Carnegie Mellon

17-356/17-766 SOFTWARE ENGINEERING FOR STARTUPS

institute for SOFTWARE RESEARCH | Carnegie Mellon University School of Computer Science

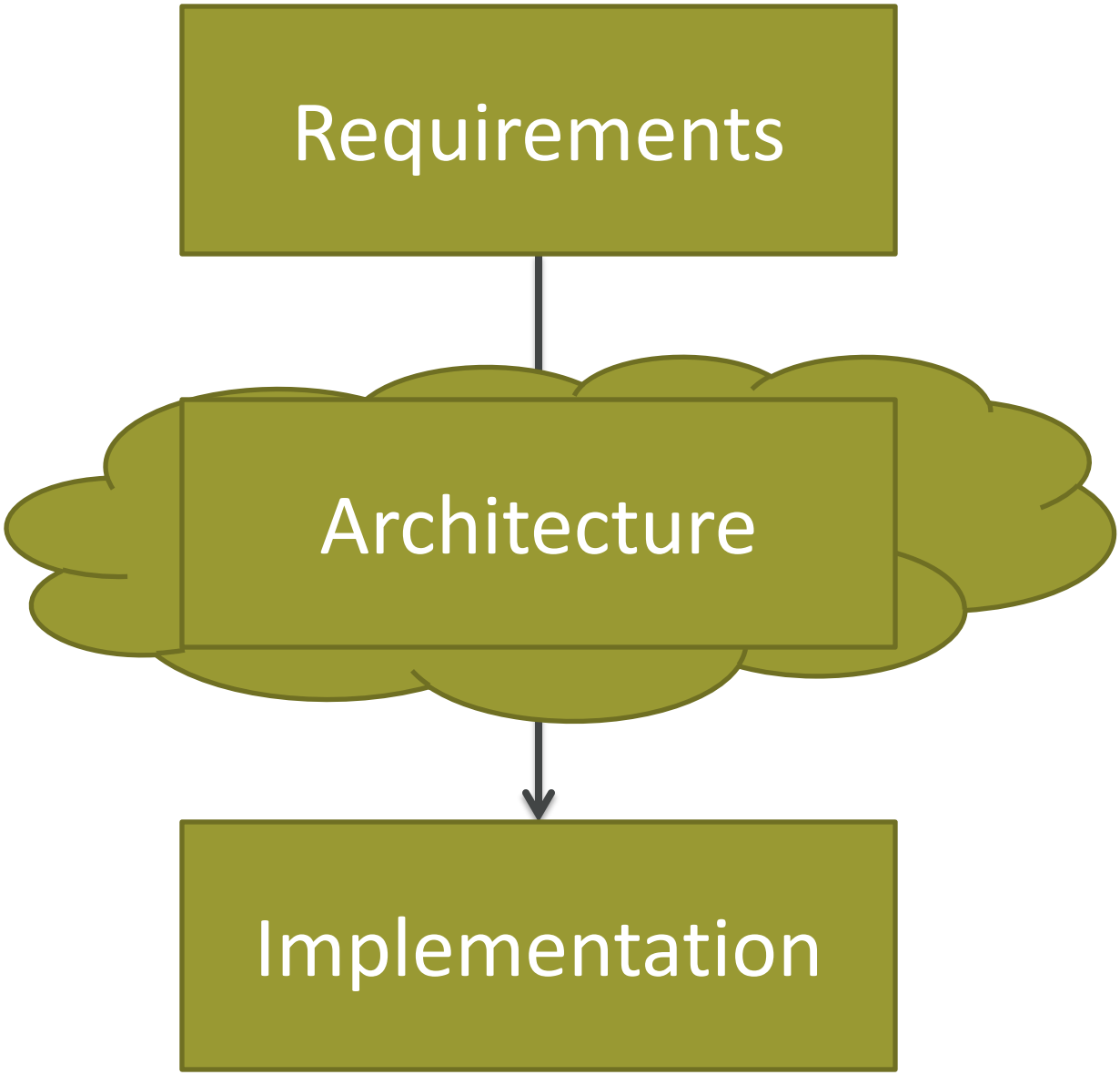
Michael Hilton & Heather Miller

Administrativa

HW2: don't delay on deployment!

limit your backlog to 40 hours

"write a business-style email justifying your choice of framework to your investors."



Software Architecture

The software architecture of a program or computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them.

[Bass et al. 2003]

