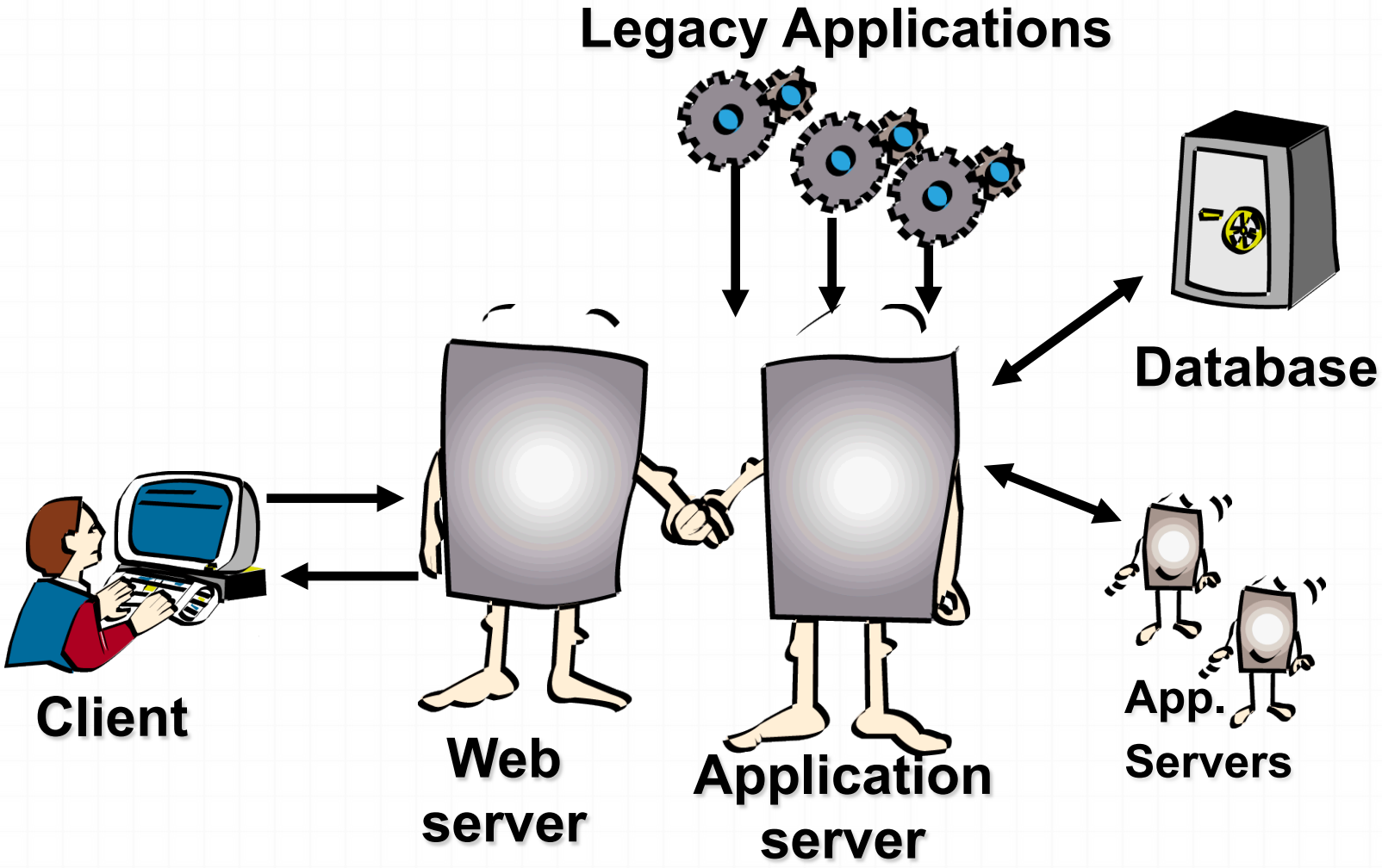
HTML 5

An Introduction

But before… a quick introduction to Web applications

Reference Web Application Architecture



Web Application Standards
HTTP

- HyperText Transfer Protocol
  - Application-level networking protocol for the exchange of multimedia documents
  - a request-response protocol in the client-server computing model.
- Defines the format of
  - Resources identification (URL)
  - Requests
  - Responses
- Versions: HTTP/0.9, 1.0, 1.1

Web Application Standards HTML (1)

- HyperText Markup Language
  - A markup language used to describe the content and structure of an hypertext
  - The first publicly available description of HTML was a document called "HTML Tags", first mentioned on the Internet by Berners-Lee in late 1991
- HTML markup consists of several key components
  - elements (and their attributes)
  - character-based data types
  - character references and entity references
  - Type declaration
    - triggers standards mode rendering.

