MOBILITY HTML 5

Parameswari.E

What is HTML5?

- **HTML5** is a markup language for structuring and presenting content for the World Wide Web and a core technology of the Internet.
- The previous version of HTML, HTML 4.01, came in 1999. The web has changed a lot since then.
- It is the fifth revision of the HTML standard (created in 1990 and standardized as HTML 4 as of 1997 and, as of December 2012, is a W3C Candidate Recommendation).

What is HTML5?

History of HTML5

- It was first published as an Internet draft in 1993.
- The ’90s saw an enormous amount of activity around HTML, with version 2.0, versions 3.2, and 4.0 (in the same year!), and finally, in 1999, version 4.01.
- After the rapid delivery of these four versions though, HTML was widely considered a dead-end;

History of HTML5

- Wanting to take the web platform to a new level, a small group of people started the Web Hypertext Application Working Group (WHATWG) in 2004.
- They created the HTML5 specification.
- The W3C became involved with HTML again in 2006 and published the first working draft for HTML5 in 2008.

History of HTML5

- Rough Timeline of Web Technologies1991 HTML
- 1994 HTML 2
- 1996 CSS 1 + JavaScript
- 1997 HTML 4
- 1998 CSS 2
- 2000 XHTML 1
- 2002 Tableless Web Design
- 2005 AJAX
- 2008 HTML 5

The combined timelines for HTML 5.0, HTML 5.1 and HTML 5.2:

Plan 2014

	2012	2013	2014	2015	2016
HTML 5.0	Candidate Rec	Call for Review	Recommendation		
HTML 5.1	1st Working Draft		Last Call	Candidate Rec	Recommendation
HTML 5.2				1st Working Draft	

The Myth of 2022 and Why It Doesn't Matter

- The HTML5 specification that we see today has been published as a working draft—it is not yet final.
- So when does it get cast in stone?
- Here are the key dates that you need to know.
- The first is 2012, which is the target date for the *candidate recommendation*.
- The second date is 2022, which is the *proposed recommendation*.

The Myth of 2022 and Why It Doesn't Matter

	2012	2013	2014	2015	2016
HTML 5.0	Candidate Rec	Call for Review	Recommendation		
HTML 5.1	1st Working Draft		Last Call	Candidate Rec	Recommendation
HTML 5.2				1st Working Draft	

Who Is Developing HTML5?

- *Web Hypertext Application Technology Working Group (WHATWG)*: Founded in 2004 by individuals working for browser vendors Apple, Mozilla, Google, and Opera, WHATWG develops HTML and APIs for web application development and provides open collaboration of browser vendors and other interested parties.

- *World Wide Web Consortium (W3C)*: The W3C contains the HTML working group that is currently charged with delivering their HTML5 specification.

- *Internet Engineering Task Force (IETF)*: This task force contains the groups responsible for Internet protocols such as HTTP. HTML5 defines a new WebSocket API that relies on a new WebSocket protocol, which is under development in an IETF working group.

A New Vision

- HTML5 is based on various design principles, spelled out in the WHATWG specification, that truly embody a new vision of possibility and practicality.
  - Compatibility
  - Utility
  - Interoperability
  - Universal access

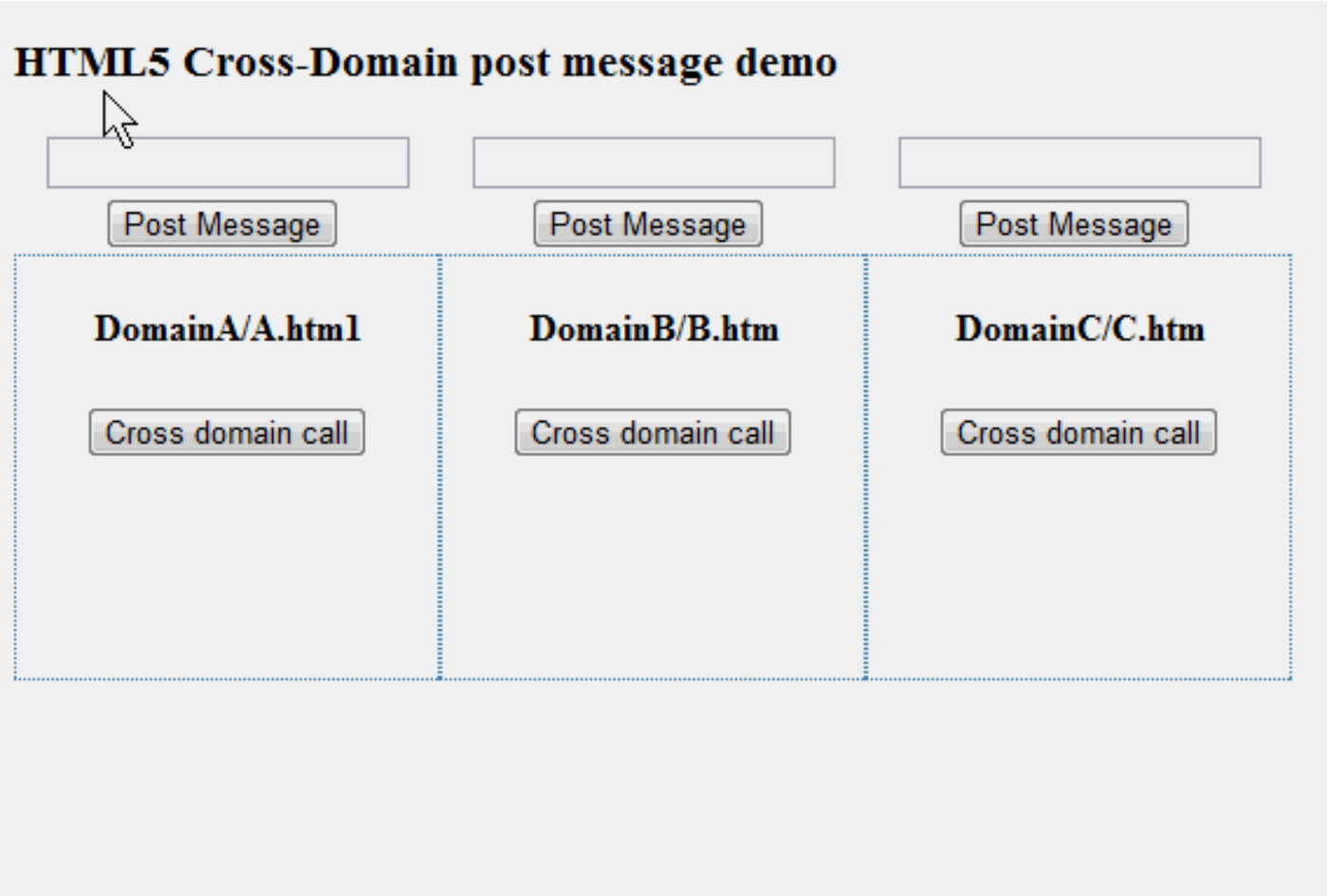
Compatibility and Paving the Cow Paths

- Its core principles is to keep everything working smoothly.
- If HTML5 features are not supported, the behavior must degrade gracefully.
- Google analysed millions of pages to discover the common ID and Class names for DIV tags and found a huge amount of repetition.
- For example, many people used DIV id="header" to mark up header content.
- HTML5 is all about solving real problems, So why not simply create a <header> element?

Utility and the Priority of Constituencies

- The HTML5 specification is written based upon a definite *Priority of Constituencies*
- "the user is king." the specification values users over authors, over implementers (browsers), over specifiers (W3C/WHATWG), and over theoretical purity.
- As a result, HTML5 is overwhelmingly practical.

Secure by Design



Separation of Presentation and Content

- The problems with presentational markup are:

• Poor accessibility

• Unnecessary complexity (it's harder to read your code with all the inline styling)

• Larger document size (due to repetition of style content), which translates into slower-loading pages

Separation of Presentation and Content

- HTML5 takes a giant step toward the clean separation of presentation and content.
- HTML5 strives to create this separation wherever possible, and it does so using CSS.

Interoperability Simplification

- HTML5 is all about simplification and avoiding needless complexity.
- Native browser ability instead of complex JavaScript code(validations, Date Type etc.,)
- A new, simplified DOCTYPE
- A new, simplified character set declaration
- Powerful yet simple HTML5 APIs

Universal Access

- This principle is divided into three concepts:

• *Accessibility*: To support users with disabilities, HTML5 works closely with a related standard called Web Accessibility Initiative (WAI) Accessible Rich Internet Applications (ARIA). WAI-ARIA roles, which are supported by screen readers, can be already be added to your HTML elements.

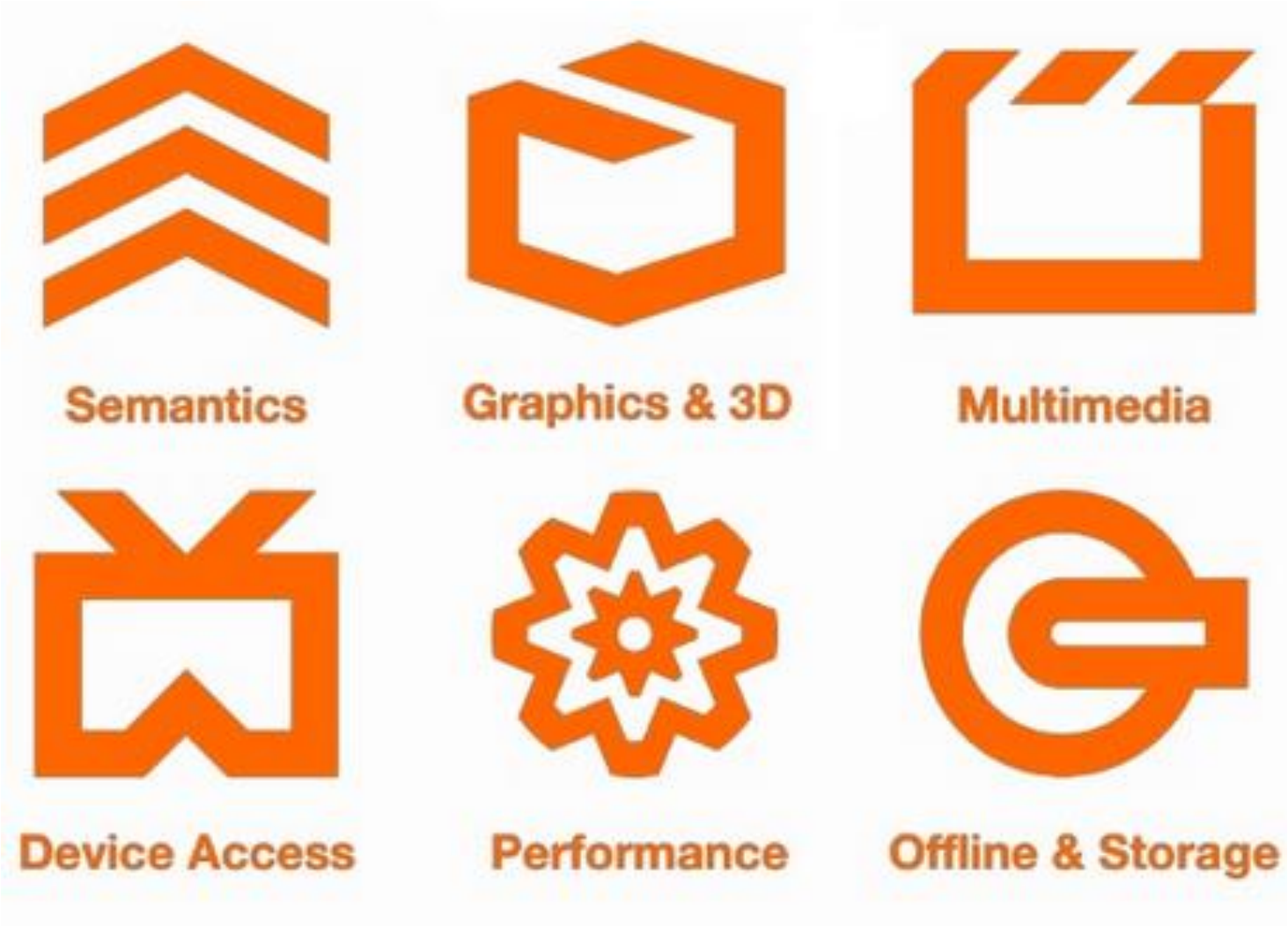
• *Media Independence*: HTML5 functionality should work across all different devices and platforms if at all possible.

• *Support for all world languages*: For example, the new <ruby> element supports the Ruby annotations that are used in East Asian typography.

A Plugin–Free Paradigm

- HTML5 provides native support for many features that used to be possible only with plugins or complex hacks (a native drawing API, native video, native sockets, and so on).

What you can do with HTML5



What's In and What's Out?

- Canvas (2D and 3D)
- Cross-document messaging
- Geolocation
- Audio and Video
- Forms
- MathML
- Microdata
- Server-Sent events
- Scalable Vector Graphics (SVG)
- WebSocket API and protocol
- Web origin concept

What's In and What's Out?

- Web storage
- Indexed database
- Application Cache (Offline Web Apps)
- Web Workers
- Drag and Drop

What's New in HTML5?

- New DOCTYPE and Character Set
- HTML4
  - <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  - "http://www.w3.org/TR/html4/loose.dtd">
- HTML5
  - <!DOCTYPE html>

What's New in HTML5?

- New DOCTYPE and Character Set
- HTML4
  - the character set declaration has also been abbreviated. It used to be
  - <meta http-equiv="Content-Type" content="text/html; charset=utf-8“>
- HTML5
  - <meta charset="utf-8">

New and Deprecated Elements

Table 1-1. HTML5 Content Types

Content Type	Description
Embedded	Content that imports other resources into the document, for example `audio`, `video`, `canvas`, and `iframe`
Flow	Elements used in the body of documents and applications, for example `form`, `h1`, and `small`
Heading	Section headers, for example `h1`, `h2`, and `hgroup`
Interactive	Content that users interact with, for example `audio` or `video` `controls`, `button`, and `textarea`
Metadata	Elements—commonly found in the `head` section— that set up the presentation or behavior of the rest of the document, for example `script`, `style`, and `title`
Phrasing	Text and text markup elements, for example `mark`, `kbd`, `sub`, and `sup`
Sectioning	Elements that define sections in the document, for example `article`, `aside`, and `title`

Semantic Markup

Table 1-2. New Sectioning HTML5 Elements

Sectioning Element	Description
`header`	Header content (for a page or a section of the page)
`footer`	Footer content (for a page or a section of the page)
`section`	A section in a web page
`article`	Independent article content
`aside`	Related content or pull quotes
`nav`	Navigational aids

Simplifying Selection Using the Selectors API

Table 1-4. New QuerySelector Methods

Function Description		Example	Result
`querySelector()`	Return the first element in the page which matches the specified selector rules(s)	`document.querySelector("input.error");`	Return the first input field with a style class of "error"
`querySelectorAll()`	Returns all elements which match the specified rule or rules	`document.querySelectorAll("#results td");`	Return any table cells inside the element with id `results`

How Did HTML5 Get Started?

- As of May 2012, the specification is back to Working Draft state at the W3C. Ian Hickson of Google is the editor of HTML5.
- The criterion for the specification becoming a W3C Recommendation is "two 100% complete and fully interoperable implementations". Many parts of the specification are stable and may be implemented in products.
- In July 2012, WHATWG (Web Hypertext Application Technology Working Group)and W3C have decided on a degree of separation.

JavaScript Logging and Debugging

- The first great tool for analyzing web pages and the code running in them was the Firefox add-on, Firebug.
- Safari's Web Inspector, Google's Chrome Developer Tools, Internet Explorer's Developer Tools, and Opera's Dragonfly.
- Many of the debugging tools offer a way to set breakpoints to halt code execution and analyze the state of the program and the current state of the variables.
- The console.log API has become the de facto logging standard for JavaScript developers.

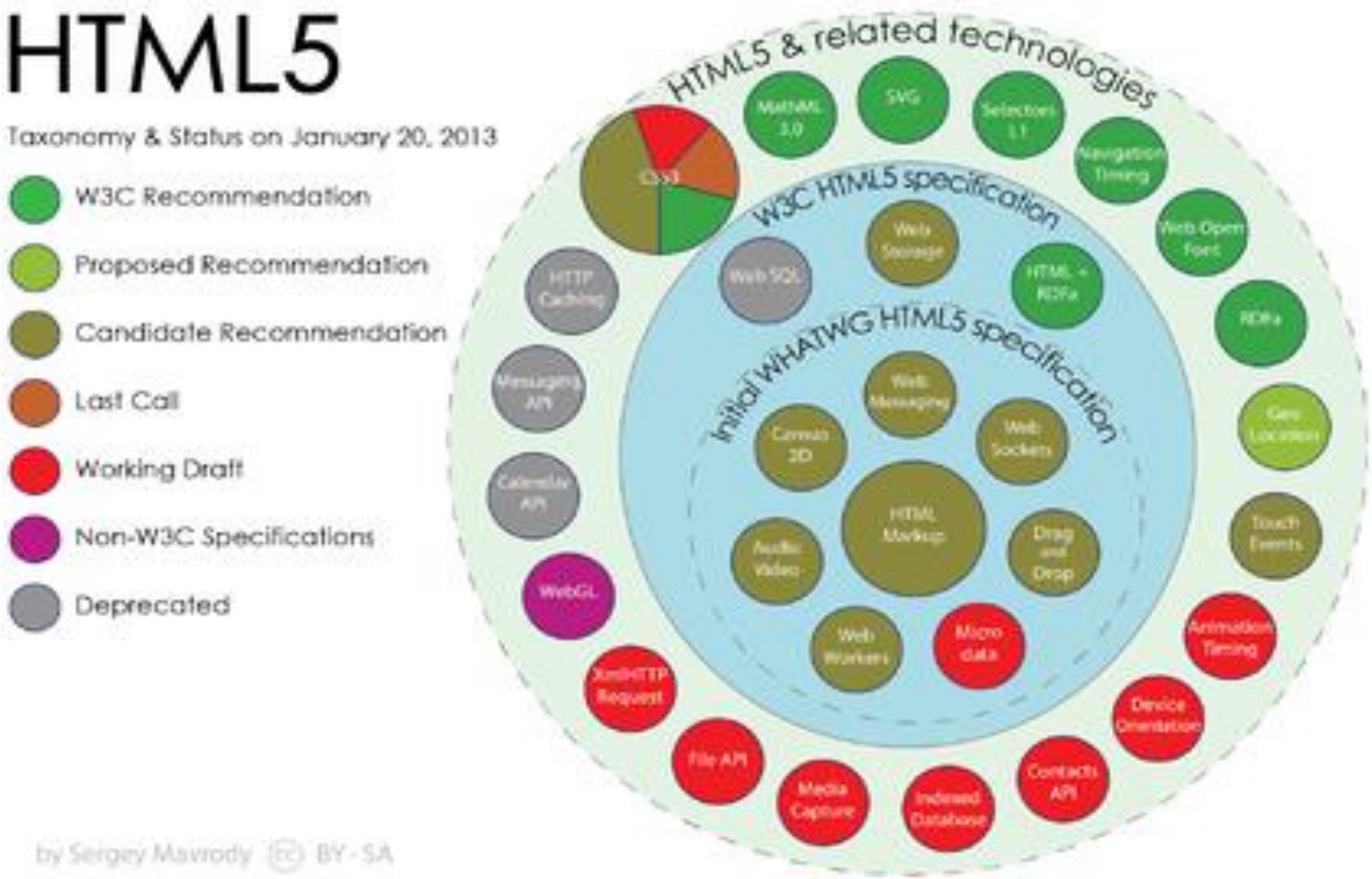
window.JSON

- JSON is a relatively new and increasingly popular way to represent data.
- It is a subset of JavaScript syntax that represents data as object literals.
- Due to its simplicity and natural fit in JavaScript programming, JSON has become the de facto standard for data interchange in HTML5 applications.
- The canonical API for JSON has two functions, parse() and stringify() (meaning serialize or convert to string).

DOM Level 3

- Internet Explorer 9 (IE9) now supports DOM Level 2 and 3 features, so you can finally use the same code for DOM manipulation and event handling in all HTML5 browsers.
- This includes the ever-important addEventListener() and dispatchEvent() methods.

Features


HTML5
Taxonomy & Status on January 20, 2013
W3C Recommendation
Proposed Recommendation
Candidate Recommendation
Last Call
Working Draft
Non-W3C Specifications
Deprecated
HTML5 & related technologies
W3C HTML5 specification
Initial WHATWG HTML5 specification
MathML 3.0
SVG
Selectors 1.1
Navigation Timing
Web Open Font
RDFa
Geo Location
Touch Events
Animation Timing
Device Orientation
Contacts API
Indexed Database
Media Capture
File API
XmlHTTP Request
Micro data
Web Workers
Drag and Drop
Audio Video
HTML Markup
Web Sockets
Canvas 2D
Web Messaging
Web Storage
Web SQL
HTML + RDFa
HTTP Caching
Messaging API
Calendar API
WebGL
by Sergey Mavrody
BY - SA

Deprecated Elements

- The following HTML 4.01 elements are removed from HTML5:

  <acronym>
  <applet>
  <basefont>
  <big>

  <dir>
  <font>
  <frame>
  <frameset>
  <noframes>
  <strike>
  <tt>

HTML5 Browser Support

- Chrome 10.0 was the big winner coming in at 301 points out of 400 possible.
- They have already implemented most of the functionality to be a compatible HTML5 browser.
- In addition, many of the features that are not added are partially added.
- So Google Chrome is definitely way out in front in the race for HTML compatible browsers.

HTML5 Browser Support

- Firefox 4.0 is next in line and they scored pretty decently getting 249 out 400 possible points.
- They are still missing a lot of key elements but got bonus points for the audio and video implementation as well as their parsing rules.
- Internet Explorer 8 is pretty far behind the curve.
- IE scored a total of 32 points out of 400.

The HTML5 <!DOCTYPE>

- In HTML5 there is only one <!doctype> declaration, and it is very simple:
- < !DOCTYPE html>

Minimum HTML5 Document

```
<!DOCTYPE html>
   < html>
   < head>
   < title>Title of the document</title>
   < /head>

   < body>
   The content of the document.....
   < /body>

   < /html>
```

Browser Support for HTML5

- HTML5 is not yet an official standard, and no browsers have full HTML5 support.
- But all major browsers (Safari, Chrome, Firefox, Opera, Internet Explorer) continue to add new HTML5 features to their latest versions.

USING THE CANVAS API

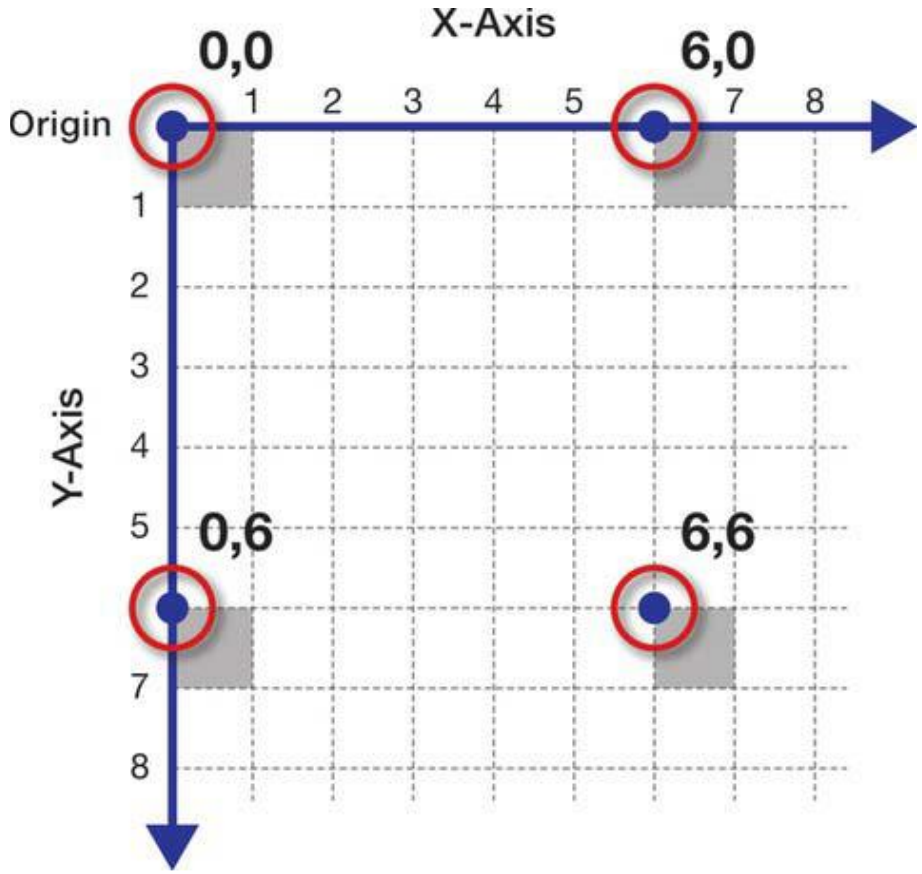
Overview of HTML5 Canvas

- The canvas concept was originally introduced by Apple to be used in Mac OS X WebKit to create *dashboard widgets*.
- Before the arrival of canvas, you could only use drawing APIs in a browser through plug-ins such as Adobe plug-ins for Flash and Scalable Vector Graphics (SVG), Vector Markup Language (VML) only in Internet Explorer, or other clever JavaScript hacks.

What Is a Canvas?

- When you use a canvas element in your web page, it creates a rectangular area on the page.
- By default, this rectangular area is 300 pixels wide and 150 pixels high, but you can specify the exact size and set other attributes for your canvas element.
- *A Basic Canvas Element*
  - <canvas></canvas>

Canvas Coordinates



When Not to Use Canvas

- Although the canvas element is great and very useful, you should *not* use the canvas element when another element will suffice.
- For example, it would not be a good idea to dynamically draw all the different headings for an HTML document on a canvas instead of simply using heading styles (H1, H2, and so on) that are meant for that purpose.

Fallback Content

- In case your web page is accessed by a browser that does not support the canvas element or a subset of the Canvas API features, it is a good idea to provide an alternate source.
- For example, you can provide an alternate image or just some text that explains what the user could be enjoying if they actually used a modern browser.
  - <canvas>
    - Update your browser to enjoy canvas!
  - </canvas>

CSS and Canvas

- As with most HTML elements, CSS can be applied to the canvas element itself to add borders, padding, margins, etc.
- Additionally, some CSS values are inherited by the contents of the canvas; fonts are a good example.

Browser Support for HTML5 Canvas

- With the arrival of Internet Explorer 9, all browser vendors now provide support for HTML5 Canvas, and it is already in the hands of a majority of users.
- This is a major milestone in web development, allowing 2D drawing to thrive on the modern Web.

Using the HTML5 Canvas APIs

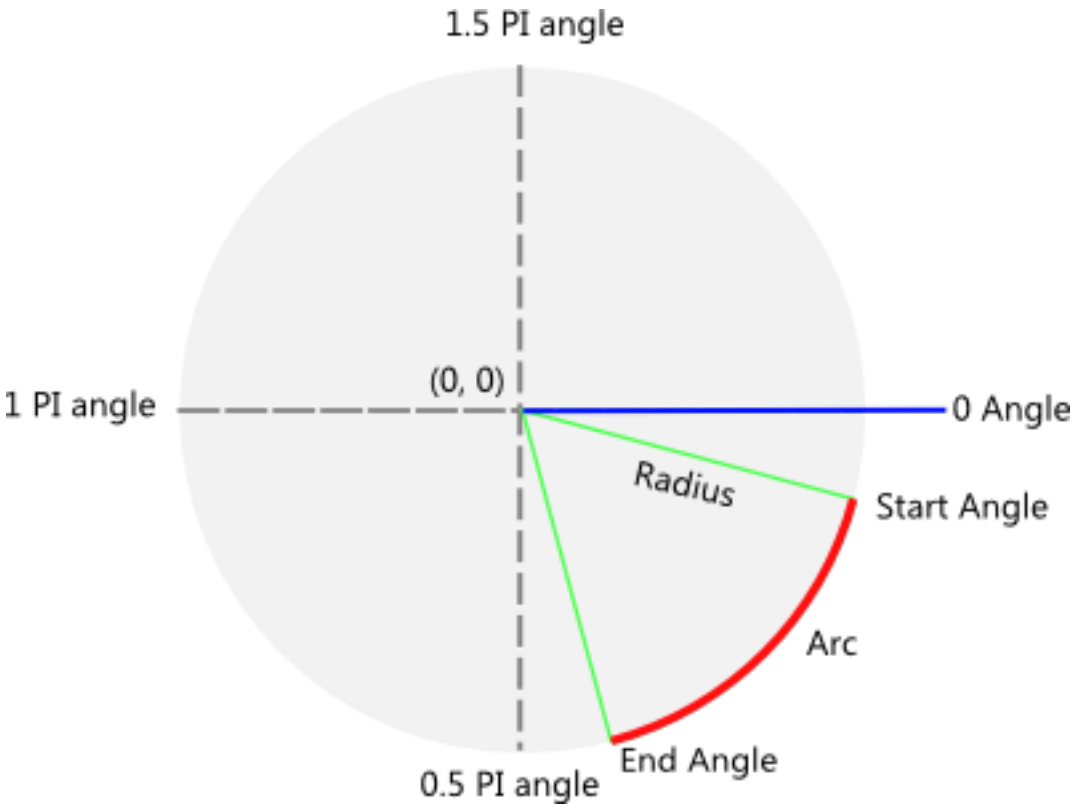
- **ctx.fillRect(Xstart,Ystart,Width,Height);**
- **strokeRect(Xstart,Ystart,Width,Height) – To draw a rectangular outline (No fill)**
- **clearRect(Xstart,Ystart,Width,Height) : To clear the specified rectangle making it fully transparent**

Canvas Paths

- beginPath()
- closePath()
- stroke()
- fill()

```
ctx.beginPath();
ctx.moveTo(50, 50);
ctx.lineTo(200, 200);
ctx.closePath();
ctx.stroke();
```

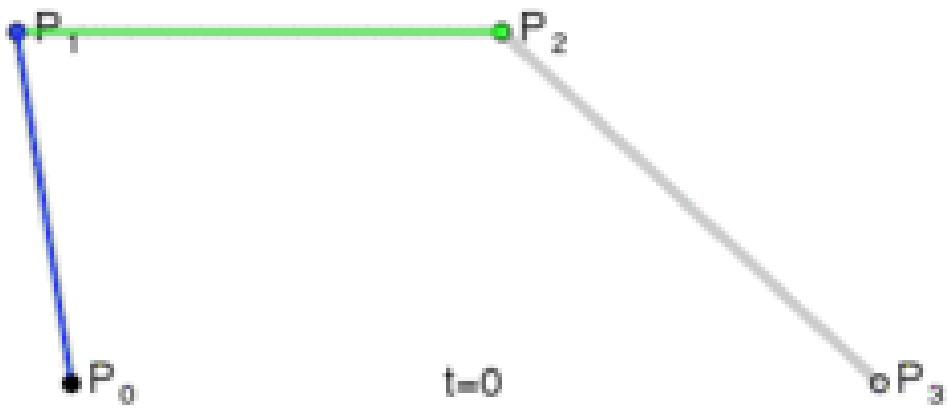
Drawing a circle in canvas



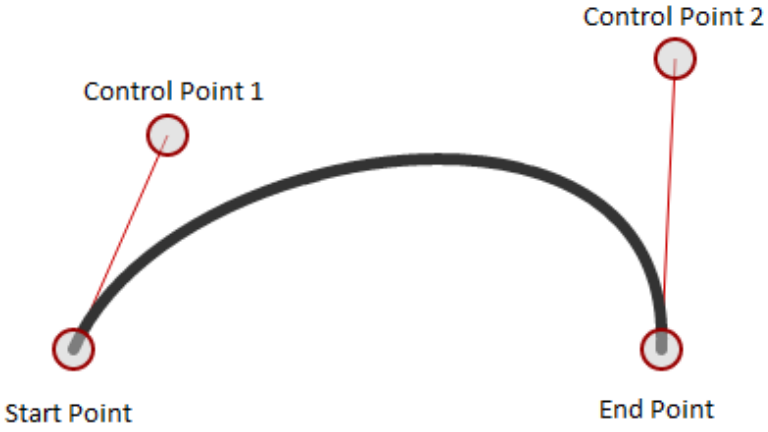
Drawing a circle in canvas

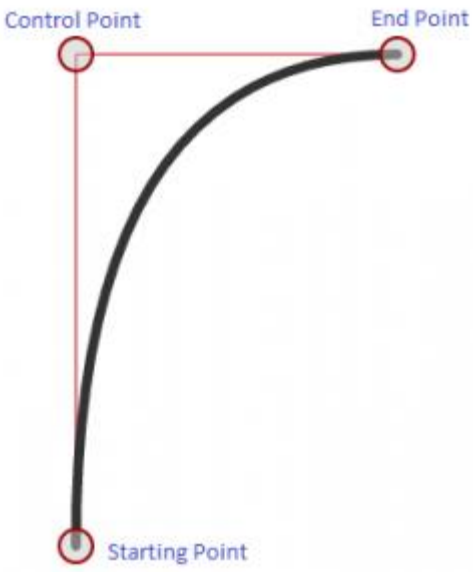
- **arc(x, y, radius, stratAngle, endAngle, *anticlockwise*)**

Drawing Curves on HTML5 Canvas



Drawing Curves on HTML5 Canvas





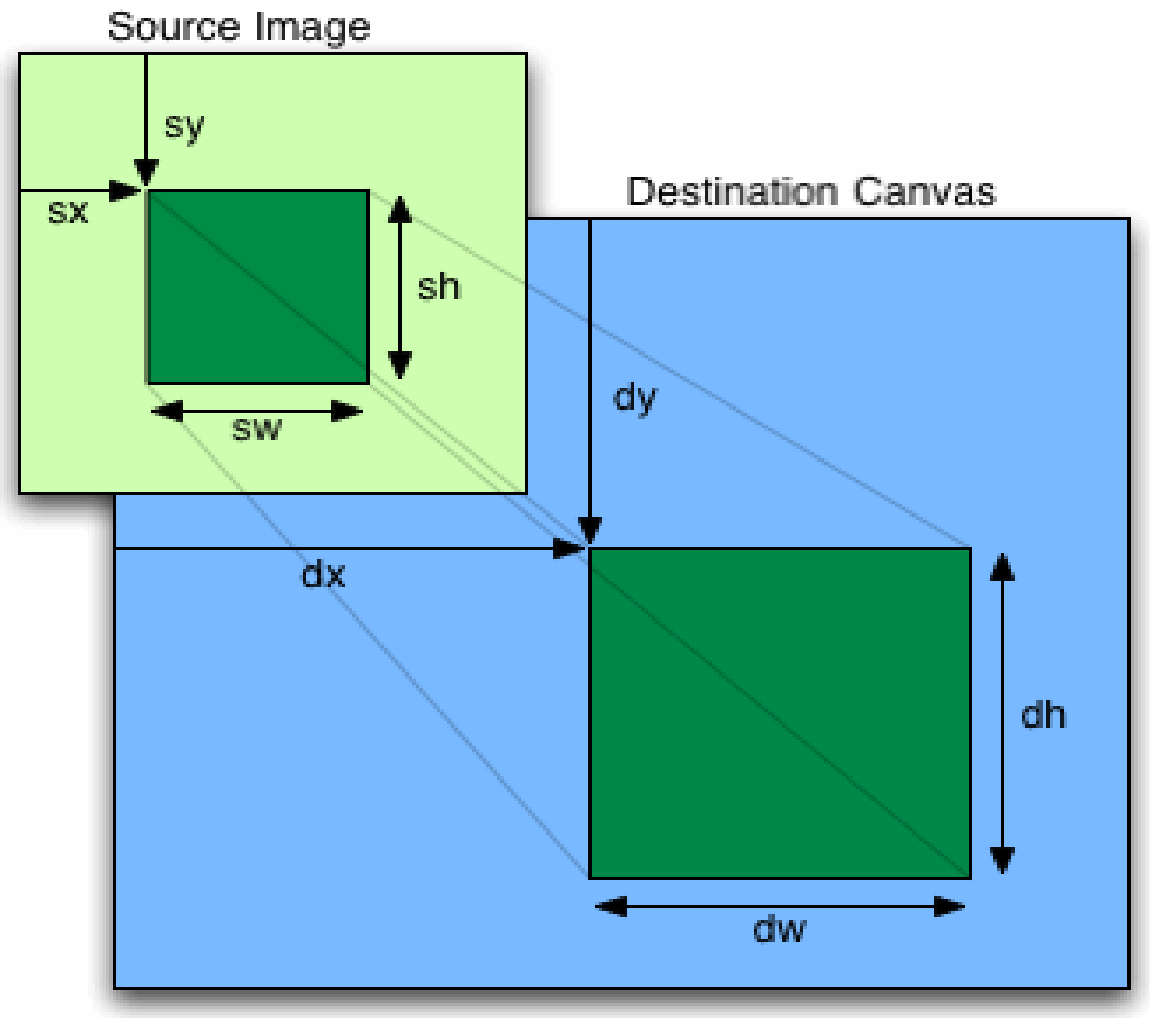
Colors in HTML5 canvas

- ctx.fillStyle = “red”;
- ctx.fillStyle = “#FF0000”;
- ctx.fillStyle = “rgb(255,0,0)”;
- ctx.fillStyle = “rgba(255,0,0,1)”;

Font and Text in Canvas

- **fillText(text, x, y)**
- **text – Any string you wish to display in your canvas**
- **x, y - Co-ordinates of the starting point of text**

Using the HTML5 Canvas APIs

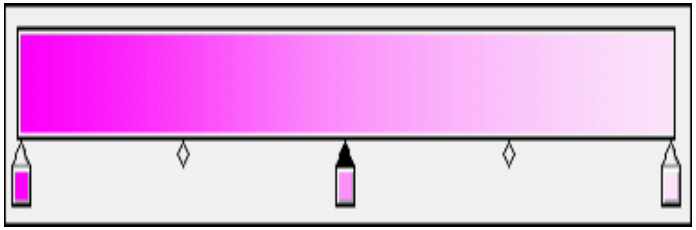


Using the HTML5 Canvas APIs

```
<canvas id="e" width="177" height="113">
</canvas>
<script>
 var canvas = document.getElementById("e");
var context = canvas.getContext("2d");
 var cat = new Image();
cat.src = "images/cat.png";
cat.onload = function()
{for (var x = 0, y = 0; x < 500 && y < 375;
x += 50, y += 37)
 { context.drawImage(cat, x, y, 88, 56); } };
 </script>
```

Canvas Gradients

Gradient



Definition

- Gradients consist of continuously smooth color transitions along a vector from one color to another, possibly followed by additional transitions along the same vector to other colors.