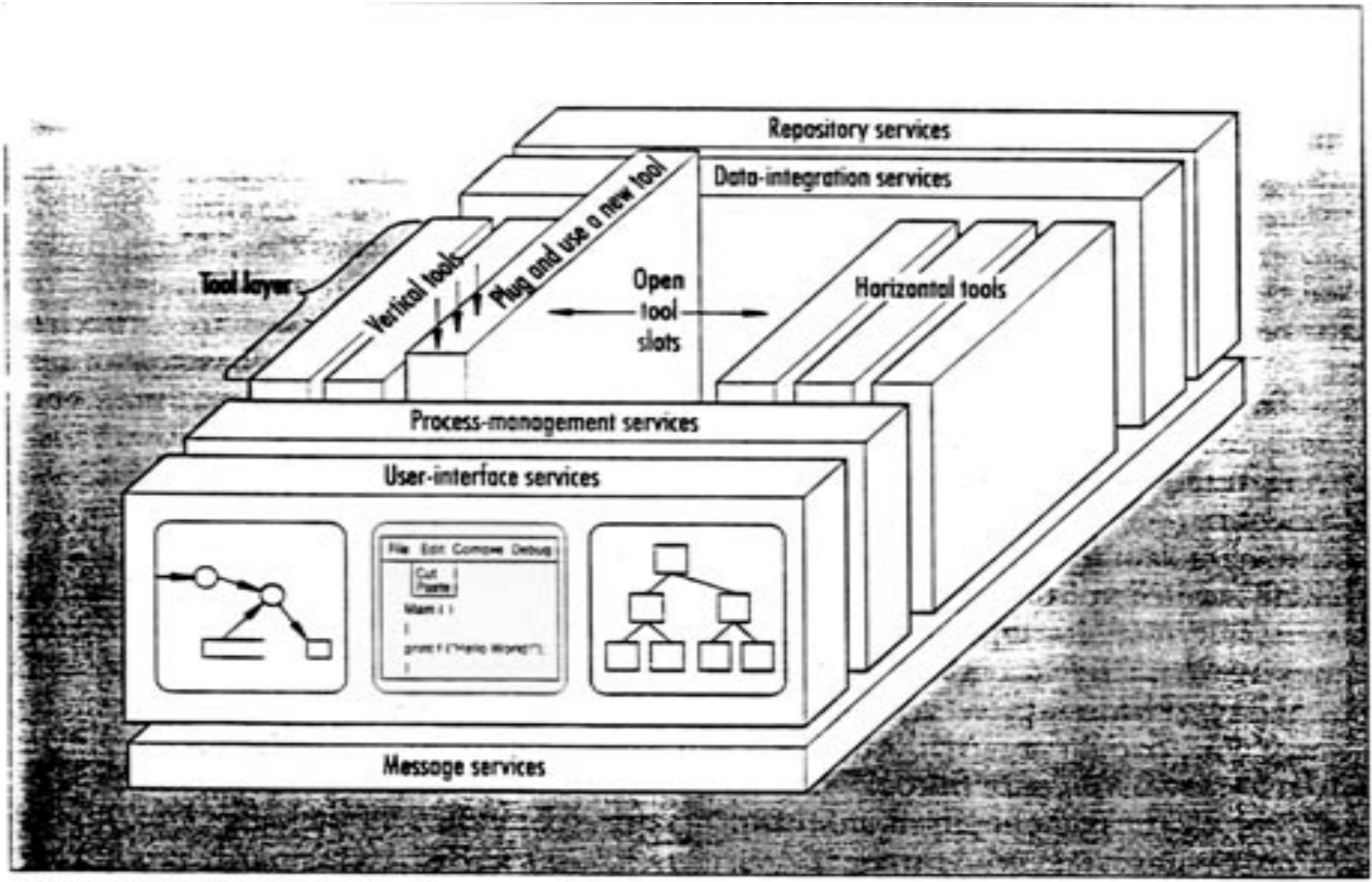
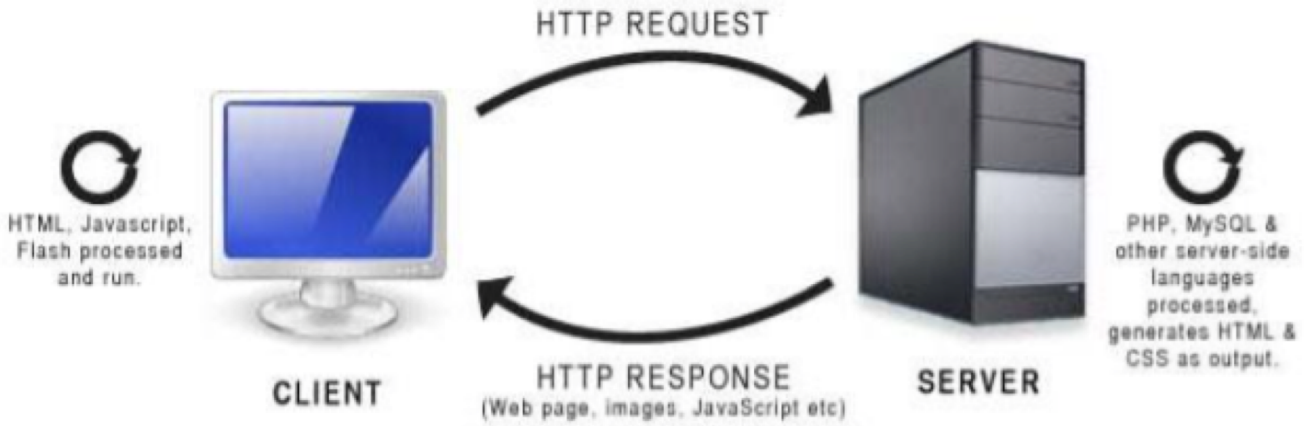


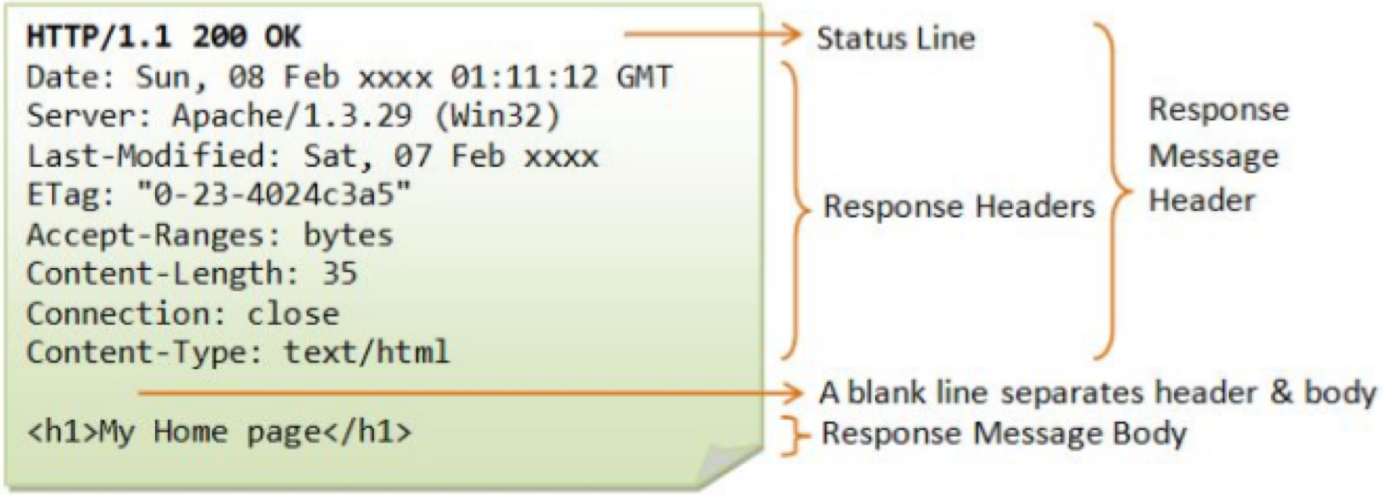
Figure 1. The NIST/ECMA reference model.