Web Application Standards HTML (2) - Example

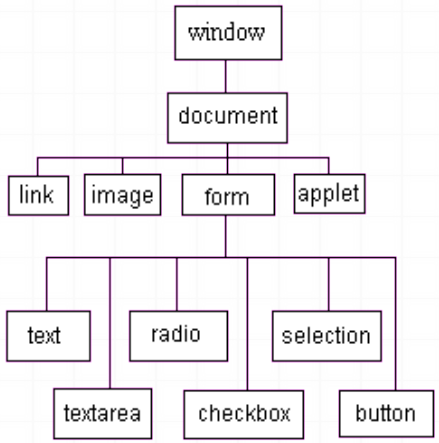
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://
www.w3.org/TR/html4/strict.dtd">
<HTML>
    <HEAD>
        <TITLE>The document title</TITLE>
    </HEAD>
    <BODY>
        <H1>Main heading</H1>
        <P>A paragraph.</P>
        <UL>
            <LI>A list item.</LI>
            <LI>Another list item.</LI>
        </UL>
        <A href="http://www.w3schools.com”>Link</A>
    </BODY>
</HTML>
```

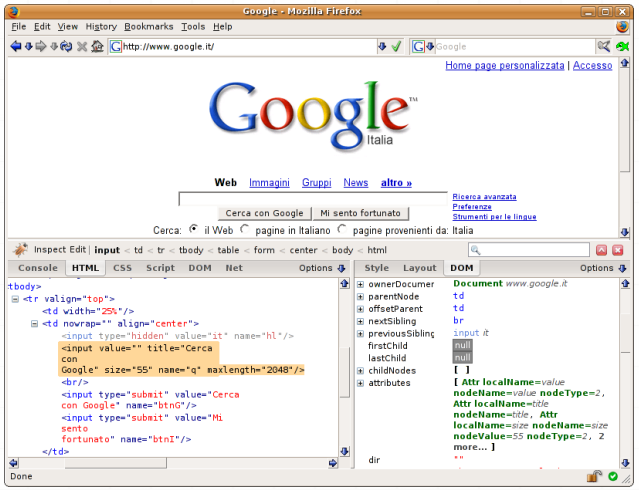
Web Browser

- An application able to:
  - Access Internet using the HTTP protocol
  - Requesting resources (identified by an URL) to Web servers
  - Interpreting resources and rendering them to the user
    - **HTML** pages, pictures, video, etc.
  - Receive input from the user, and send them to the server

Web Application Standards
DOM

- DOM (Document Object Model)
  - Standard object model (W3C) for the representation of HTML and XML documents
  - 4 levels (da L0 a L3)
  - Browsers supports L2 and some L3 (tree-based navigation, content manipulations, events)





Web Application Standard Javascript

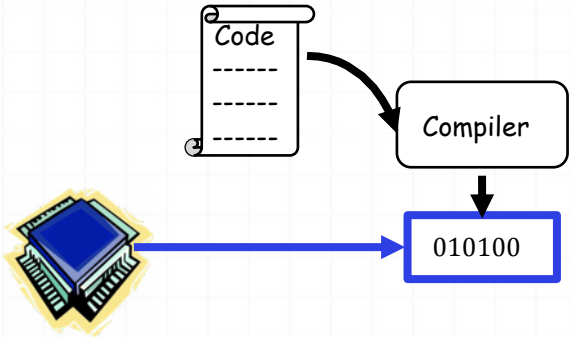
- JavaScript is an important language because it is the language of Web browsers
- The standard that defines JavaScript (aka JScript) is the third edition of *The ECMAScript Programming Language, which is available from http://www.ecmainternational.org/publications/files/ecma-st/ECMA-262.pdf.*

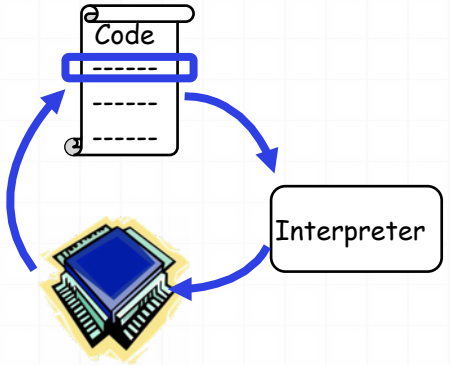
Javascript is not Java!

- JavaScript: Object Oriented scripting language
  - Interpreted and executed in a browser
  - No plug-in
- Java: multi-platform, general purpose programming language
  - Compiled in a bytecode and then executed in a VM
  - As a standalone program, or in a browser as an applet
  - Strongly typed

Javascript Features

- Most programming languages today demands strong typing (e.g. Java)
  - Compilers can detect a large class of errors at compile time
- JavaScript is a **loosely typed**, **interpreted** language
  - JavaScript interpreters cannot detect type errors but
  - No need to form complex class hierarchies
  - No casting
- Functions are first class objects with lexical scoping





Client Side Scripting

- Initially used for simple functionalities:
  - Image animations and menus(onMouseOver)
  - Simple widgets (e.g., clock, calendars)
  - Form validations
- Now: pillar of Web application development, mostly thanks to AJAX (Asynchronous JavaScript And XML)
  - Reactive Interfaces
    - Asynchronous client to server communication
  - Distribution of computation and data from server to client
    - Communication reduced to the minimum

Client Side Scripting

- Browser
  + Script Interpreter
  + Document Object Model (DOM)
  = Client-side JavaScript

- JavaScript operates on the DOM to give a dynamic behavior to HTML documents

Client Side Scripting Example

Definizione funzione

Navigazione DOM

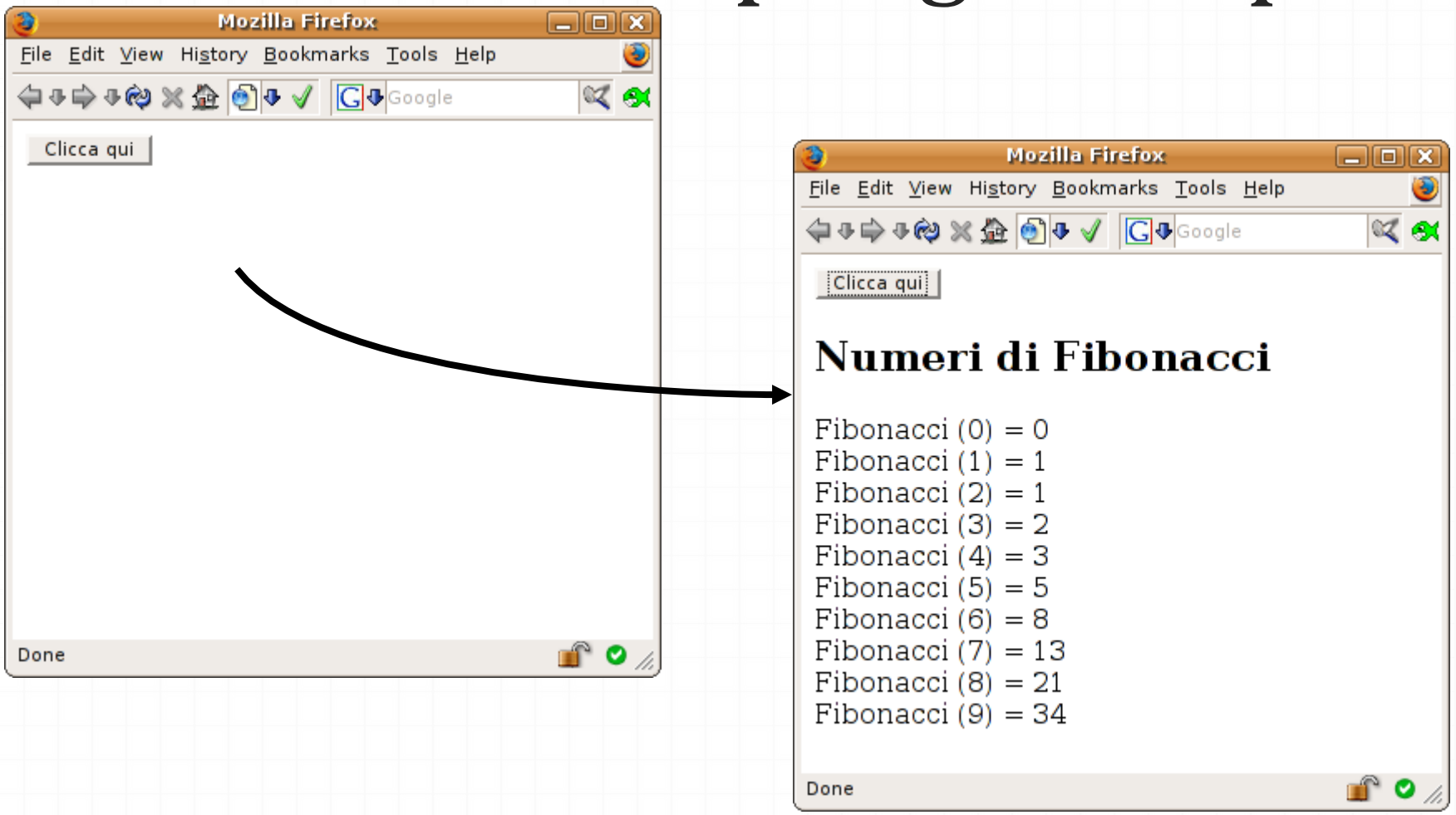
```
<html>
 <head>
 <script language="JavaScript">
  function fibo(){
   var c = document.getElementById("content");
   var header = document.createElement("<h2>");
   header.appendChild(document.createTextNode("Numeri di Fibonacci"));
   c.appendChild(header);
   for (i=0, j=1, k=0, fib =0; i<10; i++, fib=j+k, j=k, k=fib){
     c.appendChild(document.createTextNode("Fibonacci ("  + i +  ") = " + fib));
     c.appendChild(document.createElement("<br>"));
   }
  }
 </script>
 </head>
 <body>
    <button onclick="fibo();">Clicca qui</button>
    <div id="content"></div>
