

# INE5646 Programação para Web

- Tópico :

Desenvolvimento de Aplicações para Web

(estes slides fazem parte do material didático da disciplina  
INE5646 Programação para Web)

# Sumário

- Conceitos Básicos
- Desenvolvimento de Aplicações para Web
- Arquitetura em Camadas
- Arquitetura em Camadas para Aplicações para Web:
  - Aplicações em 2 camadas
  - Aplicações em 3 camadas
  - Aplicações em 4 camadas
- Evolução: da Web 1.0 para a Web 2.0
- Single Page Applications
- Padrão de Projeto MVC:
  - MVC no contexto web 1.0
  - MVC no contexto web 2.0
- Linguagens de Programação (LP):
  - LP usadas no lado Cliente
  - LP usadas no lado Servidor
- Linguagens de Representação de Dados:
  - XML
  - JSON
- Frameworks de Apoio à Programação

# Conceitos Básicos

- Aplicações para web são, antes de tudo, aplicações (“programas de computador”).
- Aplicações para web são aplicações (sistemas) distribuídas: o sistema é composto por pelo menos dois programas que são executados em dois computadores que se comunicam.
- Em um sistema distribuído é preciso definir a função de cada programa e como eles se comunicam.
- Definida a função, resta decidir como desenvolvê-la (arquitetura, protocolos, linguagens, etc)

# Conceitos Básicos

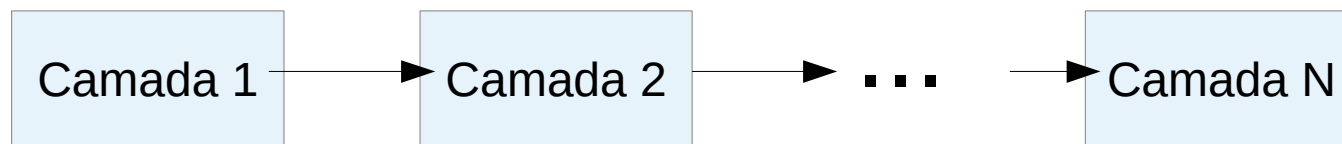
- Como em qualquer aplicação, há muito tempo se sabe que desenvolver software significa pensar sobre:
  - Como se dará a interação com o usuário.
  - Como o problema em questão pode ser resolvido.
- As duas questões, **aparência** e **essência**, devem ser desenvolvidas de forma **independente** ou, pelo menos, com a menor dependência possível.

# Desenvolvendo de Aplicações para Web

- O desenvolvimento da **aparência** (interface com o usuário) e da **essência** (lógica do domínio do problema) pode ser feito usando-se:
  - Uma única linguagem de programação (por exemplo, Java ou JavaScript)
  - Mais de uma linguagem de programação:
    - No sentido estrito (por exemplo, Java e JavaScript)
    - Em um sentido mais amplo (considerando, por exemplo, HTML uma linguagem usada para programar a aparência da aplicação)
- A escolha da(s) linguagen(s) deve, porém, ser precedida da definição da arquitetura da aplicação.

# Arquitetura em Camadas

- Observação: A ideia de organizar um software em camadas existe há bastante tempo (muito antes das aplicações para web terem sido criadas). Exemplo: As 7 camadas do Modelo OSI (Redes de Computadores) criadas em 1984.
- Motivação: softwares grandes e complexos são difíceis de manter.
- Uma camada é uma parte da aplicação com uma finalidade específica que se materializa na forma de um conjunto de serviços oferecidos.
- **Princípio fundamental**: na implementação dos serviços de uma camada utiliza-se os serviços oferecidos pela camada de baixo.
- Na figura abaixo, a camada  $i$  se utiliza dos serviços da camada  $i + 1$ .



# Arquitetura em Camadas para Aplicações para Web

- As camadas, no contexto das aplicações para web:
  - Tipicamente, a aplicação é organizada em 2, 3 ou 4 camadas.
  - No início predominavam as arquiteturas em 2 camadas.
  - Atualmente predominam as arquiteturas em 3 ou 4 camadas.

# Aplicações em 2 camadas

- **Camada 1: Cliente**

- Executada no computador do usuário.
- Responsável pela interação com o usuário (óbvio!) e pela lógica do domínio do problema.
- Tipicamente, o software é uma aplicação específica.

- **Camada 2: Servidor**

- Executada em um computador longe do usuário, em um (ou vários) servidor(es).
- Responsável pelo armazenamento dos dados e/ou acesso a outras aplicações (sistemas legados).
- Tipicamente, o software é um SGBD (banco de dados) e/ou um software legado.



# Aplicações em 3 camadas

- **Camada 1: Cliente**

- Executada no computador do usuário.
- Responsável apenas por exibir e coletar dados digitados do usuário.
- Não há quase nenhum processamento. No início, nem mesmo validações simples eram executadas (do tipo verificar se o usuário preencheu ou não um campo de um formulário ou verificar se o dado digitado é compatível com o tipo da informação esperada (“o valor digitado é número?”)).
- Tipicamente, o software é um navegador (browser) cuja função resume-se a renderizar um documento HTML gerado na camada 2.
- Expressão usada para caracterizar a camada: “thin client” (cliente magro).
- Esta camada “burra” (sem processamento no computador do cliente) caracteriza a primeira geração de aplicações para web.

# Aplicações em 3 camadas

- **Camada 2: Lógica ou de Negócio**

- Executada em um servidor.
- Contém a programação:
  - Da lógica do domínio do problema (as chamadas regras de negócio)
  - Da lógica da interface com o usuário (montagem das páginas HTML a serem exibidas pela camada 1 (navegador) e recepção dos dados enviados pela camada 1.
- Vantagens em relação às aplicações em 2 camadas:
  - Mais segurança: dados críticos (por exemplo para conexão com banco de dados) ficam no servidor (não são enviados à camada 1)
  - Atualização automática de versões: basta atualizar o servidor que os usuários passam a usar a nova versão sem nenhum esforço (instalação).
- O grande desafio é fazer com que o servidor consiga atender todos os usuários que estão acessando simultaneamente a aplicação. Problemas: consumo de memória (dados para cada usuário) e tempo de CPU.

# Aplicações em 3 camadas

- **Camada 3: Dados ou Legada**
  - Executada, normalmente, em um computador exclusivo.
  - Responsável por armazenar os dados da aplicação.
  - O software normalmente é algum SGBD (sistema gerenciador de bases de dados).
  - Pode ser um sistema legado (conter algumas regras de negócio). Exemplo: as “stored procedures” dos bancos de dados.

# Aplicações em 4 camadas

- Variação da arquitetura em 3 camadas.
- A diferença está na camada 2 que é dividida em 2 camadas.
- **Camada 1: Cliente:**
  - Mesmos conceitos da camada 1 das aplicações em 3 camadas.
- **Camada 4: Dados ou Legada:**
  - Mesmos conceitos da camada 3 das aplicações em 3 camadas.