IN PRACTICE, IN MODERN DEVELOPMENT, ESPECIALLY WEB APPS, VERY LITTLE OF THAT MATTERS.

WEB APPLICATION DEVELOPMENT CRASH COURSE

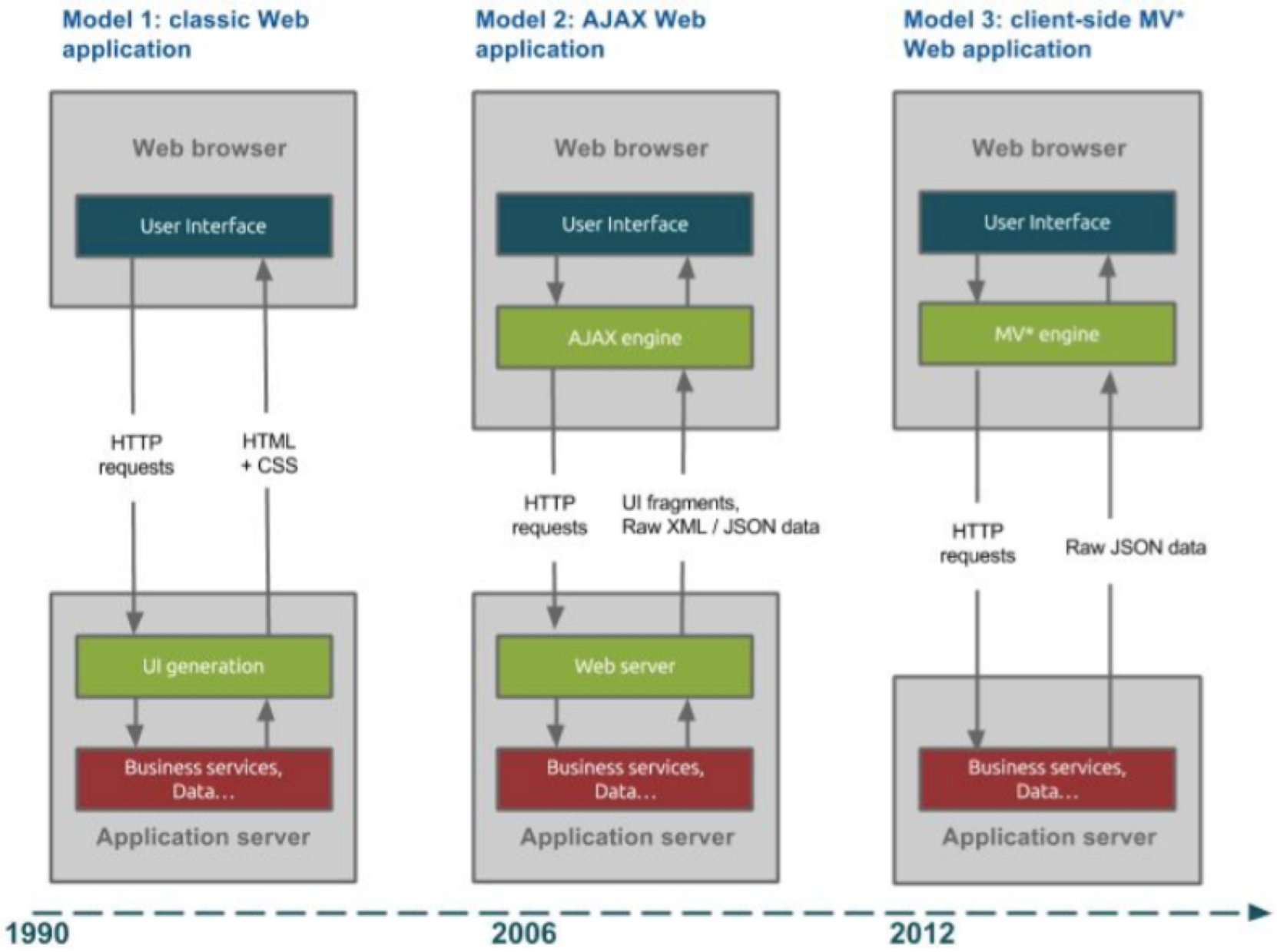
institute for SOFTWARE RESEARCH



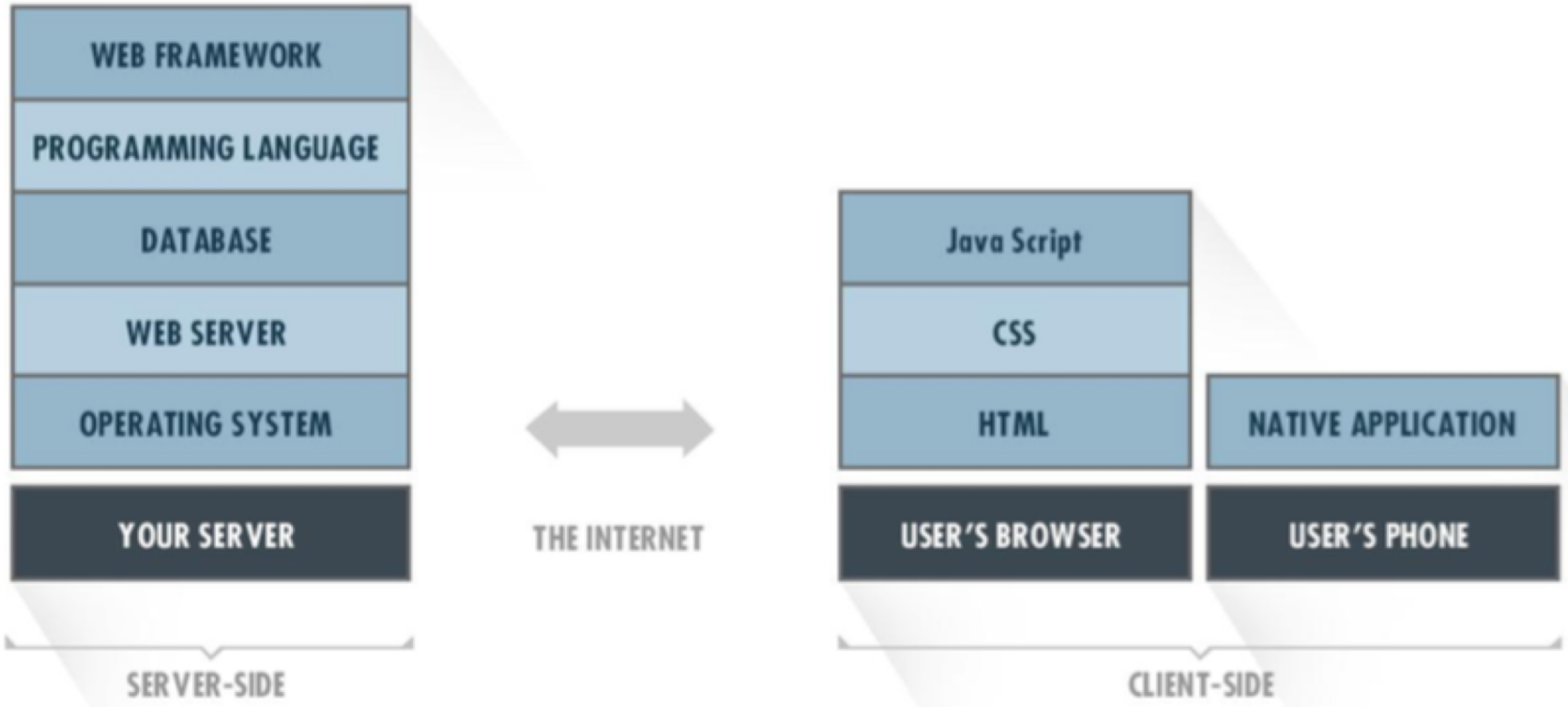


Slides borrowed wholesale and liberally from Alexey Zagalsky, Uvic
https://speakerdeck.com/alexeyza/web-development-overview-2016

https://www.thoughtworks.com/insights/blog/implications-tech-stack-complexity-executives