 </body>
</html>
```

Aggiunta di un nuovo elemento

Evento

Event Handler

Client Side Scripting Example

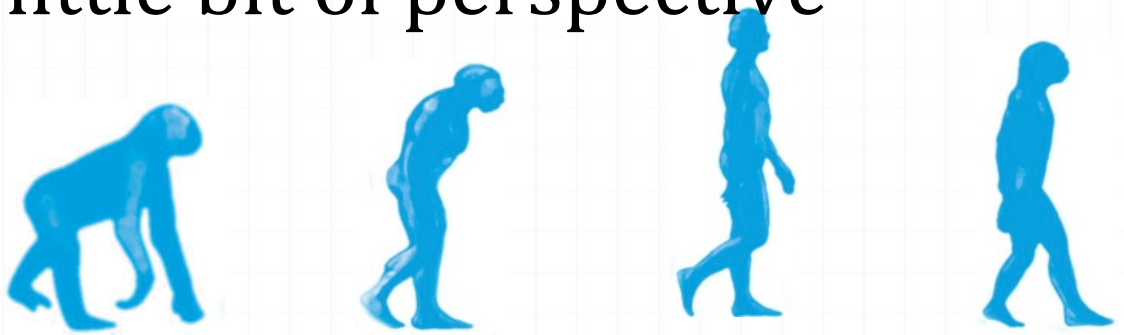


Problems with Web 1.0 Web applications

- Really complex to cope with the requirements of modern user experience
  - Page-centric user interface (multi-steps, page flipping, etc.)
  - Insufficient interactive data visualization capabilities (drill down on charts)
  - Data filtering just on server-side
  - Full page refresh, interruptive communication with servers
- From a usability point of view, HTML was a huge step backward

Rich Internet Applications

- Let's get a little bit of perspective



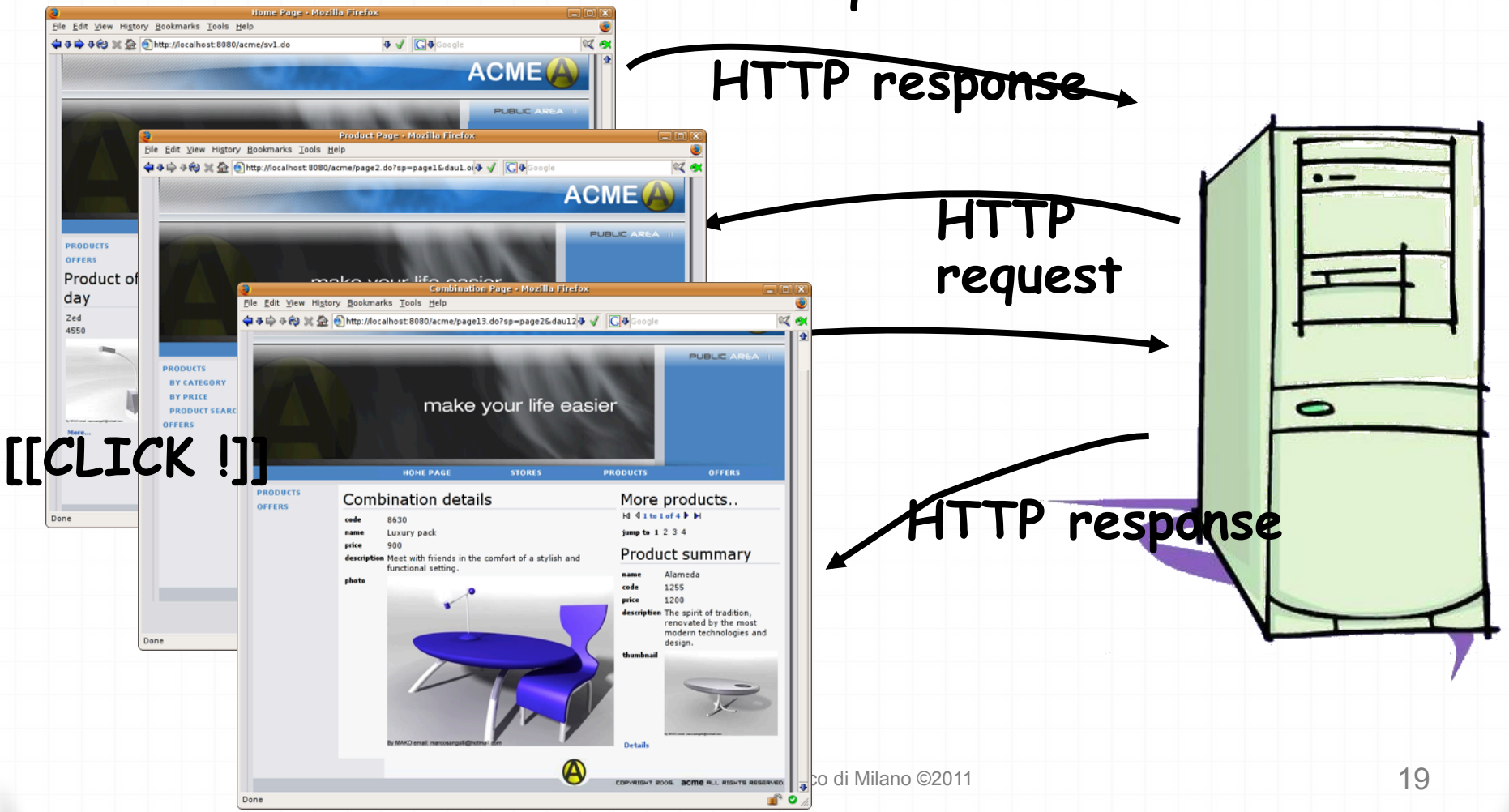
		Mainframe	Desktop	Client/Server	Websites
User Interface	Interactivity	None: dumb green-screen or command-line terminals	High: Drag-and-drop, point-and-click	High: Drag-and-drop, point-and-click	Low: Point-and-click, form fill-in
	Flexibility	None: No customization possible	High: resizable components, configurable display, local data, custom shortcuts	Medium: resizable components, configurable display, server-side data	Low: limited customization of page appearance
	Power	None: only displays data sent by server	Medium: real-time computation, complicated information visualization	High: real-time computation coupled with access to server-side data	None: only displays data sent by server
Deployment		Low cost	High cost	High cost	Low cost

Slide Source: Forrester Research, Inc.

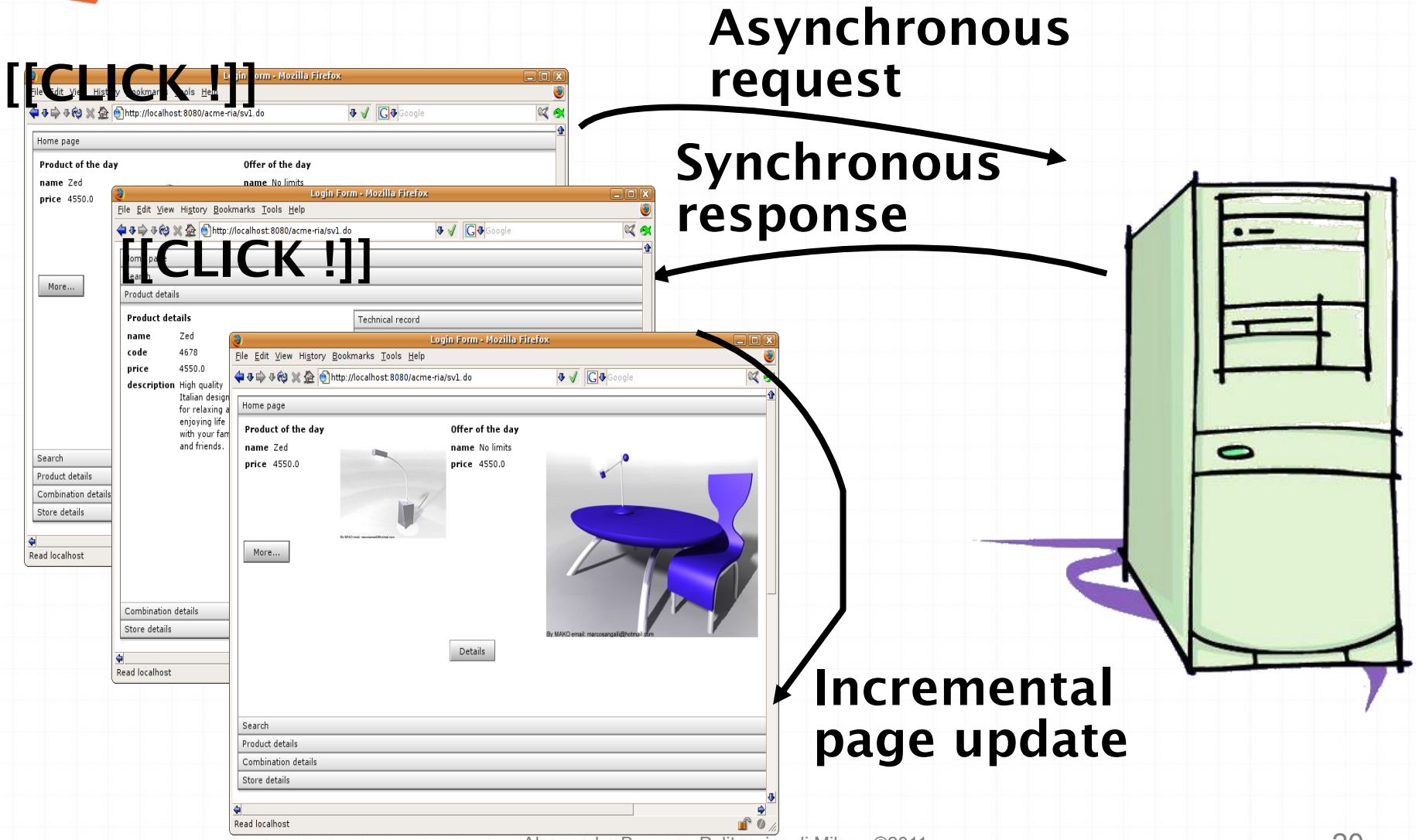
AJAX

- AJAX is a technique for building Rich Internet Applications (RIA)
  - Thanks to XMLHttpRequest object
- A RIA:
  - Is a Web application
  - With visual and interaction features similar to the ones of a Desktop application
  - Executed within a browser
  - No installation required
  - (possibly) working on-line and off-line

AJAX (2)
Traditional Web interaction



AJAX (3)
RIA interaction



Javascript / Ajax References

- Javascript: The Good Parts
  - Douglas Crockford
  - O'Reilly Media/Yahoo Press, 2008
- JavaScript Cookbook
  - Shelley Powers
  - O'Reilly Media
- http://javascript.crockford.com/
  - **videos**!
- W3C Javascript Tutorial
  - http://www.w3schools.com/js/default.asp
- HTML.it Javascript
  - http://javascript.html.it/
- "Head Rush AJAX" di Brett McLaughlin
- "AJAX Hacks" di Bruce W. Perry
- API XmlHttpRequest
  - http://www.w3.org/TR/XMLHttpRequest/

Introducing HTML 5

What is HTML 5?

- More a *movement* than a unique, identifiable technology
  - (Ongoing) standardization effort for several new features and existing de-facto standards
  - Continuous innovation

- **More semantic to HTML pages**!
  - New tags and semanrics
- **More power to the browser**!
  - Browser as a programming platform → new Javascript APIs

Why is HTML5 important?

- HTTP5 + CSS3 + Javascript overcome some of the drawbacks of the "old" HTTP + HTML4 standards
  - State transfer (no local storage, no off-line functioning)
  - HTML4 tags have no implicit semantics
    - **Huge problem for Search Engines**

- Advantages
  - Increased expressiveness
  - Machine readability
  - Increased cross-browser compatibility
  - More advanced features delegated to the browser
  - More power to the client
  - More engaging user experience

Basics HTML5 Elements

Example HTML 5 Page

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8"/>
    <title>My HTML 5 Page</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
    <link rel="alternate" type="……" />
  </head>
  <body>
    <header> … </header>
    <article> … </article>
    ……
  </body>
</html>
```