# Aplicações em 4 camadas

- **Camada 2: Web:**

- Responsável apenas pela:
  - Construção da interface com o usuário (montagem das páginas HTML a serem exibidas pela camada 1 (cliente)).
  - Recepção e análise (validações) dos dados enviados pelo usuário por meio da camada 1.
- Tipicamente, implementa o protocolo HTML.

# Aplicações em 4 camadas

- **Camada 3: Lógica ou de Negócios:**
  - Responsável apenas pela implementação das regras do negócio.
  - O desenvolvedor desta camada não deve saber qual tecnologia/linguagem/framework é usada na camada 2.
  - Em aplicações com baixa taxa de utilização, as camadas 2 e 3 podem ser executadas no mesmo computador.

# Aplicações em 4 camadas: exemplo

- Uma aplicação para web baseada na plataforma Java JEE6 é formada pelas seguintes tecnologias (situação típica):
  - **Camada 1** (Cliente): browser executando código em JavaScript e HTML. Comunica-se com a camada 2 por meio do protocolo HTTP/HTTPS.
  - **Camada 2** (Web): servlet container, como o Tomcat ou o Grizzly, executando código Java para as tecnologias servlet, JSP, JSF. Comunica-se com a camada 3 via RMI caso as camadas 2 e 3 estejam em JVM diferentes ou por referência direta (ponteiro) caso as camadas 2 e 3 estejam na mesma JVM.
  - **Camada 3** (Negócios): servidor de aplicação, como o Glassfish ou o Jboss, executando código Java para as tecnologias EJB e JPA. Comunica-se com a camada 4 por meio do protocolo JDBC.
  - **Camada 4** (Dados): qualquer data source, tipicamente SGBD relacional como MySQL.

# Evolução: da Web 1.0 para Web 2.0

- Na primeira geração das aplicações, o que hoje podemos chamar de aplicações 1.0, a interface com o usuário era modelada como sendo formada por páginas HTML. A grande novidade de então era a capacidade das tecnologias gerarem páginas HTML dinamicamente no servidor para serem exibidas no navegador do usuário.
- A ideia deu tão certo que surgiu uma inevitável pressão sobre os desenvolvedores para que as aplicações desktop tivessem também uma versão para web.
- Isto levou a um conflito: páginas HTML são, na sua origem, textos (ou hiper-textos). Textos foram feitos para serem lidos pelos usuários. As aplicações, de outro lado, demandam interação com os usuários.
- A interface com o usuário de uma aplicação não é formada por textos mas sim por uma série de elementos visuais (menus, botões, campos, etc) que reagem a estímulos (uso do mouse e do teclado).
- Consequências:
  - Modelar e implementar a interface com o usuário de uma aplicação ficou complexo, como sempre foi para as aplicações desktop.
  - Revisão do papel das camadas 1 e 2 das arquiteturas de 3 e 4 camadas.
- Para dar conta desta nova forma de entender aplicações para web, que ficou conhecida como web 2.0, Voltou-se a falar no padrão de projeto MVC (que já existia desde meados dos anos 1980).
- A camada 1 passou a realizar processamento e não apenas exibir texto em HTML.
- Mais recentemente, surgiu a tendência da camada 1 ser formada por uma única página HTML. Surgiu então o conceito de **single page applications**. (aplicações de página única).

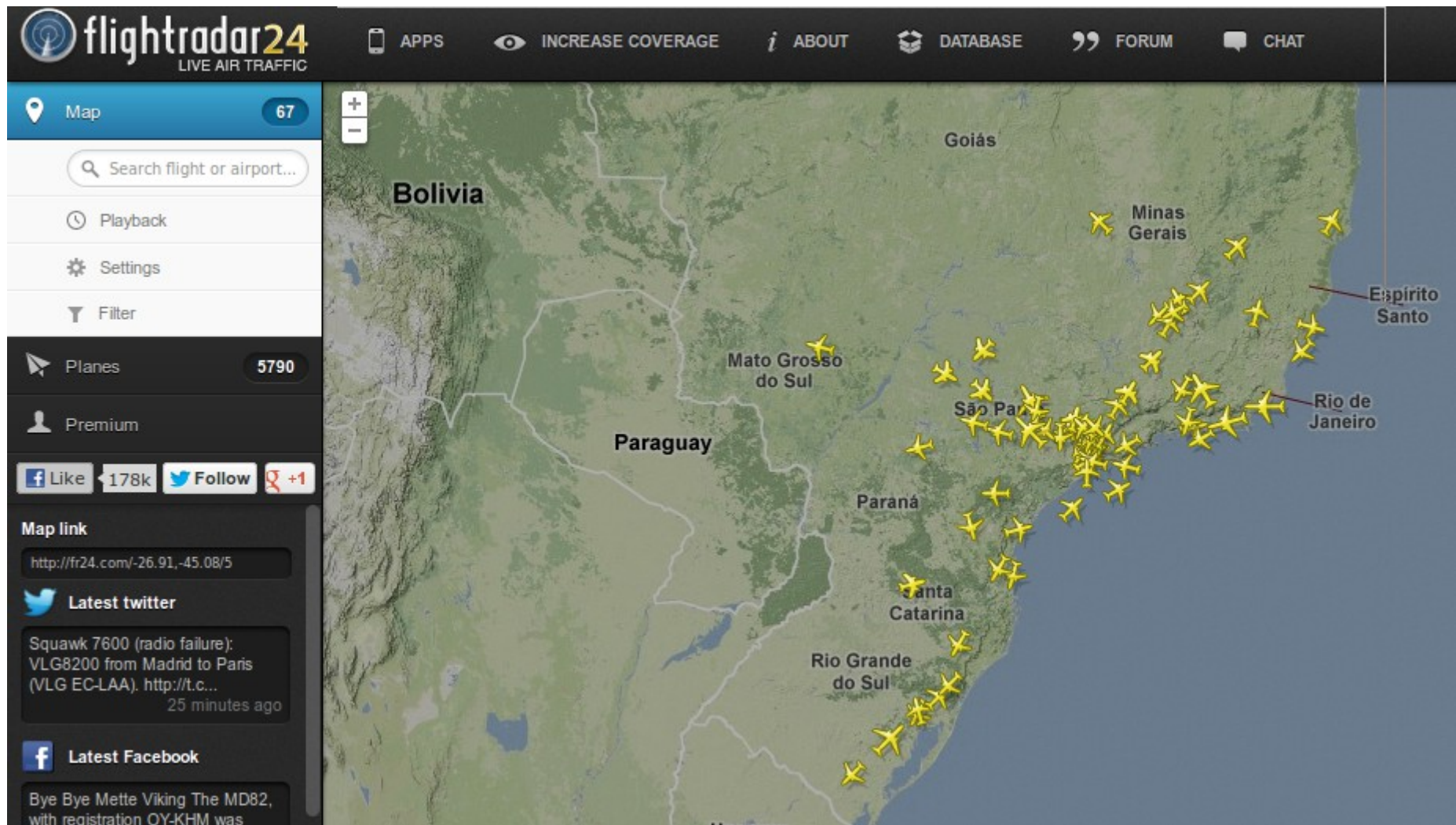


# Single Page Applications (SPA)

- A “página inicial” é a única página.
- A primeira requisição HTTP feita a partir do browser traz, como resposta, toda a interface com o usuário (fragmentos de página HTML e código JavaScript).
- Toda interação com o usuário que não dependa da camada 2 acontece na camada 1.
- A camada 1 recebe dados (e não páginas HTML) da camada 2.