Canvas Gradients

- Linear gradient : A linear gradient defines color change along a line between two points.

Syntax :

```
var gradient = ctx.createLinearGradient(x0, y0, x1, y1);
```

x0 - The x-coordinate (in pixels), of the start point of the gradient.
y0 - The y-coordinate (in pixels), of the start point of the gradient.
x1 - The x-coordinate (in pixels), of the end point of the gradient.
y1 - number The y-coordinate (in pixels), of the end point of the gradient

Canvas Gradients

- **addColorStop(offset, color)**
  This method adds a color stop with the given color to the gradient at the given offset. Here 0.0 is the offset at one end of the gradient, 1.0 is the offset at the other end.

- **createRadialGradient(x0, y0, r0, x1, y1, r1)**
  This method returns a CanvasGradient object that represents a radial gradient that paints along the cone given by the circles represented by the arguments.
- The first three arguments define a circle with coordinates (x1,y1) and radius r1 and the second a circle with coordinates (x2,y2) and radius r2.

HTML5 Inline SVG

- SVG stands for Scalable Vector Graphics
- SVG is used to define vector-based graphics for the Web
- SVG defines the graphics in XML format
- SVG graphics do NOT lose any quality if they are zoomed or resized
- Every element and every attribute in SVG files can be animated
- SVG is a W3C recommendation

SVG Advantages

- Advantages of using SVG over other image formats (like JPEG and GIF) are:
- SVG images can be created and edited with any text editor
- SVG images can be searched, indexed, scripted, and compressed
- SVG images are scalable
- SVG images can be printed with high quality at any resolution
- SVG images are zoomable (and the image can be zoomed without degradation)

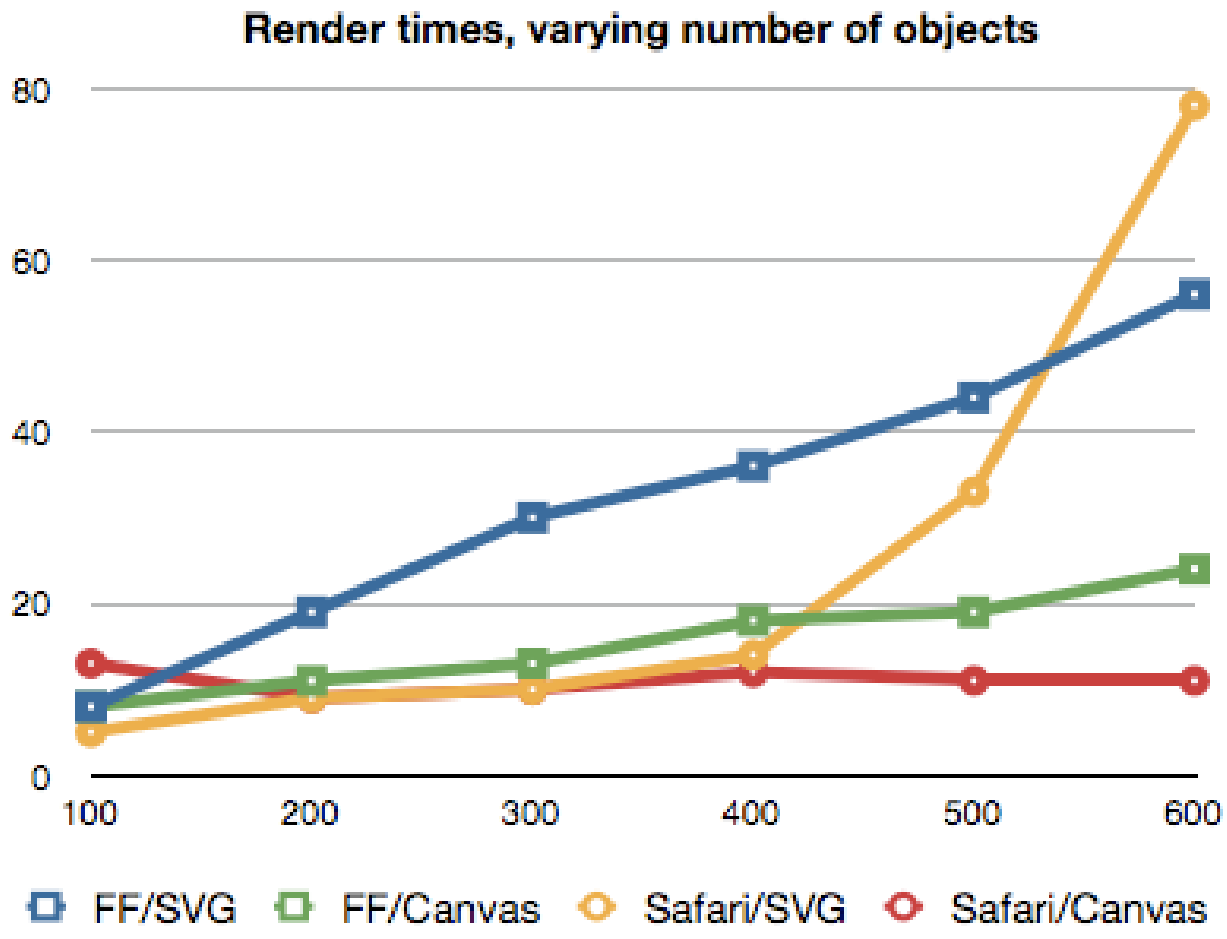
Comparison of Canvas and SVG

Canvas	SVG
•Resolution dependent •No support for event handlers •Poor text rendering capabilities •You can save the resulting image as .png or .jpg •Best suited for graphic-intensive games where many objects are redrawn frequently	•Resolution independent •Support for event handlers •Best suited for applications with large rendering areas (Google Maps) •Slow rendering if complex (anything that uses the DOM a lot will be slow) •Not suited for game applications

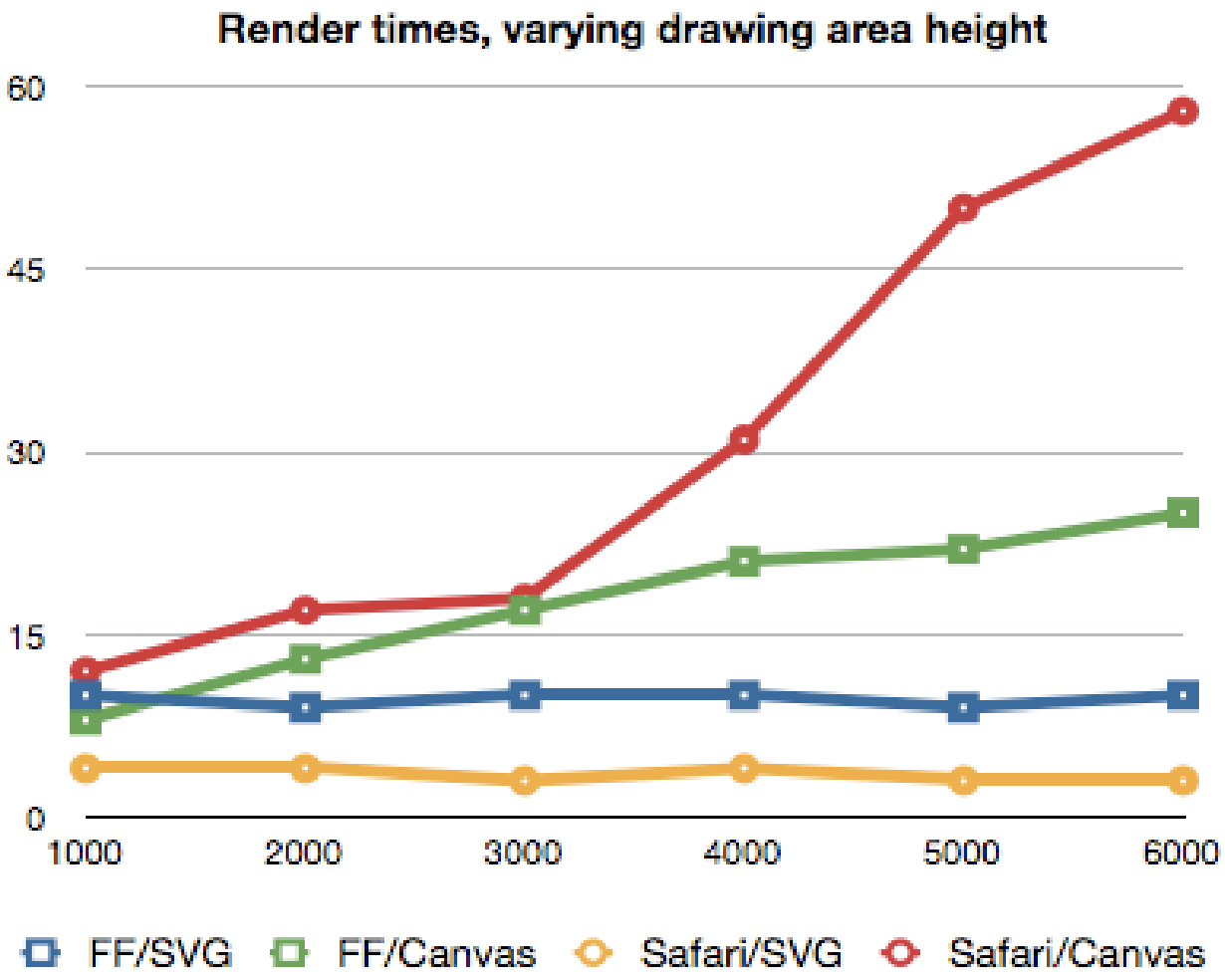
Performance of Canvas versus SVG

Canvas	SVG
Pixel based (Dynamic .png)	Shape based
Single HTML element	Multiple graphical elements, which become part of the DOM
Modified through script only	Modified through script and CSS
Event model/user interaction is granular (x,y)	Event model/user interaction is abstracted (rect, path)
Performance is better with smaller surface, a larger number of objects (>10k), or both	Performance is better with smaller number of objects (<10k), a larger surface, or both

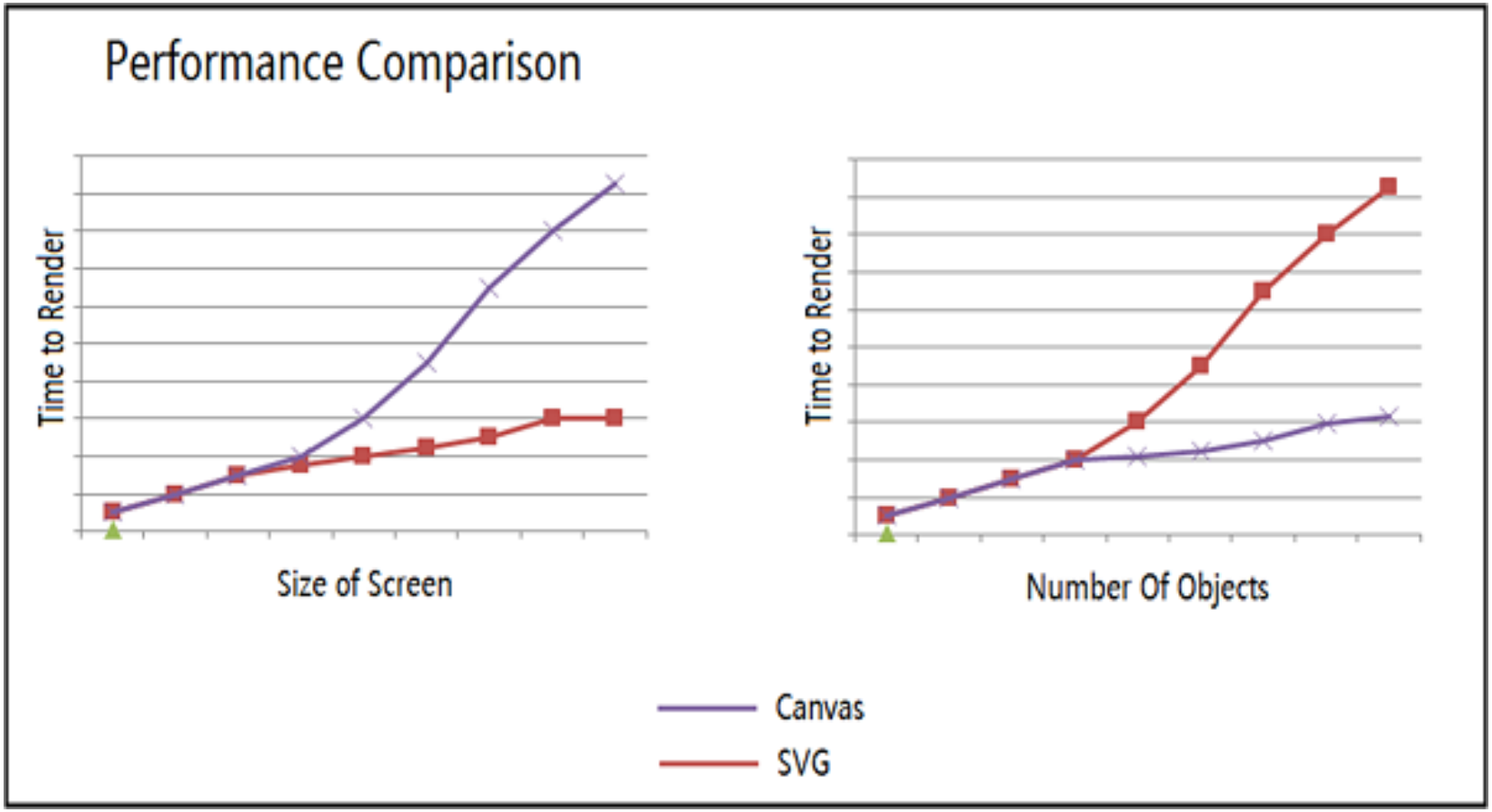
Performance of Canvas versus SVG



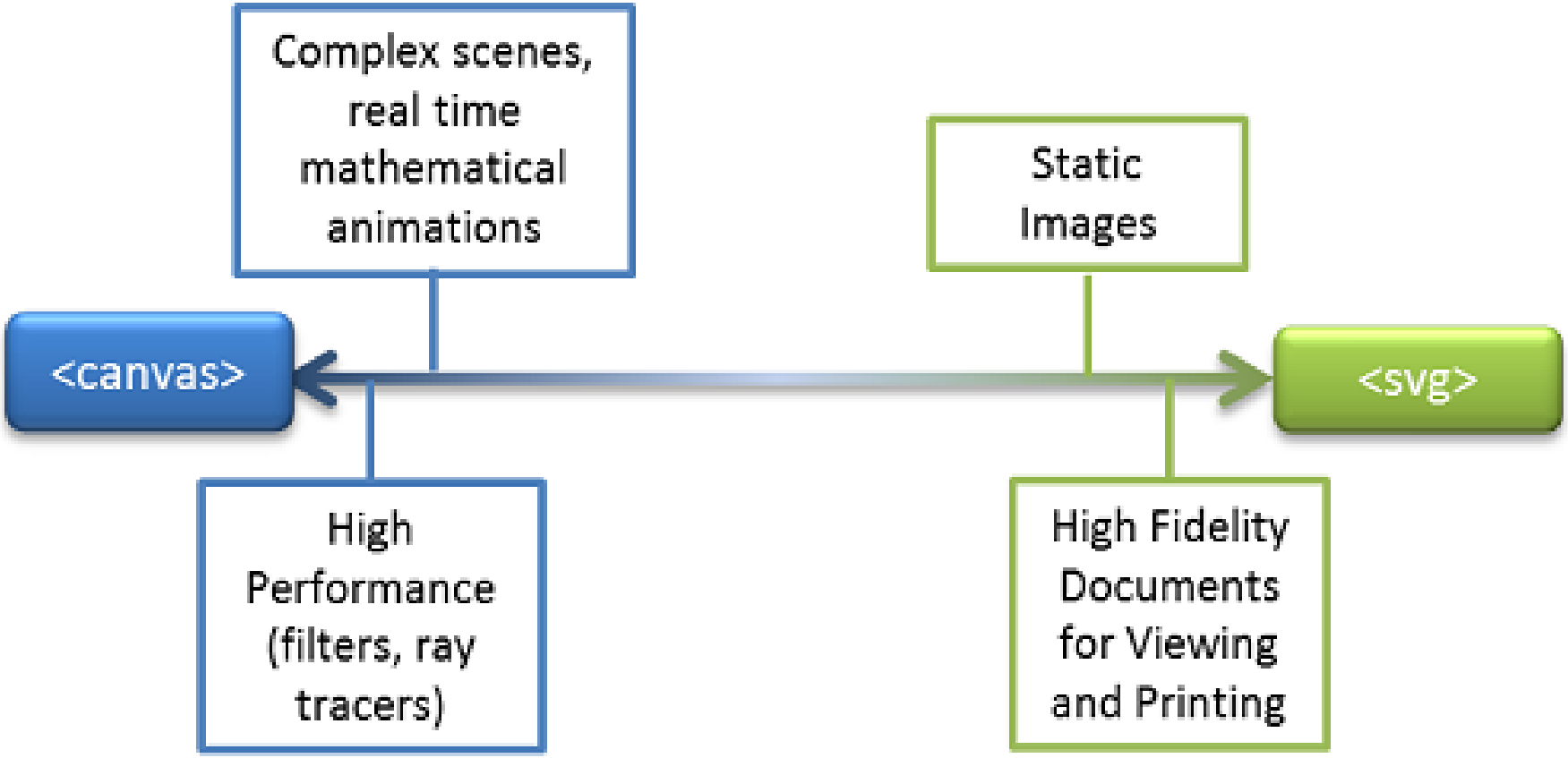
Performance of Canvas versus SVG



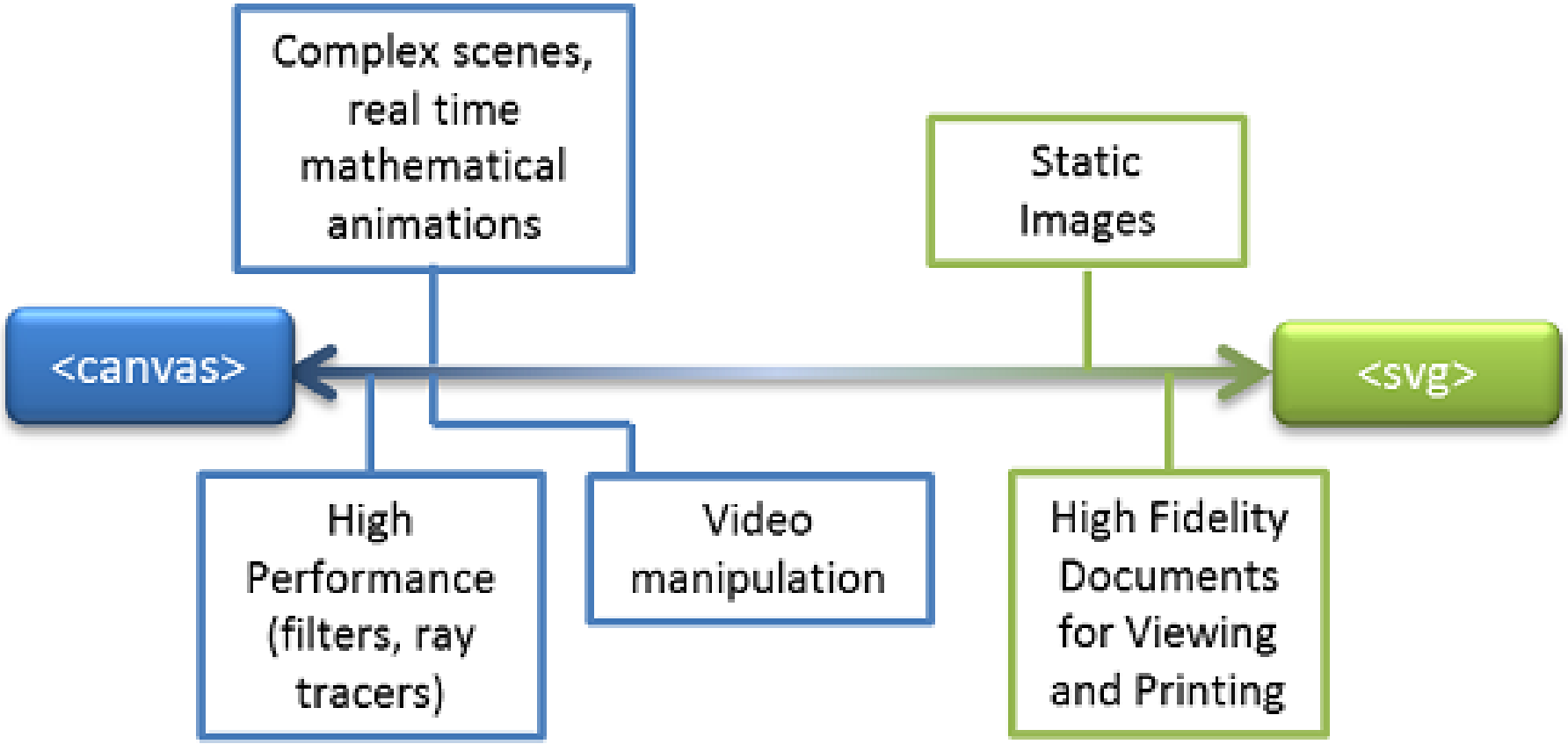
Performance of Canvas versus SVG



Performance of Canvas versus SVG



Performance of Canvas versus SVG



Learning and reusing SVG

- <svg xmlns="http://www.w3.org/2000/svg"> <circle id="greencircle" cx="30" cy="30" r="30" fill="green" /> </svg>

Creating basic shapes

Element	Description
line	Creates a simple line.
polyline	Defines shapes built from multiple line definitions.
rect	Creates a rectangle.
circle	Creates a circle.
ellipse	Creates an ellipse.
polygon	Creates a polygon.
path	Allows for the definition of arbitrary paths.

Add the SVG Element

```
<svg version="1.1"
      baseProfile="full"
      xmlns="http://www.w3.org/2000/svg"
      height="100%" width="100%">

</svg>
```

Create the Definitions

```
<defs>
  <linearGradient id="labelGrad1" x1="0%" y1="0%" x2="100%"
y2="0%">
    <stop offset="0%" style="stop-color:#990000; stop-opacity:1" />
    <stop offset="20%" style="stop-color:#ff6600; stop-opacity:1" />
    <stop offset="40%" style="stop-color:#cccc00; stop-opacity:1" />
    <stop offset="60%" style="stop-color:#009900; stop-opacity:1" />
    <stop offset="80%" style="stop-color:#000099; stop-opacity:1" />
    <stop offset="100%" style="stop-color:#990099; stop-opacity:1" />
</linearGradient>

</defs>
<rect width="90%" height="90%" x="5%" y="5%" fill="url(# labelGrad1)"
/>
```

Create the Definitions

```
<defs>
  <radialGradient id="recordGrad" cx="50%" cy="50%" r="50%"
fx="50%" fy="50%">
    <stop offset="30%" style="stop-color:#000000; stop-opacity:1" />
    <stop offset="35%" style="stop-color:#222222; stop-opacity:1" />
    <stop offset="45%" style="stop-color:#000000; stop-opacity:1" />
    <stop offset="85%" style="stop-color:#000000; stop-opacity:1" />
    <stop offset="95%" style="stop-color:#222222; stop-opacity:1" />
    <stop offset="100%" style="stop-color:#000000; stop-opacity:1" />
</radialGradient>

</defs>
<circle cx="50%" cy="50%" r="33%" fill="#000000"
filter="url(#ecordGrad)" />
```

SVG Logo

```
<svg height="130" width="500">

   <defs>
      <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
          <stop offset="0%" style="stop-color:rgb(255,180,0);stop-opacity:1" />
          <stop offset="100%" style="stop-color:rgb(255,150,0);stop-opacity:1" />
      </linearGradient>
      </defs>
    <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
   <text fill="#ffffff" font-size="45" font-family="Verdana" x="30"
y="86">Home</text>
     Sorry, your browser does not support inline SVG.
  </svg>
```

SVG Logo

```
<svg height="150" width="500">
  <ellipse cx="240" cy="100" rx="220" ry="30"
style="fill:purple" />
  <ellipse cx="220" cy="70" rx="190" ry="20"
style="fill:lime" />
  <ellipse cx="210" cy="45" rx="170" ry="15"
style="fill:yellow" />
</svg>
```

SVG Stroke Properties

- stroke
- stroke-width
- stroke-linecap
- stroke-dasharray

G Element

- The g element is a container used to group objects.
- Transformations applied to the g element are performed on all of its child elements.
- Attributes applied are inherited by child elements.

SVG <path>

- **SVG Path - <path>**
- The <path> element is used to define a path.
- The following commands are available for path data:
- M = moveto
- L = lineto
- H = horizontal lineto
- V = vertical lineto
- C = curveto
- S = smooth curveto
- Q = quadratic Bézier curve
- T = smooth quadratic Bézier curveto
- A = elliptical Arc
- Z = closepath

SVG Stroke

```
<svg height="80" width="300">
  <g fill="none">
    <path stroke="red" d="M5 20 l215 0" />
    <path stroke="blue" d="M5 40 l215 0" />
    <path stroke="black" d="M5 60 l215 0" />
  </g>
</svg>
```

SVG Stroke Width

```
<svg height="80" width="300">
  <g fill="none" stroke="black">
    <path stroke-width="2" d="M5 20 l215 0" />
    <path stroke-width="4" d="M5 40 l215 0" />
    <path stroke-width="6" d="M5 60 l215 0" />
  </g>
</svg>
```

Applying Styles

Line Settings

SVG Stroke Line Cap

```
<svg height="80" width="300">
  <g fill="none" stroke="black" stroke-width="6">
    <path stroke-linecap="butt" d="M5 20 l215 0" />
    <path stroke-linecap="round" d="M5 40 l215 0" />
    <path stroke-linecap="square" d="M5 60 l215 0" />
  </g>
</svg>
```

SVG Stroke Dash Array

```
<svg height="80" width="300">
  <g fill="none" stroke="black" stroke-width="4">
    <path stroke-dasharray="5,5" d="M5 20 l215 0" />
    <path stroke-dasharray="10,10" d="M5 40 l215 0"
/>
    <path stroke-dasharray="20,10,5,5,5,10" d="M5 60
l215 0" />
  </g>
</svg>
```

SVG Rounded Rectangle

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20"
width="150" height="150"
  style="fill:red;stroke:black;stroke-
width:5;opacity:0.5" />
</svg>
```

SVG Line

```
<svg xmlns="http://www.w3.org/2000/svg"
version="1.1" height="800">
    <line x1="100" y1="100" x2="400" y2="100"
style="stroke:rgb(255,0,0);stroke-width:4" ></line>

</svg>
```

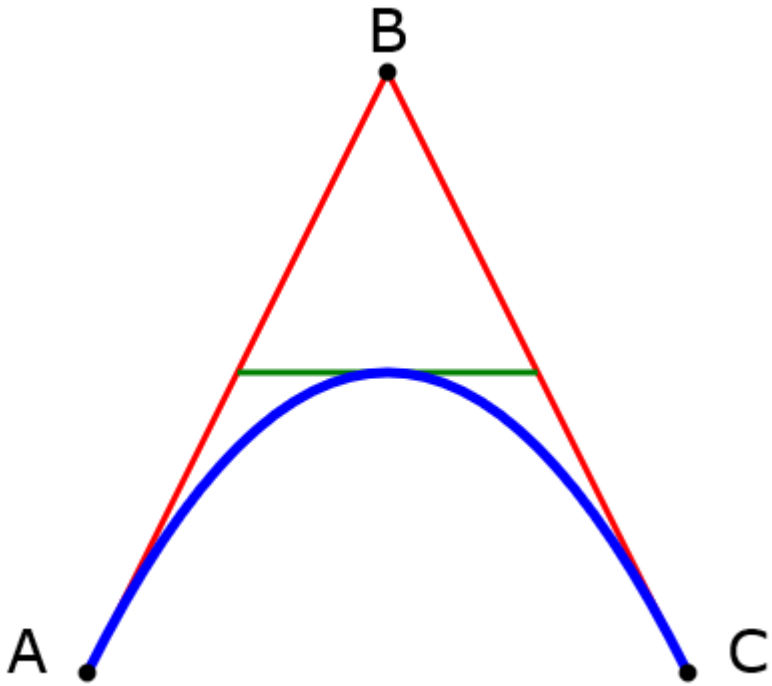
SVG Polygon

```
<svg height="210" width="500">

    <polygon points="200,100 500,100 300 200"
style="fill:lime;stroke:purple;stroke-width:1" />

</svg>
```

SVG <path>

- <svg height="210" width="400">
  <path d="M150 0 L75 200 L225 200 Z" />
  </svg>

SVG <path>



SVG <path>

- <path id="lineAB" d="M 100 350 l 150 -300" stroke="red" stroke-width="3" fill="none" />
- <path id="lineBC" d="M 250 50 l 150 300" stroke="red" stroke-width="3" fill="none" />
- <path d="M 175 200 l 150 0" stroke="green" stroke-width="3" fill="none" />

SVG <path> Quad Curve

- <svg height="400" width="450">
- <path d="M 100 350 q 150 - 300 300 0" stroke="blue"
- stroke-width="5" fill="none" />

SVG <text>

- <svg height="60" width="200">
  <text x="0" y="15" fill="red" transform="rotate(30 20,40)">Welcome</text>
  </svg>

SVG Animations

- SVG graphics can be animated using **animation elements.**
- <animate> - which allows you to animate scalar attributes and properties over a period of time.
- <set> - which is a convenient shorthand for animate, which is useful for assigning animation values to non-numeric attributes and properties, such as the visibility property.
- <animateMotion> - which moves an element along a motion path.
- <animateColor> - which modifies the color value of particular attributes or properties over time.

The SVG extensions

- <animateTransform> - allows you to animate one of SVG's transformation attributes over time, such as the transform attribute.
- path (attribute) - allows any feature from SVG's path data syntax to be specified in a path attribute to the animateMotion element.
- <mpath> - used in conjunction with the animateMotion element to reference a motion path that is to be used as a path for the motion.

The SVG extensions

- keypoints (attribute) - used as an attribute for animateMotion to provide precise control of the velocity of motion path animations.
- rotate (attribute) - used as an attribute for animateMotion to control whether an object is automatically rotated so that its x-axis points in the same direction as the directional tangent vector of the motion path.

SVG animate- Height

- <svg>
- <rect width="200" height="200" fill="slategrey">

- <animate attributeName="height" from="0" to="200" dur="3s"/>
- </rect>
- </svg>

SVG animate- Radius

- <svg>
- <circle r="50" cx="350" cy="100" fill="slategrey">
- <animate attributeName="r" from="0" to="50" dur="3s"/>
- </circle></svg>

SVG animate- Movement

- <svg>
- <rect x="0" y="250" width="200" height="200" fill="slategrey">
- <animate attributeName="x" from="0" to="200" dur="3s" fill="freeze"/>
- </rect></svg>

SVG animate- Multiple

- <circle r="50" cx="500" cy="100" fill="slategrey" stroke="#000" stroke-width="7">
- <animate attributeName="r" from="0" to="50" dur="3s"/>
- <animate attributeName="stroke-width" from="0" to="20" dur="7s"/>
- </circle>

SVG with CSS

- <svg xmlns="http://www.w3.org/2000/svg" >
  <style type="text/css" >
- <![CDATA
- [circle { stroke: #006600; fill: #00cc00; }]
-]
- >
- </style>
- <circle cx="40" cy="40" r="24"/>
- </svg>

SVG with CSS class

- <svg xmlns="http://www.w3.org/2000/svg" > <style type="text/css" >
- <![CDATA
- [circle.myGreen
- { stroke: #006600; fill: #00cc00; }
- circle.myRed { stroke: #660000; fill: #cc0000;
- }]]>
- </style>
- <circle class="myGreen" cx="40" cy="40" r="24"/> <circle class="myRed" cx="40" cy="100" r="24"/>
- </svg>

SVG with External stylesheet

- <?xml-stylesheet type="text/css" href="svg-stylesheet.css" ?>
- <svg xmlns="http://www.w3.org/2000/svg" > <circle cx="40" cy="40" r="24" style="stroke:#006600; fill:#00cc00"/>
- </svg>

GEO LOCATION API

- Geolocation is the art of figuring out where you are in the world and (optionally) sharing that information with people you trust.
- There is more than one way to figure out where you are — your IP address, your wireless network connection, which cell tower your phone is talking to, or dedicated GPS hardware that calculates latitude and longitude from information sent by satellites in the sky.

Location-based service

- Location-based services (LBS) are a general class of computer program-level services that use location data to control features.
- LBS is an information service.
- It is accessible with mobile devices through the mobile network.
- It uses information on the geographical position of the mobile device.

The Geolocation API

Geolocation API support						
IE	Firefox	Safari	Chrome	Opera	iPhone	Android
9.0+	3.5+	5.0+	5.0+	10.6+	3.0+	2.0+

Check for Browser compatibility

```
if (navigator.geolocation) {
    // Get the user's current position
} else {
    alert('Geolocation is not supported in your browser');
}
```

Get the user's current location

- if (navigator.geolocation) {
-
- // Get the user's current position
- navigator.geolocation.getCurrentPosition(showPositio n, showError, optn);
- } else {
- alert('Geolocation is not supported in your browser');
- }

Position Object

Property	Type	Notes
coords.latitude	double	decimal degrees
coords.longitude	double	decimal degrees
coords.altitude	double or null	meters above the reference ellipsoid
coords.accuracy	double	meters
coords.altitudeAccuracy	double or null	meters
coords.heading	double or null	degrees clockwise from true north
coords.speed	double or null	meters/second
timestamp	DOMTimeStamp	like a Date() object

Success callback function

- function showPosition(position) {
- document.write('Latitude: '+position.coords.latitude+'Longitude: '+position.coords.longitude);
- }

Error callback function