http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/



http://svsg.co/how-to-choose-your-tech-stack/

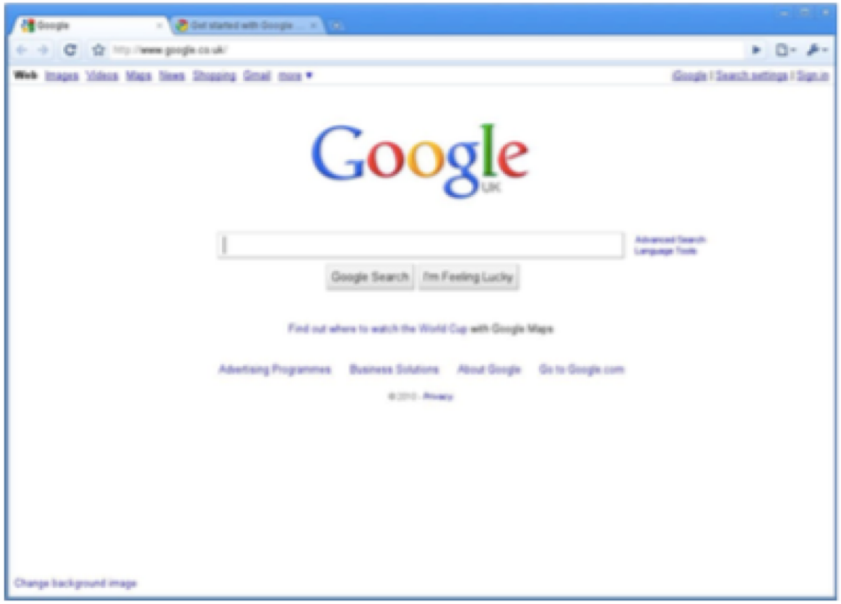
Web browser, native/iOS/Android/etc

CLIENT SIDE

Roles of the Browser

Networking

- HTTP



Rendering

- HTML
- CSS

Business Logic

- VM (Java, Silverlight, Flash)
- Javascript

HyperText Markup Language (HTML)

HTML is the standard **markup language** used to create web pages.