Doctype

Root Element

Character Encoding

Link Relations

New Semantic Elements in HTML5

Doctype

- There are > 70 page doctypes that instructs browsers about the right “rendering mode” to activate
  - Quirks Mode: browsers violate Web format specifications to avoid “breaking” pages
  - Standards Mode: browsers try to giver conforming documents the right treatment to documents, according to their doctype

- There are 15 doctypes that trigger standard mode

  <!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 String//EN”
  http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd >

- In HTML5 there is only one doctype

  <!DOCTYPE html>

The Root Element

- The root element of an HTML page is always

```
<html>
```

- A typical root definition in XHTML is

```
<html xmlns:"http://www.w3.org/1999/xhtml"
                              lang="en" xml:lang="en">
```

- In HTML5
  - there is no need for namespaces, as all the HTML5 elements are always in its namespace
  - The lang attribute suffices to define the language of the HTML5 page

```
<html lang="en">
```

Character Encoding

- Every piece of text in a computer is stored using a given *character encoding (CE)*
  - Whenever someone gives you a sequence of bytes claiming that it is *text*, the *CE* provides a mapping between what the computer manages and what you see
- The CE of a Web page is usually specified by the Web server in the HTTP header

```
Content-Type: text/html; charset="utf-8"
```

- But authors do not always have control on the Web server. The <meta> tag specifies the CE for a Web page

```
<meta charset="utf-8" />
```

- But the the HTTP header, if present, **overrides** the <meta> tag

Link Relations

- Link relations explain ***why*** a page points to another resource
  - Typically specified by/for browsers, blogging platforms, search engines, etc.

- `rel="stylesheet"` → link to a file containing CSS rules for the page

```
<link rel="stylesheet" href="file.css" type="text/css"/>
```

- `rel="alternate"` → used to indicated that the same content is available in other formats

```
<link rel="alternate" href="/feed/"
      type="application/atom+xml" title="My Weblog feed"/>
```

  - Typically used to enable "feed autodiscovery"

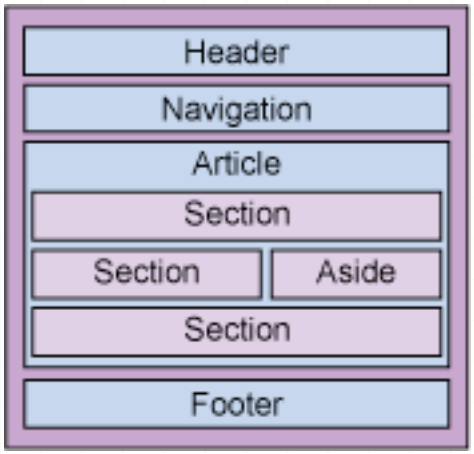
(some) Other Link Relations

- `rel="author"` → link information about the author of the page. It can be a `mailto:` address
- `rel="icon"` → link to associate a small icon with the page
- `rel="license"` → license term for the page's content
- `rel="nofollow"` → indicates that the link is not endorsed by the original author or publisher of the page
- `rel="pingback"` → specifies the address of a "pingback" server
- `rel="search"` → the referenced document provides an interface for searching the document and its related resources (typically an *Open Search* document)

New Semantic Elements in HTML5

- Goal: enrich HTML with semantics
  - Typical pattern
    - , ,
- Why?:
  - Ease automatic analysis
  - Factorize best practices into a standard
  - …

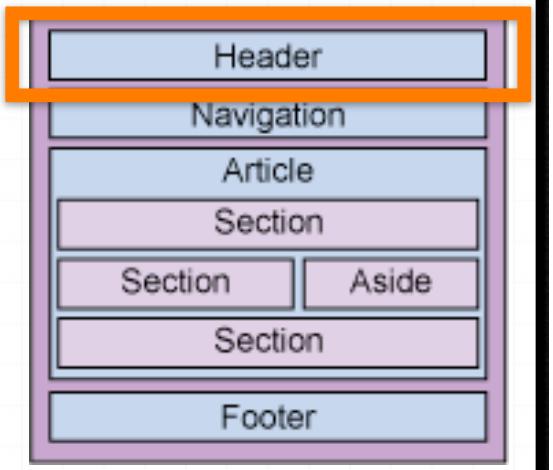
<section>
<nav>
<article>
<aside>
<hgroup>
<header>
<footer>
<time>
<mark>



<header> - <hgroup>

- Semantic elements that define
  - the *header* of a content <header>
  - a wrapper for two or more *related* heading elements <hgroup>
- Used to go beyond HTML 4 in defining the outline of a document
  - Recall that in HTML4, <Hx> elements define *structure*

```
<header>
    <hgroup>
      <h1><a href="#">home</a></h1>
      <h2><a href="#">blog</a></h2>
    </hgroup>
    ……
</header>
```

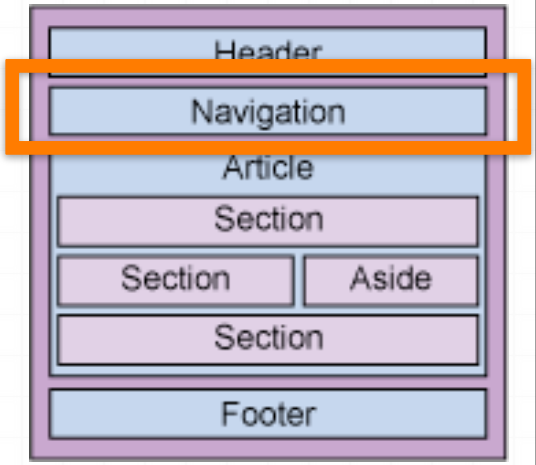


- <h2> can be used to create *taglines*, without the need for dummy structural nodes