# Single Page Applications (SPA)

- <http://www.flightradar24.com>



# Single Page Applications (SPA)

- **Usa a tecnologia Ajax** para fazer a comunicação com a camada 2.
- Usando Ajax:
  - Trafegam na rede apenas dados, representados nos formatos JSON e/ou XML.
  - A interface com o usuário não fica “congelada” aguardando a resposta do servidor.
  - Quando a camada 1 envia (requisição do protocolo HTTP) dados à camada 2, registra-se qual função JavaScript deverá tratar os dados contidos na resposta. Estas funções são conhecidas como **callback functions**.
  - Quando a camada 2 responde (resposta do protocolo HTTP), os dados são consumidos pela função callback que deverá atualizar a interface com o usuário.
- A experiência de usabilidade é semelhante às aplicações desktop pois reduz-se a comunicação com a camada 2 ao mínimo necessário.
- Exemplo mais simples (em termos de interface com o usuário) de SPA: Trello (<http://www.trello.com>)

# Padrão de Projeto MVC

- Padrão já existe há muito tempo (anos 1980).
- Do inglês: **M**odel – **V**iew – **C**ontrol
- Modelo – Visualização – Controle
- Motivação: separar a forma como os dados são representados da forma como são visualizados, mantendo sincronia entre as duas formas. Exemplo: uma lista de dados pode ser visualizada na forma textual e na forma de gráfico.
- O controle faz a sincronia (alterando o modelo atualiza as visualizações, alterando a visualização (usuário altera um dado) atualiza o modelo).

# MVC no contexto web 1.0

- Aparece na **camada 2** (web) das arquiteturas em 3 e em 4 camadas.
- **Modelo**: representação dos dados do domínio do problema, tipicamente objetos e coleções de objetos.
- **Visualização**: páginas HTML ou fragmentos de código HTML.
- **Controle**: tipicamente um objeto, recebe uma requisição HTTP da camada 1, analisa os dados da requisição, aciona o objeto do domínio do problema responsável pelo processamento dos dados e, em função do resultado decide qual página (view) deve ser retornada para visualização na camada 1 (browser).
- No contexto das tecnologias Java, um servlet faz o papel do controlador.

# MVC no contexto web 2.0

- Aparece na **camada 1** (cliente) das aplicações em 3 e em 4 camadas.
- **Modelo**: representação dos dados do domínio do problema, tipicamente objetos e coleções de objetos.
- **Visualização**: páginas HTML ou fragmentos de código HTML.
- **Controle**: um objeto (ou função) JavaScript é responsável por manter a sincronia entre o modelo (objetos JavaScript) e a página HTML (representação visual do modelo).
- Como é um padrão de projeto, define uma forma de modelar a camada 1. Pode-se seguir esta “filosofia” ou então adotar algum framework que incorpore a filosofia MVC. Exemplo: o framework Backbone (<http://backbonejs.org/>)

# Linguagens de Programação (LP)

- Linguagem de programação, definida aqui em um sentido amplo, é qualquer linguagem usada na implementação de uma aplicação para web.
  - A definição acima permite classificar HTML como um tipo de linguagem de programação pois é usada para programar a interface com o usuário.
- **Existem dois contextos:** linguagens usadas para criar programas que serão executados no computador do usuário e os que serão executados no servidor.

# LP Usadas no Lado Cliente

- **JavaScript** é a linguagem de programação mais usada para desenvolver a camada 1 das aplicações para web:
  - O desenvolvedor programa diretamente em JavaScript ou em outra linguagem que então é compilada para JavaScript, como Java ou CoffeeScript (<http://coffeescript.org/>).
- Historicamente, cada navegador implementou a sua linguagem de programação para a camada 1. Exemplo: o IE da Microsoft implementa JScript. JavaScript foi criada para o navegador Netscape nos anos 1990.
- A falta de padrão (cada navegador com a sua linguagem) levou ao caos: deveria haver uma versão da camada 1 da aplicação para cada navegador.
- O problema foi diminuído, mas não resolvido, com a criação em 1996 do padrão ECMAScript (ECMA é uma associação internacional de indústrias dedicada à padronizações na área de comunicação), atualmente na versão 5.1 de junho de 2011.
- Cada empresa desenvolvedora de navegador implementa, nem sempre completamente, alguma versão da especificação ECMAScript. Este é o motivo do sucesso de bibliotecas cross-browser como jQuery.
- Outras linguagens proprietárias, como ActionScript (Flash) da Adobe, também são usadas, embora estejam em declínio devido ao crescimento de HTML5.
- **HTML** é, naturalmente, bastante utilizada, especialmente porque incorpora um estilo de programação por eventos. Exemplo: quando o usuário clicar em um botão então um fragmento de código JavaScript deve ser executado.



# LP Usadas no Lado Servidor

- Por definição, qualquer linguagem de programação pode ser usada.
- Porém, alguns motivos devem ser levados em consideração na escolha de uma linguagem em particular:
  - Algumas linguagens, como PHP, foram criadas especificamente para o desenvolvimento de aplicações para web.
  - A existência de bibliotecas/frameworks específicos para aplicações para web.
  - O suporte oferecido por comunidades de desenvolvedores. Quanto maior a comunidade, mas fácil/rápido será para tirar dúvidas.

# LP Usadas no Lado Servidor

- Java:
  - Criada pela empresa americana Sun (depois vendida para a Oracle), é utilizada desde 1997 quando da criação dos servlets (biblioteca de classes).
- PHP:
  - Criada pelo “The PHP Group”, **PHP: Hypertext Preprocessor** (PHP) é utilizada desde 1998 (baseada em PHP/FI de 1995), foi criada especificamente para o desenvolvimento web (<http://www.php.net/>).
- VBScript:
  - Criada pela Microsoft, é utilizada desde 1996 quando da criação da tecnologia ASP (Active Server Pages).
- Ruby:
  - Criada pelo japonês Yukihiro Matsumoto, é utilizada desde 2003 quando da criação do framework para desenvolvimento web Ruby on Rails (<http://rubyonrails.org/>).
- JavaScript:
  - Apesar de ter sido criada para ser usada no lado cliente, começa a aparecer também no lado servidor. O argumento é que o desenvolvedor é mais produtivo se utilizar uma única linguagem de programação. Exemplo: Node.js (<http://nodejs.org/>)

# Linguagens para Representação de Dados: XML e JSON

- Relembrando...
  - Com a tecnologia Ajax, base do modelo web 2.0, o objetivo da comunicação entre o cliente e o servidor é o **envio e recepção de dados** (e não de páginas).
  - O corpo de uma mensagem HTTP, seja na requisição seja na resposta, não define como os dados devem ser representados. Envia-se e recebe-se apenas texto.
- Se os dados estiverem representados em um formato pré-definido é mais fácil:
  - Fazer a sua verificação e validação.
  - Fazer seu processamento.
- XML e JSON são as duas formas de representação de dados mais usadas.