```
function showError(error) {
    switch(error.code) {
        case error.PERMISSION_DENIED:
            alert("User denied the request for Geolocation.");
            break;
        case error.POSITION_UNAVAILABLE:
            alert("Location information is unavailable.");
            break;
        case error.TIMEOUT:
            alert("The request to get user location timed out.");
            break;
        case error.UNKNOWN_ERROR:
            alert("An unknown error occurred.");
            break;
    }
}
```

Show My Location on Google Maps

- var googlePos = new google.maps.LatLng(position.coords.latitude, position.coords.longitude);

Show My Location on Google Maps

- var mapOptions = {
- zoom : 12,
- center : googlePos,
- mapTypeId : google.maps.MapTypeId.ROADMAP
- };
- var mapObj = document.getElementById('mapTxt');
- var googleMap = new google.maps.Map(mapObj, mapOptions);

Show My Location on Google Maps

```
var markerOpt = {
        map : googleMap,
        position : googlePos,
        title : 'Hi , I am here',
        animation : google.maps.Animation.DROP
        };
    var googleMarker = new google.maps.Marker(markerOpt);
```

New Media Elements

Tag	Description
<audio>	Defines sound content
<video>	Defines a video or movie
<source>	Defines multiple media resources for <video> and <audio>
<embed>	Defines a container for an external application or interactive content (a plug-in)
<track>	Defines text tracks for <video> and <audio>

HTML5 <video> - Methods, Properties, and Events

Methods	Properties	Events
play()	currentSrc	play
pause()	currentTime	pause
load()	videoWidth	progress
canPlayType	videoHeight	error
	duration	timeupdate
	ended	ended
	error	abort
	paused	empty
	muted	emptied
	seeking	waiting
	volume	loadedmetadata
	height	
	width	

Poster Attribute

- <video controls poster="/images/w3html5.gif">
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
  Your browser does not support the video tag.
  </video>

Video Element Support

<video> element support						
IE	Firefox	Safari	Chrome	Opera	iPhone	Android
9.0+	3.5+	3.0+	3.0+	10.5+	1.0+	2.0+

Browser Support CODECS

	Android	Chrome		Firefox		Internet Explorer	iOS	Opera		Safari	
Codec		Win	Mac	Win	Mac	Win		Win	Mac	Win	Mac
MP4 or H.264	3.0	9	7	X	X	9	3	X	X	5	5
ogg/Theora	2.3	9	7	3.6	3.6	X	X	10.63	10.63	X	X
WebM	2.3	9	7	3.6	3.6	9 (with components)	X	10.63	10.63	X	X

Browser Support CODECS

Video	IE8	IE9+	Chrome	Safari	Mobile	Firefox	Opera
MP4 (h.264/ AAC)	no	yes	yes	yes	yes	no	no
WebM (VP8/Vorbis)	no	install	yes	no	no	yes	yes

Video-Specific Attributes

- The **video** element can be laid out easily by using CSS
- The **width** and **height** attributes specify the dimensions of the viewing area
- The video always maintains its innate aspect ratio
- The **poster** frame image is shown before the video is played

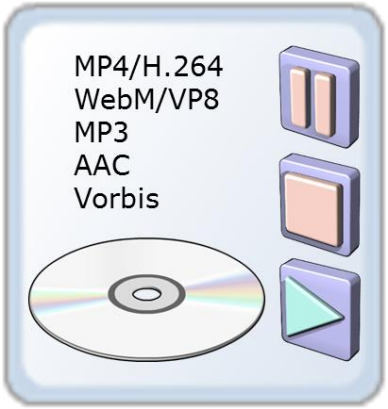
```
<video src="video.m4v" poster="first_frame.png" width=”100px”
height=”100px”></video>
```

Alternative Sources

- The **source** element specifies alternate media resources
- Multiple **source** elements can be specified
- The browser uses only the first **source** element it supports
- Specifying the MIME type with the **type** attribute avoids downloading unsupported files
- The **media** attribute constrains the environment by using *media queries*

```
video>
  <source src="video.m4v" " type="video/mp4" />
</video>
<audio>
  <source src="audio.m4a" " type="video/mp4" />
</audio>
```

Video and Audio Formats



Format	IE	Firefox	Opera	Chrome	Safari
Ogg	No	Yes	Yes	Yes	No
MP3	Yes	No	No	Yes	Yes
Wav	No	Yes	Yes	Yes	Yes

Controlling Playback

- The **play** method starts playback
- The **pause** method pauses playback
- The **playbackRate** property changes the playback speed
- The **currentTime** property reports the current position and seeks
- The **duration** property reports the length of the video or audio
- The **buffered** property lists the currently downloaded data
- Many properties and events report status and progress

```
var video = document.getElementById("video");
video.play();
window.setTimeout(function () {
    video.pause();
}, 3000);
```

Integrating Video and the Canvas

- Video uses time, and the canvas does not
- Time loops are used to simulate time
- The **setInterval** and **setTimeout** methods drive the time loop
- The Video and Canvas APIs work well together

```
function frame() {
    var context =
document.getElementyId("canvas").getContext("2d");
    var video = document.getElementById("video");
    context.drawImage(video, 0, 0);
    window.setTimeout(frame, 100);
}

frame(); // calling the function directly triggers the
first iteration
```

HTML5 Audio

```
<audio controls="controls">
  <source src="horse.ogg" type="audio/ogg" />
  <source src="horse.mp3" type="audio/mpeg" />
  Your browser does not support the audio element.
  </audio>
```

HTML5 Audio

Audio Formats and Browser Support

Currently, there are 3 supported file formats for the <audio> element: MP3, Wav, and Ogg:

Browser	MP3	Wav	Ogg
Internet Explorer 9	YES	NO	NO
Firefox 4.0	NO	YES	YES
Google Chrome 6	YES	YES	YES
Apple Safari 5	YES	YES	NO
Opera 10.6	NO	YES	YES

HTML5 Audio Tags

Tag	Description
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and < audio>

Media Elements and Global Attributes

- accesskey
- class
- contenteditable
- contextmenu
- dir
- draggable
- dropzone
- hidden
- id
- Spellcheck
- tabindex
- title
- data-*

Video-Only Attributes and Video Resolutions

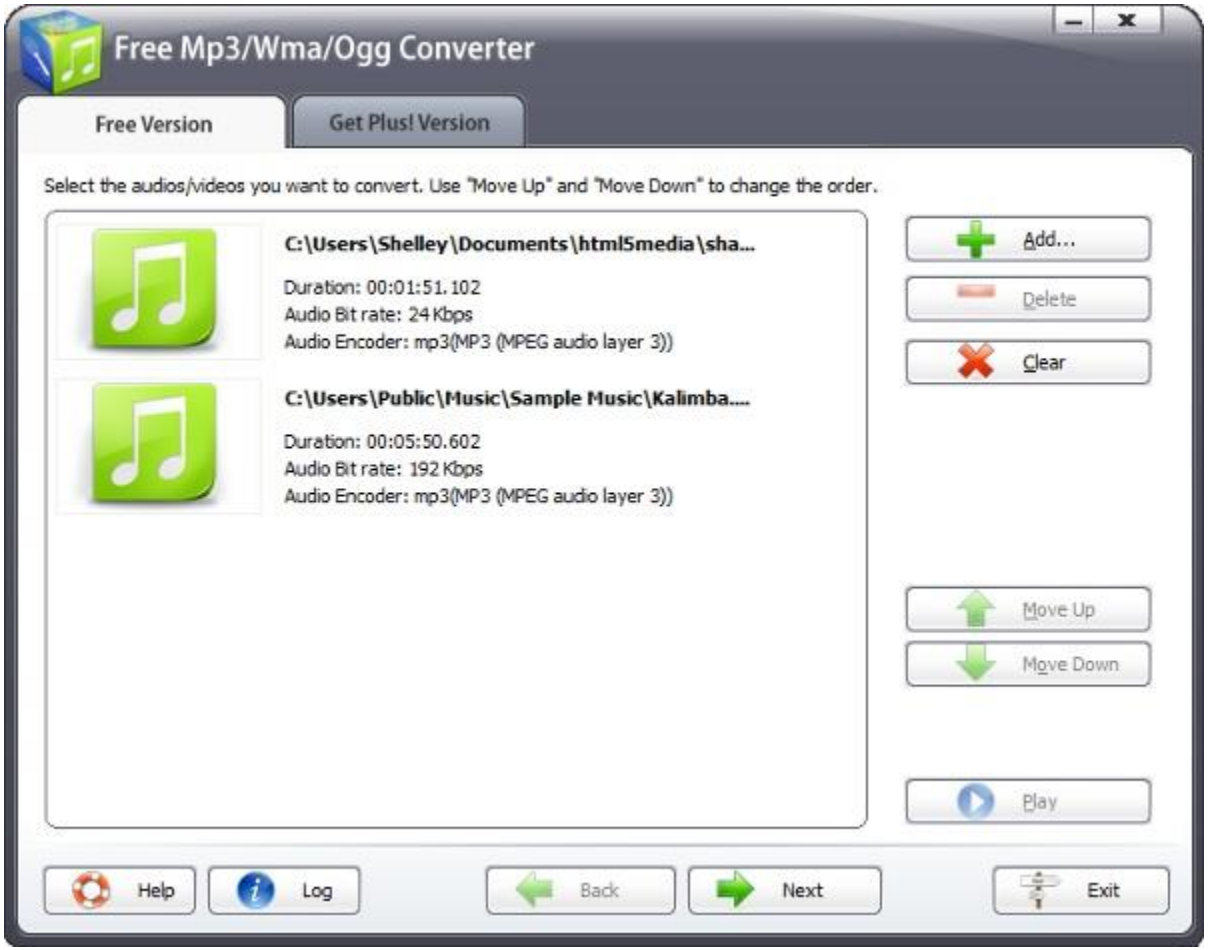
- poster
- width, height

Audio and Video in Mobile Devices and Media Profiles

- Apple has been a big fan of HTML5 from the beginning, deciding against support for Flash on iOS devices in favor of HTML5 video.
- However, some things that work on the desktop don't in an Apple mobile environment.
- As an example, using the poster attribute caused the video element to fail in iOS 3, though this problem has been fixed in iOS 4.

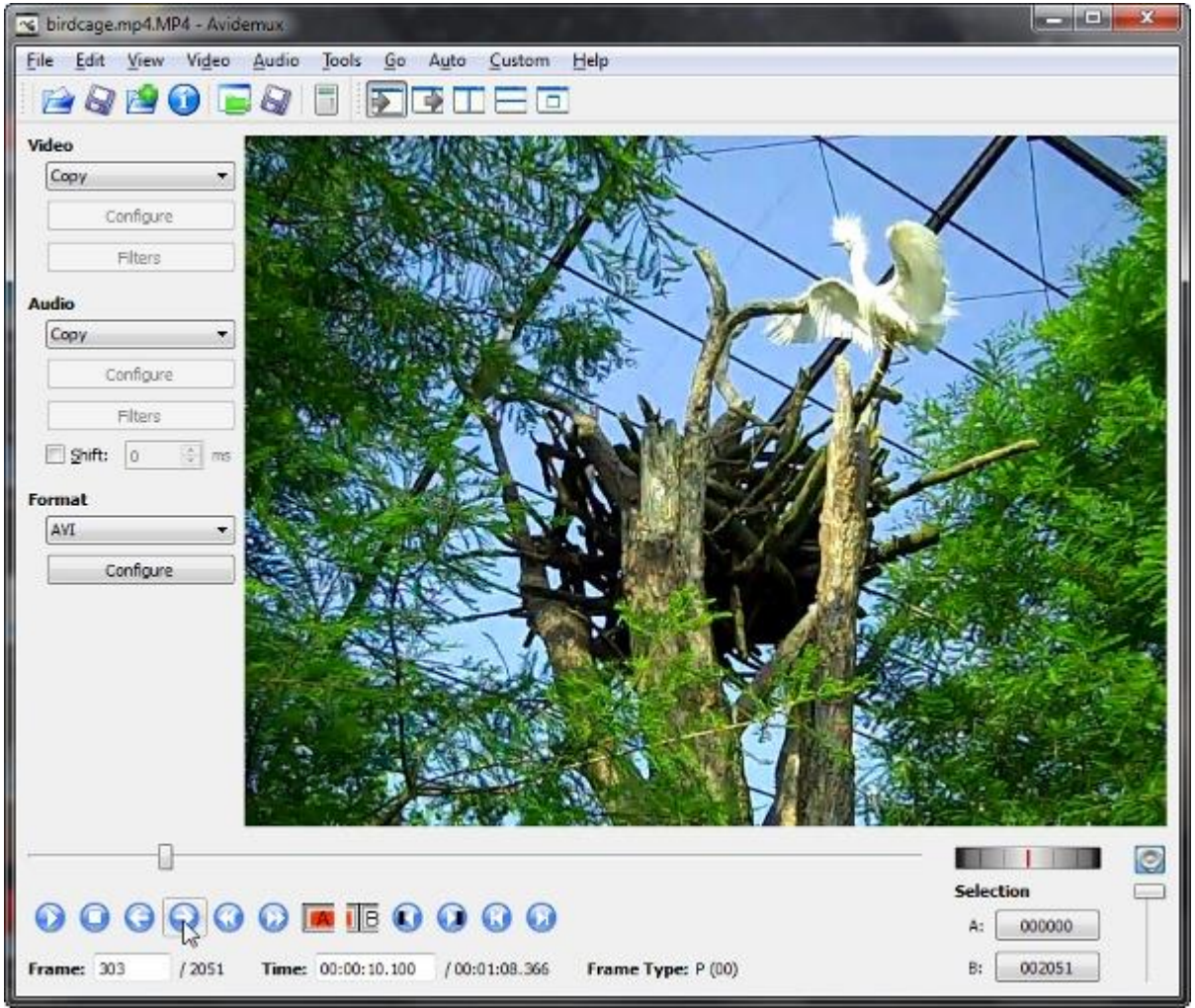
Converting Audio and Video Content

- **The Free Mp3/Wma/Ogg Converter**



Converting Audio and Video Content

- **Using a Frame Grabber**



Getting started with the HTML5 track element

- The track element provides a simple, standardized way to add subtitles, captions, screen reader descriptions and chapters to video and audio.

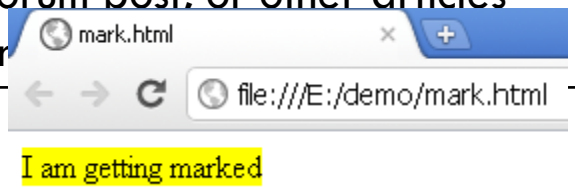
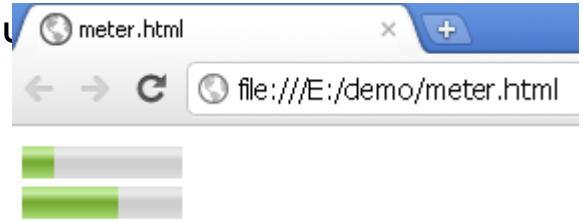
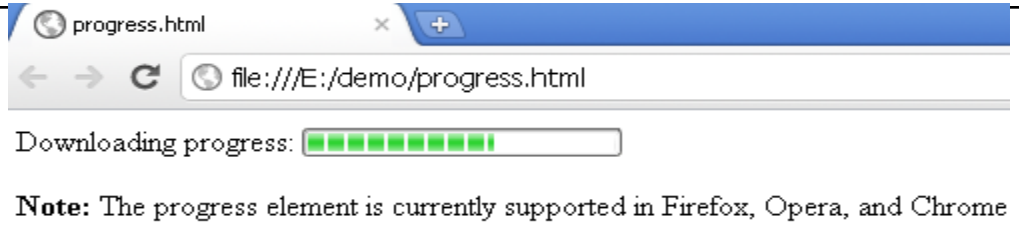
Summary

- Introduced some of the most exciting features in HTML5. The multimedia elements—**video** and **audio**
- Many accessibility features.
- Understanding APIs that programmers can use to control
- multimedia.

New Elements in HTML5

- HTML5 includes new elements for better structure, drawing, media content and better form handling
- New media elements like <audio>, <video>,<source>
- Canvas element <canvas> uses JavaScript to make drawing on a web page
- The Input elements type attribute has many new values, for better input control before sending it to the server

Some new markup elements

<article>	Used to specify a news-article, blog post, forum post, or other articles which can be distributed independently fro
<mark>	For text that should be highlighted 
<meter>	For a measurement, used only if the maximu known 
<progress>	The <progress> tag 

Summary

- At the end of this sub module:
  - Understand HTML 5
  - Different browser support for HTML 5
  - Different HTML 5 elements

HTML 5 Structural Elements

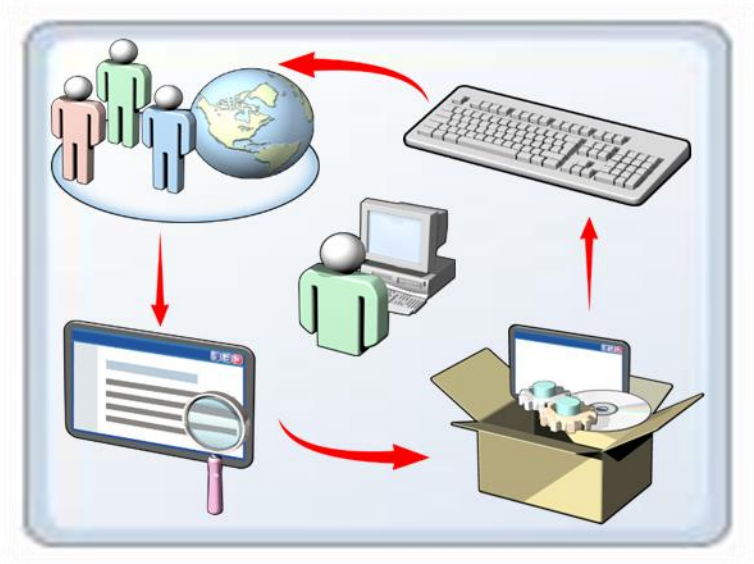
Objectives

- Explain the rationale behind the new HTML5 elements:
- Describe where and how to use the new HTML5 structural elements
- Describe what are the new semantic block-level elements
- Describe what are the new inline semantic elements
- Describe what are the new interactive elements

New HTML 5 – Rationale Behind

New HTML5 elements enable:

- Accessibility
- Searchability
- Internationalization
- Interoperability



The New Structural Elements

- New structural HTML5 elements:
  - **Article**
  - **Footer**
  - **Header**
  - **Hgroup**
  - **Section**
  - **Address**
  - **Nav**
  - **Aside**

```
<body>
    <header>
        <nav>
            <ul>
                <li>Navigation</li>
            </ul>
        </nav>
    </header>
    <section>
        <article>
            <p>Article content</p>
        </article>
    </section>
    <footer>
        <p>Copyright</p>
    </footer>
</body>
```

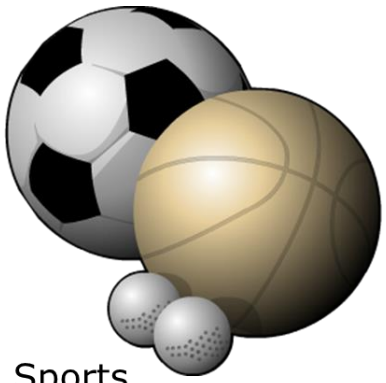
- No need for **div** elements all over the webpage

Semantic Block-Level Elements

Semantic Block-Level Elements

- Block elements that have a specific semantic meaning
- Include the figure element

```
<figure>
   <img src="imageurl" alt="image alt">
   <figcaption>Sports</figcaption>
</figure>
```



Sports

Inline Semantic Elements

- Inline elements affect their content.
- Examples:
  - **Mark**
  - **Meter**
  - **Progress**
  - **Output**
  - **Time**

```
This is <mark>marked</mark> text
<meter value="70" min="0" max="100" low="54" high="100" optimum="100">70</meter>
Your download is <progress max="100" value="60">60%</progress> complete
Output: <input type="text" value="1" /> + <input type="text" value="3" /> =
<output>4</output>
The time is now <time datetime="2011-09-28T18:36">6:36 P.M. on November 28th</time>
```

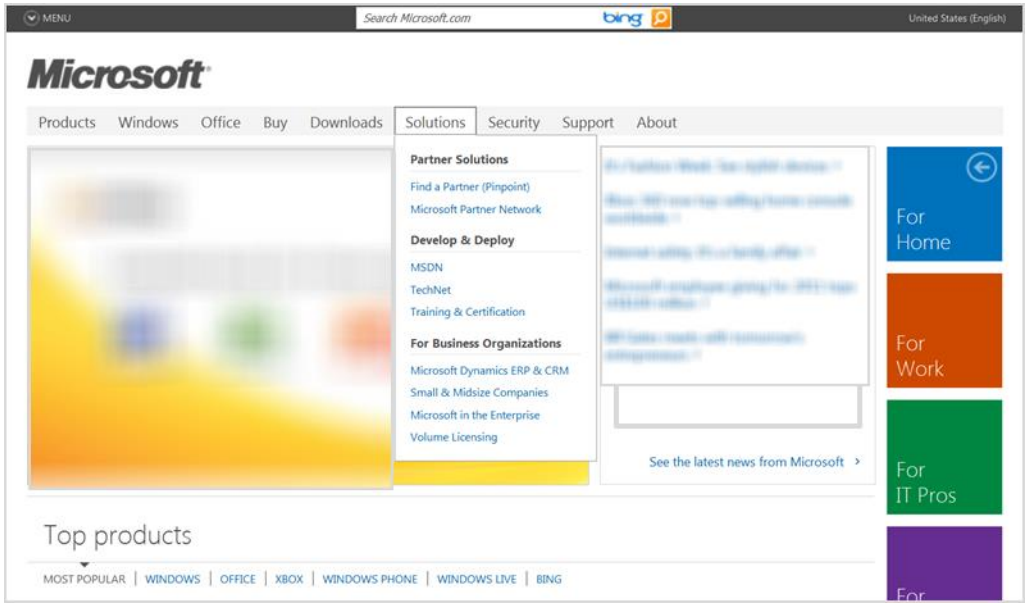
This is marked text

The time is now 6:36 P.M. on November 28th

Interactive Elements

- Interactive elements include:
  - **Details**
  - **Summary**
  - **Menu**
  - **Command**

```
<details>
   <summary>The summary</summary>
   <p>Some details</p>
</details>
```



Summary

In the module, you learned about:

- The new HTML5 elements and the rationale behind the creation of these elements.
- How and when to use them.
- The use of navigation and menus.

Creating Form Input and Validation

Objectives

At the end of this sub-module you will know:

- HTML 5 new Input types
- HTML 5 new form elements
- HTML 5 new form attributes
- Using Browser Detection, Feature Detection, and Modernizr

Challenges with HTML 4.0 Forms

- The HTML 4 form fields are not validated
- The validation to the input fields can be done by
  - Writing JavaScript code in the Client side Code
  - Or the validation needs to be done at the Server side
- Almost every web page has some kind of form with search, email, sign-up etc
- It would be great if browsers had built in validation for some of the common data types we collect

HTML – New Input types

- HTML 5 has lot of new input types

- These new features allow for better input control and validation
  - email
  - url
  - number
  - range
  - Date pickers (date, month, week, time, datetime-local)
  - search
  - color

Input type - email

- It represents a control for editing a list of e-mail addresses given in the element's value
- HTML email address validation will only work in browsers which have support for it

```
<input type="email" placeholder="Please enter your email" required
multiple="multiple">
```

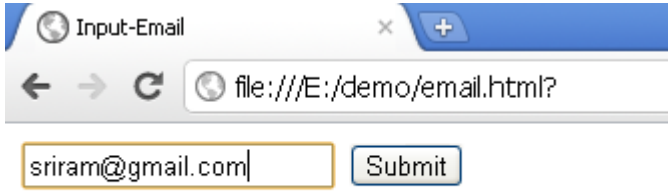
- required - specifies that the element is a required part of form submission
- placeholder - a short phrase to help the user when entering data into the control
- multiple - specifies that the element allows multiple values

Input type – email (Contd.).

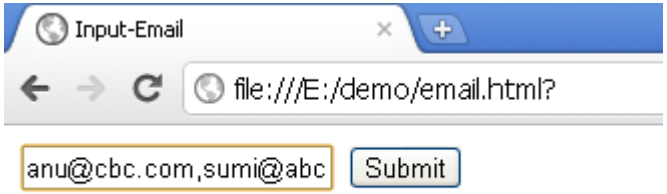
Field is empty

The value should be an email

Valid email – with a single email



Valid-More than 1 email



Input type - url

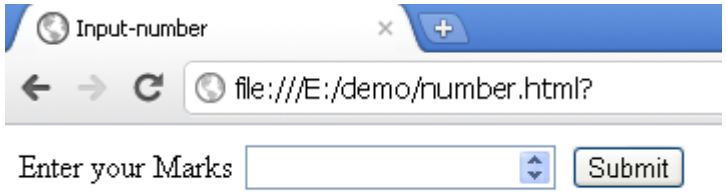
```
<input type="url" name="URL">
```

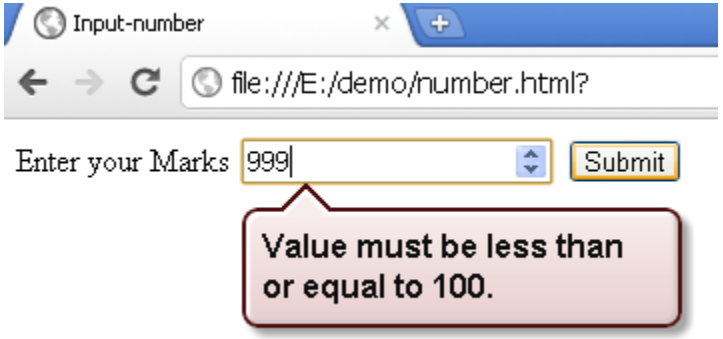
- It is used for input fields that should contain a URL address
- The value of the url field is automatically validated when the form is submitted

Input type - number

```
<input type="number" min="0" max="100">
```

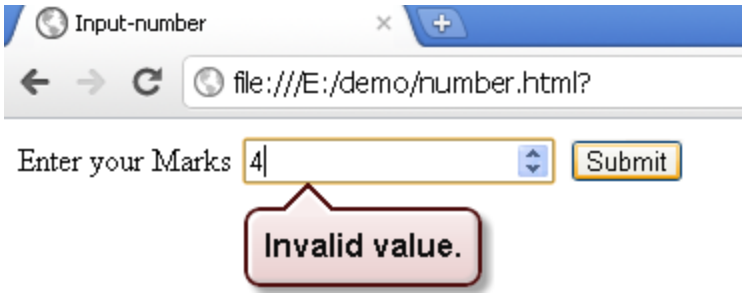
- Used to take a number as input






```
<input type="number" min="0 " max="100" step="3">
```

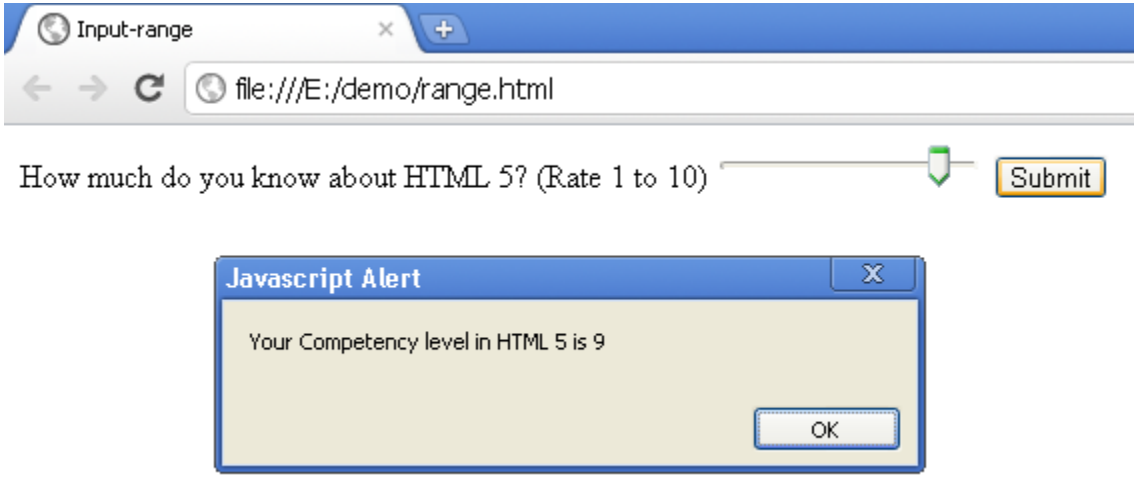
- step - specifies legal number intervals (if step="3", legal numbers could be -3,0,3,6, etc)

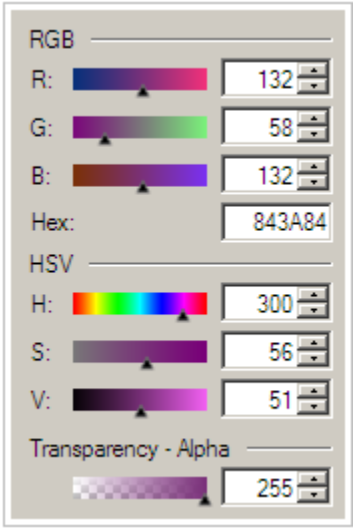


Input type - range

- Slider control is a very useful user interface to set a number within a range

```
<input type="range" name="Range" min="1" max="10" required>
```

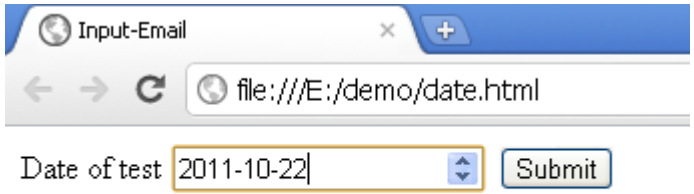


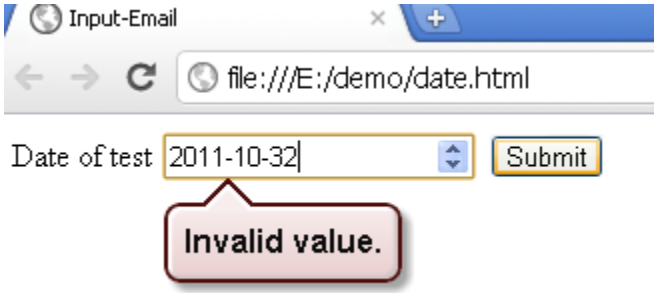


Input type – Date pickers - date

- Date and time field can be easily found in many web forms
- Typical applications are like ticket booking, appointment booking etc
- HTML5 the web browser ensures user can only enter a valid date time string into the input textbox

```
<input type="date" name="set_date" />
```

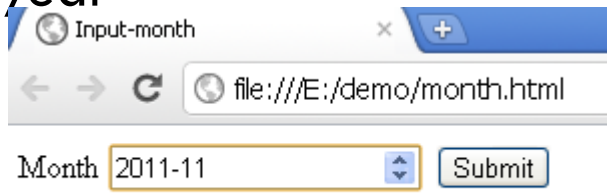


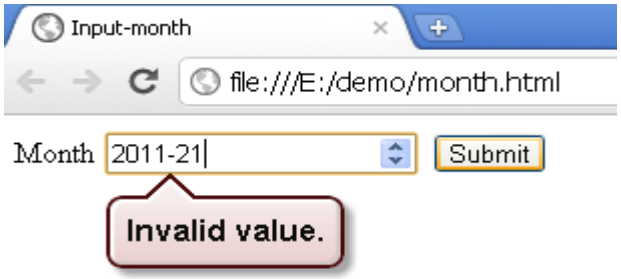


Input type – Date pickers – date (Contd.).

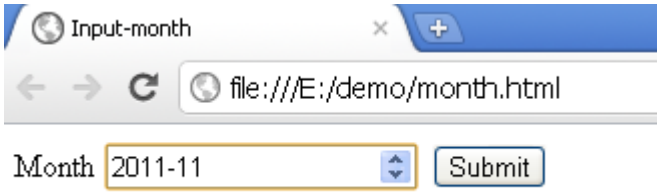
```
<input type="month" name="month" />
```

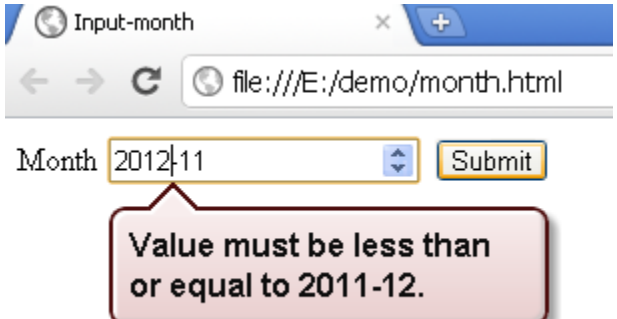
- Creates a date input control for specifying a particular month in a year





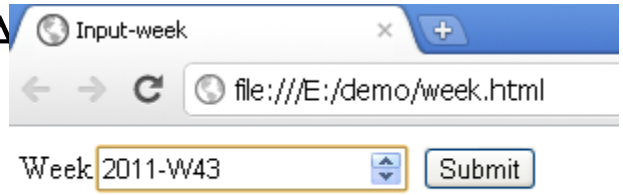

```
<input type="month" name="month" min="2011-1" max="2011-12"/>
```

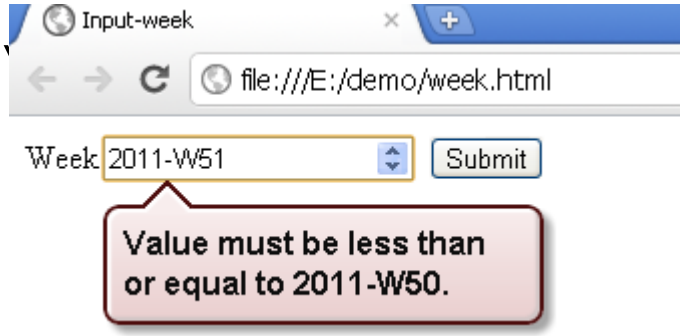




Input type – Date pickers – week

```
<input type="week" name="week" min="2011-W15"
  max="2011-W50"/>
```

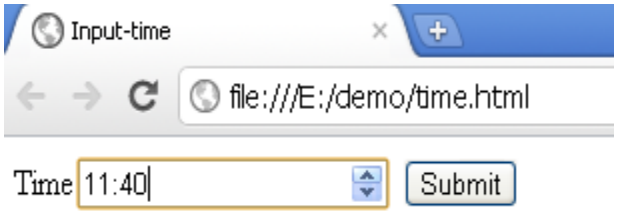
- A 

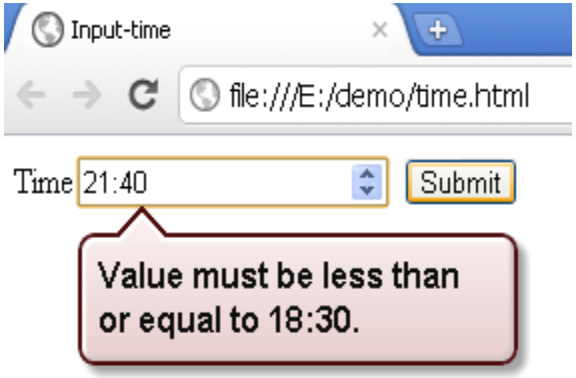
of a 

Input type – Date pickers – time

```
<input type="time" name="Time" min="11:40" max="18:30">
```

- Allows entry and validation of a valid time

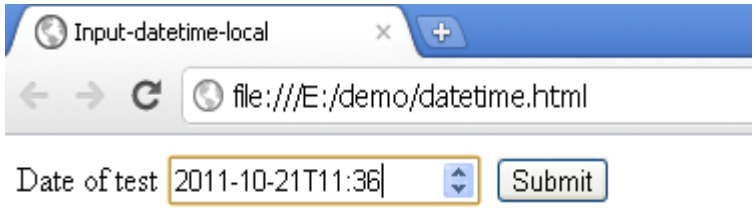




Input type – Date pickers – datetime-local