<nav>

- A section of a page that links to other pages or to parts within the page
- Only sections that consist of major navigation blocks are appropriate for the `<nav>` element

```
<nav>
    <ul>
     <li><a href="#">home</a></li>
     <li><a href="#">blog</a></li>
     <li><a href="#">gallery</a></li>
     <li><a href="#">about</a></li>
    </ul>
</nav>
```

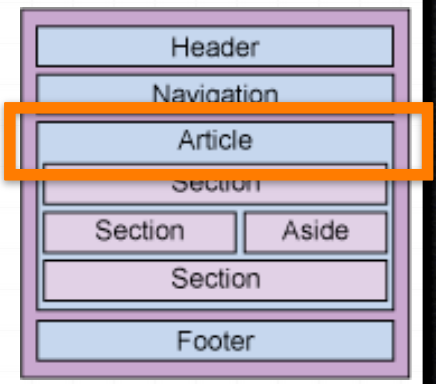


- Why the semantics of site navigation is important?
  - People with disabilities (e.g. screenreader)

<article>

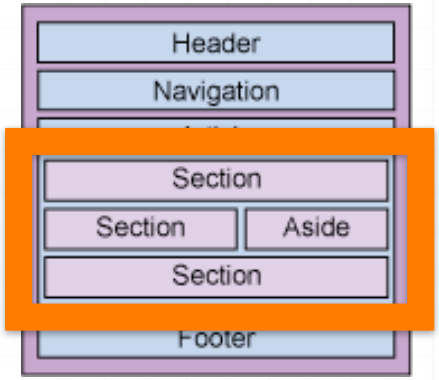
- A *self-contained composition* in a document, page, application or site that is intended to be *independently distributable* or *reusable*
  - E.g. forum post, newspaper article, blog entry, comments, a widget, etc.

```
<article>
    <header>
        <p class="post-date">October 22, 2009</p>
        <h1><a href="#" rel="bookmark"
                  title="link to this post">Travel day </a>
        </h1>
    </header>
        ...
</article>
```



- An <article> element creates a new section, that is, a new node in the document outline.
  - In HTML5, each section can have its own <h1> element

<section>



- The <section> element represents a generic document or application section.
  - A blob of content that you could store as an individual record in a database.
  - A thematic grouping of content, typically with a *heading*
  - Examples:
    - Chapters in a book
    - Tabbed pages
    - Sections in a page (e.g. intro, contact, etc)
- *<section>* should not be used if there is no natural heading for it
- Typically used to group similar information

<aside>

- Used for content *tangentially* related to the content surrounding it
  - E.g. related reading links, glossaries, etc.
- Also used to hold secondary content (when not nested within an article element)

```
<body>
  <header><h1>My Blog</h1></header>
  <article>
    <h1>My Blog Post</h1>
    <p>….</p>
    <aside>
      <!-- the content of this aside is directly related to the article itself. -->
      <h1>Glossary</h1>
       ……
    </aside>
  </article>
  <aside>
    <!-- This aside is outside of the article.Its content is related to the page -->
    <h2>Blogroll</h2>
    ……
  </aside>
</body>
```

<footer>

- It represents a *footer* for its nearest ancestor:
  - It contains information about its section (e.g., author, links to related documents, copyright …)
  - When *footer* element contains entire sections, they represent appendixes, license agreements, etc.
  - It may also contain navigation elements

```
<footer>
    <nav>
       <h1>Navigation</h1>
       <ul>
            <li><a href="/">Home</a>
            …
       </ul>
    </nav>
    <section>
       <h1>Contacts</h1>
       ...
    </section>
    <p class="copyright">Copyright @2009</p>
</footer>
```

<time>

- It represents either a time on a 24h clock or a precise date in the Gregorian calendar, optionally with a time and a timezone offset

```
<time datetime="2011-04-17T13:59:47-04:00" pupdate="pubdate">
      April 17, 2011
</time>
```

- It contains three parts:
  - A machine-readable timestamp
  - Human-readable text content
  - An optional pubdate attribute that means:
    - If the <time> element is in an <article> element -> timestamp of the article
    - Otherwise, timestamp of the entire document

<meter>

- A numeric value in a specific range
  - helps browsers and other clients recognize amounts in HTML pages.

```
<p>Your score was
     <meter value="88.7" min="0" max="100" low="65"
high="96" optimum="100">B+</meter>.
</p>
```

This indicates that the student's score was 88.7 out of a possible 100. The lowest possible grade was 0, but the lowest actual grade anyone got was 65. The highest grade anyone got was 96, although of course the ideal score was 100

<mark>

- The <mark> element can be used to identify text which should be highlighted.
- Default styles are applied to make the text appear highlighted.

<dialog>

- A conversation between several people
  - The <dt> element indicates the speaker
  - The <dd> element indicates the speech.

```
<dialog>
     <dt>Simplicius </dt>
     <dd>According to …</dd>

     <dt>Sagredo </dt>
     <dd>But I should take neither of them…</dd>

     <dt>Salviati </dt>
     <dd>
          <p> Your choice and the reason you adduce for it …</p>
     </dd>
</dialog>
```

HTML5 Page Structure

http://elankeeran.com/wp/2011/04/html5-page-structure/

```
<!DOCTYPE HTML>
<html>
 <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
        <title>Your Website</title>
 </head>
 <body>
        <header>
                <nav>
                        <ul>
                                <li>Your menu</li>
                        </ul>
                </nav>
        </header>
        <section>
                <article>
                        <header>
                                <h2>Article title</h2>
                                <p>Posted on <time datetime="2011-04-15T16:31:24+02:00">April 15th 2011</time> by <a href="#">Writer</a> -
                                  <a href="#comments">6 comments</a></p>
                        </header>
                        <p>Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.</p>
                </article>
                <article>
                        <header>
                                <h2>Article title</h2>
                                <p>Posted on <time datetime="2011-04-15T16:31:24+02:00">April 15th 2011</time>  by <a href="#">Writer</a> -
                                  <a href="#comments">6 comments</a></p>
                        </header>
                        <p>Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.</p>
                </article>
        </section>
        <aside>
                <h2>About section</h2>
                <p>Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo.</p>
        </aside>
        <footer>
                <p>Copyright 2009 Your name</p>
        </footer>
 </body>
</html>z
```

Examples of applied HTML5 semantic elements

- HTML5 Fundamentals
  - http://www.ibm.com/developerworks/apps/download/index.jsp?contentid=660316&filename=HTML5Fundamentals.zip&method=http&locale=
- HTML5 Semantic Notepad
  - http://ie.microsoft.com/testdrive/HTML5/SemanticNotepad/
- Coding a HTML5 Layout from scratch
  - http://www.smashingmagazine.com/2009/08/04/designing-a-html-5-layout-from-scratch/

HTML5 Form Elements

New input types

- HTML5 several new input types and features for forms
- WARNING: working properly only in newer browsers (but gracefully degrading in older ones)

Placeholder in text fields

- *Text* displayed in the input field as long as the field is empty and not focused
  - No HTML markup (but some browser support CSS extensions)

- The placeholder disappears on focus

```
<form>
    <input name="q" placeholder="Search Bookmarks and History">
    <input type="submit" value="Search”>
</form>
```

Autofocus fields

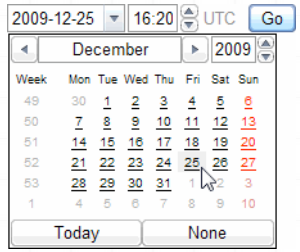
- Several applications needs a default input to be *"focused"* to improve usability and usage efficiency
  - E.g. Google
  - Behavior typically enforced via Javascript
  - But not always "usable"
    - Power users or users with disabilities
- HTML5 *autofocus* attribute: as soon as the page load, it moves the focus on an input field

```
<form>
  <input name="q" autofocus>
  <input type="submit" value="Search">
</form>
```

- Directly in the HTML markup -> it can be disabled

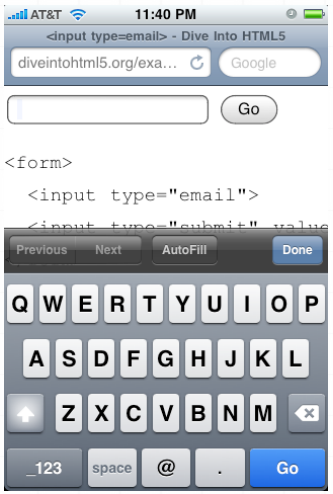
New field types

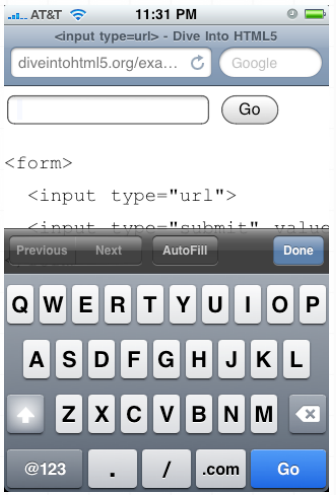
- Email address `<input type="email">`
- Web addresses `<input type="url”>`
- Number as spinboxes `<input type="number" min="0” max="10" step="2" value="6">`
  - Also with JS method to control form
- Number as sliders `<input type="range" min="0" max="10" step="2" value="6">`
- Date Pickers `<input type="date"> <input type="datetime">`
- Search Boxes `<input name="q" type="search">`
- Color Pickers `<input type="color”>`



Why new types be useful?