HTML elements:

- Head: title, base, link, style, meta, script
- Body: titles, paragraphs, text style, lists, tables, images, links, forms

Every HTML element can be assigned with an **id attribute** to allow manipulation with JavaScript or CSS.

```
<html>
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h1 id="greetings">Hi</h1>
        <p>Minimal <b>hello world</b> page.</p>
    </body>
</html>
```

Cascading Style Sheets (CSS)

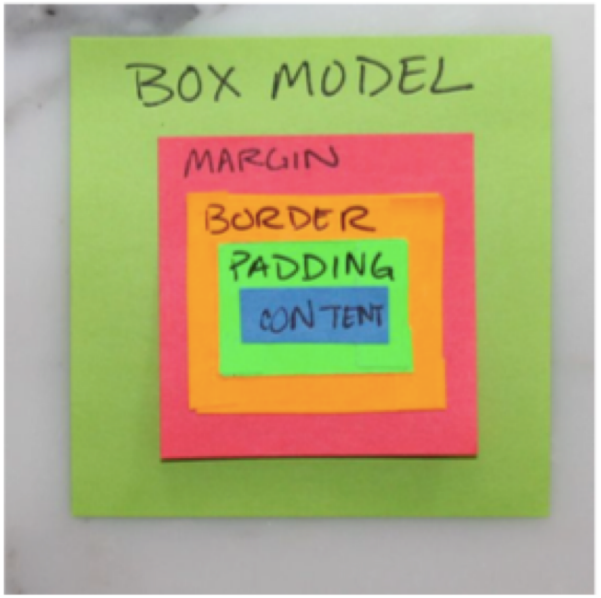
Semantic Tags vs. Design Tags

<p>, <h1> vs. **<font>, <b>**

No model-view separation

CSS Zen Garden (http://www.csszengarden.com/)

Preprocessors: **SCSS** vs **LESS**

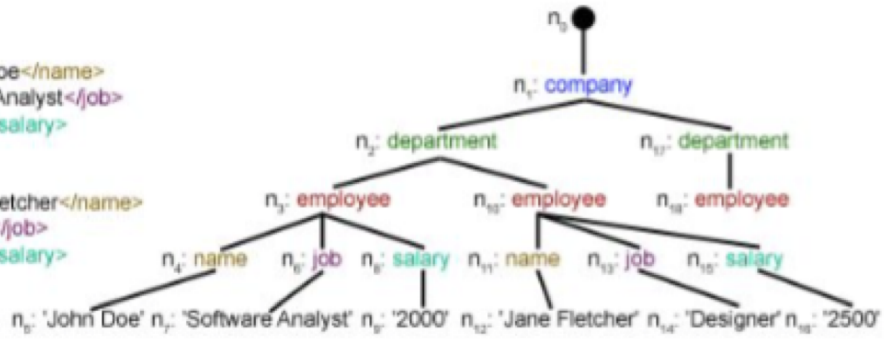


http://www.labnol.org/internet/learn-css/29126/

Extensible Markup Language (XML)

Labeled ordered **tree**

```
<company>
 <department>
  <employee>
   <name>John Doe</name>
   <job>Software Analyst</job>
   <salary>2000</salary>
  </employee>
  <employee>
   <name>Jane Fletcher</name>
   <job>Designer</job>
   <salary>2500</salary>
  </employee>
 </department>
 <department>
  <employee>
  </employee>
 </department>
</company>
```



Many available tools to:

generate, parse, transform, send

Mainly used for:

Configuration

Declarative (meta) programming

Platform independent data exchange (RSS feeds)

JavaScript "The programming language of the Web"

Prototype-based **scripting language** with dynamic typing

Fully fledged programming language: variables, arrays, objects, methods, conditional statements, loops, and **events**.

Simple and forgiving syntax:

Type declaration optional

Lambda notation

JSON

```
<html>
     <script>
          var x = 3;
          var y = 47;
          alert( " 3 x 47 = " + (x*y) );
     </script>
</html>
```

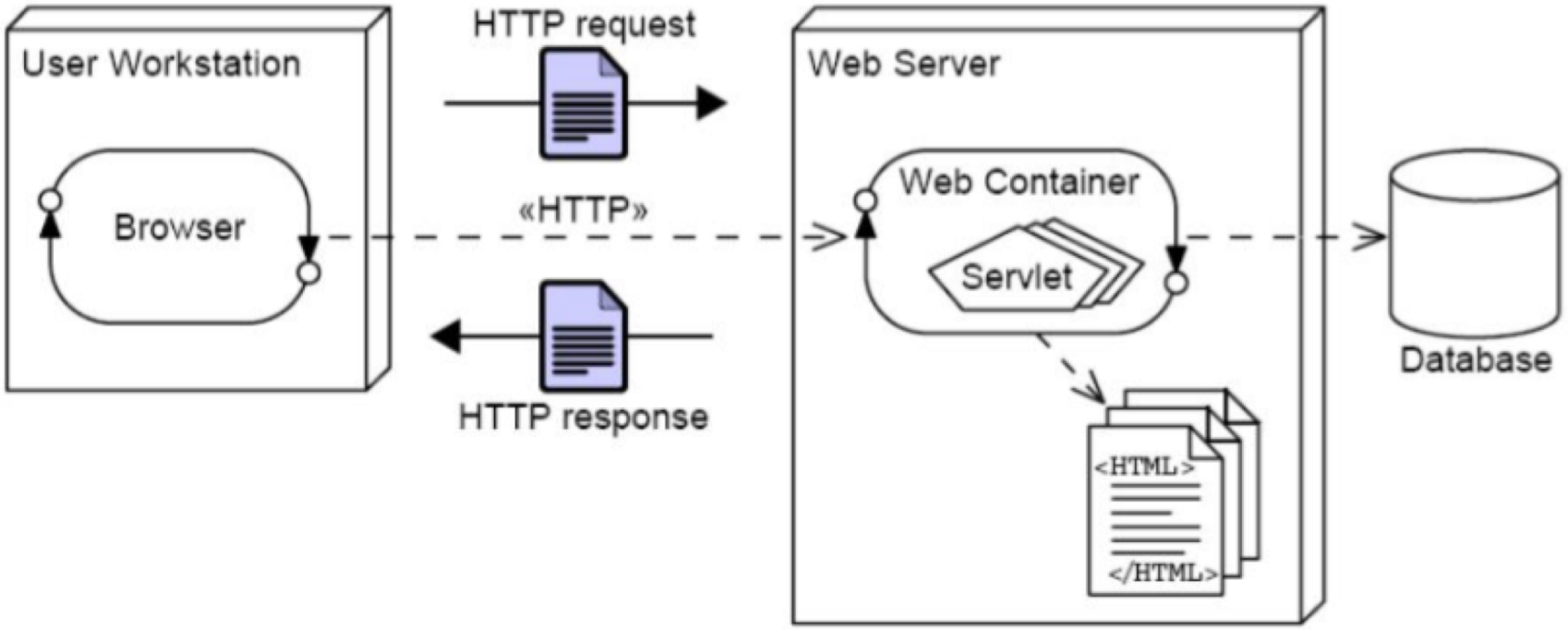
Compatibility issues between different browsers (JavaScript, JScript, ECMAScript)

```
<h1 onMouseOver="style.color='red';"onMouseOut="style.color='blue';">Hello </h1>
```