```
<input type="datetime-local" name="setdate" />
```

- It creates a combined date/time input field
- The value is an ISO formatted date and time



Input type - Search

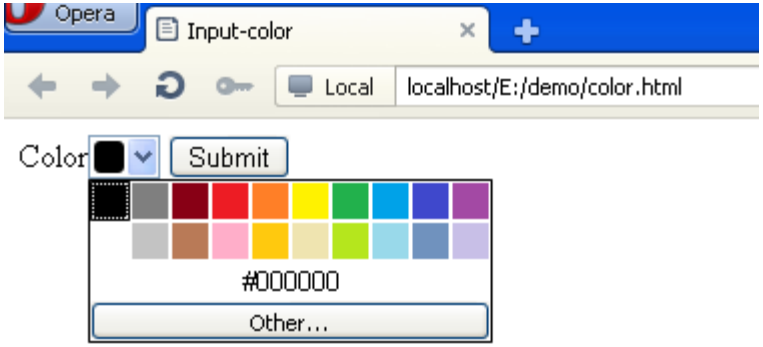
```
<input type="search" name="Search">
```

- It is used for search fields, like a site search, or Google search
- The difference between search and text type is only stylistic
- It takes the operating system's default rounded-corners style for search
- Currently Google Chrome does not support this

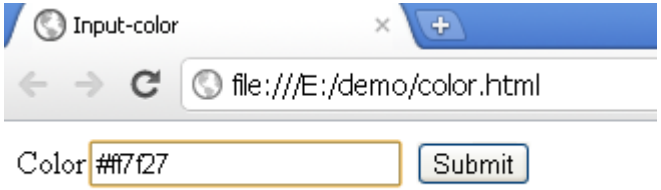
Input type - Color

- Color <input type="color" name="Color" required>
- Used to create a color control for selecting a color value
- The Opera browser will allow you to select a color from a color picker
- Color<input type="color" name="Color" required>

In Opera 11.52



In Google chrome 14.0



HTML 5 Form Elements

- New Form elements added to HTML 5 are
  - <datalist> - A list of options for input values
  - <output> - For different types of output, such as output written by a script

HTML Form Attributes

- New Form Attributes
  - autocomplete
  - novalidate
- New Input Attributes
  - autocomplete
  - autofocus
  - form
  - formoverrides
  - height and width
  - list
  - min,max and step
  - multiple
  - pattern (regexp)
  - placeholder
  - required

Autocomplete attributes

- The **autocomplete** attribute instructs the browser to remember field values
- It work with <form> and the following input types
  - text, search, url, telephone, email, password, date pickers, range and color
- It can have two values on and off
- When the user starts to type in an auto complete field the browser should display options to fill in the field

Enabling Auto-Completion for a Form Field

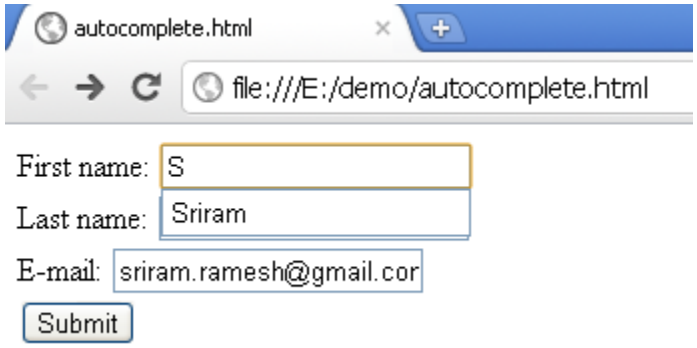
```
<input type="text" name="Username" autocomplete="on" />
```

Disabling Auto-Completion for a Sensitive Form Field

```
<input type="text" name="CreditCardNumber" autocomplete="off" />
```

Autocomplete attribute (Contd.).

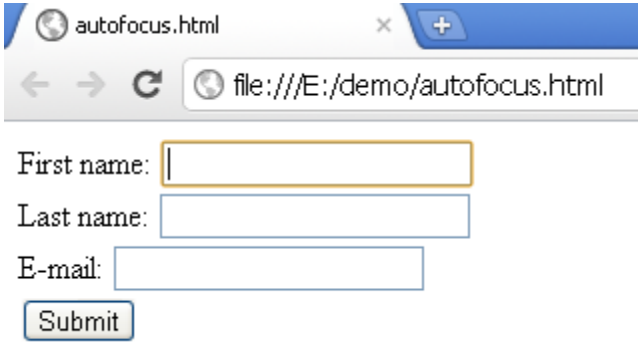
```
<form action="/webapp/demo1" method="get" autocomplete="on">
First name: <input type="text" name="fname" /><br />
Last name: <input type="text" name="lname" /><br />
E-mail: <input type="email" name="email" autocomplete="off" /><br />
<input type="submit" />
</form>
```



autofocus attribute

- It specifies that a field should automatically get focus when a page is loaded
- This attribute works with all <input> types

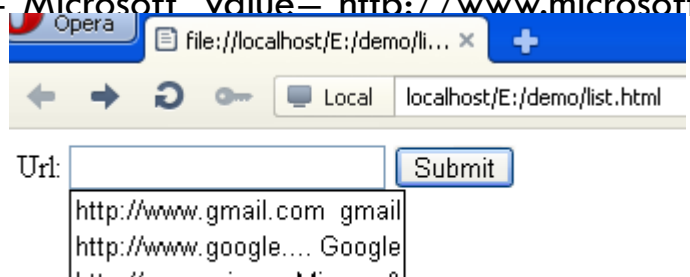
```
First name: <input type="text" name="fname" autofocus="autofocus" />
```



list attribute

- The list attribute specifies a datalist for an input field
- A datalist is a list of options for an input field
- The list attribute works with the following <input> types
  - text, search, url, telephone, email, date pickers, number, range, and color

```
Url: <input type="text" list="url_list" name="url" />
<datalist id="url_list">
    <option label="gmail" value="http://www.gmail.com" />
    <option label="Google" value="http://www.google.com" />
    <option label="Microsoft" value="http://www.microsoft.com" />
</datalist>
```

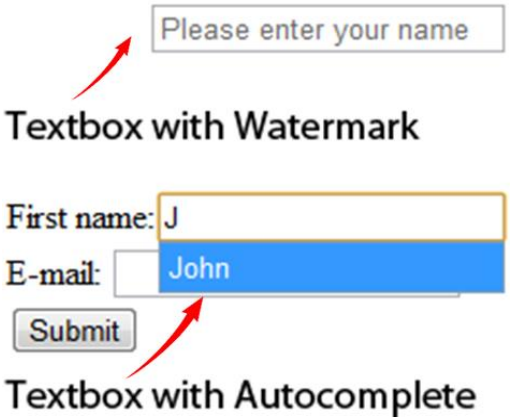


list Attribute and the datalist Element

- The **list** attribute provides another mechanism for auto-completion.
- Whereas the **autocomplete** attribute instructs the browser to suggest values based on historical entries that the browser itself manages, the **list** attribute gives the web developer more control over the suggestions.
- A list of pre-defined values can be specified in a **datalist** element somewhere on the page, and referenced by any text field.
- The browser then uses the pre-defined values as suggestions when the user fills the text field.
- To specify the suggestions made available by a **datalist** element, use **option** elements.
- The value of the **list** attribute is the ID of the **datalist** element. A **datalist** element contains **option** elements representing its values.

Usability Attributes

```
<input type="text" placeholder="Enter your full name" name="FullName" />
<input type="text" name="FullName" autofocus="autofocus" />
<input type="text" name="Username" autocomplete="on" />
```



form attribute

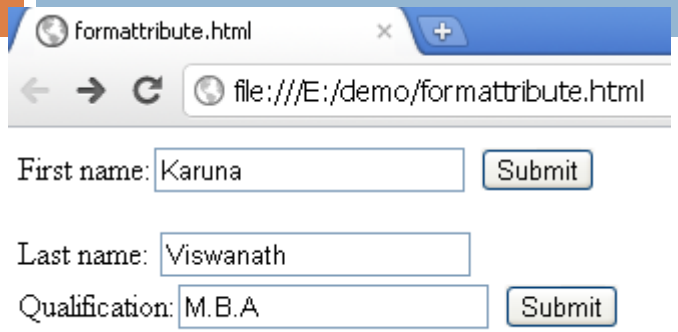
- In an HTML page there can be many number of forms but only one form can be submitted
- HTML 5 allows an element to use the form attribute and specify the form to which it belongs to

```
<form
  action="http://localhost:10000/webApp/default.aspx"
  method="get" id="user_form">
    First name:<input type="text" name="fname" />
    <input type="submit" />
</form>
Last name: <input type="text" name="lname"
  form="user_form" />
```

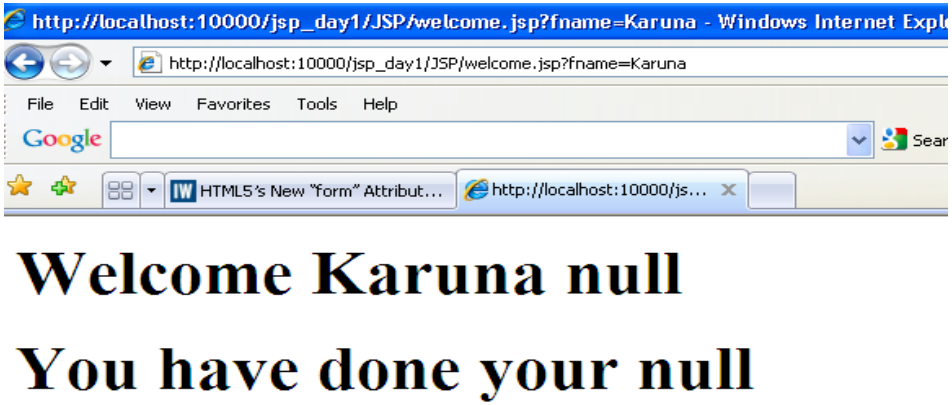
form attribute

```
<form
  action="http://localhost:10000/webApp/default1.aspx" method="get" id="user_form2">
  Qualification:<input  type="text"  name="qualification" form="user_form"/>
  <input type="submit" /></form>
```

form attribute (Contd.).



It works like this in an unsupported browser



Output in a browser with form attribute support

formoverrides attribute

- It allows you to override some of the attributes set for the form element
- The various form override attributes are:
  - formaction – It overrides the form action attribute
  - formmethod – It overrides the form method attribute
  - formnovalidate – It overrides the form novalidate attribute
  - formtarget – It overrides the form target attribute
- The form override attributes works with the following <input> types

  submit and image

formaction attribute

- It overrides the form action attribute

```
<form action="/WebApp/user.aspx" method="get">
Email: <input type="email" name="userid">
<input type="submit" value="user">
<input type="submit" formaction="/WebApp/admin.aspx"
   value="admin">
</form>
```

- If the user clicks the user button it submits the form to user.aspx
- If the user clicks the admin button it submits the forms to admin.aspx

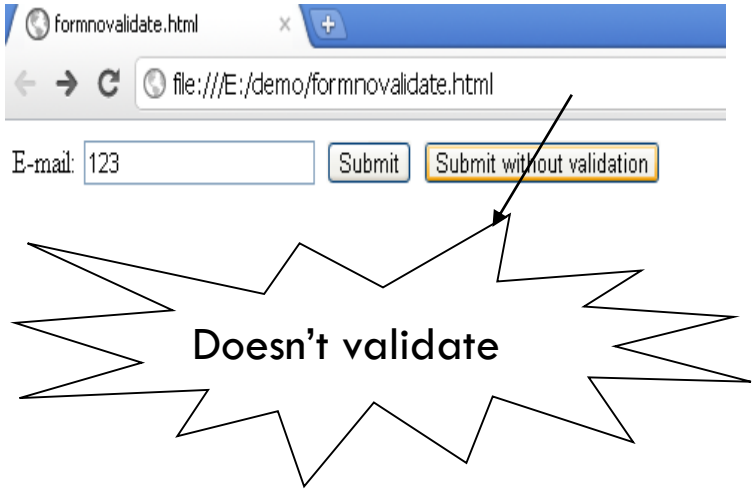
formmethod attribute

- Helps to override the form method attribute (get / post)

```
<form action="one.aspx" method="get">
FName: <input type="text" name="firstname" /><br />
LName: <input type="text" name="lastname" /><br />
<input type="submit" value="Submit" />
<input type="submit" formmethod="post"formaction="two.aspx"
  value="Submit using POST" />
</form>
```

formnovalidate attribute

```
<form action="demo.aspx" method="get">
E-mail: <input type="email" name="userid">
<input type="submit">
<input type="submit" formnovalidate="true"
  value="Submit without validation">
</form>
```



formtarget attribute

- Used to override the target attribute of the form element

```
<input formtarget="_blank|_self|_parent|_top|framename" />
```

_blank	Submits the form to a document in a new window or tab
_self	Submits the form to a document in the same frame (this is default)
_parent	Submits the form to a document in the parent frame
_top	Submits the form to a document in the full body of the window
framename	Submits the form to a document in a named iframe

formtarget attribute (Contd.).

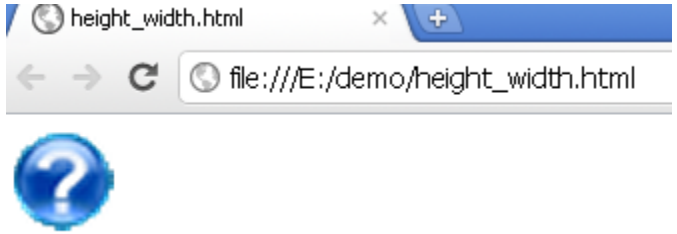
```
<form action="demo1.aspx" method="get">
  First name: <input type="text" name="firstname" />
<br />
  Last name: <input type="text" name="lastname" />
<br />
<input type="submit" value="Submit" />
<input type="submit" formtarget="_blank"
value="Submit to a new window/tab" />
</form>
```

height and width attribute

- height - specifies the height of the image
- width - specifies the width of the image

```
<input type="image" src="q-mark.gif" />
```

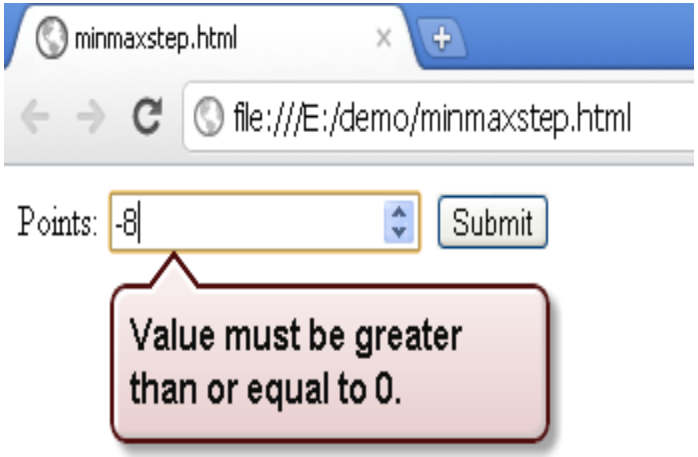
```
<input type="image" src="q-mark.gif" height="50" width="50" />
```



min, max and step attribute

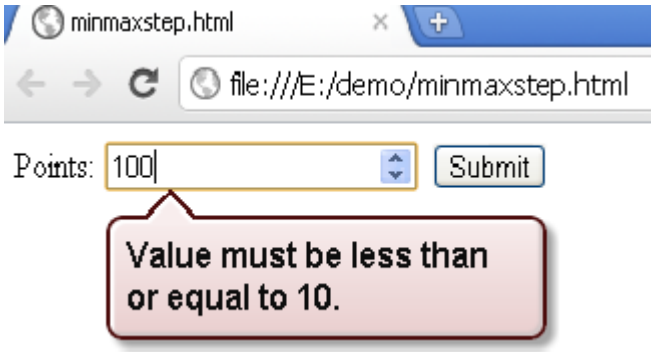
- max – used to specify the maximum value allowed for the input field
- min - used to specify the minimum value allowed for the input field
- step - specifies the legal number intervals for the input field (if step="4", legal numbers could be -4,0,4,8, etc)
- The min, max, and step attributes works with the following <input> types
  - date pickers, number, and range

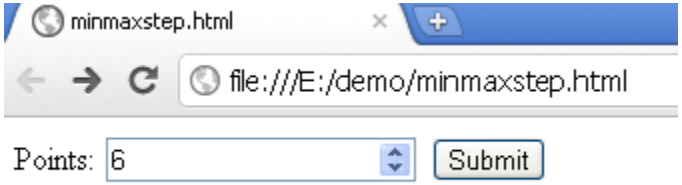
```
Points: <input type="number" name="pts" min="0" max="10" step="3"/>
```

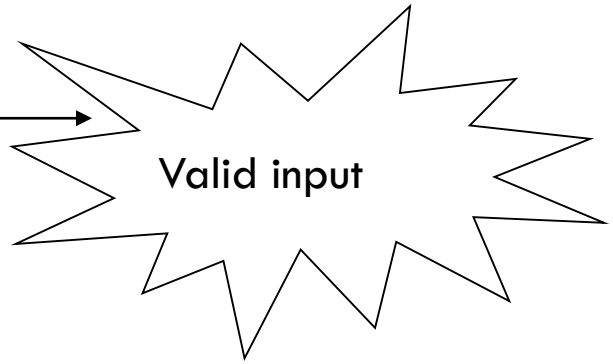


min,max and step attribute (Contd.).

```
Points: <input type="number" name="pts" min="0" max="10" step="3"/>
```





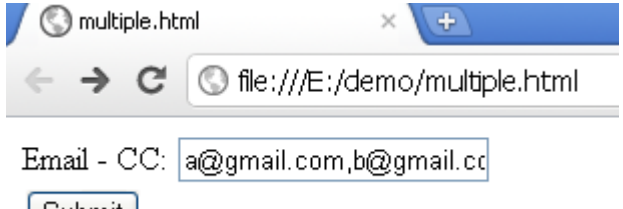


multiple attribute

- The multiple attribute is used to indicate if the user can be allowed to specify more than one value

```
Email - To: <input type="email" name="to" />
```

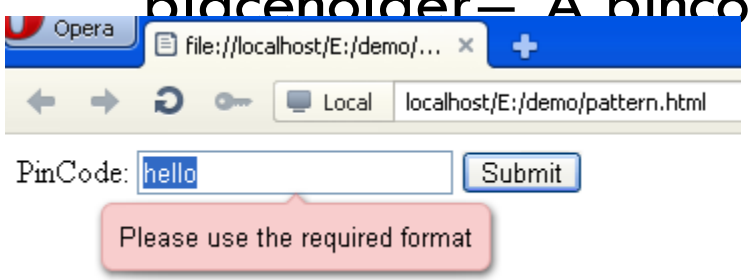
```
Email - CC: <input type="email" name="cc" multiple />
```



pattern attribute

- The pattern attribute is used to specify a pattern to validate an input field
- The pattern is a regular expression
- The pattern attribute works with the following <input> types
  - text, search, url, telephone, email, and password

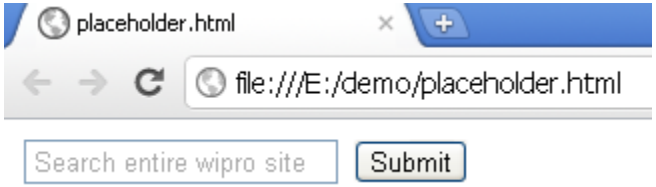
```
<input type="text" pattern="[0-9]{6}" name="pincode"
placeholder="A pincode is a 6 digit number"/>
```



placeholder attribute

- It provides description about the expected value of an input field
- It works with the following <input> types
  - text, search, url, telephone, email, and password
- The hint is displayed in the input field when it is empty, and disappears when the field gets focus

```
<input type="search" name="usr_search" placeholder="Search entire wipro site" />
```



required attribute

- The required attribute specifies that an input field must be filled out before submitting
- It works with the following <input> types
  - text, search, url, telephone, email, password, date pickers, number, checkbox, radio, and file

```
Name: <input type="text" name="user_name" required="required" />
```

Cross-Browser Interoperability

- One of the main problems with earlier versions of HTML was cross-browser interoperability
- Developers strive for code that will work in all browsers
  - This is impossible because of legacy browsers and the difference between browser engines

Browser Detection

- A technique to check which browser is running the code
- Can make dozens of assumptions based on this single check
- Should be avoided most of the time
- Should only be used when applying code to a specific legacy browser of a known version

```
<!--[if IE 7]><div>rendered in Internet Explorer 7 only</div><![endif]-->
```

Feature Detection

- A technique to check the availability of a feature before it is used
- Standard features should be looked for first
  - Browsers sometimes support both the standards and a legacy implementation
  - Looking for standart features first will ensure the use of standards if available
- Avoid making any assumptions about existing features

```
if (window.addEventListener) {
    // HTML that will be rendered if addEventListener is available
}
else {
    // HTML that will be rendered if addEventListener is not available
}
```

Modernizr

- A small JavaScript library
- Detects availability of HTML5 and CSS3 specifications and whether features are natively implemented in the browser
- Uses feature detection under the hood

```
if (Modernizr.canvas) {
    // HTML that will be rendered if the Canvas element is available
}
```

Summary

- Describe the new HTML5 input types.
- Apply date pickers on webpages.
- Use the new text box types on webpages.
- Use the new interactive text box types on webpages.

Before Web Workers

Web Workers

- When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.
- A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page.
- You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Web Workers for which scenarios?

- **image processing** by using the data extracted from the <canvas> or the <video> elements. You can divide the image into several zones and push them to the different Workers that will work in parallel. You'll then benefit from the new generation of multi-cores CPUs. The more you have, the faster you'll go.
- **big amount of data** retrieved that you need to parse after an XMLHTTPRequest call. If the time needed to process this data is important, you'd better do it in background inside a Web Worker to avoid freezing the UI Thread. You'll then keep a reactive application.

Web Workers for which scenarios?

- **background text analysis**: as we have potentially more CPU time available when using the Web Workers, we can now think about new scenarios in JavaScript. For instance, we could imagine parsing in real-time what the user is currently typing without impacting the UI experience. Think about an application like Word (of our Office Web Apps suite) leveraging such possibility: background search in dictionaries to help the user while typing, automatic correction, etc.
- **concurrent requests against a local database**. IndexDB will allow what the Local Storage can't offer us: a thread-safe storage environment for our Web Workers.

Types of Web Workers

- Dedicated Workers
- Shared Workers

Types of Web Workers

- Dedicated Workers
- A dedicated worker is accessible from the parent script that created it
- Wide browser support: All
- It is simply tied to its main connection

Types of Web Workers

- Shared Workers
- A shared worker can be accessed from any script of the same origin.
- Limited browser support: Chrome 4.0+, Safari 5.0+ and Opera 10.6+.
- It can work with multiple connections.

Web Worker Browser Support

- Browser Version
- IE 10.0
- Chrome 12+
- Firefox 5+
- Safari 4+
- Opera 11+
- iOS Safari 5+
- Opera Mobile 11+
- Android 2.1
- Chrome for Android Beta

Check Web Worker Support

```
if(typeof(Worker) !== "undefined") {
    // Yes! Web worker support!
    // Some code.....
} else {
    // Sorry! No Web Worker support..
}
```

Create a Web Worker File

```
var i = 0;

function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}

timedCount();
```

Create a Web Worker Object

- ```
  if(typeof(w) == "undefined") {
      w = new Worker("demo_workers.js");
  }
  ```
- ```
  w.onmessage = function(event){
      document.getElementById("result").innerHTML =
  event.data;
  };
  ```

Terminate a Web Worker

- w.terminate();

Reuse the Web Worker

- w = undefined;

Data Parsing in Web Worker

```
var worker = new Worker("worker-parser.js");
//when the data is fetched (e.g. in our xhr) -> this event handler
//is called to action
worker.onmessage = function(event){
//let's get our JSON structure back
var jsonObj = event.data;
//work with the JSON object
showData(jsonObj);
};
//send the 'huge' JSON string to parse
worker.postMessage(jsonText);
```

HTML5 Custom Data Attributes (data-*)

- The data-* attributes is used to store custom data private to the page or application.
- The data-* attributes gives us the ability to embed custom data attributes on all HTML elements.
- The stored (custom) data can then be used in the page's JavaScript to create a more engaging user experience (without any Ajax calls or server-side database queries).
- The data-* attributes consist of two parts:
- The attribute name should not contain any uppercase letters, and must be at least one character long after the prefix "data-"
- The attribute value can be any string.

Cross-Document Messaging

- In order to send a cross-document message, the first effort you would like to put is to create a new browsing context.
- You may accomplish it either by referencing an iframe or by generating a new window.
- Now, you can successfully send a message via it employing the postMessage() method, which requires two arguments for cross-document messaging.

Cross-Document Messaging - Sending

- var iframe = document.querySelector('iframe');
- var button = document.querySelector('button');
- var clickHandler = function(){
- // iframe.contentWindow refers to the iframe's window object.
- iframe.contentWindow.postMessage('The message to send.','http://dev.opera.com');
- }
- button.addEventListener('click',clickHandler,false);

Cross-Document Messaging -Receiving

- var messageEventHandler = function(event){// check that the origin is one we want.if(event.origin == 'http://dev.opera.com'){alert(event.data);}}window.addEventListener('message', messageEventHandler,false);

HTML5 Server-Sent Events

- A server-sent event is when a web page automatically gets updates from a server.
- This was also possible before, but the web page would have to ask if any updates were available.
- With server-sent events, the updates come automatically.
- Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Possible Applications

- A real-time chart of streaming stock prices
- Real-time news coverage of an important event (posting links, tweets, and images)
- A live Twitter wall fed by Twitter's streaming API
- A monitor for server statistics like uptime, health, and running processes

Overview of the API

- new EventSource(url) — this creates our EventSource object, which immediately starts listening for events on the given URL.
- •readyState — as with XHR, we have a readyState for the EventSource that tells us if we're connecting (0), open (1), or closed (2).
- •onopen, onmessage — two events that we can listen for on the new EventSource object. By default, the message event will fire when new messages are received, unless the server explicitly sets the event type.
- •addEventListener — not only can we listen for the default message event, but we can also listen for custom messages using the addEventListener on the EventSource object, just as if we were listening for a click event.
- •event.data — as with most messaging APIs, the contents of the message reside in the data property of the event object. This is a string, so if we want to pass around an object, we need to encode and decode it with JSON.
- •close — closes the connection from the client side.

WEB SOCKET API:

- Web Sockets is a next-generation bidirectional communication technology for web applications which operates over a single socket and is exposed via a JavaScript interface in HTML 5 compliant browsers.
- Once you get a Web Socket connection with the web server, you can send data from browser to server by calling a **send()** method, and receive data from server to browser by an **onmessage** event handler.

WebSocket Attributes:

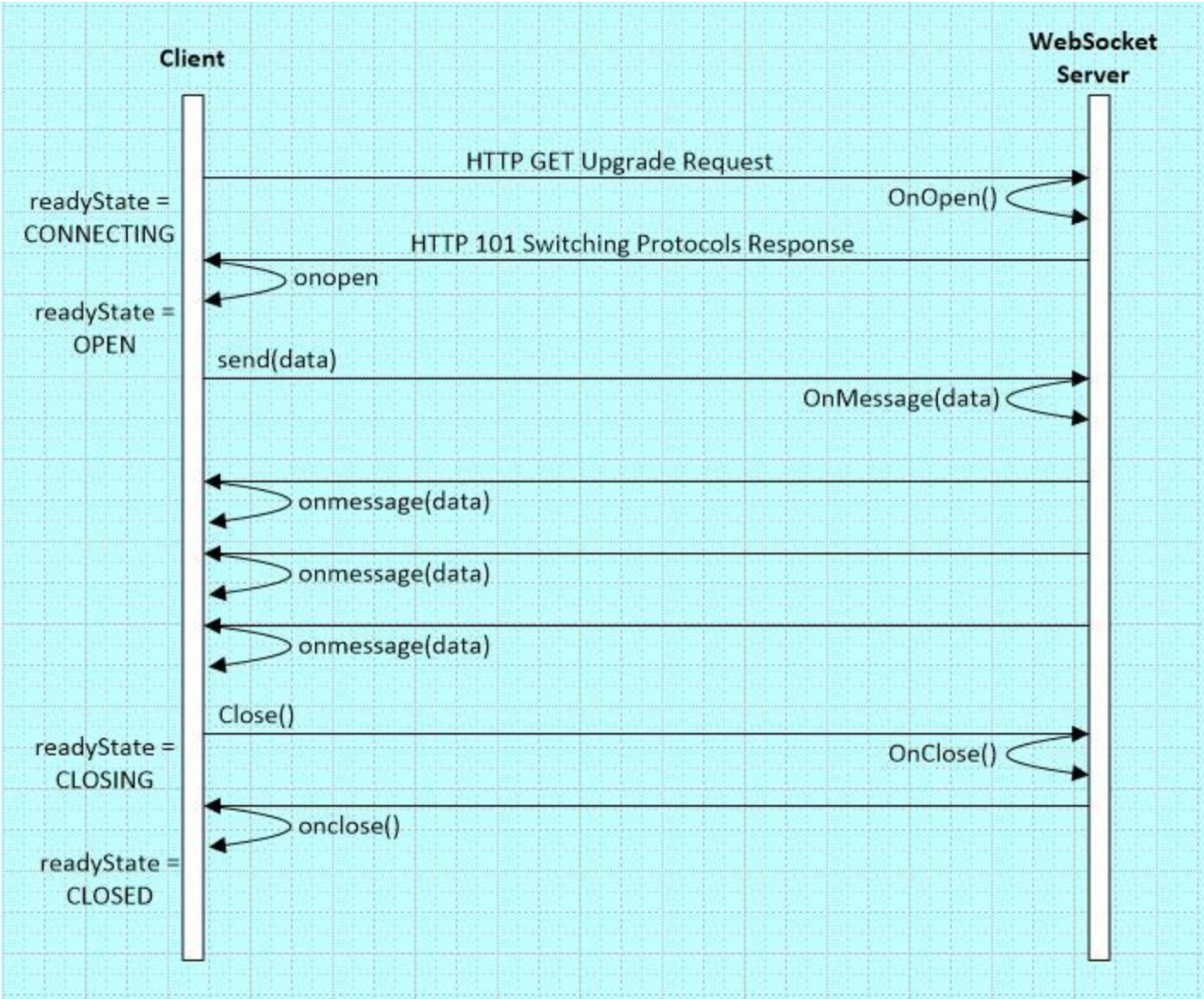
Attribute	Description
Socket.readyState	The readonly attribute readyState represents the state of the connection. It can have the following values: 1.A value of 0 indicates that the connection has not yet been established. 2.A value of 1 indicates that the connection is established and communication is possible. 3.A value of 2 indicates that the connection is going through the closing handshake. 4.A value of 3 indicates that the connection has been closed or could not be opened.
Socket.bufferedAmount	The readonly attribute bufferedAmount represents the number of bytes of UTF-8 text that have been queued using send() method.

WebSocket Events:

Event	Event Handler	Description
open	Socket.onopen	This event occurs when socket connection is established.
message	Socket.onmessage	This event occurs when client receives data from server.
error	Socket.onerror	This event occurs when there is any error in communication.
close	Socket.onclose	This event occurs when connection is closed.

WebSocket Methods:

Method	Description
Socket.send()	The send(data) method transmits data using the connection.
Socket.close()	The close() method would be used to terminate any existing connection.


Client
WebSocket Server
HTTP GET Upgrade Request
readyState = CONNECTING
OnOpen()
HTTP 101 Switching Protocols Response
onopen
readyState = OPEN
send(data)
OnMessage(data)
onmessage(data)
onmessage(data)
onmessage(data)
Close()
readyState = CLOSING
OnClose()
onclose()
readyState = CLOSED

Web RTC API:

- Imagine a world where your phone, TV and computer could all communicate on a common platform.
- Imagine it was easy to add video chat and peer-to-peer data sharing to your web application.
- That's the vision of WebRTC.
- WebRTC is available now in Google Chrome, Opera and Firefox.

Web RTC API:

- A good place to start is the simple video chat application at apprtc.appspot.com:
- Open apprtc.appspot.com in Chrome, Opera or Firefox.
- Click the Allow button to let the app use your web cam.
- Open the URL displayed at the bottom of the page in a new tab or, better still, on a different computer.

WEB STORAGE & File System API:

- With local storage, web applications can store data locally within the user's browser.
- Before HTML5, application data had to be stored in cookies, included in every server request.
- Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- Local storage is per domain. All pages, from one domain, can store and access the same data.

WEB STORAGE & File System API:

- **The sessionStorage Object**
- The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session.
- The data is deleted when the user closes the browser window.

WEB STORAGE & File System API:

- With the FileSystem API, a web app can create, read, navigate, and write to a sandboxed section of the user's local file system.
- The API is broken up into various themes:
- Reading and manipulating files: File/Blob, FileList, FileReader
- Creating and writing: Blob(), FileWriter
- Directories and file system access: DirectoryReader, FileEntry/DirectoryEntry, LocalFileSystem

IndexedDBAPI

- The IndexedDB, a new HTML5 web database which allows HTML5 web application to store data inside a user's browser.
- This makes the application to load faster and more responsive than ever.
- IndexedDB creates an object store for a type of data and simply persist JavaScript objects to that store.

IndexedDB Interfaces

- Connecting to a database
- IDBEnvironment Provides access to IndexedDB functionality. It is implemented by the window and worker objects.
- IDBFactoryProvides access to a database. This is the interface implemented by the global object indexedDB and is therefore the entry point for the API.
- IDBOpenDBRequestRepresents a request to open a database.
- IDBDatabaseRepresents a connection to a database. It's the only way to get a transaction on the database.
- IDBRequestGeneric interface that handles database requests and provides access to results.

Retrieving and modifying data

- IDBTransactionRepresents a transaction.
- You create a transaction on a database, specify the scope (such as which object stores you want to access), and determine the kind of access (read only or readwrite) that you want.
- IDBObjectStoreRepresents an object store that allows access to a set of data in an IndexedDB database, looked up via primary key.
- IDBIndex Also allows access to a subset of data in an IndexedDB database, but uses an index to retrieve the record(s) rather than the primary key.
- This is sometimes faster than using IDBObjectStore.
- IDBCursorIterates over object stores and indexes.
- IDBCursorWithValueIterates over object stores and indexes and returns the cursor's current value.
- IDBKeyRangeDefines a range of keys that can be used to retrieve data from a database in a certain range.

CSS FUNDAMENTALS FOR ADVANCED USE

- A CSS (cascading style sheet) file allows you to separate your web sites (X)HTML content from it's style.
- As always you use your (X)HTML file to arrange the content, but all of the presentation (fonts, colors, background, borders, text formatting, link effects & so on...) are accomplished within a CSS.

HTML5-Data-*Attributes

```
<ul>
  <li data-animal-type="bird">Owl</li>
  <li data-animal-type="fish">Salmon</li>
  <li data-animal-type="spider">Tarantula</li>
</ul>
```

Application Cache <html manifest="demo.manifest">

- AppCache is intended for proper web applications whereas the browser's cache is for normal web pages in general.
- The normal cache will cache pretty much any page whereas the AppCache will only cache pages which are specifically included in the manifest file.
- Plus, the normal cache is unreliable, as we dont know which pages (and which resources within those pages) will be available for sure.

The manifest file

- CACHE MANIFEST
- #You can also use the CACHE: section header to explicitly declare the following three files.

- style.css
- script.js
- index.htm

Network

- CACHE MANIFEST

- CACHE:
- style.css
- script.js
- index.htm

- NETWORK:
- style2.css

Fallback

- CACHE MANIFEST

- CACHE:
- style.css
- script.js
- index.htm

- NETWORK:
- style2.css

- FALLBACK:
- main_image.jpg backup_image.jpg

Status

- 0 - uncached
- If the page is not linked to the application cache. Also, the very first time the application cache is being downloaded, then during the time it is being downloaded, the AppCache will have a status of uncached.
- 1 - idle
- When the browser has the latest version of the AppCache, and there aren no updated versions to download, then the status is set to Idle.
- 2 - checking
- The duration of when the page is checking for an updated manifest file, then the status is set to Checking.
- 3 - downloading
- The duration of when the page is actually downloading a new cache (if an updated manifest file has been detected), the status is set to downloading

Status window.applicationCache.status

- 4 - updateready
- Once the browser finishes downloading that new cache, it is ready to be used (but still not being used yet). During this time, the status is set as updateready
- 5 - obsolete
- In case the manifest file cannot be found, then the status is set to obsolete and the application cache gets deleted. It is important to know, that in case the manifest file (or any file mentioned in the manifest file except those which have a fallback) fail to load, then it will be counted as an error and the old application cache will continue to be used.

CROSS DOCUMENT MESSAGING API

Internal Stylesheet