- HTML5 does not mandate any specific UI for new input types, but, for instance…



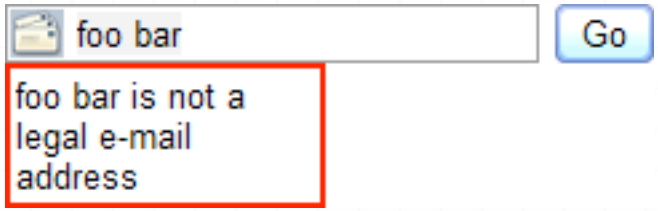


- *Automatic input validation*

Automatic Input Validation

- Typically performed
  - Server-side (always useful – and suggested)
  - Client-side with Javascript
    - Cost of additional script
    - JS-disabled in the user browser
- Offload the functionality for standard types to the browser

```
<form>
    <input type="email" id="addr" required>
    <input type="submit" value="Subscribe">
</form>
```



The HTML5 <canvas> element

The <canvas> element

"A resolution-dependent bitmap canvas that can be used for rendering graphs, game graphics, or other visual images on the fly"

- A <canvas> element is an empty rectangle in a page in which one can use JavaScript to draw everything
  - no content and no border of its own

```
<canvas id="myCanvas" width="300" height="225"></canvas>
```

- If the element has an id, it can be easily accessed using JS DOM API

```
Var my_canvas = document.getElementById("myCanvas");
```

Draw shapes

1. Get a reference to the element
2. Get a reference to the *canvas drawing context*
   - *2D* or *3D* (not standard yet)
3. Draw

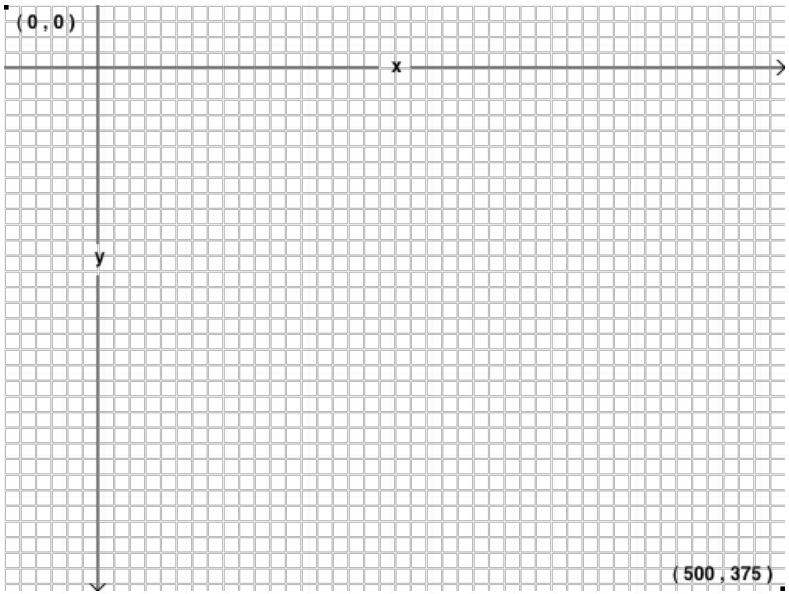
```
1. var my_canvas = document.getElementById(“myCanvas");

2. Var my_canvas_context = my_canvas.getContext("2d");

3. my_canvas_context.<DRAWING PRIMITIVE>;
```

Canvas Coordinates

- The canvas is a two-dimensional grid, with origin (0,0) in the upper-left corner




```
var c = document.getElementById("c");
var ctx = c.getContext("2d");
draw_grid(ctx);
draw_arrows(ctx);
draw_labels(ctx);
draw_dots(ctx)
```

Drawing the grid

```
function draw_grid(ctx) {
 try {
      /* vertical lines */
      for (var x = 0.5; x < 500; x += 10){
            ctx.moveTo(x, 0); ctx.lineTo(x, 375);
      }
      /* horizontal lines */
      for (var y = 0.5; y < 375; y += 10) {
            ctx.moveTo(0, y); ctx.lineTo(500, y);
      }
      /* draw it! */
      ctx.strokeStyle = "#eee";
      ctx.stroke();
 } catch(err) {}
}
```

```
function draw_arrows(ctx) {
 try {
      /* x-axis */
      ctx.beginPath();
      ctx.moveTo(0, 40);ctx.lineTo(240, 40);
      ctx.moveTo(260, 40);ctx.lineTo(500, 40);
      ctx.moveTo(495, 35);ctx.lineTo(500, 40);
      ctx.lineTo(495, 45);
      /* y-axis */
      /* draw it! */
      ctx.strokeStyle = "#000";
      ctx.stroke();
} catch(err) {}
}
```

```
function draw_labels(ctx) {
 try {
   ctx.font = "bold 12px sans-serif"; ctx.fillText("x", 248, 43); ctx.fillText("y", 58, 165);
 } catch(err) {}
 try {
   ctx.textBaseline = "top"; ctx.fillText("( 0 , 0 )", 8, 5);
     } catch(err) {}
 try {
    ctx.textAlign = "right"; ctx.textBaseline = "bottom"; ctx.fillText("( 500 , 375 )", 492, 370);
 } catch(err) {}
}
```

http://www.html5tutorial.info/html5-canvas-text.php

Gradients

- A *gradient* is a smooth transition between two or more colors
  - createLinearGradient(x0, y0, x1, y1) paints along a line from (x0, y0) to (x1, y1)
  - createRadialGradient(x0, y0, r0, x1, y1, r1) paints along a cone between two circles

```
var d_canvas = document.getElementById("d");
var context = d_canvas.getContext("2d");
var my_gradient = context.createLinearGradient(0, 0, 300, 0);
my_gradient.addColorStop(0, "black");
my_gradient.addColorStop(1, "white");
context.fillStyle = my_gradient;
context.fillRect(0, 0, 300, 225);
```

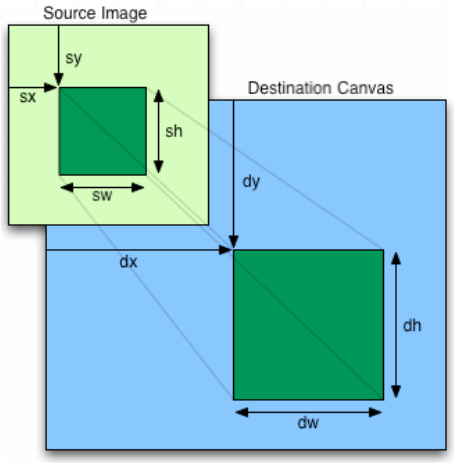
Images

- The canvas drawing context allows the drawing of an image in the canvas
  - drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh) takes an image, clips it to the rectangle (sx, sy, sw, sh), scales it to dimensions (dw, dh), and draws it on the canvas at coordinates (dx, dy).