# XML

- Recomendação do W3C para versão 1.0 em 1998, inspirada em SGML.
- Representação textual dos dados.
- Um dos principais objetivos é facilitar a intercomunicação entre diferentes aplicações executadas em diferentes plataformas.
- Possibilita que um documento possa ser verificado (se obedece ao formato XML) e validado (contém todas as informações e estas estão corretas).

# XML - Exemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Um exemplo de documento XML.
-->
<pedido>
  <numero>75</numero>
  <descricao>
    <item codigo="TA" quantidade="20"/>
    <item codigo="TB" quantidade="14"/>
  </descricao>
</pedido>
```

- Um pedido é caracterizado pelo seu número (75) e por sua descrição. A descrição envolve dois itens: o primeiro tem código TA e quantidade 20; o segundo tem código TB e quantidade 14.
- Documentos XML são de fácil leitura para as pessoas e podem ser processados de modo bastante eficiente.

# JSON

- JSON: **J**ava**S**cript **O**bject **N**otation.
- Apesar do nome, não tem relação com nenhuma linguagem de programação pois é apenas uma forma textual de representação de dados (<http://json.org/>).
- Tem este nome porque foi originalmente baseada em JavaScript. Oficialmente, o formato foi incorporado à especificação ECMAScript. É por isso que JSON integra-se tão naturalmente na linguagem JavaScript.

# JSON

- Há duas formas para representar os dados:
  - **Objeto**.
  - **Lista de valores**.
- Objeto: uma coleção de pares nome-valor.
- Valor: strings, números, true, false, null, objetos, listas de valores.
- Para as linguagens de programação mais conhecidas, há bibliotecas que permitem converter objetos e listas de valores JSON em alguma estrutura da linguagem e vice-versa. Por exemplo, um objeto JSON pode ser representado como um objeto ou como um mapeamento (tabela de hash).
- No caso de JavaScript, a conversão é automática.

# JSON - Exemplos

- Objeto:

**{“municipio”: “Itá”, “UF”: “SC”, “pop”: 5000}**

- O município de Itá, em Santa Catarina, possui uma população de 5 mil habitantes.

- Lista de valores:

**[1,2,3,4,5,6]**

- Os números 1, 2, 3, 4, 5 e 6.



# JSON - Exemplos

- Objeto:

```
{“nome”: “Fulano de Tal”,  
  “login”: “fulano”,  
  “email”: fulano@kmail.com,  
  “nivel”: [2, 5, 8]}
```

- Lista de valores:

```
[{“inscritos”: 75, “aprovados”: 70},  
 {“login”: “admin”, “senha”: “secreta2”}]
```

- Uma lista contendo dois objetos.

# Frameworks de Apoio à Programação

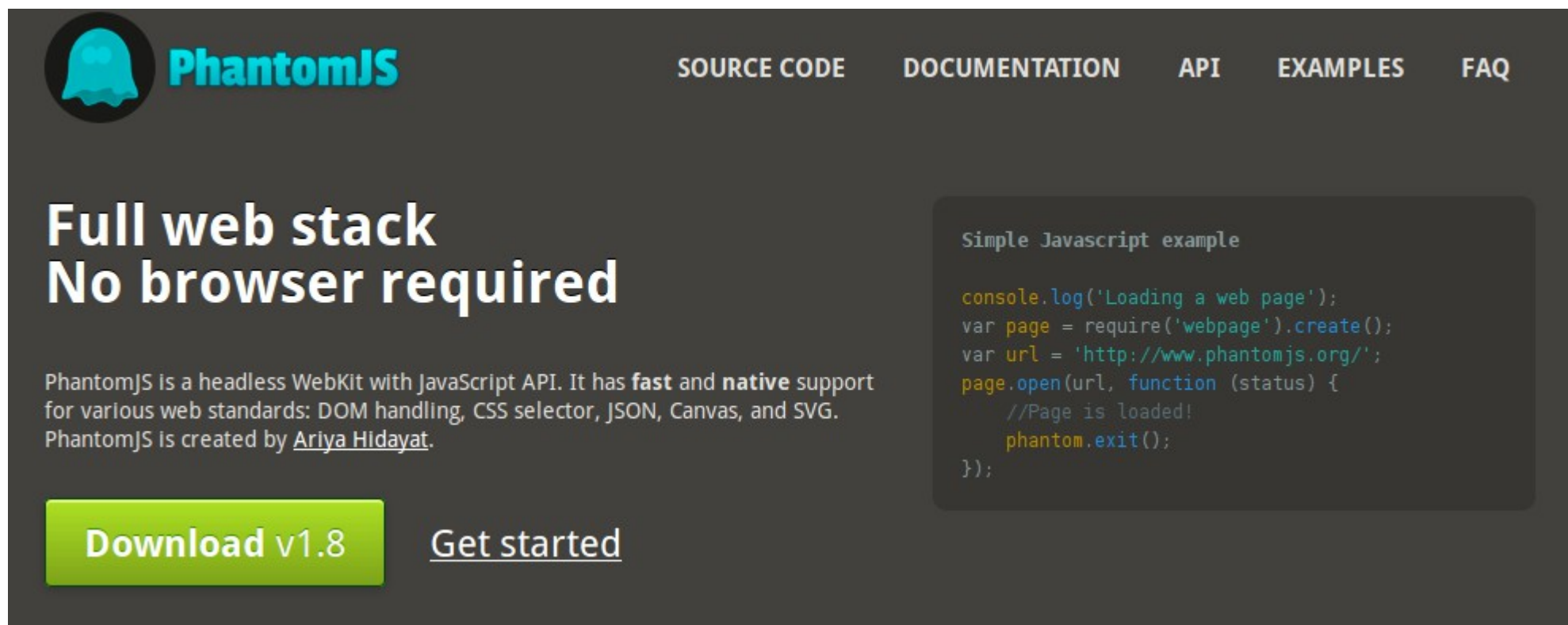
- A área de desenvolvimento de aplicações para web está em franca expansão.
- Lado positivo:
  - Desenvolver tem ficado mais fácil pois os novos frameworks incorporam as boas ideias/práticas aprendidas com a experiência.
  - Novos frameworks conseguem explorar as novas arquiteturas de hardware (múltiplos núcleos/CPU's) (paralelismo).
- Lado negativo:
  - Aumento do risco associado ao custo de manutenção: quais frameworks vão sobreviver e quais vão morrer?

# Frameworks de Apoio à Programação

- A seguir são apresentados alguns frameworks gratuitos (com licença open-source ou não).
- Para cada framework, o importante é compreender:
  - O tipo de benefício proporcionado ao desenvolvedor.
  - Como se posiciona na arquitetura da aplicação.

# Frameworks de Apoio à Programação

- <http://phantomjs.org>



The screenshot shows the PhantomJS website with a dark grey background. At the top left is the PhantomJS logo, a teal ghost icon next to the text 'PhantomJS'. To the right of the logo are navigation links: 'SOURCE CODE', 'DOCUMENTATION', 'API', 'EXAMPLES', and 'FAQ'. Below the navigation links, the main heading reads 'Full web stack' and 'No browser required' in large white text. Underneath this, a paragraph describes PhantomJS as a headless WebKit with JavaScript API, mentioning its fast and native support for various web standards like DOM, CSS, JSON, Canvas, and SVG. It also credits Ariya Hidayat as the creator. At the bottom left, there is a green button labeled 'Download v1.8' and a link labeled 'Get started'. On the right side, there is a code block titled 'Simple Javascript example' containing a snippet of JavaScript code that demonstrates how to use the PhantomJS API to load a web page and log the status.

**PhantomJS**

SOURCE CODE DOCUMENTATION API EXAMPLES FAQ

## Full web stack No browser required

PhantomJS is a headless WebKit with JavaScript API. It has **fast** and **native** support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG. PhantomJS is created by [Ariya Hidayat](#).

[Download v1.8](#) [Get started](#)

Simple Javascript example