```
<head>
<title><title>
<style type="text/css">
CSS Content Goes Here
</style>
</head>
<body>
```

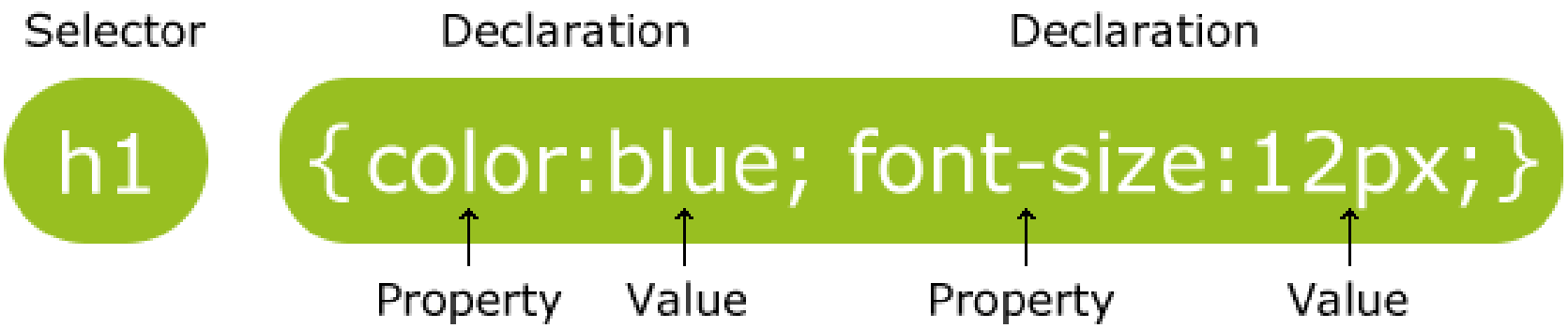
External style sheets

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

Here are a few reasons this is better.

- Easier Maintenance
- Reduced File Size
- Reduced Bandwidth
- Improved Flexibility

CSS Syntax

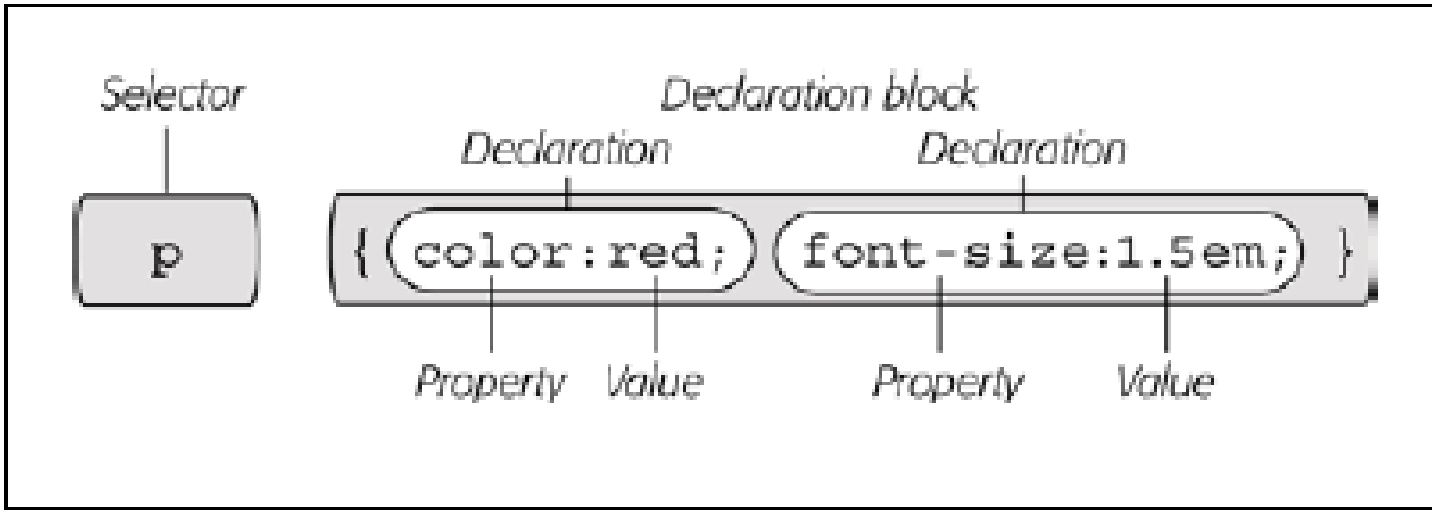

Selector
Declaration
Declaration
h1
{color:blue; font-size:12px;}
Property
Value
Property
Value

CSS Syntax

- selector { property: value }

```
body {
background: #eeeeee;
font-family: "Trebuchet MS", Verdana, Arial, serif;
}

p
    {
    color:red;
    /*This is another comment*/
    text-align:center;
    }
```

Understanding Style Sheets



CSS Selectors

- Element Selector
- p
  {
  text-align:center;
  color:red;
  }
- Id Selector
- #para1
  {
  text-align:center;
  color:red;
  }

CSS Selectors

- class Selector
- .center
  {
  text-align:center;
  color:red;
  }
- p.center
  {
  text-align:center;
  color:red;
  }

CSS Background

- CSS properties used for background effects:
- background-color
- background-image
- background-repeat
- background-attachment
- background-position

CSS Background

- body {background-color:#b0c4de;}
- h1 {background-color:#6495ed;}
  p {background-color:#e0ffff;}
  div {background-color:#b0c4de;}
- body {background-image:url("paper.gif");}
- body
  {
  background-image:url("gradient2.png");
  background-repeat:repeat-x;
  }

CSS Text

- h1 {text-align:center;}
  p.date {text-align:right;}
  p.main {text-align:justify;}
- h1 {text-decoration:overline;}
  h2 {text-decoration:line-through;}
  h3 {text-decoration:underline;}
- p.uppercase {text-transform:uppercase;}
  p.lowercase {text-transform:lowercase;}
  p.capitalize {text-transform:capitalize;}
- p {text-indent:50px;}

CSS Font

- p{font-family:"Times New Roman", Times, serif;}
- p.normal {font-style:normal;}
  p.italic {font-style:italic;}
  p.oblique {font-style:oblique;}
- h1 {font-size:40px;}
  h2 {font-size:30px;}
  p {font-size:14px;}
- h1 {font-size:2.5em;} /* 40px/16=2.5em */
  h2 {font-size:1.875em;} /* 30px/16=1.875em */
  p {font-size:0.875em;} /* 14px/16=0.875em */

CSS Links

- The four links states are:
- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

CSS Links

- a:link {color:#FF0000;} /* unvisited link */
  a:visited {color:#00FF00;} /* visited link */
  a:hover {color:#FF00FF;} /* mouse over link */
  a:active {color:#0000FF;} /* selected link */

CSS Float

- img
  {
  float:right;
  }
- .thumbnail
  {
  float:left;
  width:110px;
  height:90px;
  margin:5px;
  }

CSS Box Model

CSS Box Model

- #buttonfld
- {
- background-color:green;
- width:600px;
- padding:10px;
- border:5px solid gray;
- margin:0px;

- }

Creating a Transparent Image

- ```
  img
  {
  opacity:0.4;
  filter:alpha(opacity=40); /* For IE8 and earlier */
  }
  ```
- ```
  img:hover
  {
  opacity:1.0;
  filter:alpha(opacity=100); /* For IE8 and earlier */
  }
  ```

Creating a Transparent Image

```
div.background
  {
  width:500px;
  height:250px;
  background:url(klematis.jpg) repeat;
  border:2px solid black;
  }
div.transbox
  {
  width:400px;
  height:180px;
  margin:30px 50px;
  background-color:#ffffff;
  border:1px solid black;
  opacity:0.6;
  filter:alpha(opacity=60); /* For IE8 and earlier */
  }
div.transbox p
  {
  margin:30px 40px;
  font-weight:bold;
  color:#000000;
  }
```

Combining Selectors

```
h1, h2, h3, h4, h5, h6 {
color: #009900;
font-family: Georgia, sans-serif;
}
```

Inheritance

- body {font-family: Verdana, serif;}
- h1 {font-family: Georgia, sans-serif;}
- p {font-family: Tahoma, serif;}

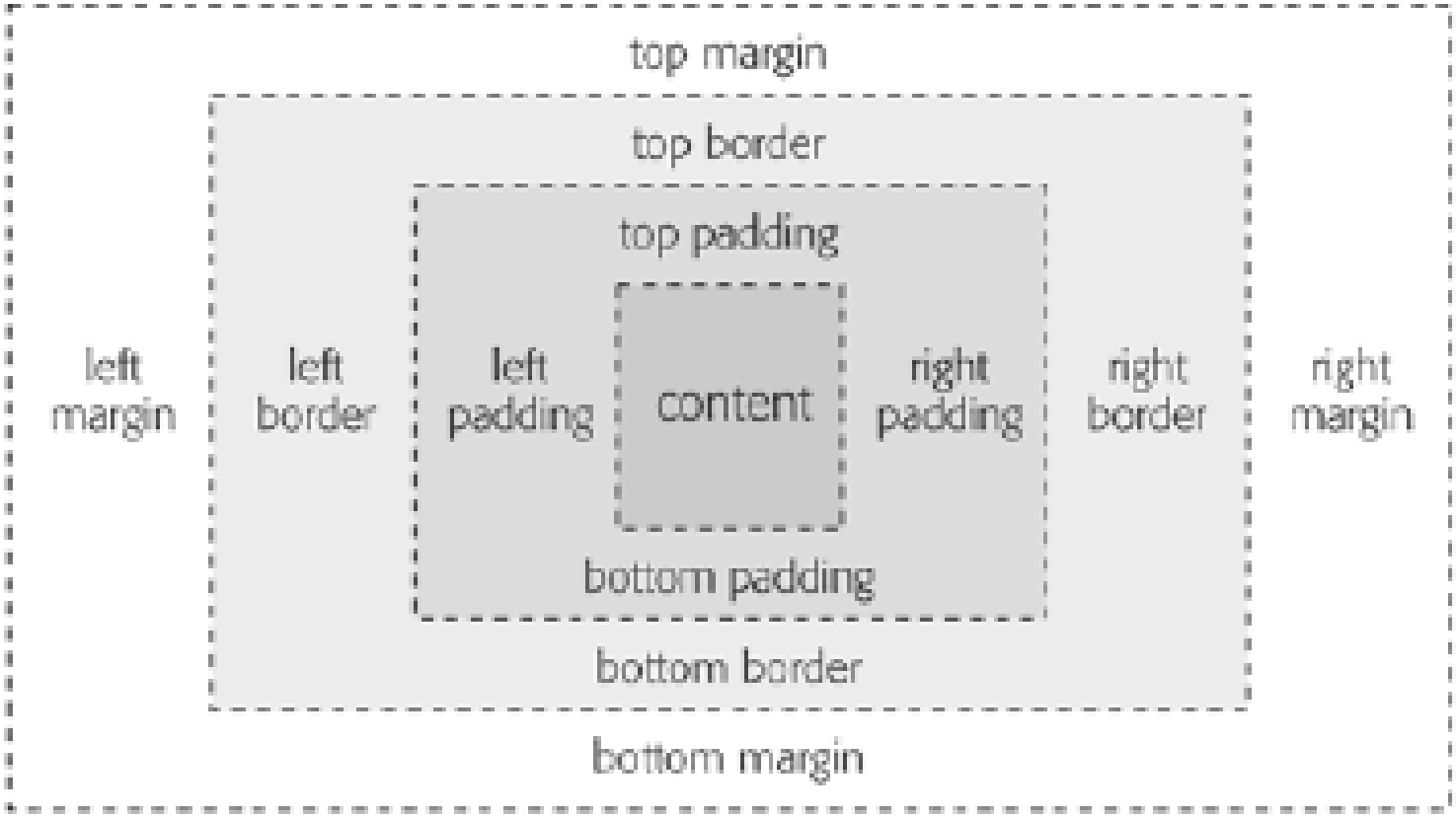
CSS Classes

```
<p>
To put it more simply, this
<span class="greenboldtext">sentence</span>
you are reading is styled in my CSS file by the following.
</p>
.greenboldtext{
font-size: small;
color: #008080;
font-weight: bold;
}
```

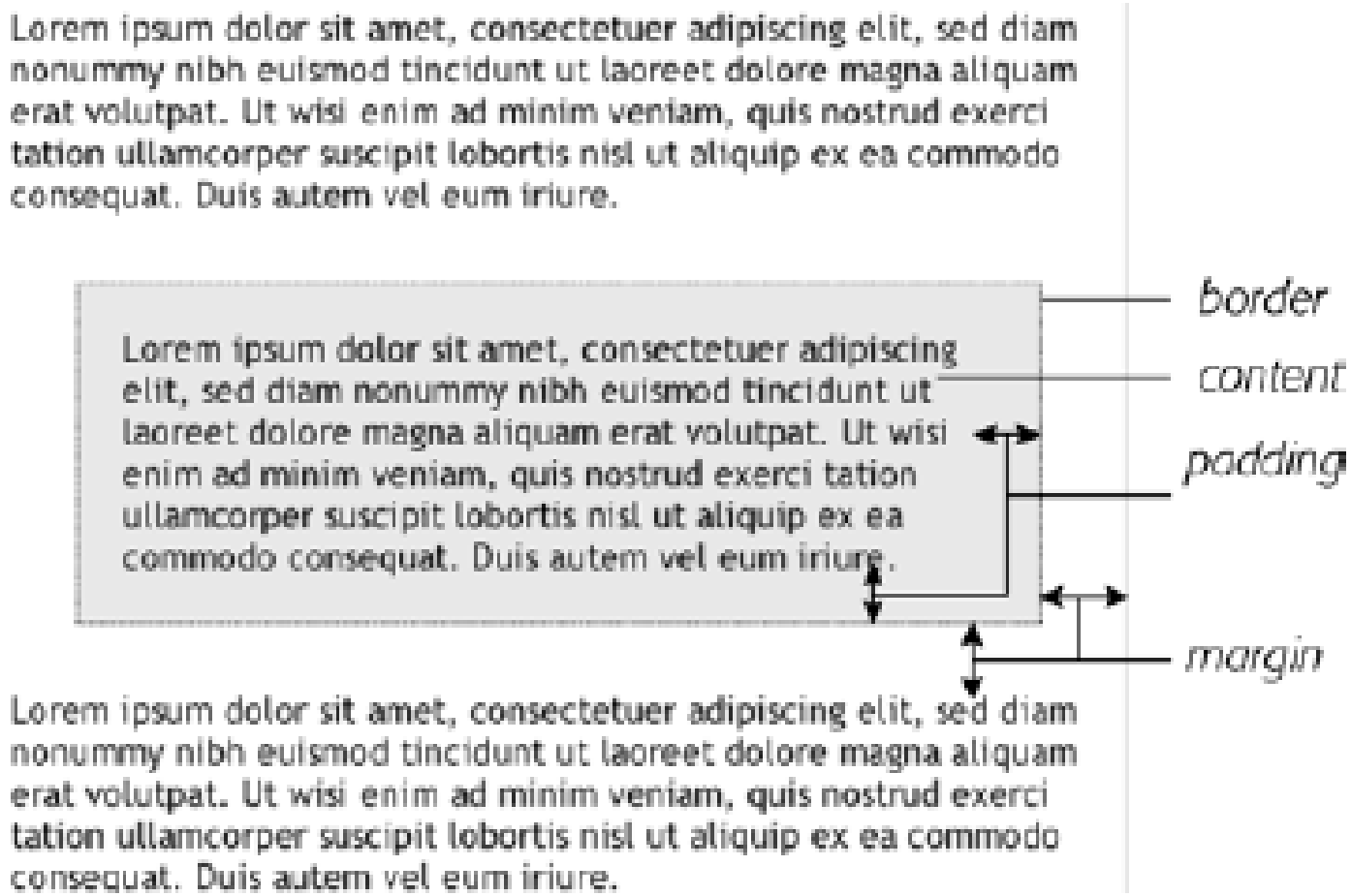
CSS IDs

```
<div id="container">
Everything within my document is inside this division.
</div>
#container{
width: 80%;
margin: auto;
padding: 20px;
border: 1px solid #666;
background: #ffffff;
}
```

CSS Margins


top margin
top border
top padding
left margin
left border
left padding
content
right padding
right border
right margin
bottom padding
bottom border
bottom margin

CSS Margins


Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.
border
content
padding
margin
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.

CSS Margins

- margin-top: length percentage or auto;
- margin-left: length percentage or auto;
- margin-right: length percentage or auto;
- margin-bottom: length percentage or auto;

- margin: 10px 10px 10px 10px;
- If you declare all 4 values as I have above, the order is as follows:
- 1. top
- 2. right
- 3. bottom
- 4. left

CSS Margins

- If only one value is declared, it sets the margin on all sides.
  - margin: 10px;
- If you only declare two or three values, the undeclared values are taken from the opposing side.
    - margin: 10px 10px; /* 2 values */
    - margin: 10px 10px 10px; /* 3 values */
- If you do not declare the margin value of an element, the margin is 0 (zero).

CSS Margins

- You can see in the example below, the elements for this site are set to be 20px (pixels) from the body.

```
body{
margin: 20px;
background: #eeeeee;
font-size: small;
font-family: Tahoma, Arial, "Trebuchet MS", Helvetica, sansserif;
text-align: left;
}
```

CSS Padding

- Padding is the distance between the border of an (X)HTML element and the content within it.
  - padding-top: length percentage;
  - padding-left: length percentage;
  - padding-right: length percentage;
  - padding-bottom: length percentage;

CSS Text Properties

- letter-spacing: value;
- text-align: value;
- color: value;
- text-decoration: value;
  - Possible values are
    - none
    - underline
    - overline
    - line through
    - blink

CSS Text Properties

- text-transform: value;
- Possible values are
  - none
  - capitalize
  - lowercase
  - Uppercase
- word-spacing: value;

CSS Font Properties

- The font property can set the style, weight, variant, size, line height and font:
- font: italic bold normal small/1.4em Verdana, sans-serif;
- **Font-Family**
- You can set what font will be displayed in an element with the font-family property.
- There are 2 choices for values:
  - family-name
  - generic family
- If you set a family name it is best to also add the generic family at the end. As this is a priortized list. So if the user does not have the specified font name it will use the same generic family. (see below)
  - font-family: Verdana, sans-serif;

Font Size

- You can set the size of the text used in an element by using the font-size property.
  - font-size: value;
- There are a lot of choices for values:
  - xx-large
  - x-large
  - larger
  - large
  - medium
  - small
  - smaller
  - x-small
  - xx-small
  - length
  - % (percent)

Font Style

- You can set the style of text in a element with the font-style property
- font-style: value;
- Possible values are
  - normal
  - itailc
  - oblique

CSS Anchors, Links and Pseudo Classes

- a:link {color: #009900;}
  a:visited {color: #999999;}
  a:hover {color: #333333;}
  a:focus {color: #333333;}
  a:active {color: #009900;}

CSS Backgrounds

- **Background**
- You can style the background of an element in one declaration with the background property.
  - background: #ffffff url(path_to_image) top left no-repeat fixed;
  - Values:
    - attachment
    - color
    - image
    - position
    - repeat

Background Attachment

- If you are using an image as a background. You can set whether the background scrolls with the page or is fixed when the user scrolls down the page with the background-attachment property
- background-attachment: value;
- Values:
  - fixed
  - scroll

Background Color

- You can specifically declare a color for the background of an element using the background-color property.
- background-color: value;
- Values:
  - color name
  - hexadecimal number
  - RGB color code
  - transparent

Background Image

- You can set an image for the background of an element using the background-image property.
- background-image: url(path_to_image);
- Values:
  - <u>url</u>
  - <u>none</u>

Background Position

- You can position an image used for the background of an element using the background-position property.
- background-position: value;
  - Values:
  - top left
  - top center
  - top right
  - center left
  - center center
  - center right
  - bottom left
  - bottom center
  - bottom right
  - x-% y-%
  - x-pos y-pos

Background Repeat

- You can set if an image set as a background of an element is to repeat (across=x and/or down=y) the screen using the background-repeat property.
- background-repeat: value;
- Values:
  - no-repeat
  - repeat
  - repeat-x
  - repeat-y

CSS Borders

- You can set the color, style and width of the borders around an element in one declaration by using the border property.
- border: 1px solid #333333;
- Values:
  - color
  - style
  - width

Border Color

- You can set the color of a border independently with the border-color property.
- border-color: value;
- Values:
  - color name
  - hexadecimal number
  - RGB color code
  - transparent

Border Style

- You can set the style of a border independently with the border-style property.
- border-style: value;
- Values:
  - dashed
  - dotted
  - double
  - groove
  - hidden
  - inset
  - none
  - outset
  - ridge
  - solid

CSS Ordered & Unordered Lists

- **List Style**
- You can control the appearance of ordered and unordered lists in one declaration with the list-style property
- list-style: value value;
- Values:
  - image
  - position
  - type

List Style Type

- list-style-type: value;
- Values
  - disc
  - circle
  - square
  - decimal
  - lower-roman
  - upper-roman
  - lower-alpha
  - upper-alpha
  - none

Display

- You can control how an element is displayed with the display property
- display: value;
- Values:
  - block
  - inline
  - list-item
  - none
- **Now, what does all that mean?**
- **Block**
  - Creates a line break before and after the element
- **Inline**
  - No line break is created
- **List Item**
  - Creates a line break before and after the element and adds a list item marker
- **None**
  - Makes an element not display on the page

Float

- The float property changes how text and or images within an element are displayed
- float: value;
- Values:
  - left
  - right
  - none
- **Now, what does all that mean?**
- **Left**
  - The image/text is displayed to the left of the parent element
- **Right**
  - The image/text is displayed to the right of the parent element
- **None**
  - There is no change in the way the image/text is displayed

Overflow

- You can control what an elements contents will do if it overflows it boundaries with the overflow property
- overflow: value;
- Values:
  - auto
  - hidden
  - visible
  - scroll

Visibility

- You can control if an element is visible or not with the visibility property
- visibility: value;
- Values:
  - <u>hidden</u>
  - <u>visible</u>

Z-Index

- You can control the layer order of positioned elements with the z-index property
- z-index: value;
- Values:
  - auto
  - number
  - The higher the number the higher the level. Negative numbers are allowed

CSS 3 Features

- **Border Radius**

```
#my_id { height: 100px; width: 100px; border: 1px solid
   #FFF; /* For Mozilla: */ -moz-border-radius: 15px; /*
   For WebKit: */ -webkit-border-radius: 15px; }


div
   {
   border:2px solid;
   border-radius:25px;
   }
```

CSS 3 Features

Property	IE	chrome	firefox	safari	opera
border-radius	9.0	5.0 4.0 -webkit-	4.0 3.0 -moz-	5.0 3.1 -webkit-	10.5
box-shadow	9.0	10.0 4.0 -webkit-	4.0 3.5 -moz-	5.1 3.1 -webkit-	10.5
border-image	11.0	16.0 4.0 -webkit-	15.0 3.5 -moz-	6.0 3.1 -webkit-	15.0 10.5 -o-

CSS 3 Features

- **Border Images**
- #my_id { /* url, top image, right image, bottom image, left image */ border-image: url(border.png) 30 30 30 30 round round; }

CSS 3 Features

- **Box Shadow**
- #my_id { background: #FFF; border: 1px solid #000; /* For Mozilla: */ -moz-box-shadow: 5px #999; /* For WebKit: */ -webkit-box-shadow: 5px #999; }

CSS 3 Features

- **Multi-Column Layout**
- #my_id {text-size: 12px; /* For Mozilla: */ -moz-column-gap: 1em; -moz-column-rule: 1px solid #000; -moz-column-count: 3; /* For WebKit: */ -webkit-column-gap: 1em; -webkitcolumn-rule: 1px solid #000; -webkit-column-count: 3; }

CSS 3 Features

- **Multiple Backgrounds**
- #my_id { background: url(topbg.jpg) top left no-repeat, url(middlebg.jpg)center left no-repeat, url(bottombg.jpg) bottom left no-repeat; }

CSS 3 Features

- **@font-face**
- @font-face { font-family: “my-font”; src: url(my-font.tff) format(“truetype”); }
- #my_id { font-family: “my-font”, sans-serif; }

CSS 3 Features

- **Attribute Selectors**
- p[title^=t] { /* Attributes to give them. */ }

CSS 3 Features

- **:nth-child() and :nth-of-type()**
- p:nth-child(3n+1) { background: #F00; }
- /* First, Fourth, Seventh, etc..
- /*
- Only li elements */
- p li:nth-of-type(3n+1) { background: #F00; }

CSS 3 Features

- **Opacity**

- #my_id { background: #F00; opacity: 0.5; }

CSS 3 Features

- **RGBA Colors**
- #my_id { background: rgba(255, 212, 45, 0.5); }

CSS 3 Features

- **RGBA Colors**
- #my_id { background: rgba(255, 212, 45, 0.5); }

CSS3 2D Transforms

- translate()
- rotate()
- scale()
- skew()
- matrix()

Transition properties

- transition-property: width;
  transition-duration: 1s;
  transition-timing-function: linear;
  transition-delay: 2s;

CSS3 2D Transforms

- div
  {
  -ms-transform: translate(50px,100px); /* IE 9 */
  -webkit-transform: translate(50px,100px); /*
  Chrome, Safari, Opera */
  transform: translate(50px,100px);
  }

CSS3 2D Transforms

- div
  {
  -ms-transform: rotate(30deg); /* IE 9 */
  -webkit-transform: rotate(30deg); /* Chrome, Safari, Opera */
  transform: rotate(30deg);
  }

CSS3 2D Transforms

- div
  {
  -ms-transform: scale(2,4); /* IE 9 */
  -webkit-transform: scale(2,4); /* Chrome, Safari, Opera */
  transform: scale(2,4);
  }

CSS3 2D Transforms

- div
  {
  -ms-transform: skew(30deg,20deg); /* IE 9 */
  -webkit-transform: skew(30deg,20deg); /* Chrome, Safari, Opera */
  transform: skew(30deg,20deg);
  }

CSS3 2D Transforms

- div
  {
  -ms-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* IE 9 */
  -webkit-transform:matrix(0.866,0.5,-0.5,0.866,0,0); /* Chrome, Safari, Opera */
  transform:matrix(0.866,0.5,-0.5,0.866,0,0);
  }

CSS3 3D Transforms

- **The rotateX() Method**
- div
  {
  -webkit-transform: rotateX(120deg); /* Chrome, Safari, Opera */
  transform: rotateX(120deg);
  }

CSS3 3D Transforms

- **The rotateY() Method**
- div
  {
  -webkit-transform: rotateY(130deg); /* Chrome, Safari, Opera */
  transform: rotateY(130deg);
  }

The nature of output: grouping output with CSS media types

Media Type	Continuous/ Paged	Visual/Audio/ Speech/Tactile	Grid/ Bitmap	Interactive/ Static
braille	Continuous	Tactile	Grid	Both
embossed	Paged	Tactile	Grid	Static
handheld	Both	Visual, audio, speech	Both	Both
print	Paged	Visual	Bitmap	Static
projection	Paged	Visual	Bitmap	Interactive
screen	Continuous	Visual, audio	Bitmap	Both
speech	Continuous	Speech	N/A	Both
tty	Continuous	Visual	Grid	Both
tv	Both	Visual, audio	Bitmap	Both

Introduction to Jquery & Syntax

Objectives

At the end of this module you will understand

- What is jQuery?
- jQuery Features
- How jQuery Works
- Downloading jQuery Library
- Using jQuery Library
- Basic jQuery Example

What is jQuery?

- jQuery is an open-source library of JavaScript functions
- jQuery project started in 2006 by John Resig
- jQuery helps you to easily & quickly write clean JavaScript
- The jQuery library contains many common functions & plug-ins that you can extend immediately. For example:
  - Applying styles on HTML DOM elements on-the-fly
  - Use AJAX out-of-the-box
  - Add date picker, auto-complete, progress bar, menu tabs to your forms
  - Make animations almost as good as Flash can

JQuery

- jQuery is a fast, lightweight JavaScript library that is CSS3 compliant and supports many browsers.
- The jQuery framework is extensible and very nicely handles DOM manipulations, CSS, AJAX, Events and Animations.
- JavaScript is a language whereas jQuery is a library written using JavaScript.

Why jQuery?

- Consider that we have to apply zebra-stripes on a table with many rows
- This can be achieved by either using CSS or JavaScript
- Using CSS has an immediate fall-out – Row Consistency

1	Lost In The Plot	The Dears
2	Poison	The Constantines
3	Plea From A Cat Named Virtute	The Weakerthans
4	Melissa Louise	Chixdiggit!
5	Living Room	Tegan And Sara
6	Speed	Bran Van 3000
7	Fast Money Blessing	King Cobb Steelie
8	Sore	Buck 65
9	Love Travel	Danko Jones
10	You Never Let Me Down	Furnaceface

- A typical JavaScript function will require about a few dozen lines of code and could look something like this:

Zebra-striping with JavaScript

```
var stripe = function() {
  var tables = document.getElementsByTagName("table");
  for(var x=0;x!=tables.length;x++){
    var table = tables[x];
    if (! table) { return; }
    var tbodies = table.getElementsByTagName("tbody");
    for (var h = 0; h < tbodies.length; h++) {
      var even = true;
      var trs = tbodies[h].getElementsByTagName("tr");
      for (var i = 0; i < trs.length; i++) {
     this.className += " ruled"; return false
      if(even) trs[i].className += " even";
        even = !even;
      }
    }
  }
}
window.onload = stripe;
```

Zebra-striping with jQuery

```
$("tr:nth-child(even)").css('background-color','coral');
```

What does jQuery do?

- **Crunches JavaScript**
  - reduces the number of lines of JavaScript you would have to write, call it shortcut JavaScript
- **Manipulate HTML**
  - allows you to select & manipulate HTML, popularly used to manipulate <form> elements
- **Manipulate CSS**
  - allows you to apply, remove or change CSS properties applied on HTML elements, on-the-fly
- **Out-of-the-box widgets**
  - includes many commonly needed widgets. For example, calendar popups, accordion, tab panels, etc
- **Animation**
  - apply animated effects like show, hide, slide on HTML elements. Also includes some very cool animation effects on images.

Downloading jQuery Library

- Download the latest release of the library from **jQuery.com**
- 2 versions are available – minified & uncompressed

- Alternatives to downloading:
  - Google CDN:

```
<head>
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
</head>
```

  - Microsoft CDN:

```
<head>
<script type="text/javascript" src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"></script>
</head>
```

Jquery 3 Features

- *show()* and *hide() improved*
- Nomenclature for *.data()* Key Names
- *width()* and *height()* return decimal values
- *.load()*, *.unload()*, and *.error()* methods are removed
- *The recommended way is to use .on() to handle these events*
- jQuery Objects are Iterable Now
- jQuery Animations Now Use *requestAnimationFrame* API at the Backend
- Massive Speedups for Custom Selectors Like *:visible*
- Better Handling of Error Cases

Basic jQuery Usage Example

```
<html>
<head>
<script type="text/javascript" src="myJsPath/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

Add the jQuery library file in the head section

How does jQuery work?

- jQuery works first by “Selecting” the HTML element(s) and then applying your “Action” on it
- To select and apply action, use the **jQuery** object
- The jQuery object can also be called using $
- Example:

```
// Select all elements of class: "someClass" and
// apply a red border to all of them
jQuery(".someClass").css("border", "1px solid red");

// Exactly the same as above, but using $
$(".someClass").css("border", "1px solid red");
```

jQuery Syntax - $() Factory

- All types of selectors in jQuery start with $()
- The syntax is **$(selector).action()**
  - The $ sign defines jQuery
  - A selector to "query or find" HTML elements
  - A jQuery action() to perform on the HTML elements

- Examples:
  - $(this).hide → hides current element
  - $("p").hide → hides all paragraphs
  - $("p.test").hide → hides all paragraphs with class="test"
  - $("#test").hide → hides the element with id="test"

The Document Ready Function

```
$(document).ready(function(){
   // jquery function go here…
});
```

- This method of writing function, prevents the jQuery function from running before the document is finished loading (is ready)
- Note: It is not mandatory to write jQuery statements within this construct

Ready vs Load

```
$( document ).ready(function() {
    console.log( "document loaded" );
});
$( window ).load(function() {
    console.log( "window loaded" );
});
```

$(document).ready() function

- If you want an event to work on your page, you should call it inside the $(document).ready() function.
- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.
- The $(document).ready() function has some benefit for its own way to make its events to work.

- First no need for "behavioral" markup tag in the HTML.
- $(document).ready() helps your events to get load or fire before the window loads.
- Anything that you write inside its brackets is ready as soon as the DOM is registered by the browser.