- The image can be an existing <img> element or an Image JS object

```
<img id="cat" src="images/cat.png" >
<canvas id="e" width="177" height="113"></canvas>

var canvas = document.getElementById("e");
var context = canvas.getContext("2d");
var cat = document.getElementById("cat");
context.drawImage(cat, 0, 0); };
```



Media in HTML5

<figure>

- The <figure> tag can contain the <figcaption>, which in turn contains the caption for the figure contained in the <figure> tag, allowing you to enter a description that can tie the figure more closely to the content

```
<figure>
  <img src="…" width="10" height="10" alt="picture"/>
  <figcaption>Caption</figcaption>
</figure>
```

The <video> element

```
<video width="320" height="240" controls autoplay preload poster="star.png'>
    <source src="pr6.mp4" type='video/mp4; codecs="mp4a.40.2"'>
    <source src="pr6.webm" type='video/webm; codecs="vp8, vorbis"'>
    <source src="pr6.ogv" type='video/ogg; codecs="theora, vorbis"'>
</video>
```

Image to be shown while video loading

Video size

Show video control commands

Play the video as the page loads

Start downloading as the page loads

Define on video source

1. **File**
2. **Type**
3. **Video/audio codecs**

A new browser war?

- In HTML4 there is no standard way to embed video in a page
  - Third-party plug-in like Flash, QuickTime, Real Player, etc.
  - But plug-ins are don't work in all platforms
- HTML5 includes a <video> tag to embed video in pages
  - No restrictions on video (audio) codecs
  - But incompatibility problems are now at codec level

Codec/ Container	IE	FF	Safari	Chrome	Opera	iPhone	Android
Theora/ Vorbis/Ogg		3.5+	(extra plugin)	5.0	10.5+		
H264+AAC +MP4			3.0+	5.0 (to drop)		3.0+	2.0+
WebM			(extra plugin)	6.0	10.6+		

Video and HTML

- The video is not treated as an embedded foreign object
  - focus is on the player
  - Native Javascript interaction
  - Interaction with other HTML5 primitives

Video and Javascript

- Event binding

```
video.addEventListener('canplay', function(e) {
  this.volume = 0.4;
  this.currentTime = 10;
  this.play();
}, false);
```

- Control trough API

```
<video id='video'>
<button onclick="document._video.load()">load()</button>
<button onclick="document._video.play()">play()</button>
<button onclick="document._video.pause()">pause()</button>
<button onclick="document._video.currentTime+=10">currentTime+=10</
button>
<button onclick="document._video.currentTime-=10">currentTime-=10</
button
```

http://www.w3.org/2010/05/video/mediaevents.html

Video and CSS

- The <video> tag is a first-class citizen in the DOM.
  - It can be styled using traditional CSS (e.g. border, opacity, etc)
  - But also with the latest CSS3 properties
    - reflections, masks, gradients, transforms, transitions and animations.
    - http://www.satine.org/research/webkit/video/trailers.html (With SAFARI)

```
-webkit-box-reflect: below 1
-webkit-gradient(linear, left top, left bottom, color-stop
(0.5, transparent), color-stop(1.0, rgba(255, 255, 255, 0.5)));

-webkit-transition-property: opacity;
-webkit-transition-duration: 550ms;
-webkit-transition-timing-function: ease-in-out;
```

Video and Canvas

- <video> and <canvas> were designed to work together
- <canvas> has a *drawImage* method which lets you import images from three different sources:
  - an image element
  - another canvas element
  - **a *<video> element ->*** every time the method is invoked, the current frame in the video is imported and rendered into the canvas
  - **If the method is invoked as the same framerate of the video, nice effects can be obtained**
    - http://craftymind.com/factory/html5video/CanvasVideo.html
    - http://html5doctor.com/video-canvas-magic/

HTML5 APIs

What is an API?

- *API = application programming interface*
  - a collection of programming instructions and standards for accessing a software application
- The HTML5 movement wants to standardize several functionalities that have been introduced over the years
  - Interaction with local device
  - Multi-threading
  - Storage
  - Media interaction

HTML5 APIs

- ~ 40 proposed specifications
  - http://dret.typepad.com/dretblog/html5-api-overview.html
  - Mostly only partially supported
    - http://html5test.com/results.html
    - http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(HTML5)

Contacts, IndexedDB, WebWorkers, WebStorage, WebSockets, FileAPI, GeoLocation, Cavas, Microdata, Media Capture, Messaging, Clipboard, Calendar, BatteryStatus, TouchEvents, Timing, DeviceOrientation, Vibration, Cache, etc.

Geo-location API (1)

- Geo-location -> Finding the geographical location of a user
  - IP-Address
  - GPS coordinates
  - WI-FI hotspot coordinates
  - Cellular tower(s) coordinates
- Why Geolocation?
  - Customized services (e.g. Google Search)
  - Social Networks (e.g. Foursquare)
  - Advertising

Geo-location API (2)

- New property on the global navigator object:
  `navigator.geolocation`
- Geolocation option has two methods
  - `getCurrentPosition`: returns the current position of the user
  - `watchPosition`: called *every time the user's location changes*. There is no need to actively poll their position. The device will determine the optimal polling interval, and it will call your callback function whenever it determines that the user's position has changed.

Geo-location API (3)

```
navigator.geolocation.getCurrentPosition(callback,
errorObject, PositionOptions);


navigator.geolocation.watchPosition(callback,
errorCallback, PositionOptions)

function callback(position) {
// Handle position object
}
```

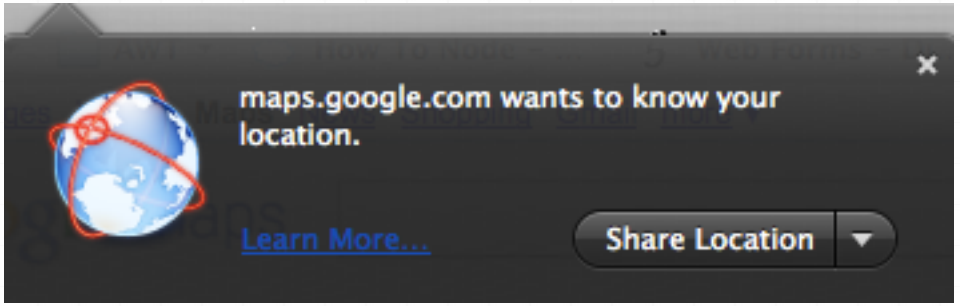
Geo-location API (4)

- Position Object properties
  - `coords.latitude` *double* decimal degrees → GUARANTEED
  - `coords.longitude` *double* decimal degrees → GUARANTEED
  - `coords.altitude` *double* or *null* meters → OPTIONAL
  - `coords.accuracy` *double* meters → OPTIONAL
  - `coords.altitudeAccuracy` *double or null* meters → OPTIONAL
  - `coords.heading` *duble or null* degrees clockwise from true north → CALCULATED BY DEVICE
  - `coords.speed` *double or null* meters/second → CALCULATED BY DEVICE
  - `Timestamp` *DOMTimeStamp* like a Date() object → GUARANTEED

- The `PositionOption` argument allows the definition of "quality constraints" for the returned location
  - `enableHighAccuracy` : if true, and the device can support high precision location detection (e.g. GPS), then the device will try to provide it
  - `timeout`: the number of milliseconds your web application is willing to wait for a position
  - `maximumAge` : allows the device to answer immediately with a cached position (if you are ok with small changes in location)

Geo-location API Privacy Concerns

- "User Agents must not send location information to Web sites without the express permission of the user."
- Sharing your location MUST BE OPTIONAL
  - If you don't want to, you don't have to.