Web server, databases, authentication, etc…

SERVER SIDE

Web Containers / Web Frameworks



Roles of web container:

Networking, Thread management, Servlet management, HTTP protocol management

GET vs. POST

The **HTTP GET** method is used when:

- The processing of the request is idempotent
- The amount of form data is small
- You want to allow the request to be bookmarked

The **HTTP POST** method is used when:

- The processing of the request changes the state of the server, such as data in a database
- The amount of form data is large
- The contents of the data should not be visible in the URL (e.g. passwords)

Web Template System

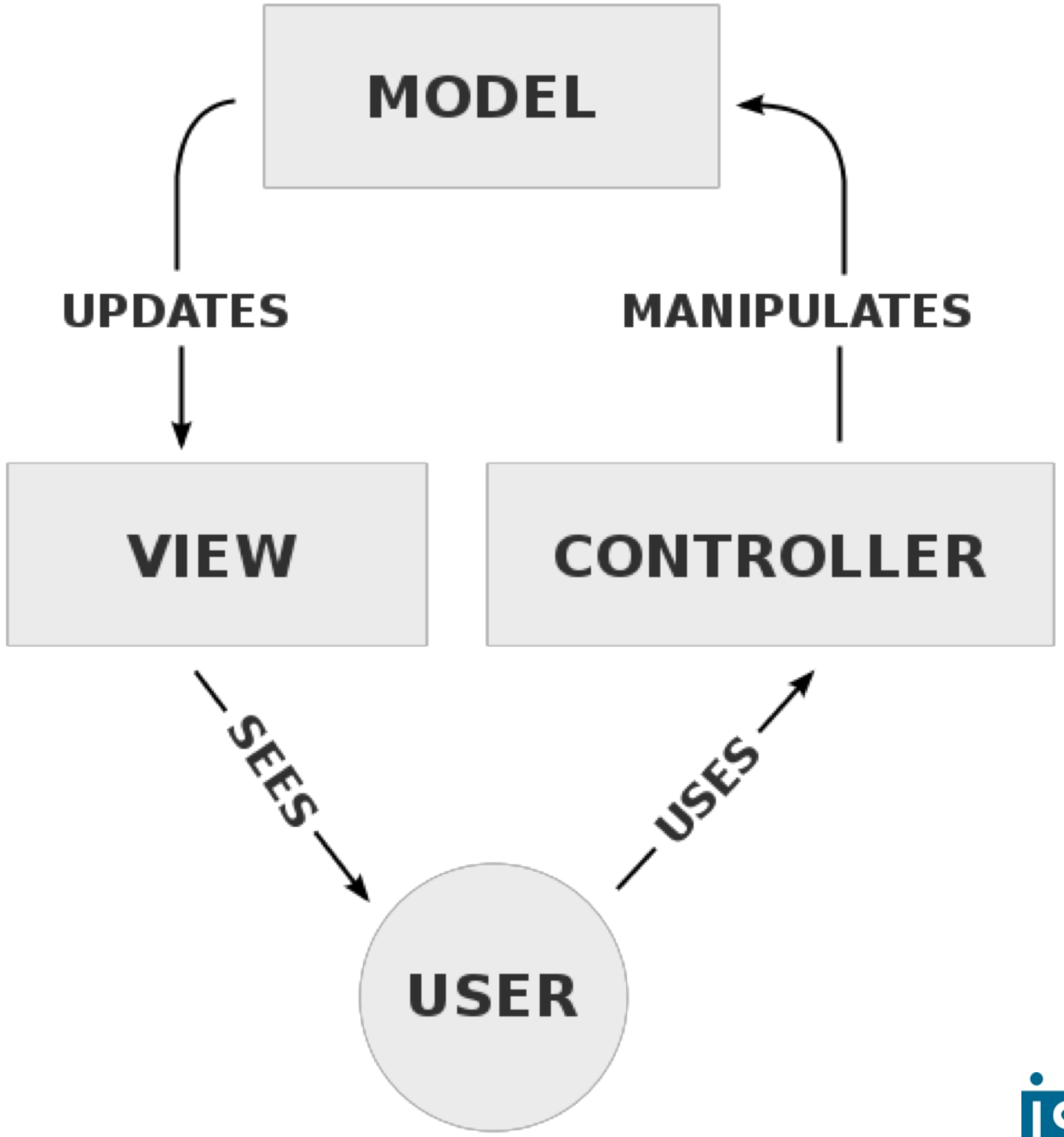
A web template system is composed of:

- A template engine (EJS, DJango)
- Content resource (DB)
- Template resource (JSP, PHP)

Benefits:

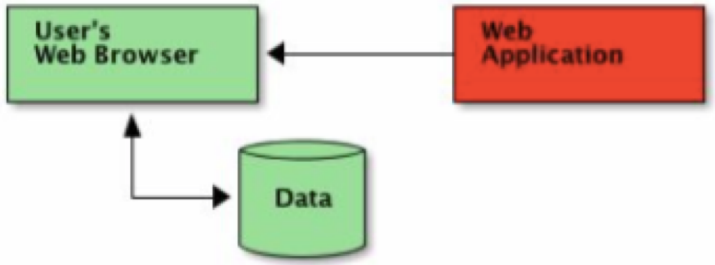
- Separation of concerns (Model/View)
- Mass production of content
- Flexible presentation
- Reusability
- Style standardization

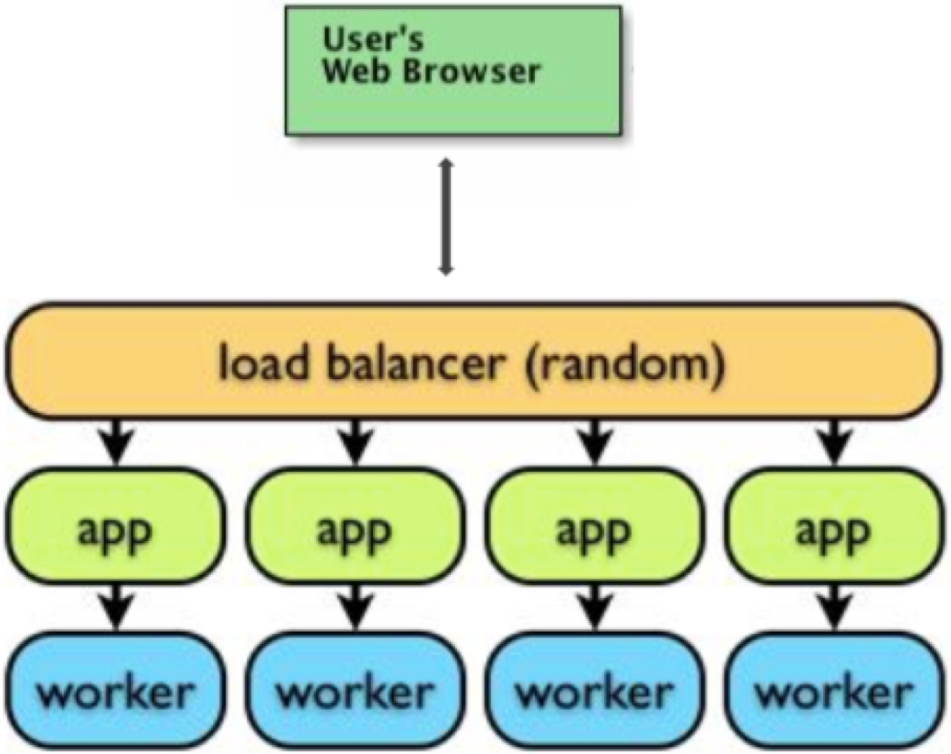
```
</script>
</head>
<body bgcolor="#ffffff">

<h1>Office Locations</h1>
<table border="1">
  <tr>
    <th>Location ID</th>
    <th>Street Address</th>
    <th>City</th>
  </tr>
  {% for loc in locations %}
  <tr>
    <td><a href="http://this_link_goes_nowhere/"
        onClick="makeRequest{{{loc.location_id}}}); retu
        {{loc.location_id}}<a></td>
    <td>{{loc.street_address}}</td>
    <td>{{loc.city}}</td>
  </tr>
  {% endfor %}
</table>

<p>
<div id="outputNode"></div>
</p>

</body>
</html>
```