How to write jQuery functions

- Simple function:

```
$(document).ready(function(){
    $("p").hide();
}); // end document.ready
```

- Nested function:

```
$(document).ready(function(){
    $("button").click(function(){
        var num1 = 10;
        var num2 = 5;
        var numSum = num1 + num2;
        alert("The sum is " + numSum);
    }); // end click
}); // end document.ready
```

jQuery

jQuery Syntax

- The jQuery syntax is made for **selecting** HTML elements and perform some **action** on the element(s).
- Syntax **$(selector).action()**
- A dollar sign to define jQuery
- A (selector) to "query (or find)" HTML elements
- A jQuery action() to be performed on the element(s)

Summary

- At the end of this sub module
  - What is jQuery
  - jQuery features
  - Downloading jQuery library
  - Including jQuery in web pages
  - How jQuery works
  - The $() and jQuery() function
  - jQuery chaining
  - The $(document).ready() function
  - Writing jQuery functions

jQuery Selectors

Objective

- At the end of this sub module we will understand
  - Element Selectors
  - CSS Selectors
  - Attribute Selectors
  - Advanced Selectors

jQuery Selectors

- In HTML DOM, selectors allow you to manipulate DOM elements as a group or as a single node
- jQuery selectors allow you to select HTML elements (or groups of elements) by element name, attribute name or by content
- Types of jQuery selectors:
  - Element Selectors
  - CSS Selectors
  - Attribute Selectors

jQuery Element Selectors

- Element Selectors
  - jQuery uses the HTML tags to select HTML elements
  - Example:
    - $("p") → selects all <p> elements
    - $("a") → selects all <a> elements
    - $("ul") → selects al <ul> elements
    - $("*") → selects all elements within the document
    - $("img, span") → selects <img> and elements

jQuery CSS Selectors

- CSS Selectors
  - jQuery uses CSS IDs & classnames to select elements
  - A CSS ID is preceded with a '#' & a classname with a '.'
  - Examples:
    - $("#intro") → selects the first element with id="intro"
    - $("p.intro") → selects all <p> elements with class="intro"
    - $("a#myId.myClass") → selects <a> with id="myId" & class="myClass"
    - $("p a.specialClass") → selects <a> with class="specialClass" declared within <p> elements

jQuery Attribute Selectors

- Attribute Selectors
  - jQuery uses XPath expressions to select elements with given attributes
  - To select, put the attribute with in [ ]
  - Examples:
    - $("[href]") → selects all elements with href attributes
    - $("input[type=text]") → selects all <input> elements with a type of text

jQuery Advanced Selectors

- jQuery advanced selector uses the next generation of CSS supported by Mozilla Firefox, IE 7, Chrome, Safari, etc
- Examples:
- $('*') → Selects all elements in the document
- $(this) → Selects the current element
- $("p > *") → Selects all elements that are children of <p> element
- $("li:not(.myclass)") →Selects all elements matched by <li> that do not have class="myclass"
- $("li > ul") → Selects all elements matched by <ul> that are children of an element matched by <li>
- $("code, em, strong") → Selects all elements matched by <code> or or
- $("p:empty") → Selects all elements matched by <p> that have no children
- $(":radio") → Selects all radio buttons in the form
- $(":checked") → Selects all checked boxes in the form
- $("[href!='#']") → Selects all elements with href value NOT equal to "#"
- $("[href^='http://']") → Selects all elements with href value that starts with 'http://'
- $("[href$='.pdf']") → Selects all elements with href value that ends with '.pdf'

jQuery Advanced Selectors

- $("li:even") → Selects all elements matched by <li> that have an even index value
- $("tr:odd") → Selects all elements matched by <tr> that have an odd index value
- $("li:first") → Selects the first <li> element
- $("li:last") → Selects the last <li> element
- $("li:eq(2)") → Selects the third <li> element. Index starts at 0
- $("li:lt(2)") → Selects all elements matched by <li> element before the third one; in other words, the first two <li> elements
- $("li:gt(1)") → Selects all elements matched by <li> after the second one
- $("li:first-child") → Selects all elements matched by <li> that are the first child of their parent
- $("li:last-child") → Selects all elements matched by <li> that are the last child of their parent
- $(":parent") → Selects all elements that are the parent of another element, including text
- $("li:contains(second)") → Selects all elements matched by <li> that contain the text second
- $("li:visible") → Selects all elements matched by <li> that are visible
- $("li:hidden") → Selects all elements matched by <li> that are hidden

** The above examples are not limited to <li> & also apply to other elements like <p>, <img>, etc*

jQuery Selector Usage

```
<html>
<head>
<title>jQuery - Selectors</title>
<script type="text/javascript" src="jsLib/jquery-1.4.2.min.js"></script>
</script>
</head>

<body>
<p id="intro">This is  just a test of JQuery.</p>
<ol>
    <li>Bangalore</li>
    <li>Hyderabad</li>
    <li>Kolkata</li>
    <li>Chennai</li>
    <li>Pune</li>
    <li>Bhubaneswar</li>
</ol>
<p>This is how it is defined </p>
```

jQuery Selector Usage (cont'd)

```
<script type="text/javascript">
$(document).ready(function(){
    $("p").css("background-color", "#ff9");
    $("a").css("background-color","#f00");
    $("p#intro").css("border","3px dashed #00f");
    $("li:first").css("background-color","#66f");
    $("li:contains(jQ)").css("background-color","#0f0");
});
</body></html>
```

Attributes

- $("a").attr("href", "allMyHrefsAreTheSameNow.html");
- $("a").attr({
- title: "all titles are the same too!",
- href: "somethingNew.html"
- });

Pseudo-Selectors

- // Select all input-like elements in a form (more on this below).
- $("#myForm :input");
- $("div:visible");
- // All except the first three divs.
- $("div:gt(2)");
- // All currently animated divs.
- $("div:animated");

Refining & Filtering Selections

- // Refining selections.
- $("div.foo").has("p"); // div.foo elements that contain <p> tags
- $("h1").not(".bar"); // h1 elements that don't have a class of bar
- $("ul li").filter(".current"); // unordered list items with class of current
- $("ul li").first(); // just the first unordered list item
- $("ul li").eq(5);

Selecting Form Elements

- $("form :button");
- $("form :checkbox");
- $("form :checked");
- $("form :disabled");
- $("form :enabled");
- $("form :file");
- $("form :image");
- $("form input[name='gender']:radio");

Working with Selections

- // The .html() method used as a setter:
- $("h1").html("hello world");
- // The .html() method used as a getter:
- $("h1").html();
- // Attempting to call a jQuery method after calling a getter.
- // This will NOT work:
- $("h1").html().addClass("test");
- $("#content").find("h3").eq(2).html("new text for the third h3!");

Manipulating Elements

- .html() – Get or set the HTML contents.
- .text() – Get or set the text contents; HTML will be stripped.
- .attr() – Get or set the value of the provided attribute.
- .width() – Get or set the width in pixels of the first element in the selection as an integer.
- .height() – Get or set the height in pixels of the first element in the selection as an integer.
- .position() – Get an object with position information for the first element in the selection, relative to its first positioned ancestor. This is a getter only.
- .val() – Get or set the value of form elements.

Moving, Copying, and Removing Elements

- // Make the first list item the last list item:
- var li = $("#myList li:first").appendTo("#myList");
- // Another approach to the same problem:
- $("#myList").append($("#myList li:first"));
- // Making a copy of an element.
- // Copy the first list item to the end of the list:
- $("#myList li:first").clone().appendTo("#myList");

Moving, Copying, and Removing Elements

- There are two ways to remove elements from the page: .remove() and .detach(). Use .remove() when you want to permanently remove the selection from the page.
- Use .detach() if you need the data and events to persist. Like .remove(), it returns the selection, but it also maintains the data and events associated with the selection, so you can restore the selection to the page at a later time.

Creating New Elements

```
// Creating new elements from an HTML string.
$( "<p>This is a new paragraph</p>" );
$( "<li class=\"new\">new list item</li>" );
// Creating a new element with an attribute object.
$( "<a/>", {
    html: "This is a <strong>new</strong> link",
    "class": "new",
    href: "foo.html"
});
```

Creating New Elements

- // Getting a new element on to the page.
-
- var myNewElement = $(" New element ");
-
- myNewElement.appendTo("#content");
-
- myNewElement.insertAfter("ul:last"); // This will remove the p from #content!
-
- $("ul").last().after(myNewElement.clone()); // Clone the p so now we have two.

Creating New Elements

```
// Creating and adding an element to the page at the same time.

$( "ul" ).append( "<li>list item</li>" );

var myItems = [];

var myList = $( "#myList" );

for ( var i = 0; i < 100; i++ ) {

    myItems.push( "<li>item " + i + "</li>" );

}

myList.append( myItems.join( "" ) );
```

Manipulating Attributes

```
// Manipulating a single attribute.
$( "#myDiv a:first" ).attr( "href", "newDestination.html" );
// Manipulating multiple attributes.
$( "#myDiv a:first" ).attr({
    href: "newDestination.html",
    rel: "nofollow"
});
```

Manipulating Attributes

```
// Using a function to determine an attribute's new value.

$( "#myDiv a:first" ).attr({

    rel: "nofollow",

    href: function( idx, href ) {

        return "/new/" + href;

    }

});

$( "#myDiv a:first" ).attr( "href", function( idx, href ) {

    return "/new/" + href;

});
```

Traversing

```
<div class="grandparent">
    <div class="parent">
        <div class="child">
            <span class="subchild"></span>
        </div>
    </div>
    <div class="surrogateParent1"></div>
    <div class="surrogateParent2"></div>
</div>
```

Traversing

- // Selecting an element's direct parent:
- // returns [div.child]
- $("span.subchild").parent();
- // Selecting all the parents of an element that match a given selector:
- // returns [div.parent]
- $("span.subchild").parents("div.parent");
- // returns [div.child, div.parent, div.grandparent]
- $("span.subchild").parents();

Traversing

- // Selecting all the parents of an element up to, but *not including* the selector:
- // returns [div.child, div.parent]
- $("span.subchild").parentsUntil("div.grandparent");
- // Selecting the closest parent, note that only one parent will be selected and that the initial element itself is included in the search:
- // returns [div.child]
- $("span.subchild").closest("div");
-
- // returns [div.child] as the selector is also included in the search:
- $("div.child").closest("div");

Traversing

- // Selecting an element's direct children:
- // returns [div.parent, div.surrogateParent1, div.surrogateParent2]

- $("div.grandparent").children("div");

-

Traversing

- $("div.parent").next();
- // Selecting a prev sibling of the selectors:
- // returns [] as No sibling exists before div.parent
- $("div.parent").prev();
- // Selecting all the next siblings of the selector:
- // returns [div.surrogateParent1, div.surrogateParent2]
- $("div.parent").nextAll();

Traversing

- // returns [div.surrogateParent1]
- $("div.parent").nextAll().first();
- // returns [div.surrogateParent2]
- $("div.parent").nextAll().last();
- // Selecting all the previous siblings of the selector:
- // returns [div.surrogateParent1, div.parent]
- $("div.surrogateParent2").prevAll();
- // returns [div.surrogateParent1]

CSS, Styling, & Dimensions

```
// Getting CSS properties.
$( "h1" ).css( "fontSize" ); // Returns a string such as "19px".

$( "h1" ).css( "font-size" ); // Also works.
// Setting CSS properties.

$( "h1" ).css( "fontSize", "100px" ); // Setting an individual property.

// Setting multiple properties.

$( "h1" ).css({

    fontSize: "100px",

    color: "red"

});
```

Using CSS Classes for Styling

- h1.addClass("big");
- h1.removeClass("big");
- h1.toggleClass("big");
-
- if (h1.hasClass("big")) {
- ...
- }

Dimensions

- // Working with classes.
- var h1 = $("h1");
- // Basic dimensions methods.
- // Sets the width of all <h1> elements.
- $("h1").width("50px");
- // Gets the width of the first <h1> element.
- $("h1").width();

Dimensions

- // Sets the height of all <h1> elements.
- $("h1").height("50px");
- // Gets the height of the first <h1> element.
- $("h1").height();
- // Returns an object containing position information for
- // the first <h1> relative to its "offset (positioned) parent".
- $("h1").position();

Utility Methods

```
// Returns "lots of extra whitespace"
$.trim( "    lots of extra whitespace    " );
$.each([ "foo", "bar", "baz" ], function( idx, val ) {
    console.log( "element " + idx + " is " + val );
});
$.each({ foo: "bar", baz: "bim" }, function( k, v ) {
    console.log( k + " : " + v );
});
```

Utility Methods

- var myArray = [1, 2, 3, 5];
- if ($.inArray(4, myArray) !== -1) {
- console.log("found it!");
- }

Utility Methods

- var firstObject = { foo: "bar", a: "b" };
- var secondObject = { foo: "baz" };
- var newObject = $.extend(firstObject, secondObject);
- console.log(firstObject.foo); // "baz"
- console.log(newObject.foo); // "baz"

Iterating over jQuery and non-jQuery Objects

- var sum = 0;
- var arr = [1, 2, 3, 4, 5];
- $.each(arr, function(index, value){
- sum += value;
- });
- console.log(sum); //

Iterating over jQuery and non-jQuery Objects

- <ul>
- <li><a href="#">Link 1</a></li>
- <li><a href="#">Link 2</a></li>
- <li><a href="#">Link 3</a></li>
- </ul>

Iterating over jQuery and non-jQuery Objects

- $("li").each(function(index, element){
- console.log($(this).text());
- });

Using jQuery's .index() Function

- <ul>
-
- <li id="foo1">foo</li>
- <li id="bar1">bar</li>
- <li id="baz1">baz</li>
-
- </ul>

Using jQuery's .index() Function

- var foo = $("#foo1");
- console.log("Index: " + foo.index()); // 1
- var listItem = $("li");
- // This implicitly calls .first()
- console.log("Index: " + listItem.index()); // 1
- console.log("Index: " + listItem.first().index()); // 1
- var div = $("div");
- // This implicitly calls .first()
- console.log("Index: " + div.index()); // 0
- console.log("Index: " + div.first().index()); // 0

How do I replace text from the 3rd element of a list

- var thirdLink = $(this).find("li a").eq(2);
-
- var linkText = thirdLink.text().replace("foo", "bar");
-
- thirdLink.text(linkText);

How do I get the text value of a selected option?

```
<select id="myselect">
    <option value="1">Mr</option>
    <option value="2">Mrs</option>
    <option value="3">Ms</option>
    <option value="4">Dr</option>
    <option value="5">Prof</option>
</select>
```

How do I get the text value of a selected option?

- $("#myselect option:selected").text();

How do I check/uncheck a checkbox input or radio button?

- // Check #x
- $("#x").prop("checked", true);
- // Uncheck #x
- $("#x").prop("checked", false);

jQuery Event Basics

```
// Event setup using a convenience method
$( "p" ).click(function() {
    console.log( "You clicked a paragraph!" );
});
// Equivalent event setup using the `.on()` method
$( "p" ).on( "click", function() {
    console.log( "click" );
});
```

Many events and handlers

```
$( "div" ).on({
    mouseenter: function() {
        console.log( "hovered over a div" );
    },
```

Many events and handlers

```
mouseleave: function() {

        console.log( "mouse left a div" );

    },
    click: function() {

        console.log( "clicked on a div" );

    }});
```

Disconnecting Events

- // Unbinding all click handlers on a selection
- $("p").off("click");

Introduction to Effects

- // Instantaneously hide all paragraphs
- $("p").hide();
- // Instantaneously show all divs that have the hidden style class
- $("div.hidden").show();

Introduction to Effects

- / Slowly hide all paragraphs
- $("p").hide("slow");
-
- // Quickly show all divs that have the hidden style class
- $("div.hidden").show("fast");

Fade and Slide Animations

- /// Hide all paragraphs using a slide up animation over 0.8 seconds
- $("p").slideUp(800);
- // Show all hidden divs using a slide down animation over 0.6 seconds
- $("div.hidden").slideDown(600);

Fade and Slide Animations

- // Hide all paragraphs using a fade out animation over 1.5 seconds
- $("p").fadeOut(1500);
-
- // Show all hidden divs using a fade in animation over 0.75 seconds
- $("div.hidden").fadeIn(750);

Changing Display Based on Current Visibility State

- // Instantaneously toggle the display of all paragraphs
- $("p").toggle();
- // Slowly toggle the display of all images
- $("img").toggle("slow");
- // Toggle the display of all divs over 1.8 seconds
- $("div").toggle(1800);

Doing Something After an Animation Completes

- $("p.hidden").fadeIn(750).addClass("lookAtMe");

Custom Effects with .animate()

```
// Custom effects with .animate()
$( "div.funtimes" ).animate(
    {
        left: "+=50",
        opacity: 0.25
    },
    // Duration
    300,
    // Callback to invoke when the animation is finished
    function() {
        console.log( "done!" );
    }
);
```

AJAX

- AJAX is the art of exchanging data with a server, and update parts of a web page - without reloading the whole page.

Method	Description
$.ajax()	Performs an async AJAX request
$.ajaxPrefilter()	Handle custom Ajax options or modify existing options before each request is sent and before they are processed by $.ajax()
$.ajaxSetup()	Sets the default values for future AJAX requests
$.ajaxTransport()	Creates an object that handles the actual transmission of Ajax data
$.get()	Loads data from a server using an AJAX HTTP GET request

AJAX

$.getJSON()	Loads JSON-encoded data from a server using a HTTP GET request
$.getScript()	Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request
$.param()	Creates a serialized representation of an array or object (can be used as URL query string for AJAX requests)
$.post()	Loads data from a server using an AJAX HTTP POST request
ajaxComplete()	Specifies a function to run when the AJAX request completes
ajaxError()	Specifies a function to run when the AJAX request completes with an error
ajaxSend()	Specifies a function to run before the AJAX request is sent
ajaxStart()	Specifies a function to run when the first AJAX request begins
ajaxStop()	Specifies a function to run when all AJAX requests have completed

AJAX

ajaxSuccess()	Specifies a function to run when an AJAX request completes successfully
load()	Loads data from a server and puts the returned data into the selected element
serialize()	Encodes a set of form elements as a string for submission
serializeArray()	Encodes a set of form elements as an array of names and values

jQuery ajax() Method

- The ajax() method is used to perform an AJAX (asynchronous HTTP) request.
- All jQuery AJAX methods use the ajax() method. This method is mostly used for requests where the other methods cannot be used.

jQuery ajax() Method

- **Syntax**
- $.ajax({*name:value, name:value, ...* })

jQuery ajax() Method

Name	Value/Description
async	A Boolean value indicating whether the request should be handled asynchronous or not. Default is true
beforeSend(*xhr*)	A function to run before the request is sent
cache	A Boolean value indicating whether the browser should cache the requested pages. Default is true
complete(*xhr,status*)	A function to run when the request is finished (after success and error functions)
contentType	The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded"
context	Specifies the "this" value for all AJAX related callback functions
data	Specifies data to be sent to the server

jQuery ajax() Method

dataFilter(*data*,*type*)	A function used to handle the raw response data of the XMLHttpRequest
dataType	The data type expected of the server response.
error(*xhr,status,error*)	A function to run if the request fails.
global	A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true
ifModified	A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false.
jsonp	A string overriding the callback function in a jsonp request
jsonpCallback	Specifies a name for the callback function in a jsonp request
password	Specifies a password to be used in an HTTP access authentication request.
processData	A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true

jQuery ajax() Method

scriptCharset	Specifies the charset for the request
success(*result,status,xhr*)	A function to be run when the request succeeds
timeout	The local timeout (in milliseconds) for the request
traditional	A Boolean value specifying whether or not to use the traditional style of param serialization
type	Specifies the type of request. (GET or POST)
url	Specifies the URL to send the request to. Default is the current page
username	Specifies a username to be used in an HTTP access authentication request
xhr	A function used for creating the XMLHttpRequest object

jQuery - AJAX get() and post() Methods

- **HTTP Request: GET vs. POST**
- Two commonly used methods for a request-response between a client and server are: GET and POST.
- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource
- GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.
- POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

jQuery $.get() Method

- **Syntax:**
- $.get(*URL,callback*);
- The required URL parameter specifies the URL you wish to request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.

jQuery $.post() Method

- **Syntax:**
- $.post(*URL,data,callback*);
- The required URL parameter specifies the URL you wish to request.
- The optional data parameter specifies some data to send along with the request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.

Hands-On-Lab

- Create a HTML document with:
  - Add the jQuery library to your HTML document using <script>
  - <h1> header
  - <p> with some content in it
  - An ordered list <ol> with about 6 <li> elements
  - Another <p> with some content in it and few of the phrases with <a> link on it

- The first and the last <li> element should have the background color as red and green respectively.

Summary

- In this session, we learnt about:
  - Element Selectors
  - CSS Selectors
  - Attribute Selectors
  - Advanced Selectors

An Introduction to Sencha Touch

“Create amazing apps
built
on web standards”

Documents

Applications

Thin client

Thick client

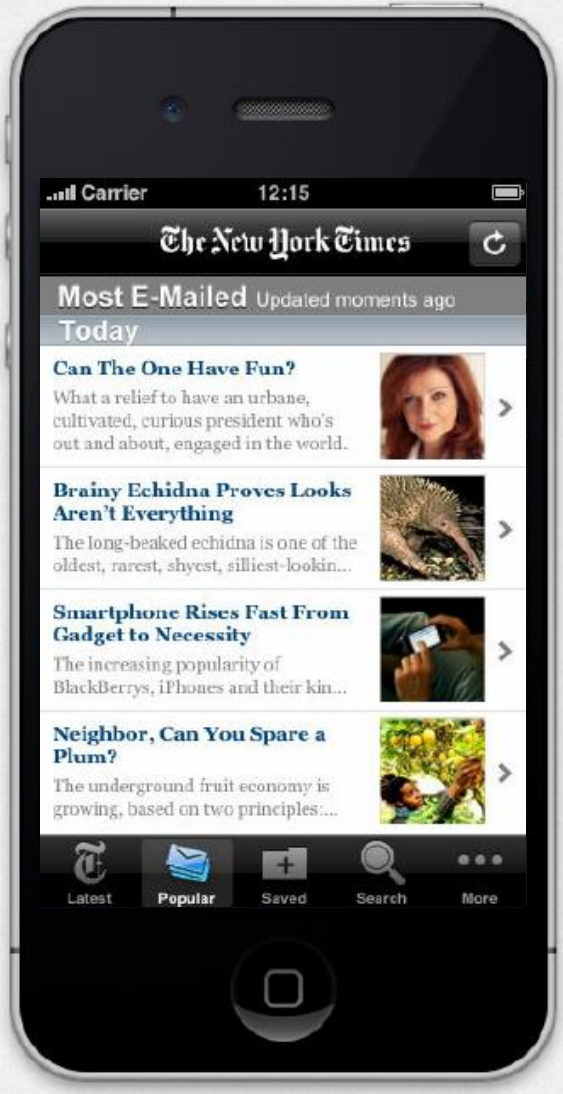
One PC

Multiple devices

Sedentary usage

Mobile usage

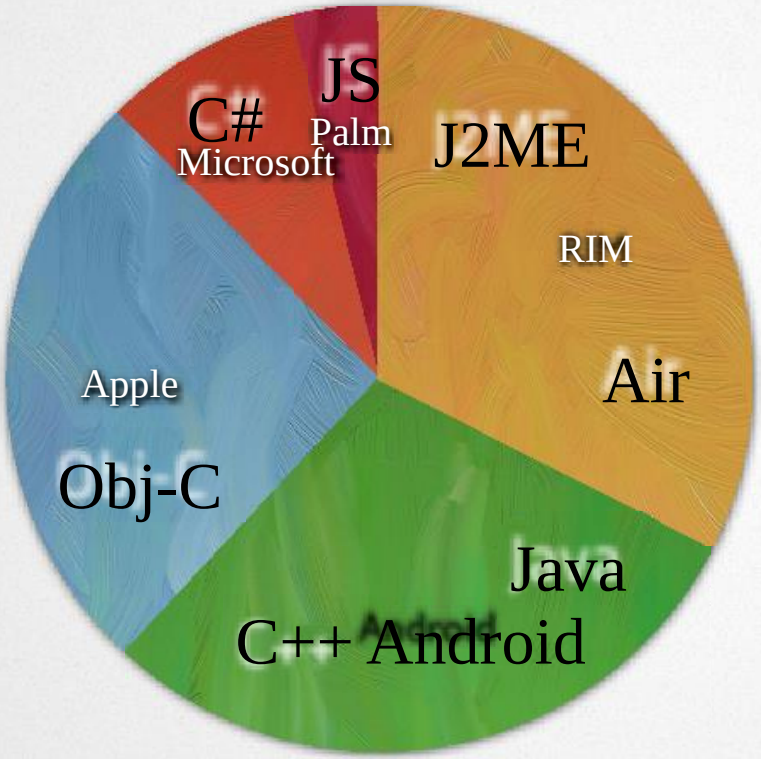
But everyone loves apps


Carrier
12:15
The New York Times
Most E-Mailed Updated moments ago
Today
Can The One Have Fun?
What a relief to have an urbane, cultivated, curious president who's out and about, engaged in the world.
Brainy Echidna Proves Looks Aren't Everything
The long-beaked echidna is one of the oldest, rarest, shyest, silliest-lookin...
Smartphone Rises Fast From Gadget to Necessity
The increasing popularity of BlackBerrys, iPhones and their kin...
Neighbor, Can You Spare a Plum?
The underground fruit economy is growing, based on two principles:...
Latest
Popular
Saved
Search
More

The New York Times
Obama Orders Improvements in Security Policies

T-Mobile
Amateur Finds New Earth-Sized Blot on Jupiter
The New York Times
Amateur Discovers Earth-Sized Blot on Jupiter
Pace of Decline Seems to Have Slowed, Fed Chief Says
U.S. Withheld Data on Risks of Distracted Driving
The Unwelcome Mat Is Out for Beckham
U.S.
Politics
N.Y./Region
Business
Technology
MENU

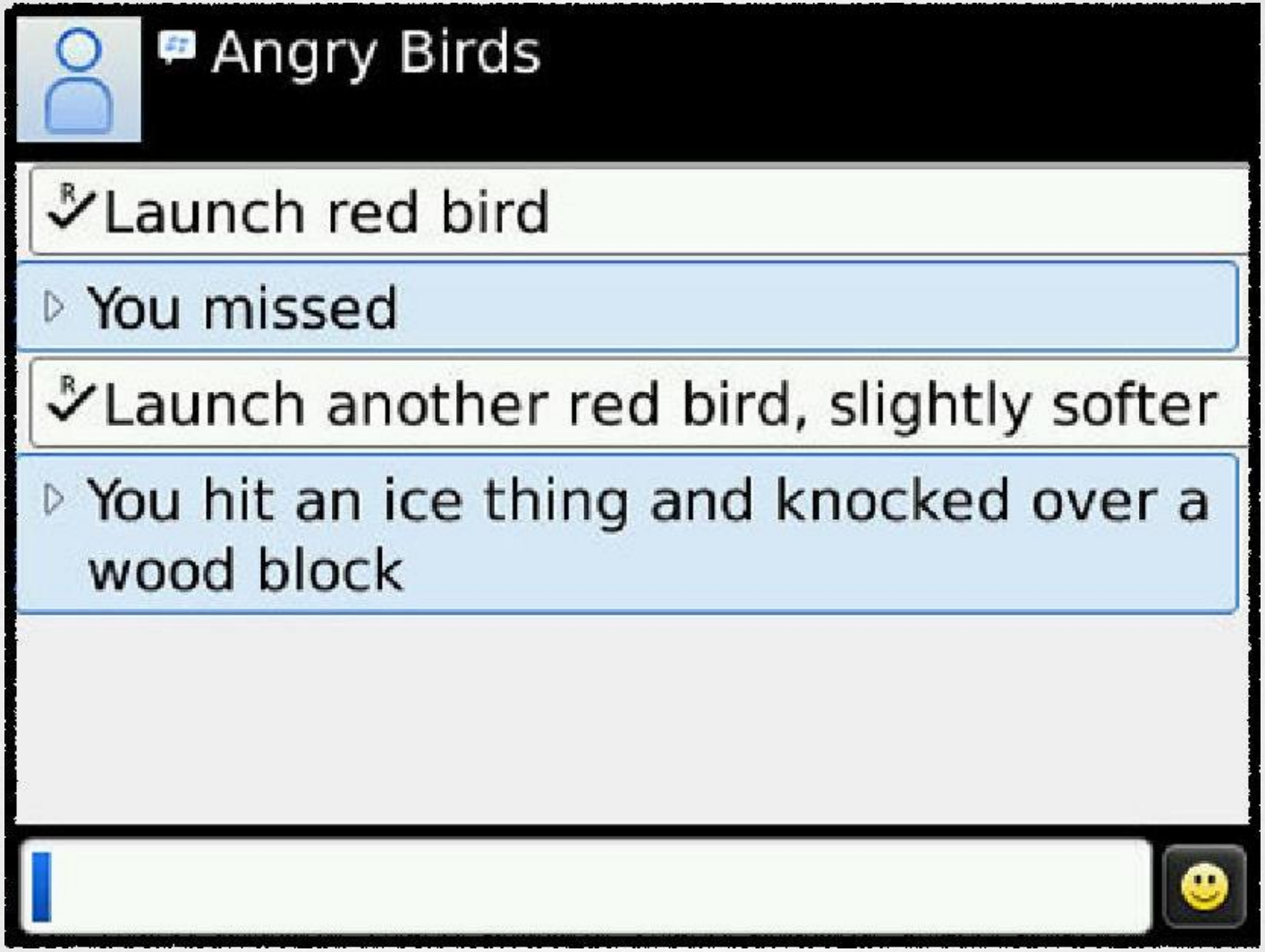
The native challenge



- RIM
- Android
- Apple
- Microsoft
- Palm

Top U.S. Smartphone Platforms,
3 Month Average Ending December 2010
comScore MobiLens 2010

HIGHSCORE: 103270
SCORE: 0
TNT
TNT
TNT

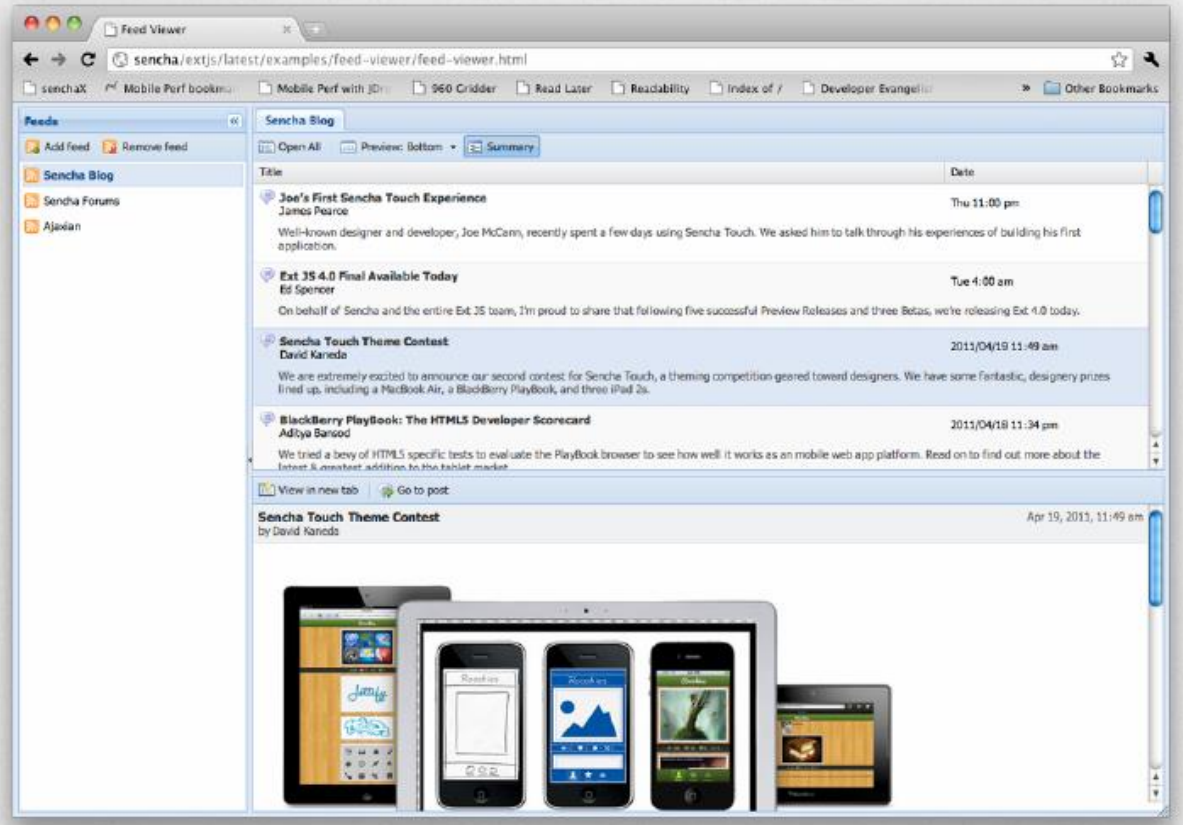

Angry Birds
Launch red bird
You missed
Launch another red bird, slightly softer
You hit an ice thing and knocked over a wood block

A return to the web

Cross-platform
Familiar skills & tools
Decentralized
Easily updated
Indexed
Well-understood technologies

HTML

A New Mobile App Stack


Feed Viewer
sencha/extjs/latest/examples/feed-viewer/feed-viewer.html
senchaX
Mobile Perf bookm
Mobile Perf with JDr
960 Gridder
Read Later
Readability
Index of /
Developer Evangeli
Other Bookmarks
Feeds
Add feed
Remove feed
Sencha Blog
Sencha Forums
Ajaxian
Sencha Blog
Open All
Preview: Bottom
Summary
Title
Date
Joe's First Sencha Touch Experience
James Pearce
Thu 11:00 pm
Well-known designer and developer, Joe McCann, recently spent a few days using Sencha Touch. We asked him to talk through his experiences of building his first application.
Ext JS 4.0 Final Available Today
Ed Spencer
Tue 4:00 am
On behalf of Sencha and the entire Ext JS team, I'm proud to share that following five successful Preview Releases and three Betas, we're releasing Ext 4.0 today.
Sencha Touch Theme Contest
David Kaneda
2011/04/19 11:49 am
We are extremely excited to announce our second contest for Sencha Touch, a theming competition geared toward designers. We have some fantastic, designery prizes lined up, including a MacBook Air, a BlackBerry PlayBook, and three iPad 2s.
BlackBerry PlayBook: The HTML5 Developer Scorecard
Aditya Bansod
2011/04/18 11:34 pm
We tried a bevy of HTML5 specific tests to evaluate the PlayBook browser to see how well it works as an mobile web app platform. Read on to find out more about the
View in new tab
Go to post
Sencha Touch Theme Contest
by David Kaneda
Apr 19, 2011, 11:49 am

htc
Sprint
10:08 AM
10
08
AM
Seattle
Sunny
Thu, Jun 01
73°
Messages
Voicemail
People
Mail
Market
Camera
Phone
menu
9:41 AM
Messages
Calendar
Photos
Camera
YouTube
Stocks
Maps
Weather
Notes
Utilities
iTunes
App Store
Settings
Phone
Mail
Safari
iPod

MOTOROLA

?

Introducing Sencha Touch

What is Sencha Touch?

- Sencha Touch is an HTML5 framework for developing mobile applications.
- It allows you to develop mobile applications that would have the same look and feel as a native application.
- Sencha Touch supports Android, iOS, Windows Phone, Microsoft Surface Pro and RT, and Blackberry devices.

Features

- UI components (Panels, Tab bar, Navigation view, buttons, pickers)
- Components can be themed depending on the target devices
- Access device capabilities like camera, accelerometer etc, with the help of PhoneGap frameworks.

The first JavaScript framework for building rich mobile apps with web standards

Carrier
3:36 PM
E-Resistible
Leverate Mob
TravelMate
VFA Mobile
Contacts
Safari
HTML

Carrier
3:10 PM
e-resistible
Quality Online Takeaways
Enter your postcode or town:
eg CV32 5QZ
Go!
or hit 'Go' for a demo
HTML

Carrier
3:30 PM
E-Resistible
Leverate Mob
TravelMate
VFA Mobile
Contacts
Safari
HTML

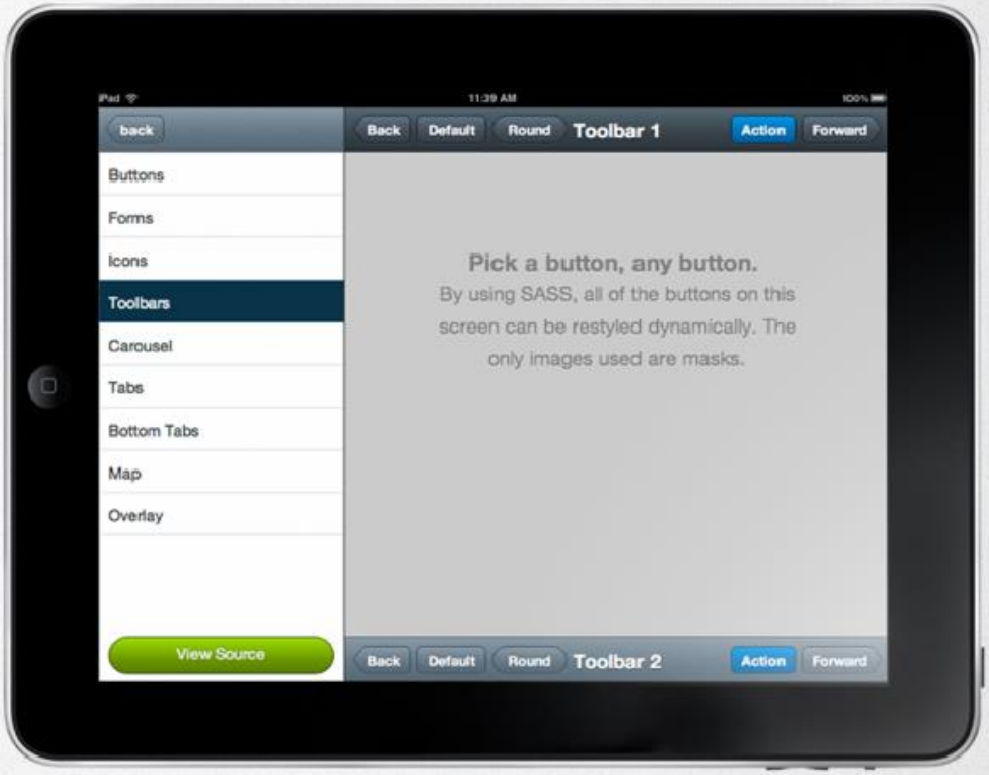
What’s in Sencha Touch?