HTML5 Storage (1)

- A.K.A. Web Storage, Local Storage, DOM Storage
  - http://dev.w3.org/html5/webstorage/

HTML5 Storage (2)

"There is no such thing as *the state*…"

- Desktop applications are able to manage (and persist) their state locally
  - Registry, INI files, plist files, XML files, database, etc.
- The only "storage" system available for Web clients were ***cookies***
  - Included in every HTTP request (with problems of performance and security)
  - Limited to 4Kb of data
  - Possibly unavailable (if cookies are disabled)

HTML5 Storage (3)
Previous solutions

- Internet Explorer
  - userData -> 64Kb per domain in a XML-based structure (trusted domain up to 640Kb)
- Flash
  - Local Shared Objects -> 100Kb of data per domain (but it allows users to increase storage size)
- Google Gears Plugin
  - Local Relations storage based on SQLite -> unlimited data, SQL statements

HTML5 Storage (4)

- A *standard* way for Web pages to store **key**/**value** pairs

  `window.localStorage`

- Data persists page navigation, user sessions, and browser session

- Limit of 5Mb for each *origin*
  - http://www.whatwg.org/specs/web-apps/current-work/multipage/origin-0.html#origin-0
  - No additional space

HTML5 Storage (5) Usage

```
interface Storage {
    getter any getItem(in DOMString key);
    setter creator void setItem(in DOMString key, in any data);
    deleter void removeItem(in DOMString key);
    void clear();
    readonly attribute unsigned long length;
    getter DOMString key(in unsigned long index);
};
```

- Data are stored as associative arrays based on a named **key**
  - The squared brackets notation can be used to set and get values
- **Key**s are strings
- **Data** can be any type supported by Javascript (e.g. strings, Booleans, integers, floats) but they are **stored as strings**
  - retrieving values require parsing to restore original type
  - Objected are stored serializing them to JSON

HTML5 Storage (6) Additional events

- Track changes to the data storage
- `storage`: triggered when something changes
- The `StorageEvent` object (associated with the fired events) will contain the following properties
  - `key` – string - the named key that was added, removed, or modified
  - `oldValue` – any - the previous value (now overwritten), or null if a new item was added
  - `newValue` – any - the new value, or null if an item was removed
  - `url` – string - the page which called a method that triggered this change

HTML5 Storage (7)
What about relational storage?

- **Web SQL Database** specification (http://dev.w3.org/html5/webdatabase/)
  - **Not standard**, as the only implementation adopted so far use the Sqlite backend – and several implementation are needed to have a standard
- **Indexed Database** specification (http://dev.w3.org/2006/webapi/IndexedDB/)
  - Idea of *object store* -> databases with records and fields, where each field has a datatype defined at DB creation
  - Only implemented in FF4, maybe it will be implemented in Chrome

Off-line Web Applications (1)

http://www.whatwg.org/specs/web-apps/current-work/multipage/offline.html

- **Web** is being online – why being offline?
  - Nomadic usage, Network outage, Network cost, etc.
- What do you need when off-line?
  - HTML pages, CSS, Javascripts
  - Images/Videos etc.
  - Local data
- Off-line enabled by
  - Caching resources
  - Switching from cache to "live" resources (and viceversa) when needed
    - A flag in the DOM tells the application whether you're online or offline
    - A dedicated event fires when the status change

Off-line Web Applications (2) Going Offline/Online

- Switching offline/online is controlled by the browser
  - Explicit “work offline” command
  - Automatically, monitoring connectivity
- Ideally, switching between modes should not result in failure

Off-line Web Applications (3) HTTP Cache Vs. AppCache

- Browsers typically implement an HTTP cache
  - Cache HTTP responses based on HTTP methods and metadata
- For an application to run off-line, all the required resources must be reliably cached

- The HTML5 caching mechanism works hand-by-hand with the traditional browser caching policy (cache every resource that you visit until they expire)
  - but it add some additional explicit policies

Off-line Web Applications (4) The HTML5 cache manifest

- The home page of the offline Web application points to **the manifest** file for the resources to cache
  - There is only one manifest file. All the application pages must point to it
- An HTML5 compliant browser will read the manifest file, download the resources, cache them locally, and automatically update changes

```
<!DOCTYPE HTML>
  <html manifest="/cache.manifest">
  <body> ... </body>
</html>
```

Off-line Web Applications (5) The cache manifest /2

- A manifest file is divided in:
  - CACHE: resources which are explicitly downloaded and cached locally and used when online
  - FALLBACK: substitution for on-line resources that will not be cached (e.g. other pages in your site, etc.)
    - It can be configured for specific application pages or folders
  - NETWORK: an *online whithelist* of resources that are never cached and are not available offline
- If the manifest has no header, all the resources are implicitly in the *CACHE* section

Off-line Web Applications (6) Monitoring Caching Behavior

- New `window` default object `window.applicationCache`
  - **`Checking`** event triggered when the browser spots a `manifest` attribute in the `<html>` element
  - If the `manifest` file is new, a `downloading` event is fired and the browser starts downloading resource
    - The browser periodically fires `progress` events to notify about the download progress (*downloaded* of *toDownload*)
    - When all the resources are downloaded, the browser sends a `cached` event
  - If the `manifest` file changed from a previous version, a new `downloading` event is fired and the browser starts downloading resource (firing `progress` events)
    - When the download is finished, the browser triggers an `updateready` event
    - The new cached resources won't be used unless the user refreshe the page or a manual swap is forced `window.applicationCache.swapCache()`

WebSocket (1)

- HTTP is a request-response protocol
- But several applications needs bi-directional communication
  - Chat
  - Multiplayer on-line games
  - Realtime updating of social streams

- Solution so far: COMET
  - An hack that provide “server push” using standard browser functionality
    - most-often implemented via Ajax with long polling
  - Inefficient

- **WebSocket**: New proposal to provide full-duplex, bi-directional client-server interaction over a single TCP connection.

WebSocket (2)

- Special socket connections between a Web browsers and the server
  - an open connection between the client and the server and both parties can start sending data at any time
  - send messages to a server and receive event-driven responses without having to poll the server for a reply
- **GOALS**:
  - Enabling near real-time push communications
  - Increasing web server connection scalability
  - Simplifying the coding task

WebSocket (3)

- Has its own protocol, handshake and headers
  - ws:// and wss://
- … but different headers

- Has (partial) support from several browsers
  - Firefox >6, Chrome >14, Opera 11, Safari 5, IE 10

- But has some security issues

WebSocket (4)
An Example

```
ws = new WebSocket("ws://127.0.0.1:8080/websocket");

ws.onmessage = function(evt) {
    tweet = $.parseJSON(evt.data)
    $("<p>"+ tweet.user.screen_name+' posted:
        '+TwitterText.auto_link(tweet.text)+
        "</p>").prependTo("#msg");
    };
ws.onclose = function() { $("#debug").append("socket closed"); };
ws.onopen = function() { $("#debug").append("socket connected");

ws.send("Hello…");
ws.close()
```

WebWorkers (1)

- Answer to Javascript single-thread limitation
- WebWorkser allows for thread-like operation with message-passing as the coordination mechanism
  - Computation-intensive tasks
  - Asynchronous download and storage of data

WebWorkers (2)

- WebWorkers run processes on a separate thread from the main page
  - Page is preserved for the main functions, such as maintaining a stable
  - They don't use the browser UI thread
  - They are not permitted access to the DOM

- WebWorksers can:
  - Access the Navigator object
  - Read-only location objects
  - Perform AJAX requests
  - Access the cache
  - Create new Workers

```
var worker = new Worker('worker.js');
```

HTML5
Missing Features

Advanced HMI

- (multi) touch screen
- Accelerometer for measuring device movement
- Vibration for giving tactile feedback

Special Graphics Capabilities

- Advanced rendering (3D canvas)
- Non-traditional bitmapped displays such as e-ink

Access to device hardware

- Phone (to make calls)
- Camera(s)
- Compass

Access to device functionalities

- Messaging
- Contacts, calendar, social networks, etc.

References

- *W3C HTML5 Introduction:* http://www.w3.org/TR/html5/introduction.html
- *HTML 5 Up and Running* – Mark Pilgrim, O'Reilly Google Press – or http://diveintohtml5.org/
- *HTML5 boilerplate* - http://html5boilerplate.com/ - "collection of some tricks to get your project off the ground quickly and right-footed"
- *HTML5 Demos* - http://html5demos.com/
- *HTML5 Rocks* - http://www.html5rocks.com/
  - http://slides.html5rocks.com/
- *HTML5 Video* - http://dev.opera.com/articles/view/everything-you-need-to-know-about-html5-video-and-audio/
- *HTML5 Foundamentals* - https://www.ibm.com/developerworks/mydeveloperworks/blogs/social-media-marketing/entry/html5_fundamentals?lang=en
- *HTML5 Tag Reference* - http://www.w3schools.com/html5/html5_reference.asp