```
console.log('Loading a web page');
var page = require('webpage').create();
var url = 'http://www.phantomjs.org/';
page.open(url, function (status) {
    //Page is loaded!
    phantom.exit();
});
```

# Frameworks de Apoio à Programação

- **PhantomJS:**
  - Biblioteca JavaScript para desenvolvimento de testes da camada 1.
  - Permite também a captura de tela via programação.

# Frameworks de Apoio à Programação

<http://docs.seleniumhq.org/>

**SeleniumHQ**  
Browser Automation

[edit this page](#) search selenium:  [Go](#)

[Projects](#) [Download](#) [Documentation](#) [Support](#) [About](#)

## What is Selenium?

*Selenium automates browsers.* That's it. What you do with that power is entirely up to you. Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should!) also be automated as well.

Selenium has the support of some of the largest browser vendors who have taken (or are taking) steps to make Selenium a native part of their browser. It is also the core technology in countless other browser automation tools, APIs and frameworks.

## Which part of Selenium is appropriate for me?



If you want to

- create quick bug reproduction scripts
- create scripts to aid in automation-aided exploratory testing



If you want to

- create robust, browser-based regression automation
- scale and distribute scripts across many environments



### Selenium is a suite of tools

to automate web browsers across many platforms.

Selenium...

- runs in [many browsers](#) and [operating systems](#)
- can be controlled by many [programming languages](#) and [testing frameworks](#).



# Frameworks de Apoio à Programação

- **SeleniumHQ:**
  - Permite automatizar testes da aplicação, especificamente da camada 1 (navegador).
  - É composto por um conjunto de ferramentas: Selenium 2 (API para uso nos testes), Selenium IDE (plugin Firefox para criação dos testes), Selenium Grid (para múltiplos testes em computadores remotos).

# Frameworks de Apoio à Programação

- <http://www.playframework.com/>



The screenshot shows the Play Framework website for Java developers. The header is green with the text "Introduction to Play Framework for Java developers". The main content area is white with green accents. On the left, there are two sections: "GET THE LATEST PACKAGE" with a green button "Download 2.1.0" and a link "or [browse all versions](#)", and "GETTING STARTED WITH" with a green button "Java & Scala" and a link "or [read full documentation](#)". On the right, there is a video player with a green background and the word "play" in white. The video player has a play button, a progress bar, and a "vimeo" logo. Below the video player, there is a headline "Play Framework makes it easy to build web applications with Java & Scala." and two paragraphs of text: "Play is based on a lightweight, stateless, web-friendly architecture." and "Built on Akka, Play provides predictable and minimal resource consumption (CPU, memory, threads) for highly-scalable applications."

Introduction to Play Framework for Java developers

GET THE LATEST PACKAGE

**Download 2.1.0**

or [browse all versions](#)

GETTING STARTED WITH

**Java & Scala**

or [read full documentation](#)

**play**

19:28 HD vimeo

**Play Framework makes it easy to build web applications with Java & Scala.**

Play is based on a lightweight, stateless, web-friendly architecture.

Built on Akka, Play provides predictable and minimal resource consumption (CPU, memory, threads) for highly-scalable applications.



# Frameworks de Apoio à Programação

- **Play Framework:**
  - Desenvolvimento de aplicações escaláveis em Java ou em Scala.
  - Adota o padrão MVC para a camada 2.
  - Obs: será visto em mais detalhes mais adiante na disciplina

# Frameworks de Apoio à Programação

- <http://nodejs.org/>



# Frameworks de Apoio à Programação

- **Node.js:**
  - Desenvolvimento de aplicações usando JavaScript no servidor.
  - Orientada a eventos e a operações de E/S não bloqueantes.

# Frameworks de Apoio à Programação

- [Http://lesscss.org](http://lesscss.org)



## The dynamic stylesheet language.

LESS extends CSS with dynamic behavior such as variables, mixins, operations and functions.

LESS runs on both the server-side (with Node.js and Rhino) or client-side (modern browsers only).



Download less.js

version 1.3.3  
changelog

## Write some LESS:

```
@base: #f938ab;

.box-shadow(@style, @c) when (iscolor(@c)) {
  box-shadow: @style @c;
  -webkit-box-shadow: @style @c;
  -moz-box-shadow: @style @c;
}

.box-shadow(@style, @alpha: 50%) when (isnumber(@alpha)) {
  .box-shadow(@style, rgba(0, 0, 0, @alpha));
}

.box {
  color: saturate(@base, 5%);
  border-color: lighten(@base, 30%);
  div { .box-shadow(0 0 5px, 30%) }
}
```

## Compile to CSS:

```
npm install -g less
lessc styles.less styles.css
```

# Frameworks de Apoio à Programação

- **LESS:**
  - Estende CSS incorporando conceitos de linguagem de programação (como variáveis e funções).
  - Pode ser usada tanto na camada 1 como na camada 2.
  - O código escrito em LESS é compilado para CSS.

# Frameworks de Apoio à Programação

- <http://www.openxava.org>

The screenshot shows the OpenXava website. At the top is a navigation bar with links: Home, Demos, Downloads, Documentation, Book, Add-ons, Support, Blog, and Credits. Below the navigation bar is a breadcrumb trail: OpenXava > Home. On the left, there is a green button with a downward arrow and the text "Download OpenXava". To the right of the button, the text reads: "OpenXava is an **AJAX Java Framework** for **Rapid Development** of Enterprise Web Applications. In OpenXava you **only** have to write the **domain classes** in plain Java to get a web **application ready for production**."

Below this text, there is a link: [Look the Demos - Follow the Quick Start guide](#).

Under the link, there are two sections:

- Write this code**: A code editor showing the following Java code:

```
import javax.persistence.*;

@Entity
class Customer {

    @Id
    int number

    String name

}
```
- Get this application**: A screenshot of a web application interface. It features a table with columns "Number" and "Name". The table contains the following data:

Number	Name
1	PHILIP GLASS
2	JUANITO VALDERRAMA
3	Pedro Grillo
14	JUAN LÓPEZ
123	First Invoice Test

The third row is highlighted in yellow. Below the table, there is a pagination control showing "1 2" and "5 rows per page". A red button labeled "Delete selected rows" is at the bottom. A status message at the bottom right says "There are 10 objects in list (Hide them)".

# Frameworks de Apoio à Programação

- **OpenXava:**
  - Desenvolvimento ágil de aplicações.
  - A partir de classes Java (o modelo do MVC), gera automaticamente o V e o C para as operações CRUD (Create, Read, Update e Delete).
  - É usada na camada 2 e gera todo o código da camada 1.

# Frameworks de Apoio à Programação

- <http://coffeescript.org>


 **CoffeeScript**

TABLE OF CONTENTS   TRY COFFEESCRIPT   ANNOTATED SOURCE

**CoffeeScript is a little language that compiles into JavaScript.** Underneath that awkward Java-esque patina, JavaScript has always had a gorgeous heart. CoffeeScript is an attempt to expose the good parts of JavaScript in a simple way.

The golden rule of CoffeeScript is: *"It's just JavaScript"*. The code compiles one-to-one into the equivalent JS, and there is no interpretation at runtime. You can use any existing JavaScript library seamlessly from CoffeeScript (and vice-versa). The compiled output is readable and pretty-printed, passes through [JavaScript Lint](#) without warnings, will work in every JavaScript runtime, and tends to run as fast or faster than the equivalent handwritten JavaScript.

**Latest Version:** [1.5.0](#)

## Overview

*CoffeeScript on the left, compiled JavaScript output on the right.*

<pre># Assignment: number = 42 opposite = true  # Conditions: number = -42 if opposite  # Functions: square = (x) -&gt; x * x  # Arrays: list = [1, 2, 3, 4, 5]  # Objects:</pre>	<pre>var cubes, list, math, num, number, opposite, race, square,     __slice = [].slice;  number = 42;  opposite = true;  if (opposite) {   number = -42; }  square = function(x) {   return x * x; };</pre>
---	--



# Frameworks de Apoio à Programação

- **CoffeeScript:**
  - Linguagem que compila para JavaScript.
  - Mais simples (menos verboso) que JavaScript.
  - Usada tanto para a camada 1 quanto para a camada 2 (quando JavaScript é usada no lado servidor).

# Frameworks de Apoio à Programação

- <http://dhtmlx.com>

The screenshot displays the DHTMLX website, which promotes its JavaScript library for building professional web applications. The main banner features the DHTMLX logo and the text "Start Building Professional Web Apps Today". It highlights "dhtmlxSuite" as a rich JavaScript library providing a complete set of UI components. A badge indicates "21 UI Controls". Below this, icons represent various components: Grid, TreeGrid, Tree, and Form. A "Screens & Demos" button is also present.

Overlaid on the banner are several UI components:

- Sales Table:** A table with columns for Sales, Book, Price, and Delivery. It lists books like "Designing Interfaces" and "JavaScript: The Good Parts" with their respective prices and delivery status.
- Calendar:** A calendar for March 2012, showing dates from 1 to 31.
- Modal Dialog:** A dialog box with a "Disable profile" checkbox and radio buttons for "Forever" and "For some time". It also has a "Get updates" section with a "Daily" button and a "Send notification" checkbox.

At the bottom, three sections are visible:

- Suite:** "Build Ajax-Based Web Apps Fast". It shows a "Book shop" interface with a table of books and a "dhtmlxWindow" component.
- Touch:** "JavaScript HTML5 Mobile Framework". It features a mobile app interface on a tablet and a JavaScript logo.
- Scheduler:** "Now for Windows 8". It shows a "Feature-Rich Event Calendar" with a table of events, including a "Meeting" on Friday, April 20, and a "Presentation" on Saturday, April 21.

# Frameworks de Apoio à Programação

- **DHTMLX:**
  - Biblioteca JavaScript que contém um grande conjunto de componentes de interface com o usuário.
  - Usada, portanto, na programação da camada 1.
  - É gratuita para aplicações gratuitas.

# Frameworks de Apoio à Programação

- <http://documentcloud.github.com/backbone>

## Backbone.js (0.9.10)

» [GitHub Repository](#)

» [Annotated Source](#)

### Introduction

### Upgrading

### Events

- on
- off
- trigger
- once
- listenTo
- stopListening
- **Catalog of Built-in Events**

### Model

- extend
- constructor / initialize
- get
- set
- escape
- has
- unset
- clear
- id



# BACKBONE.JS

Backbone.js gives structure to web applications by providing **models** with key-value binding and custom events, **collections** with a rich API of enumerable functions, **views** with declarative event handling, and connects it all to your existing API over a RESTful JSON interface.