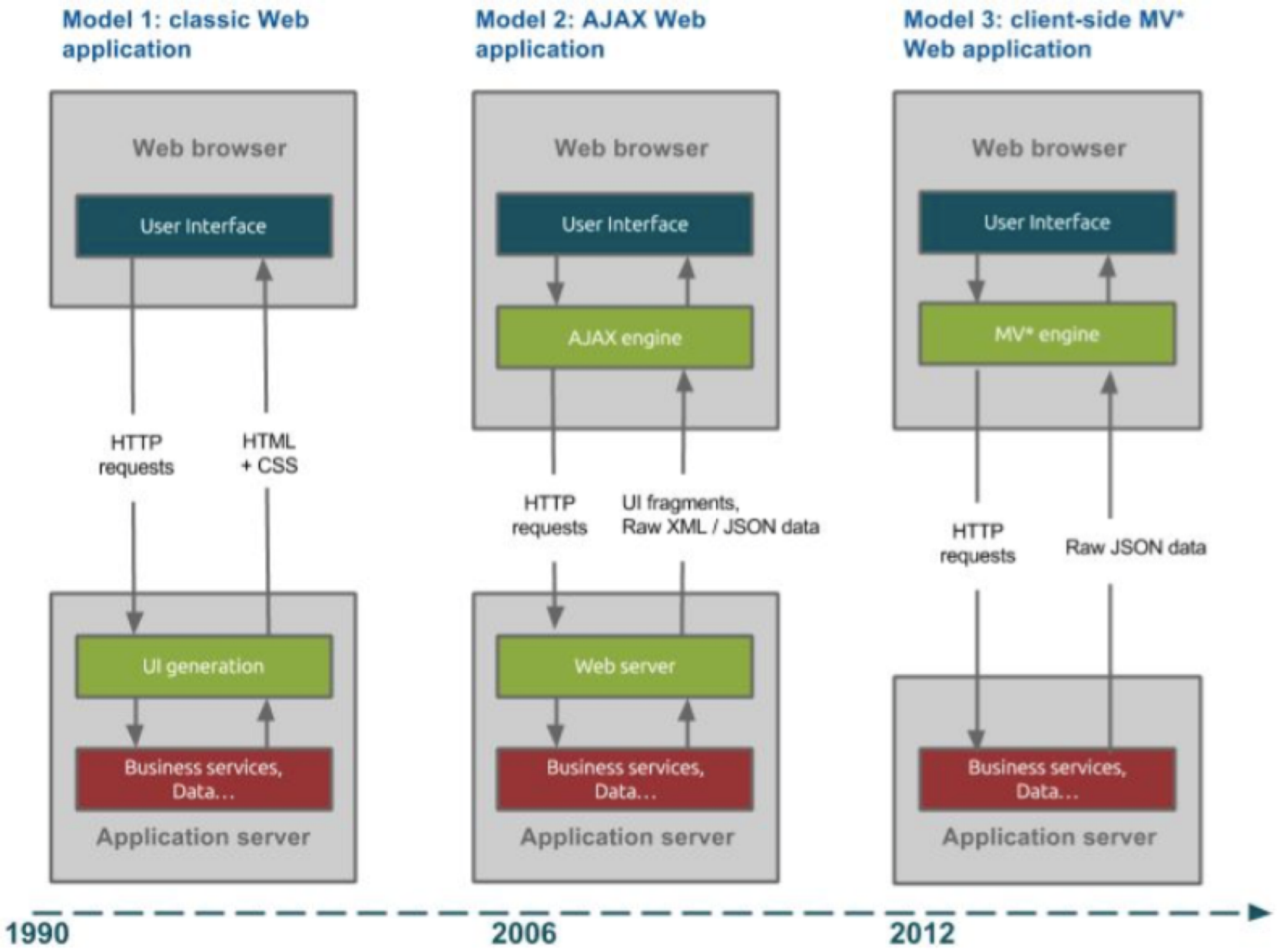


Typical LAMP hosted webapp

Serverless webapp







http://blog.octo.com/en/new-web-application-architectures-and-impacts-for-enterprises-1/

Wait, so what should we use?

Frontend

HTML, CSS, JavaScript, jQuery...

Backend

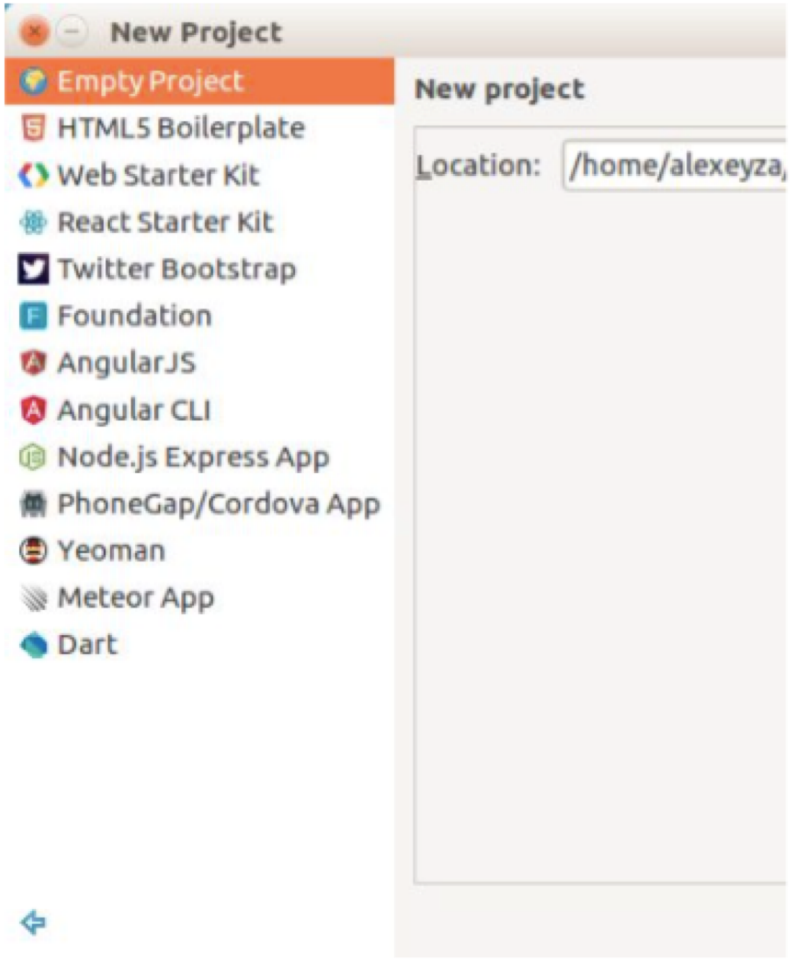
Java, Python, Node.js, Go, Ruby on Rails...

Hosting

Google App Engine, Heroku, AWS...

IDE / Text Editor

Eclipse, Webstorm, SublimeText...



"A starting architecture, with supporting libraries, to support a general programming goal, typically using inverted control."

WHAT IS A FRAMEWORK?

Examples? And how do we choose between them?

- <I'd have made a list, but I bet we can just name a bunch, eh?>

Sorry/Not Sorry

HONESTY TIME: WE'RE BOXING YOU IN

PATH TO PROTOTYPES

Product

MVP

Prototype

Prototyping a web app

- Front end (View): Paper prototypes
- Back end (Database): Data Model
- “Middle end”: API, endpoints

PROTOTYPE TO ITERATE OVER AND REFINE IDEAS…SO WHERE DO IDEAS COME FROM?

tl;dr: Geniuses don't have better ideas, they have more of them.

Creativity can be cultivated through practice.

- –Creativity style (innovative versus adaptive) doesn't correlate with how many ideas people generate.
- We are most creative when we turn off evaluations and restrictions.
- –Warning re: startup!

Alternate between exploring problems in depth versus laterally.

Accept bad ideas – "yes and" them – point out the good point of an idea if you can find one and mention how it could be improved with the and statement.

Guidelines:

- –Defer judgment
- –Generate as many options as possible (more is better)– ask what else
- –Seek novelty
- –Then, converge on the best ideas (spend about as much time on this as generating ideas)

ACTIVITY: BRAINSTORM SNOW PLOWING FUNCTIONALITY