Layouts & components
Theming & icons
Orientation & animation
Touch events & scroller
Data package
MVC framework

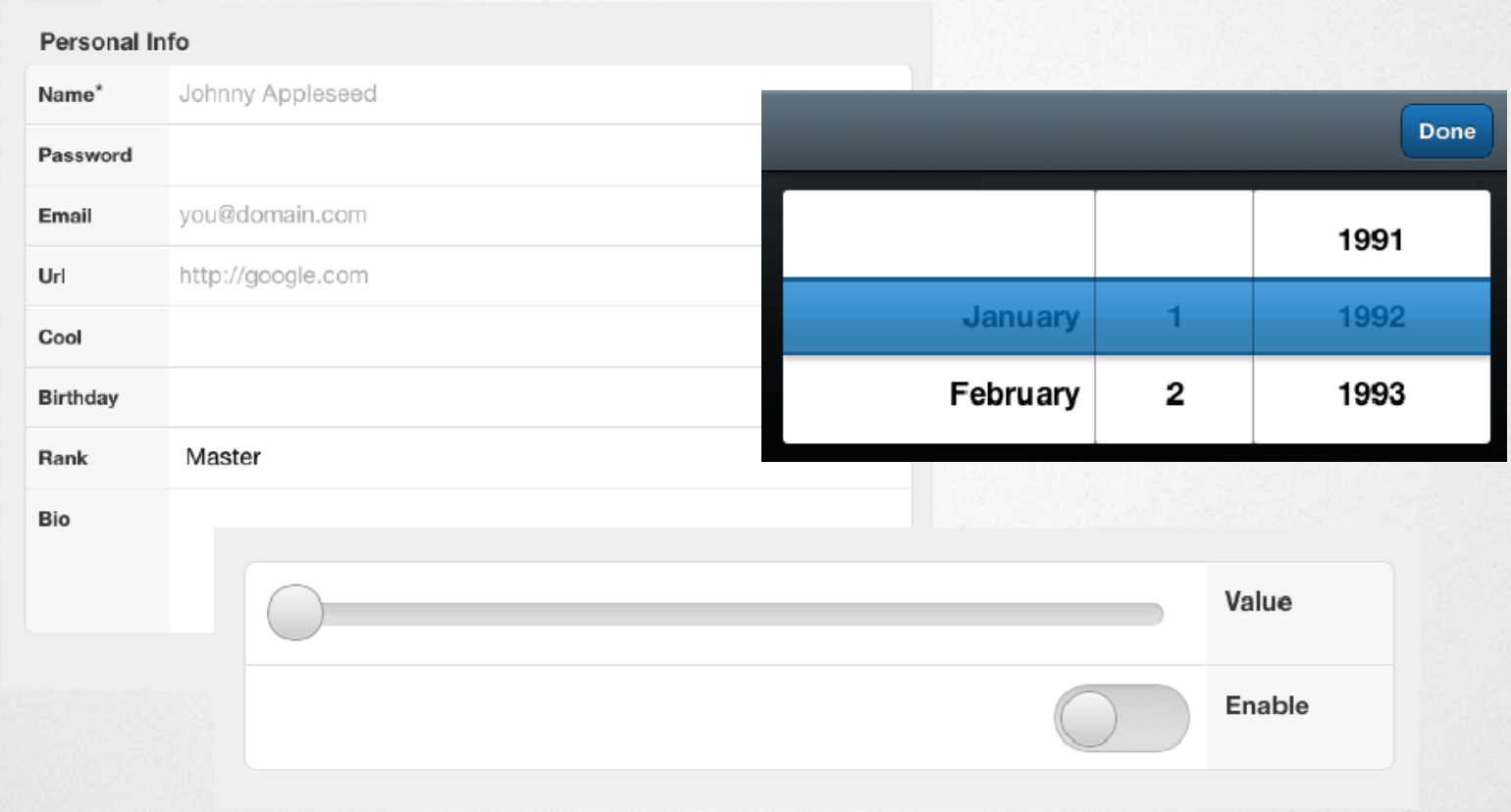
Components

Lists
- Nested, Grouped, Sortable

Carousel
Picker
Overlay
Slider
Forms & fields
Toolbars & buttons
HTML5
- Audio
- Video
- GeoLocation



Forms


Personal Info
Name*
Johnny Appleseed
Password
Email
you@domain.com
Url
http://google.com
Cool
Birthday
Rank
Master
Bio
Done
1991
January
1
1992
February
2
1993
Value
Enable

Scrolling

Momentum/bounce physics

Hardware accelerated

Throughout all components:

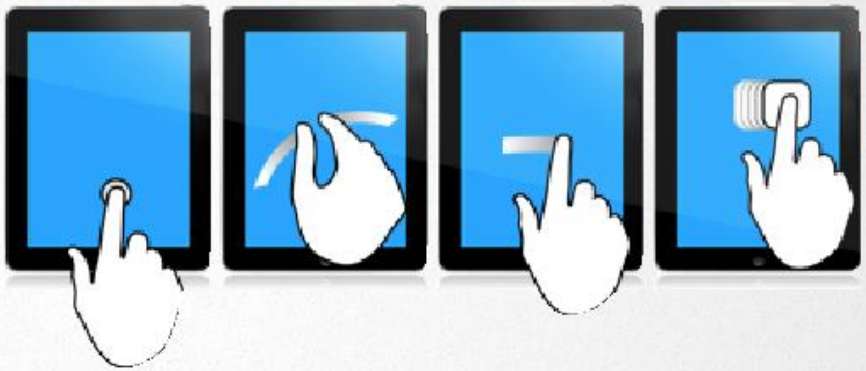
- Lists
- Carousel
- Pickers

Touch Events

Built on native events

Abstracted for performance

Additional events
- Tap
- Double tap
- Tap and hold
- Swipe
- Rotate
- Drag & drop



Data Package

Models, Stores, and Proxies

- Associations
- Validation
- Local & server storage

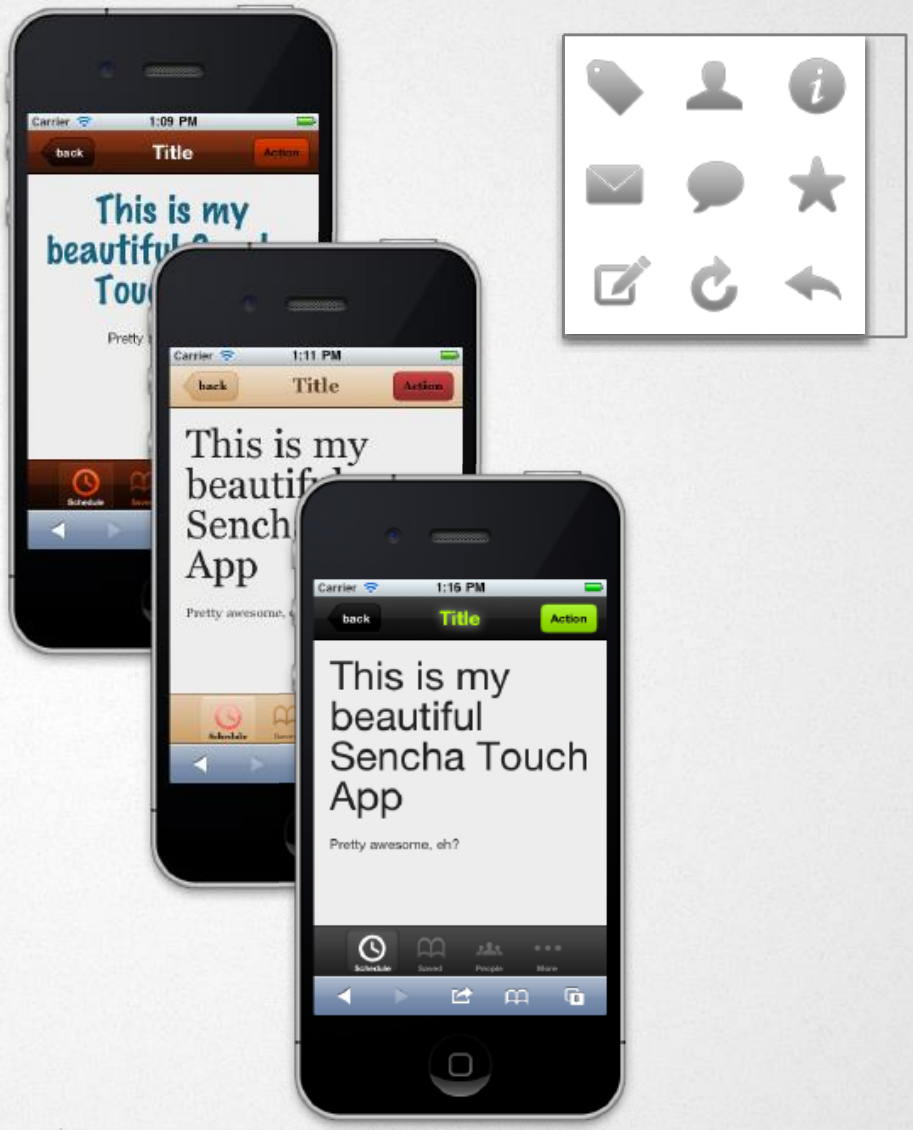
Easily consume web services

- JSON/P
- XML
- YQL

Theming

Use CSS3 & SASS

- Flexible themes
- Highly optimized



e.g.

```
$base-color: #ff6699
```

“The Kitchen Sink”

http://sencha.com/x/5e

Architectural Considerations

The classic web stack

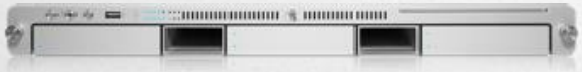
User interface

Business logic

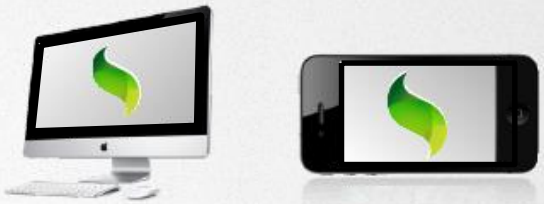
Storage

req/res

Rendering



The next web stack



User interface

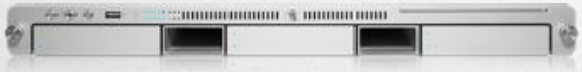
Business logic

Storage

sync

Security

Storage



Write once,
run anywhere?

Not quite.

“The Mobile Web
is not a
320px Web”

Feed Viewer
sencha/extjs/latest/examples/feed-viewer/feed-viewer.html
Feeds
Add Feed
Remove feed
Sencha Blog
Sencha Forums
Ajaxian
Sencha Blog
Open All
Preview: Bottom
Summary
Title
Date
Joe's First Sencha Touch Experience
James Pearce
Thu 11:00 pm
Ext JS 4.0 Final Available
Ed Spencer
Sencha Touch Theme
David Kaneda
BlackBerry PlayBook
Aditya Bansod
View in new tab
Sencha Touch Theme
by David Kaneda
Carrier
2:52 PM
Kitchen Sink
Back
YQL
Nested Loading
JSON P
YQL
AJAX
Load using YQL
Joe's First Sencha Touch Experience by
Ext JS 4.0 Final Available Today by
Sencha Touch Theme Contest by
BlackBerry PlayBook: The HTML5 Developer Scored
Source Conference for Sencha Web App Developers
7 in Europe by
Carrier
2:51 PM
Back
YQL
Source
Load using YQL
Joe's First Sencha Touch Experience by
Well-known designer and developer, Joe McCann, recently spent a few days using Sencha Touch. We asked him to talk through his experiences of building his first application.
Ext JS 4.0 Final

But you can re-use
a lot of code

Views

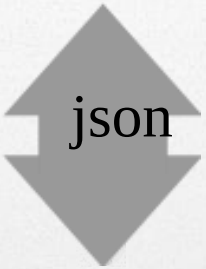
Views

Controllers

Models

Stores

Proxies



Ext JS

buttongroup
colorpalette
cycle
editor
editorgrid
grid
multislider
progress
splitbutton
treepanel
viewport
window

Sencha Touch

actionsheet
audio
carousel
list
map
nestedlist
picker
pickerslot
segbutton
sheet
tabbar
video

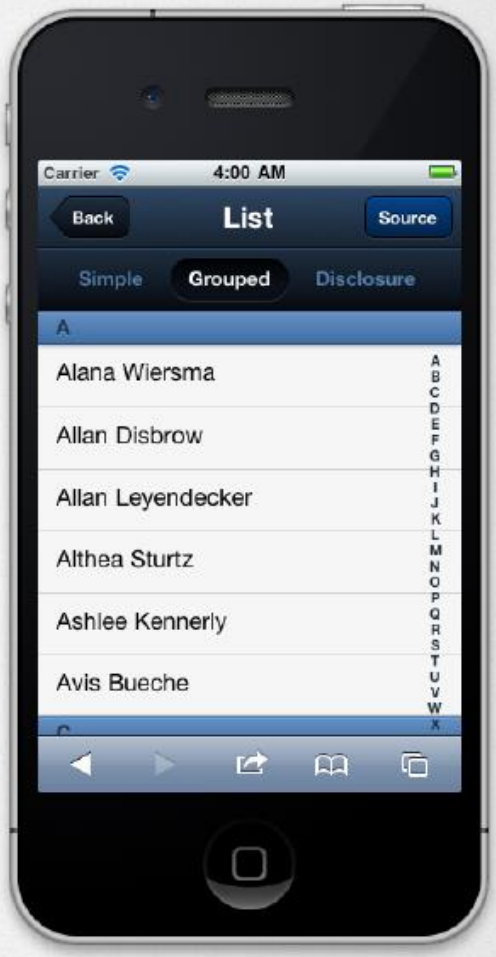
Ext JS / Sencha Touch

button
component
container
dataview
datepicker
panel
slider
spacer
toolbar
tabpanel

User Interface Components

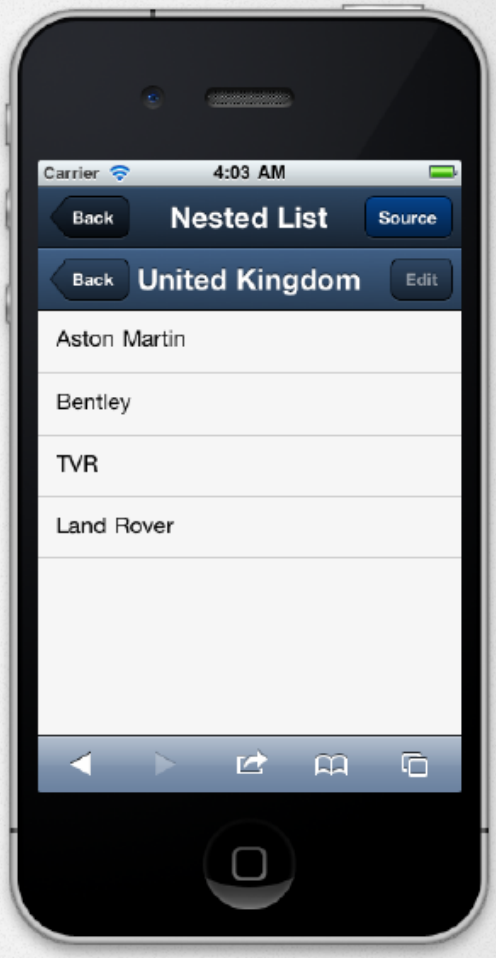
List

```
var list = new Ext.List({
  store: store,
  itemTpl:
   '{firstName} {lastName}',
  grouped: true,
  indexBar: true
});
```



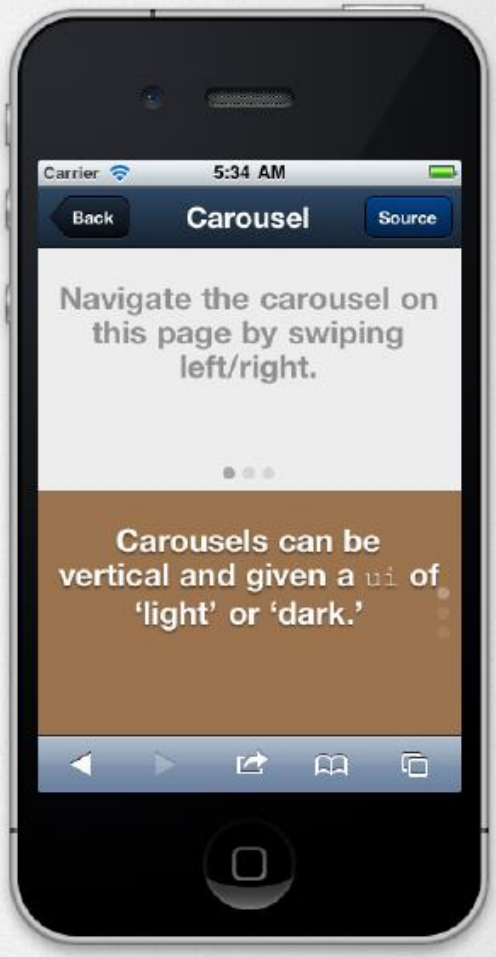
Nested List

```
var list = new Ext.NestedList({
  store: store,
  displayField: 'name',
  title: 'My List',
  updateTitleText: true,
  getDetailCard:
   function(record, parent) {..}
});
```



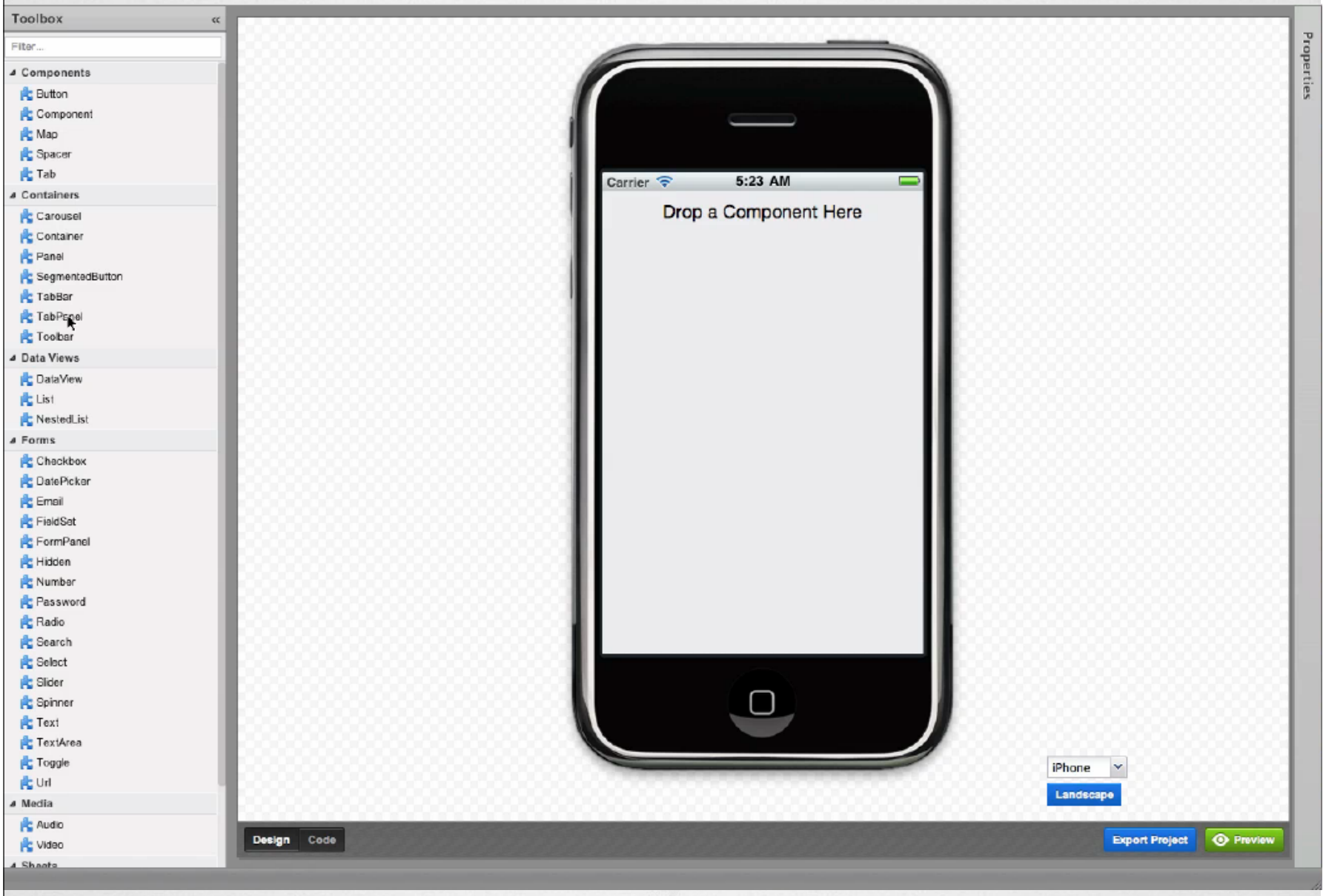
Carousel

```
var car = new Ext.Carousel({
  direction: 'horizontal',
  indicator: true,
  items: [
    ..
  ]
});
```

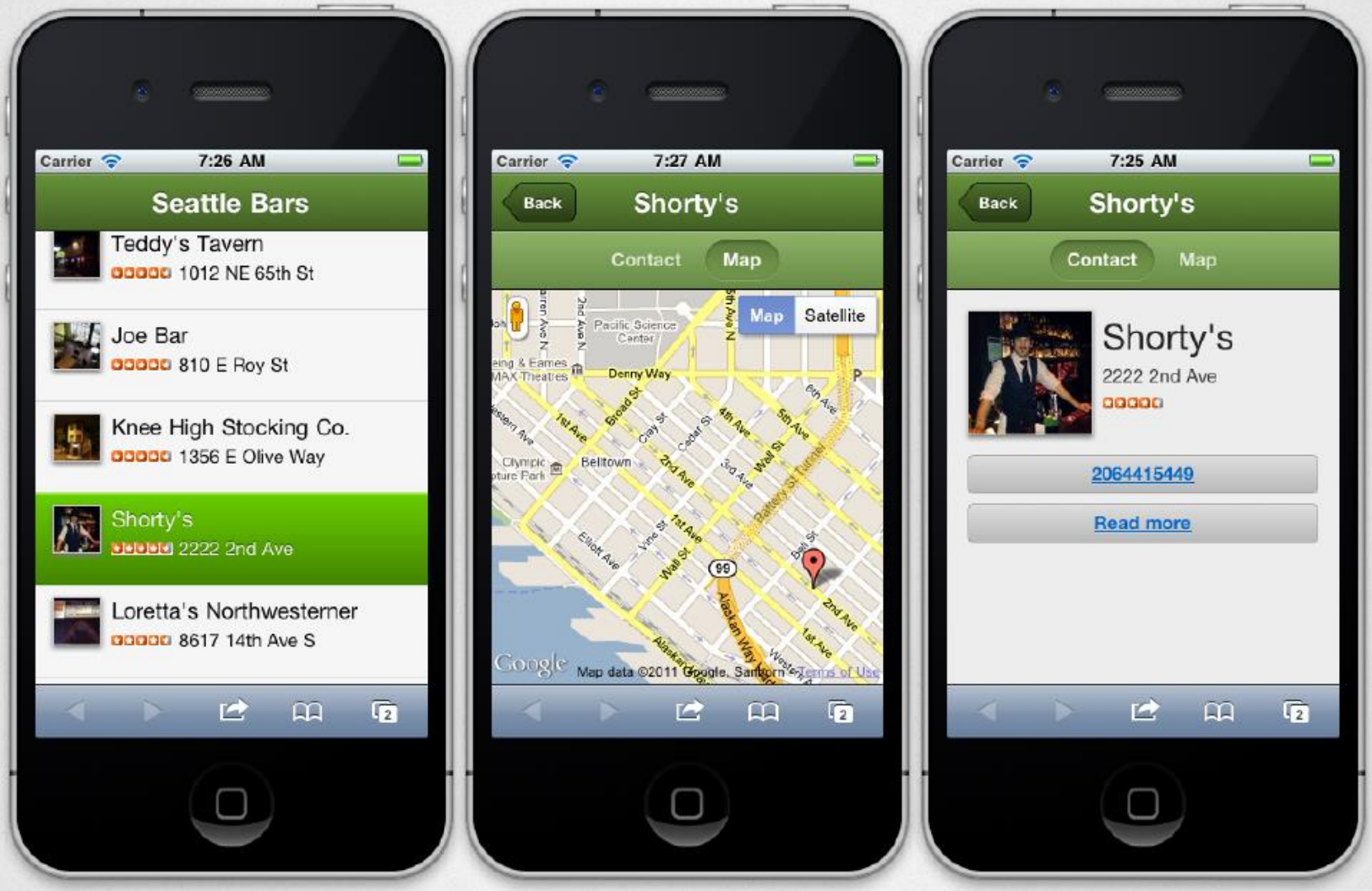


Sheets

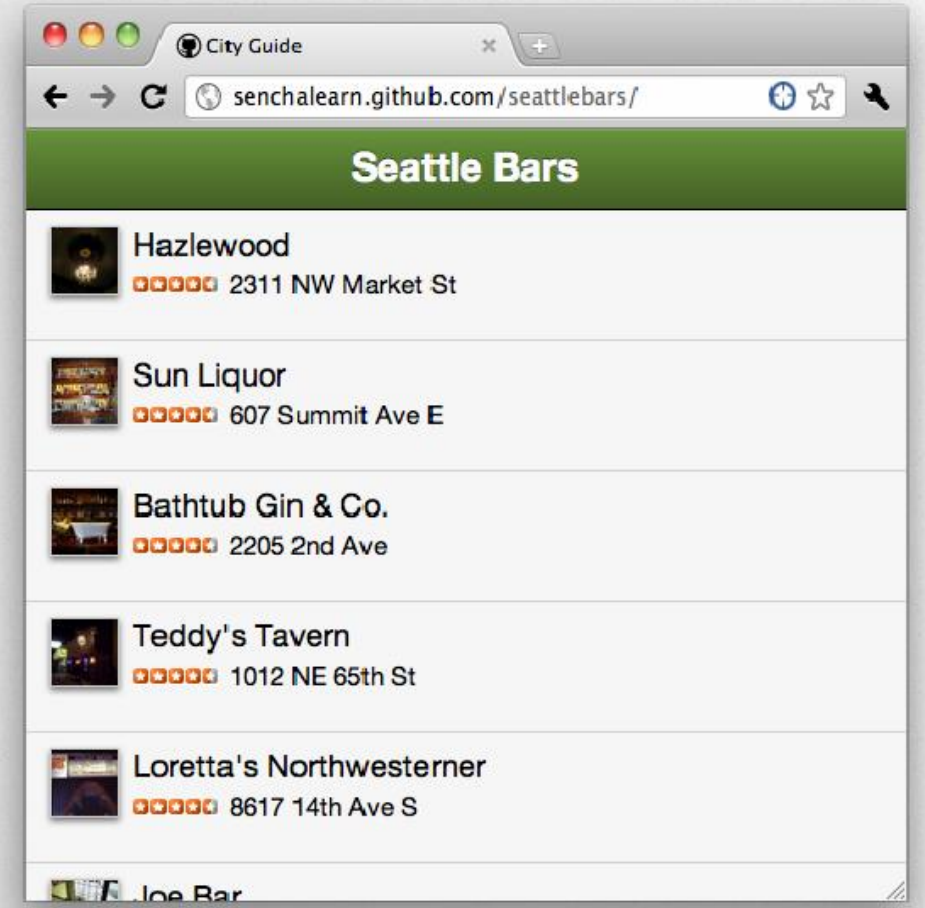
```
var sheet = new Ext.ActionSheet({
  items: [
    {
      text: 'Delete draft',
      ui: 'decline'
    }, {
      text: 'Save draft'
    }, {
      text: 'Cancel',
    }
  ]
});
sheet.show();
```

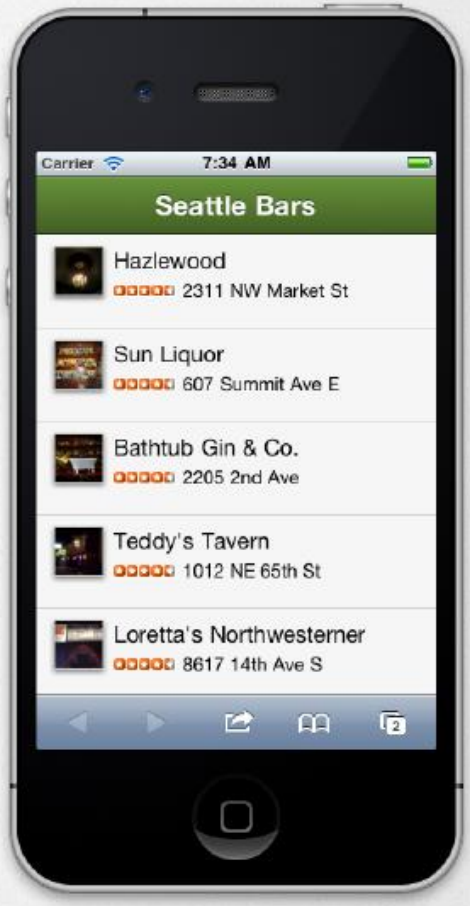

Toolbox
Filter...
Components
Button
Component
Map
Spacer
Tab
Containers
Carousel
Container
Panel
SegmentedButton
TabBar
TabPanel
Toolbar
Data Views
DataView
List
NestedList
Forms
Checkbox
DatePicker
Email
FieldSet
FormPanel
Hidden
Number
Password
Radio
Search
Select
Slider
Spinner
Text
TextArea
Toggle
Url
Media
Audio
Video
Carrier
5:23 AM
Drop a Component Here
iPhone
Landscape
Design
Code
Export Project
Preview
Properties

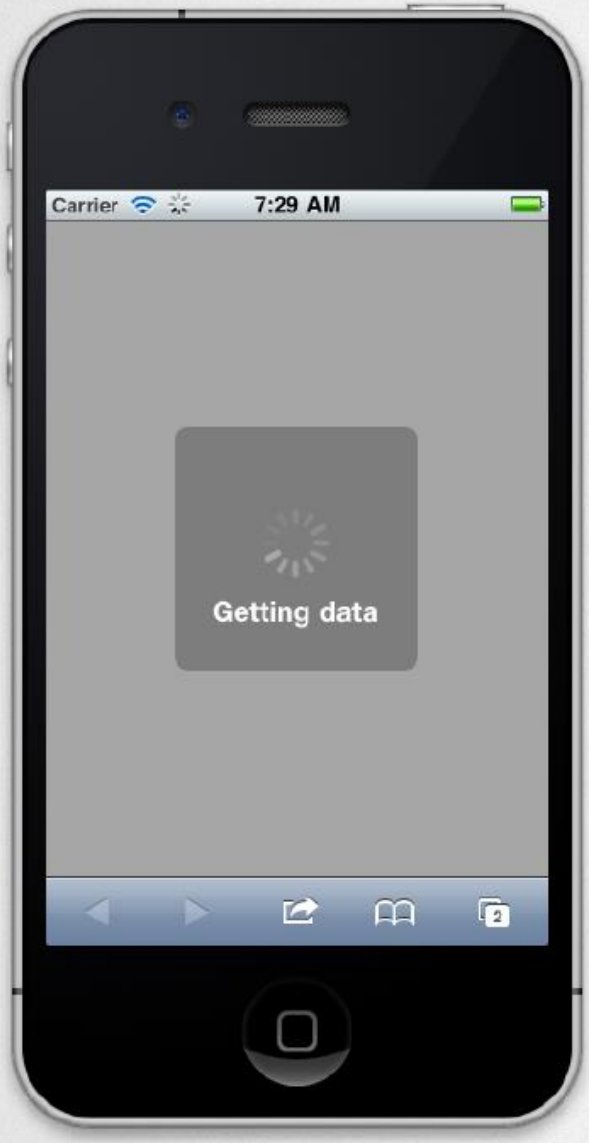
A Typical Application


Carrier
7:26 AM
Seattle Bars
Teddy's Tavern
1012 NE 65th St
Joe Bar
810 E Roy St
Knee High Stocking Co.
1356 E Olive Way
Shorty's
2222 2nd Ave
Loretta's Northwesterner
8617 14th Ave S
Carrier
7:27 AM
Back
Shorty's
Contact
Map
Map
Satellite
Pacific Science Center
Denny Way
Belltown
Map data ©2011 Google, Sanborn
Terms of Use
Carrier
7:25 AM
Back
Shorty's
Contact
Map
Shorty's
2222 2nd Ave
2064415449
Read more

http://senchalearn.github.com/seattlebars/







APIs at work...

Location API to get lat/long

MongoLabs to get city name

Yelp to get businesses

Final Thoughts

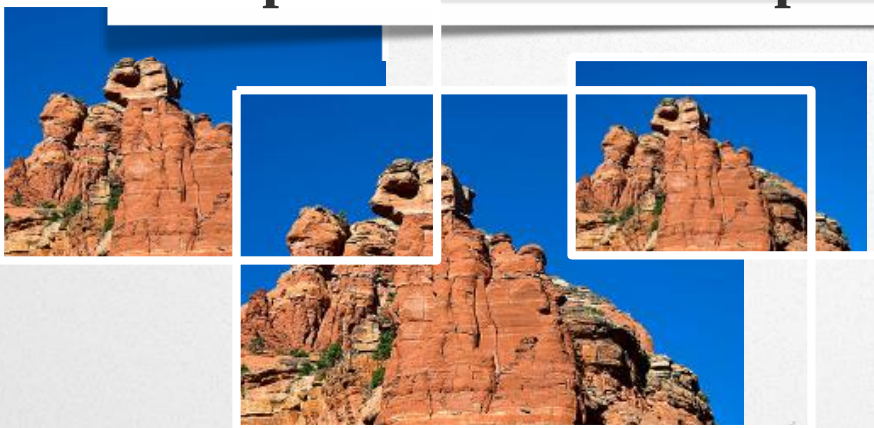
Thick client, thin server

The shortfall in the cloud

Location Services
Adaptation
Analytics
Aa
Web Fonts
Data Sync
Video Serving
Ad Serving
Image Serving
$
Commerce
Network APIs

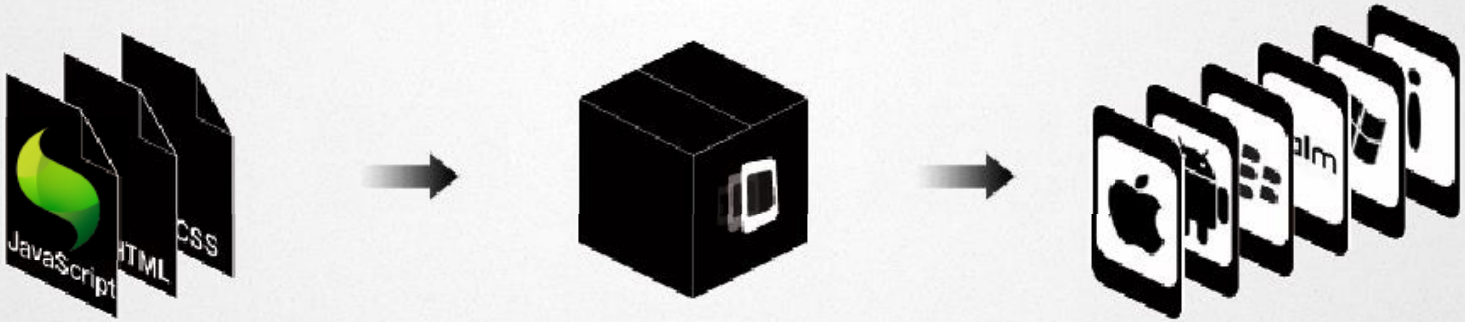
http://mysite.com/myimage.png

http://src.sencha.io/http://mysite.com/myimage.png

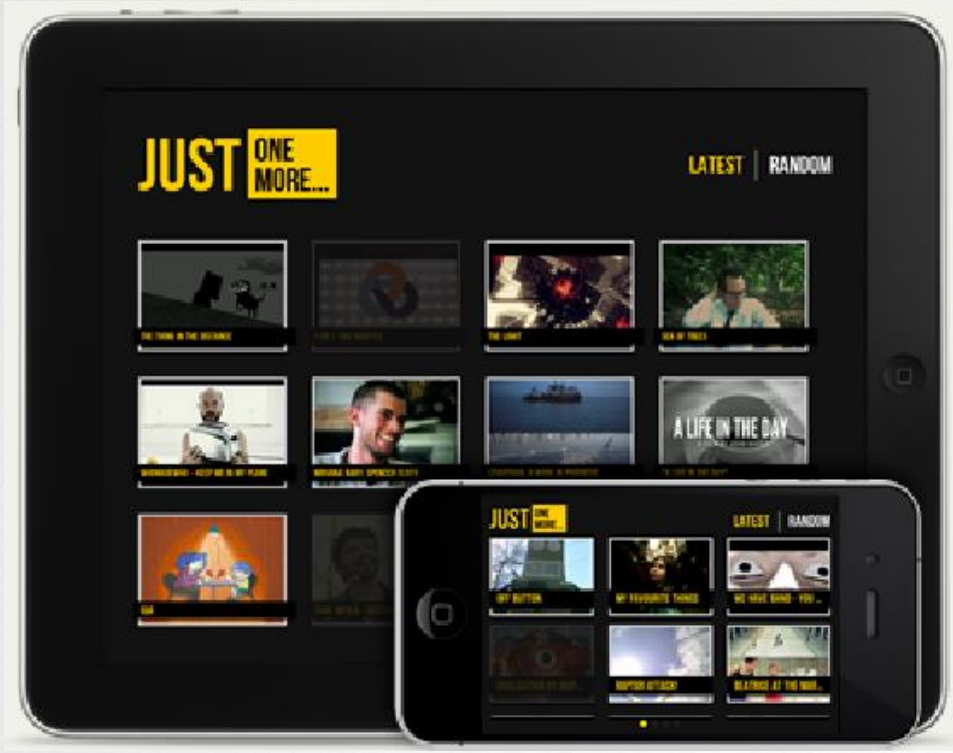


PhoneGap

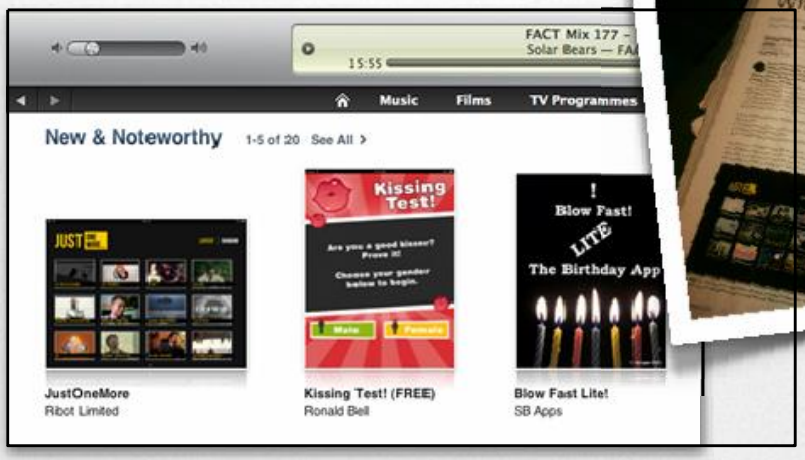
A platform that allows you to package web applications
and get access to device APIs



http://phonegap.com


JUST ONE MORE...
LATEST | RANDOM
A LIFE IN THE DAY

+


FACT Mix 177 –
Solar Bears — FA
15:55
Music
Films
TV Programmes
New & Noteworthy
1-5 of 20 See All >
JustOneMore
Ribot Limited
Kissing Test!
Kissing Test! (FREE)
Ronald Bell
Blow Fast!
LITE
The Birthday App
Blow Fast Lite!
SB Apps

build an app
with Sencha Touch

built with

Apps ~~vs~~ Web technology

EXTJS 4.2

JavaScript MVC Framework

Sencha Products

- **Ext JS** is a framework for web application based on JavaScript and Html5.
- **Sencha Touch** is a high-performance HTML5 mobile application framework
- **Sencha GXT** is the fastest, most powerful way to create rich web-based applications using Java.