The project is [hosted on GitHub](#), and the [annotated source code](#) is available, as well as an online [test suite](#), an [example application](#), a [list of tutorials](#) and a [long list of real-world projects](#) that use Backbone. Backbone is available for use under the [MIT software license](#).

You can report bugs and discuss features on the [GitHub issues page](#), on Freenode IRC in the [#documentcloud](#) channel, post questions to the [Google Group](#), add pages to the [wiki](#) or send tweets to [@documentcloud](#).

*Backbone is an open-source component of [DocumentCloud](#).*

# Frameworks de Apoio à Programação

- **Backbone.js:**
  - Biblioteca JavaScript que estrutura a camada 1 de acordo com o padrão MVC.
  - Conecta-se à camada 2 por meio de uma API RESTful usando JSON.



# Frameworks de Apoio à Programação

- <http://knockoutjs.com>

The screenshot shows the Knockout.js website homepage. At the top is a navigation bar with links: Home, Download / Install, Tutorials, Live examples, Documentation, Forum, and Source. The main header features the 'Knockout.' logo in a stylized white font on a dark red background. To the right of the logo, a text box states: 'Simplify dynamic JavaScript UIs by applying the Model-View-View Model (MVVM) pattern'. Below this text is a yellow 'Download' button labeled 'v2.2.1 - 14kb min+gz' with a downward arrow icon. Below the main header is a 'Key concepts' section with a light gray background. It contains four items, each with a circular icon and a description: 1. 'Declarative Bindings' with a red icon of two interlocking rings, described as 'Easily associate DOM elements with model data using a concise, readable syntax'. 2. 'Automatic UI Refresh' with a blue icon of two circular arrows, described as 'When your data model's state changes, your UI updates automatically'. 3. 'Dependency Tracking' with a yellow icon of two people, described as 'Implicitly set up chains of relationships between model data, to transform and combine it'. 4. 'Templating' with a green icon of a star and arrows, described as 'Quickly generate sophisticated, nested UIs as a function of your model data'.

# Frameworks de Apoio à Programação

- **Knockout:**
  - Biblioteca que estrutura a camada 1 de acordo com o padrão MVVM (variação do MVC).
  - Facilita a definição de dependências entre os dados contidos na página. Alterando um valor altera-se automaticamente os valores dependentes (como acontece nas planilhas eletrônicas).

# Frameworks de Apoio à Programação

- <http://vertx.io>

The screenshot shows the vert.x website homepage. At the top is a dark navigation bar with the 'vert.x' logo and links for Home, Download, Install, Tutorials, Examples, Documentation, Source, Google Group, Community, and Blog. The main content area has a light gray background. It features the 'vert.x' logo in large black text, followed by the tagline 'Effortless asynchronous application development for the modern web and enterprise' in a smaller green font. Below this, there are four white boxes with green headers, each describing a key feature: Polyglot, Simplicity, Scalability, and Concurrency. Each box contains a short paragraph explaining the feature.

**vert.x**  
Effortless asynchronous application development for the modern web and enterprise

**Polyglot**  
Write your application components in **JavaScript**, **CoffeeScript**, **Ruby**, **Python**, **Groovy** or **Java**. Or mix and match several programming languages in a single app.

**Simplicity**  
...without being simplistic. Create real, scalable applications in just a few lines of code. No sprawling xml config.

**Scalability**  
Scale using messaging passing and immutable shared data to efficiently utilise your server cores.

**Concurrency**  
Super-simple concurrency model frees you from the hassles of traditional multi-threaded programming.

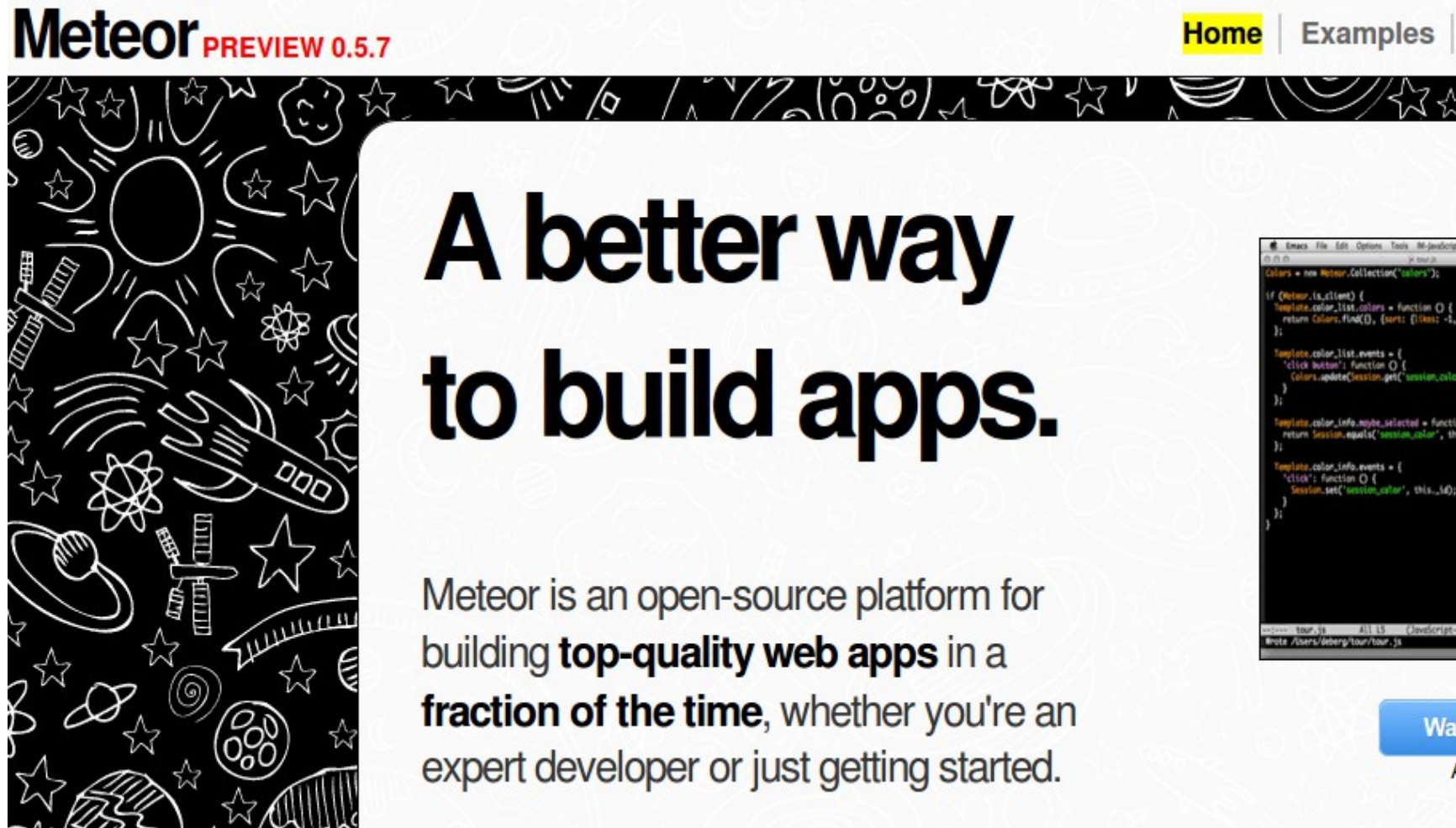


# Frameworks de Apoio à Programação

- **vert.x:**
  - Desenvolvimento de aplicações assíncronas, ao estilo Node.js.
  - Permite outras linguagens no lado servidor.
  - Atua, portanto, na camada 2.

# Frameworks de Apoio à Programação

- <http://meteor.com>



The screenshot shows the Meteor website homepage. At the top, the 'Meteor' logo is followed by 'PREVIEW 0.5.7'. Navigation links for 'Home' and 'Examples' are in the top right. The main visual is a dark banner with white space-themed doodles (stars, planets, rockets) on the left. The central text reads 'A better way to build apps.' Below this, a paragraph states: 'Meteor is an open-source platform for building **top-quality web apps** in a **fraction of the time**, whether you're an expert developer or just getting started.' On the right, a code editor window displays JavaScript code for a 'Colors' application. A blue 'Watch' button is visible at the bottom right of the code editor.

Meteor PREVIEW 0.5.7 Home Examples

## A better way to build apps.

Meteor is an open-source platform for building **top-quality web apps** in a **fraction of the time**, whether you're an expert developer or just getting started.

