwnProperty(i)) {
      child[i] = parent[i];
    }
  }
  return child;
}

var dad = {name: 'Adam'},
    kid = extend(dad);

kid.name; // => "Adam"
```

Herança por cópia de propriedades

```
function extend(parent, child) {
  var i,
      toStr = Object.prototype.toString,
      astr = '[object Array]';

  child = child || {};

  for (i in parent) {
    if (parent.hasOwnProperty(i)) {
      if (typeof parent[i] === 'object') {
        child[i] = (toStr.call(parent[i]) === astr) ? [] : {};
        extend(parent[i], child[i]);
      } else {
        child[i] = parent[i];
      }
    }
  }
  return child;
}
```

Mixins

```
// jQuery
$.extend();

var cake = $.extend(
  {eggs: 2, large: true},
  {butter: 1, salted: true},
  {fluor: '3 cups'},
  {sugar: 'sure!'}
);

console.log(cake); // => Object {eggs: 2, large: true, butter: 1 ... }
```

Design Patterns

Factory

```
CarFactory.prototype.compact = function(car) {
  car.doors = 4;
  return car;
}

CarFactory.prototype.convertible = function(car) {
  car.doors = 2;
  return car;
}
```

```
CarFactory.prototype.suv = function(car) {
  car.doors = 24;
  return car;
}

var c;
c = new CarFactory(); // => NotImplementedError
c = new CarFactory('suv');
c.drive(); // => "Vromm, I have 24 doors!"
```

```
function Car() {
  this.doors = null;
  this.drive = function() {
    return 'Vromm, I have ' + this.doors + ' doors!';
  }
}

function CarFactory(type) {
  var car = new Car();

  if (typeof this[type] !== 'function') {
    throw 'NotImplementedError';
  }

  return this[type](car);
}
```

Singleton

Literal objects

```
var Dextra = {
  property: 'value'
}
```

Namespace

Namespace

```
var DEXTRA = {};

DEXTRA.AutoCompleteEvents();

// --

DEXTRA.alert();
```

Namespace

```
var DEXTRA = DEXTRA || {};
```

Namespace

```
var DEXTRA = {
  modules: {
    module1: {},
    module2: {}
  }
};
```

Namespace

```
DEXTRA.namespace = function(ns_string) {
    var parts = ns_string.split('.'),
        parent = DEXTRA,
        current = '',
        i = 0;

    for ( ; i < parts.length; i++) {
        current = parts[i];
        parent[current] = parent[current] || {};
        parent = parent[current];
    }

    return parent;
}

var module1 = DEXTRA.namespace('modules.module1'),
    module2 = DEXTRA.namespace('modules.module2');
```

Fonte: JavaScript Patterns, por Stoyan Stefanow (O'Reilly)

Strategy

```
validator.types.isNonEmpty = {
  validate: function(value) {
    return value !== "";
  }
}

validator.config = {
  first_name: 'isNonEmpty'
}
```

```
var data = {
  first_name: 'Guilherme'
};

validator.validate(data);
validator.hasErrors();
validator.messages;
```

Fonte: http://www.slideshare.net/fgalassi/refactoring-to-unobtrusive-javascript

```
var validator = {
  types: {},
  messages: [],
  config: {},
  validate: function(data) {
    var i, msg, type, checker, resultOk = [];
    this.messages = [];
    for (i in data) {
      if (data.hasOwnProperty(i)) {
        type = this.config[i];
        checker = this.types[type];
        if (!type) continue;
        if (!checker) {
          throw { name: 'ValidatorError',  message: 'No handler to validate type' + type };
        }
        resultOk = checker.validate(data[i]);
        if (!resultOk) {
          msg = 'Invalid value for *' + i + '*, ' + checker.instructions;
          this.messages.push(msg);
        }
      }
    }
    return this.hasErrors();
  },
  hasErrors: function() {
    return this.messages.length !== 0;
  }
}
```

Fonte: JavaScript Patterns, por Stoyan Stefanow (O'Reilly)

Observer

```
var juiz = new Juiz();
var jogador = new Jogador();

juiz.addObserver(jogador);

juiz.verificarPosicaoJogadores();
// Juiz verifica posicao de jogadores...

juiz.apitar();
// Juiz apitou!
// Jogador chutou a bola!!
```

```
var Juiz = function() { // Observable
  this.changed = false;
  this.observers = [];
  this.verificarPosicaoJogadores = function() {
      console.log('Juiz verifica posicao de jogadores...');
  };
  this.apitar = function(){
      console.log('Juiz apitou!');
      this.changed = true;
      this.notifyObservers('apitou');
  };
  this.addObserver = function(obs) {
    this.observers.push(obs);
  };
  this.notifyObservers = function(msg) {
    if (this.changed) {
      this.changed = false;
     for (var i = 0; i < this.observers.length; i++) this.observers[i].update(msg);
    }
  };
};
var Jogador = function() { // Observer
    this.update = function(msg) {
        if (msg == 'apitou') this.chutar();
    };
    this.chutar = function(){
        console.log('Jogador chutou a bola!!!');
    };
};
```

OBRIGADO! :)

PERGUNTAS?

@karreiro_