```
Colors = new Meteor.Collection("colors");

if (Meteor.isClient) {
  Template.color_list.colors = function () {
    return Colors.find({}, {sort: {likes: -1}});
  };

  Template.color_list.events = {
    'click button': function () {
      Colors.update(Session.get("session_color"), {
        $inc: 'likes', 1
      });
    }
  };

  Template.color_info.maybe_selected = function () {
    return Session.equals("session_color", this._id);
  };

  Template.color_info.events = {
    'click': function () {
      Session.set("session_color", this._id);
    }
  };
}
```

Watch

# Frameworks de Apoio à Programação

- **Meteor:**
  - O servidor utiliza o Node.js
  - Usa JavaScript tanto no lado cliente (camada 1) como no lado servidor (camada 2).
  - Atualização automática e sem interromper o usuário.
  - Suporta qualquer biblioteca JavaScript.

# Frameworks de Apoio à Programação

- <http://www.jstat.org>

## jStat : about

About Download Documentation Demonstration

### jStat : a JavaScript statistical library.

jStat is a statistical library written in JavaScript that allows you to perform advanced statistical operations without the need of a dedicated statistical language (i.e. [MATLAB](#) or [R](#)).

### Get the latest version:

Choose your download:

☒ Production (40KB, Minified)

jStat uses numerous advanced statistical functions that require considerable processing power. This requirement results in differing user experiences depending upon your browser choice. Currently, the fastest browser is **Google Chrome**. However, most browsers provide satisfactory performance, with the exception of **Internet Explorer** which has a significant performance hit.

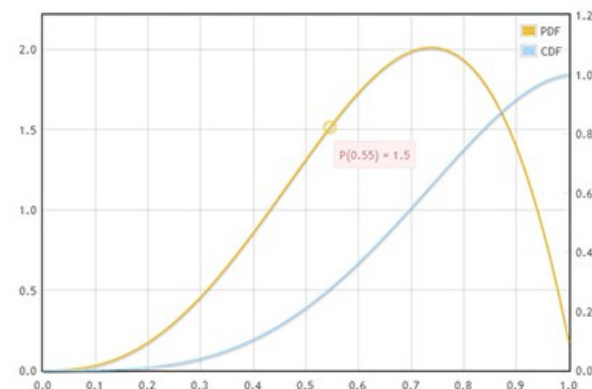
### jStat examples:

jStat was designed with **simplicity** in mind. Using an object-oriented design provides a clean API that can produce results in a few lines of code. jStat also provides a **procedural** API that mimics [R](#).

To demonstrate the **simplicity** of jStat a number of examples are provided below. Clicking on the 'execute' button will run the code of each example.

website soon.

### Screenshots:

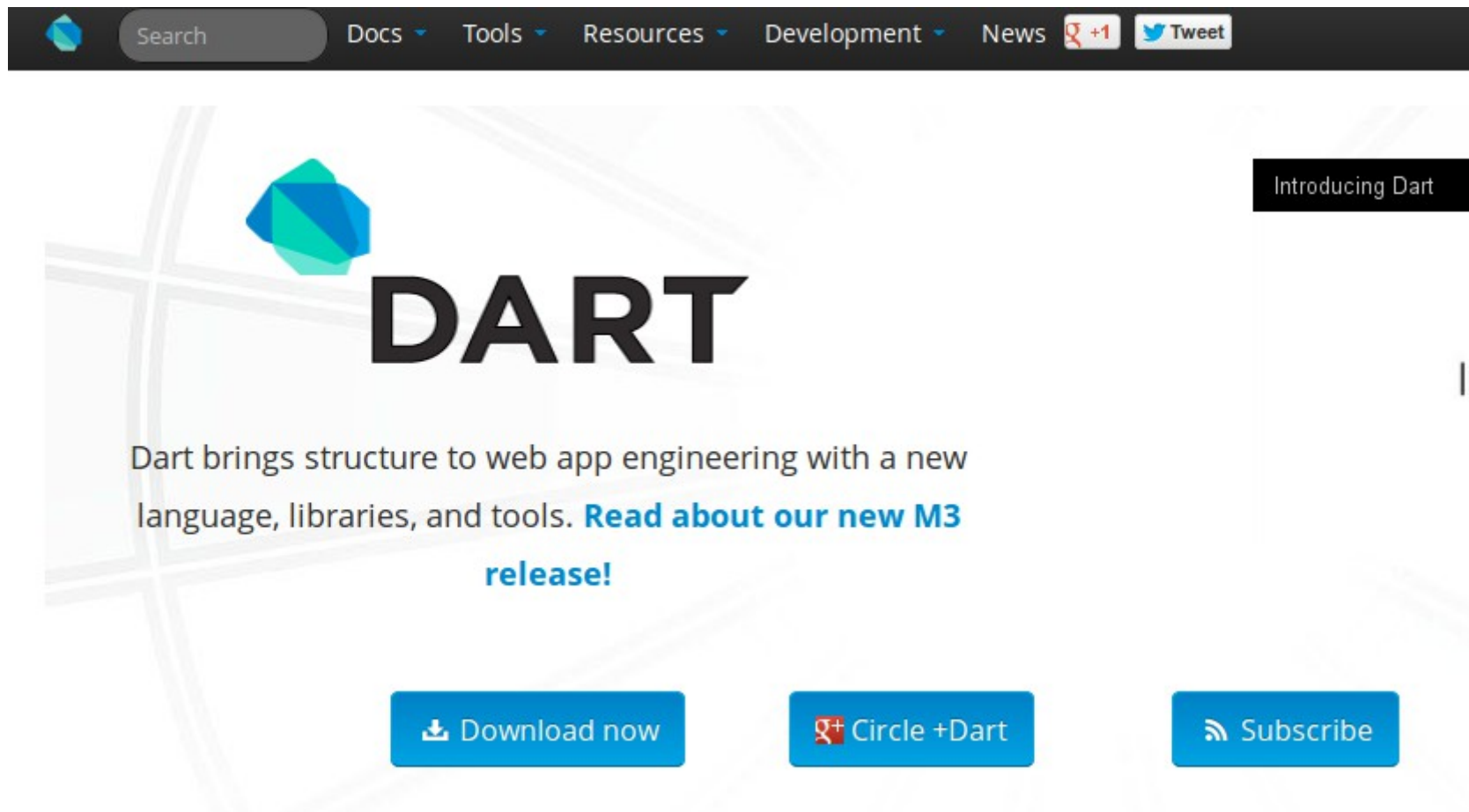


# Frameworks de Apoio à Programação

- **jStat:**
  - Biblioteca de funções estatísticas.
  - Usada na camada 1.
  - Em aplicações para web 2.0 o processamento na camada 1 (navegador) pode ser bastante intenso (uso da CPU).

# Frameworks de Apoio à Programação

- <http://www.dartlang.org>



# Frameworks de Apoio à Programação

- **Dart:**
  - Aposta do Google como sendo a linguagem de programação que (deveria) entrar no lugar de JavaScript.
  - Motivação é criar uma máquina virtual, chamada Dart VM, mais eficiente.
  - Pode ser usada tanto na camada 1 como na camada 2.