Ext JS History

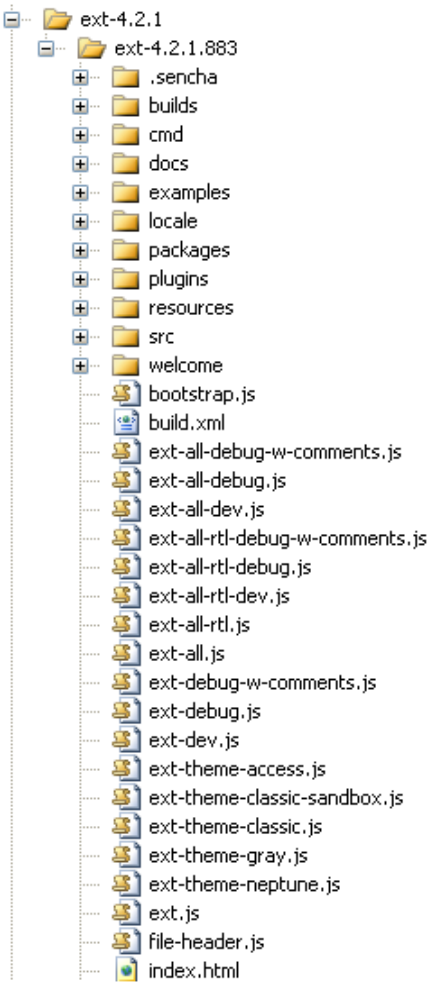
- Ext JS is a client-side, JavaScript framework for building web applications.
- In early 2006, Jack Slocum (jackslocum.com) started working on a set of extension utilities for the Yahoo User Interface (YUI) library under the name "yui-ext.“
- Eventually, Jack created an adapter layer through which any foundation library could be plugged in to provide utilities such as event handling, DOM manipulation and animated effects.
-

Ext JS History

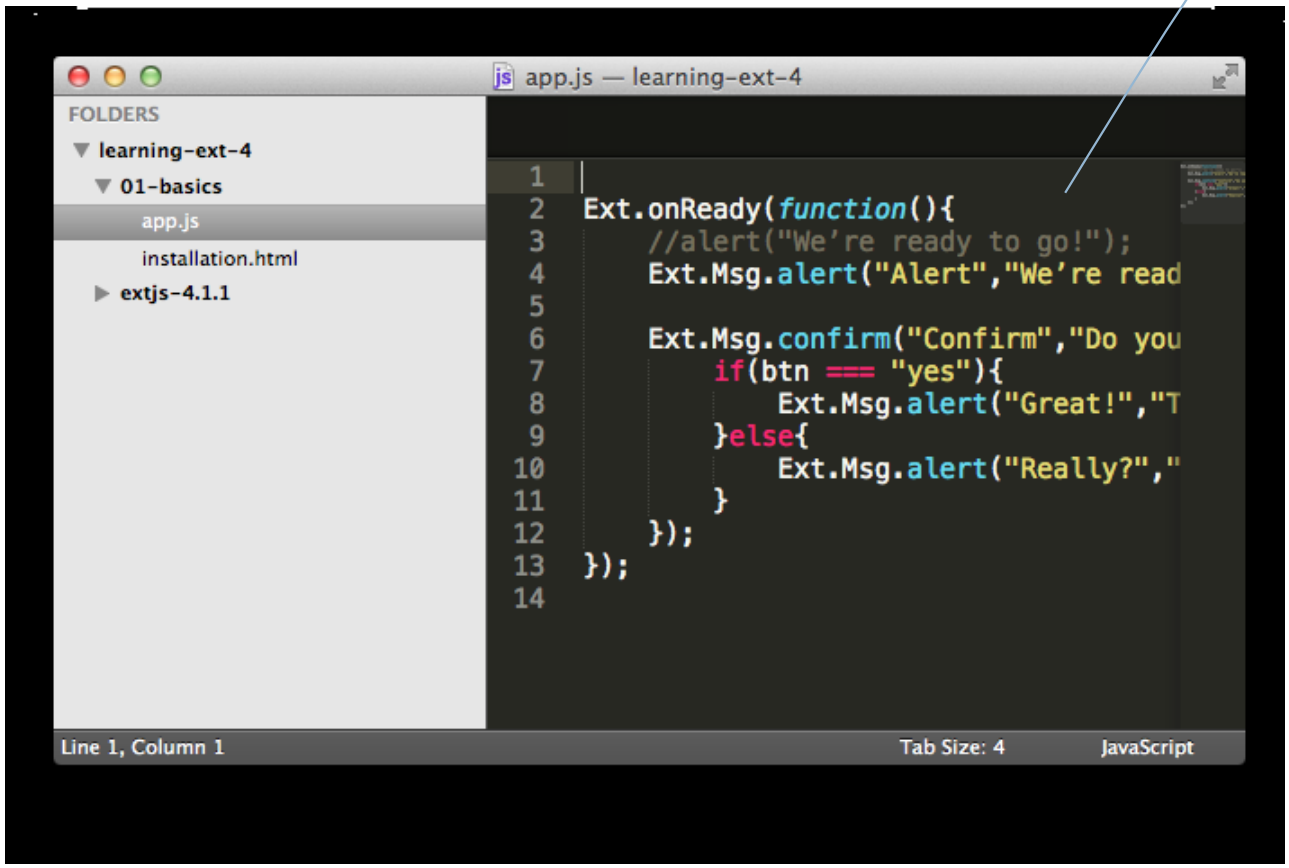
- This unanchored the library from YUI, and allowed the use of YUI, jQuery, prototype, or a new ext-base foundation written by Jack. This was version 1 of Ext JS. Current version of Ext JS is 4.2.x.
-

Setup ExtJS development environment:

- Download GPL version of Sencha ExtJS library from sencha.com to learn ExtJS 4.

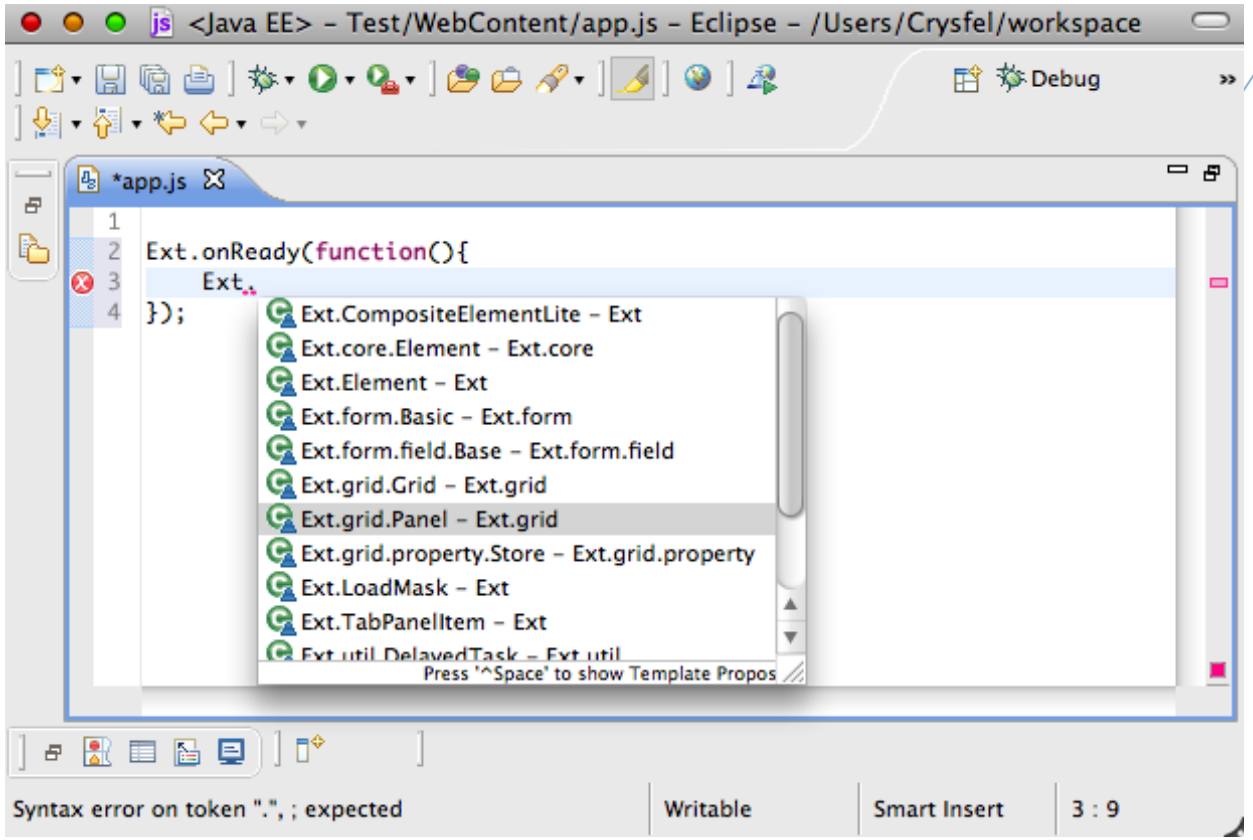


Editors



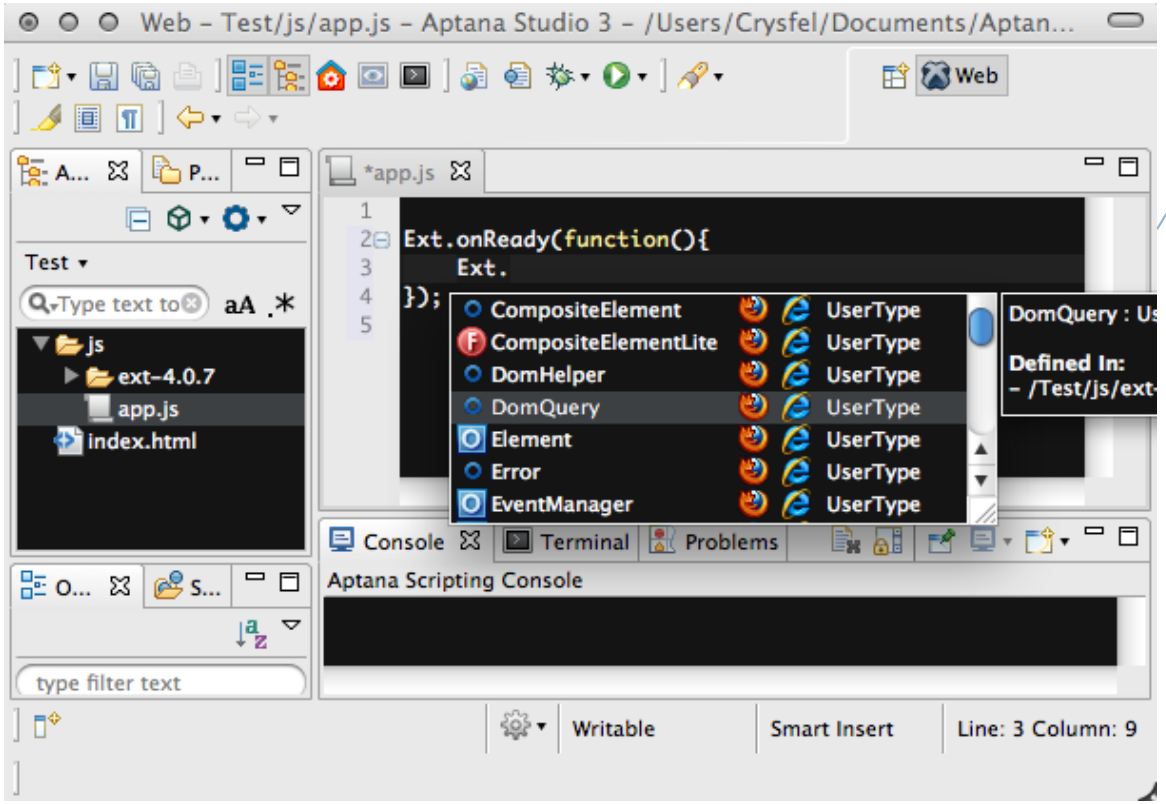
Sub lime

Editors



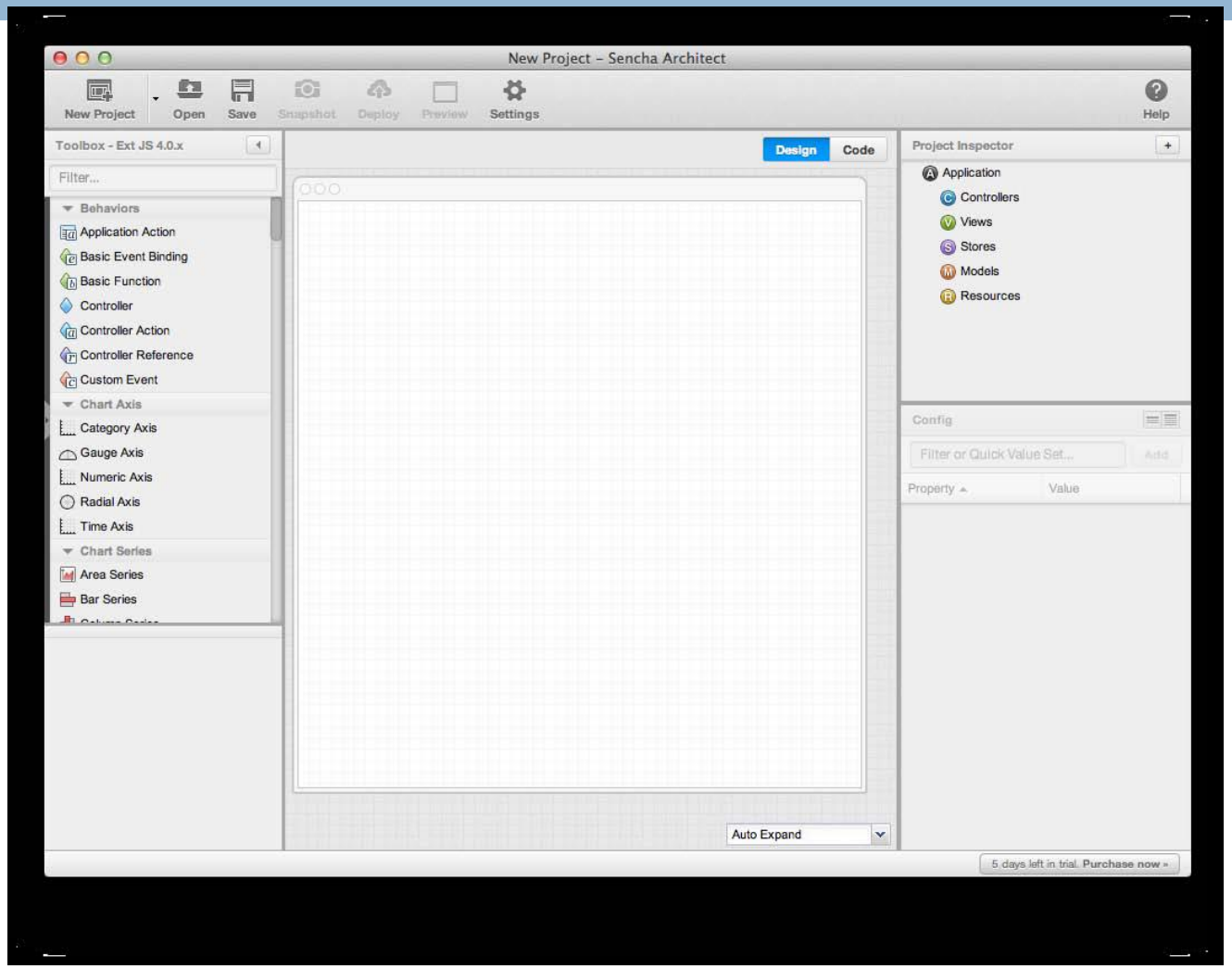
Eclipse

Editors

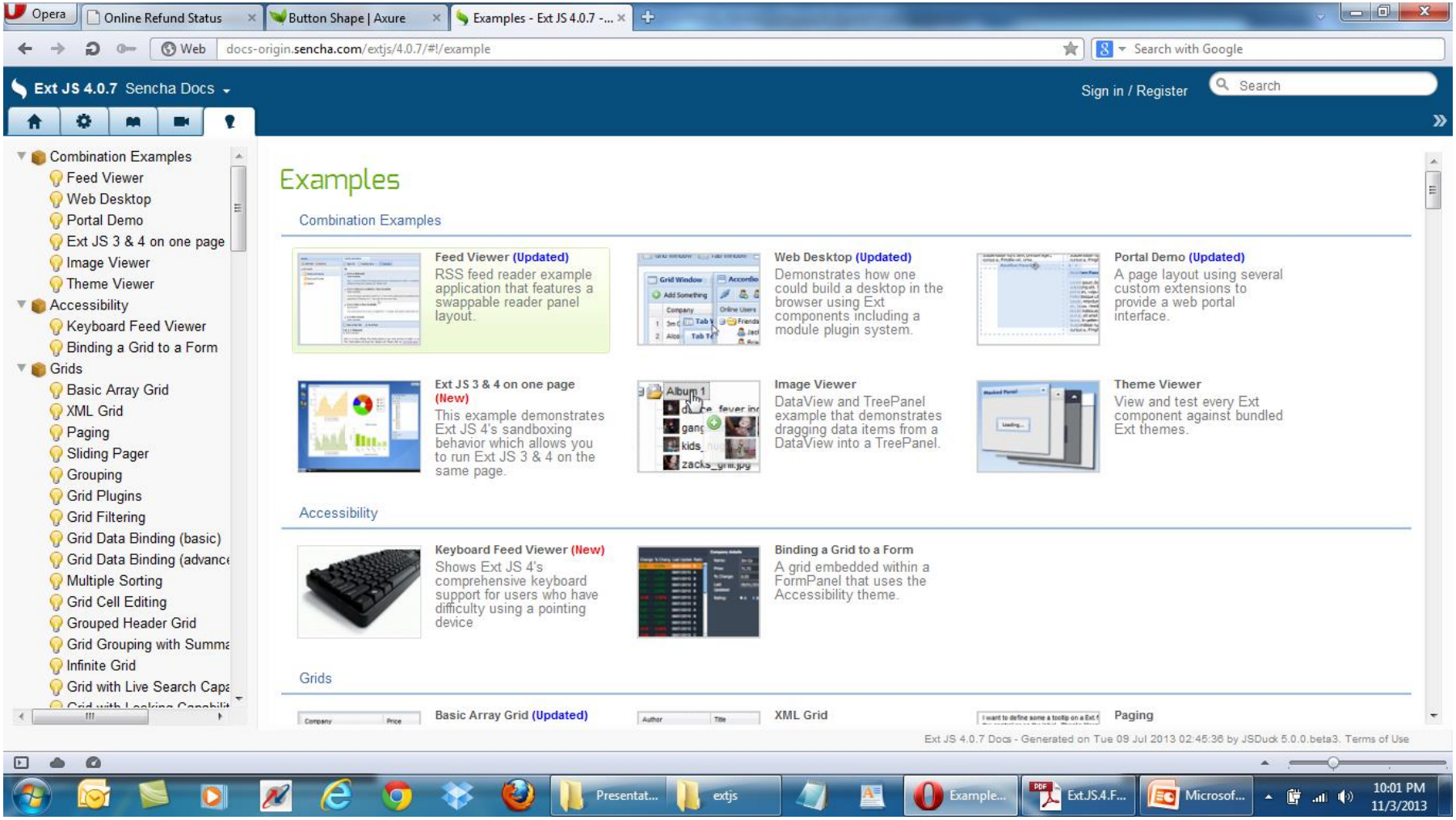


Aptana

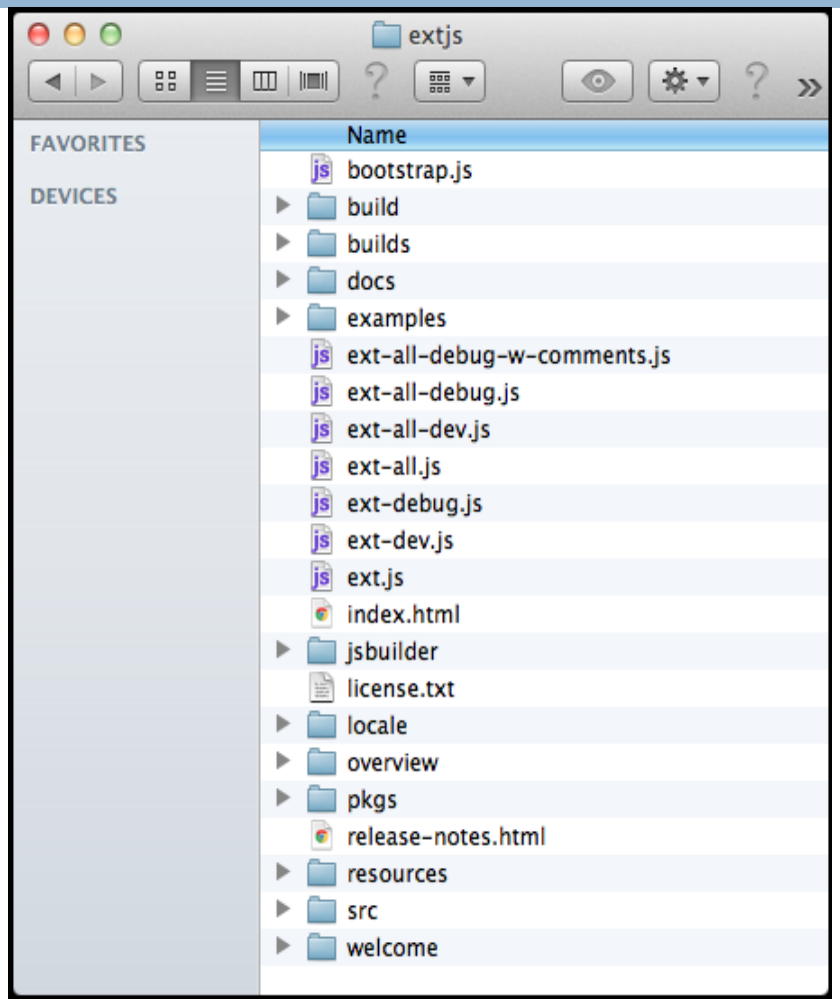
Sencha Architect



Package and namespace updates



Ext JS 4 SDK quick look



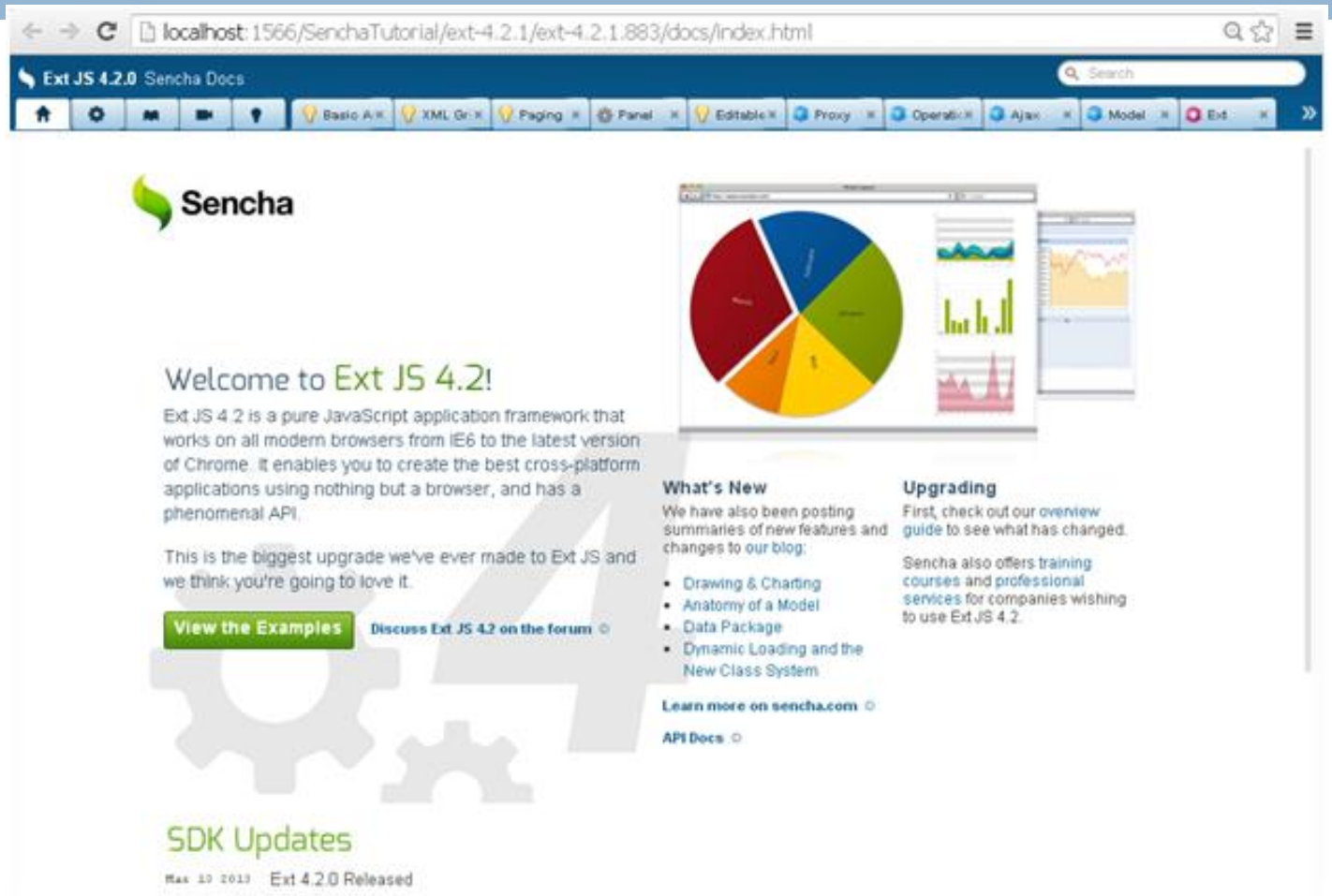
What is the difference between ext.js and ext-all.js?

- If we choose a version of the **ext-all file, the browser will load the entire Ext JS framework.**
- **If we choose a version of the ext.js file, the browser will load the minimum code required to execute the application, and we can make use of the new dynamic loading feature.**

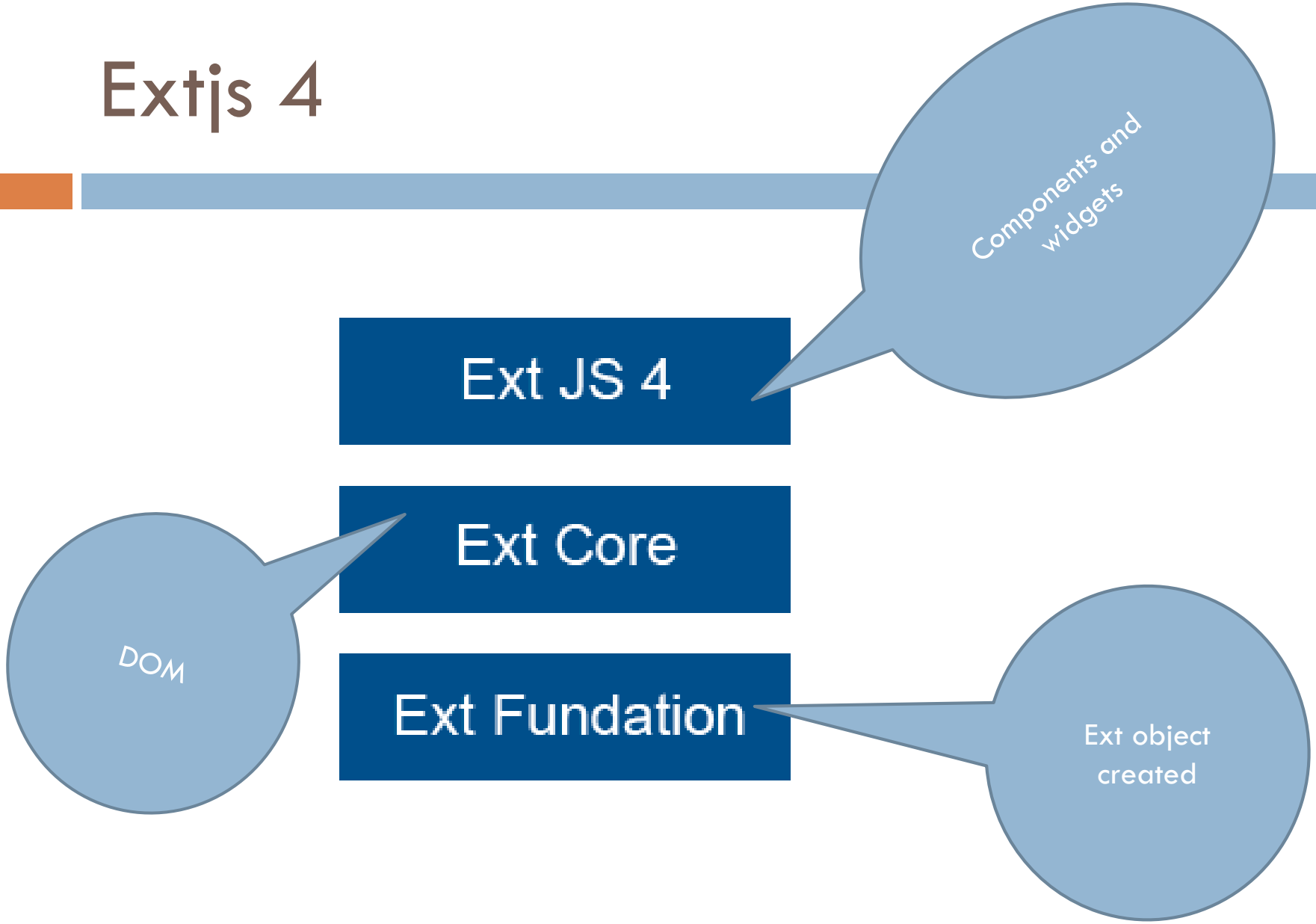
Deploying Ext JS locally

- Windows: C:\Program Files\Apache Software Foundation\Apache2.2\ htdocs
- Linux: /var/www/
- Mac OS X: /Library/WebServer/Documents/

ExtJS 4 Help Document:



Extjs 4


Components and widgets
Ext JS 4
Ext Core
DOM
Ext Fundation
Ext object created

Extjs Files and Folders

- build folder contains the descriptor files to create a custom version of the Ext JS library.
- The builds folder contains the minified version of the library;
- we can find the foundation, the core, and the Ext JS sandboxed version of the library.
- The docs folder contains documentation of the API.
- The examples folder contains a lot of examples of the components, layouts, and small applications that are built to show what we can do with the library

Extjs Files and Folders

- The locale folder has the translations of 45 languages.
- The jsbuilder folder contains the tool to build and compress our source code;
- The src folder contains all the classes of the framework.
- The resources folder is where the styles and images are located; also we can
- find the Sass files to create our custom theme in here.

Extjs Files and Folders

- The ext-all.js file is the complete library with all the components, utilities, and classes.
- The ext-all-debug.js file is the same as the ext-all.js file, but it is not minified so we can use this file to debug our application.
- The ext-all-dev.js file is similar to the ext-all-debug.js file but contains additional code to show more specific errors and warnings at development time

Extjs Files and Folders

- The ext.js file is the core and foundation layer for Ext JS.
- The ext-debug.js and ext-dev.js files follow the same concept as mentioned with the ext-all files.

ExtJS Naming convention

- ExtJS 4 or Sencha Touch is a JavaScript framework. So you should follow most of the JavaScript code & naming conventions.

Convention for	Description	Example
Class	Class name should be in CamelCased	MyCustomClass
	Class name should contain alphanumeric characters. Numbers are permitted if it belongs to technical term.	Base64

ExtJS Naming convention

	Do not use underscores, hyphens, or any other nonalphanumeric character.	
	Class name should be with atleast one unique namespace separated by dot (.).	TopLevelNamespace.MyClassName
	Classes that are not distributed by Sencha should never use Ext as the top-level namespace.	
	The top-level namespaces and the actual class names should be in CamelCased, everything else should be all lower-cased.	TopNamespace.middnamespace.CustomClass

ExtJS Naming convention

Source File	The names of the classes map directly to the file paths in which they are stored. As a result, there must only be one class per file.	Ext.util.Observable is stored in path/to/src/Ext/util/Observable.js MyTopLevelNamespace.util.JsonToXml is tored in path/to/src/MyTopLevelNamespace/util/ JsonToXml.js
Methods	Method should always be in camelCased.	encodeUsingMd5(), getHtml()
Variables	Variable should always be in camelCased.	var isGoodName var base64Encoder

ExtJS Naming convention

	Constant variable should be in upper-case	var SALARY = 1000
	Private variable should start with underscore '_'	var _privateVariable
Properties	Properties should always be in camelCased. Static properties should bein upper-case.	Ext.MessageBox.YES = "Yes"

What's New in Ext JS 4?

- How to get started with Ext JS 4
- How the Ext JS platform is now organized
- The new Ext JS 4 class system
- Ext JS 4 SDK
- Ext JS 3 versus Ext JS 4 compatibility
- Migrating from Ext JS 3 to Ext JS 4
- A quick overview of new components

Getting started with Ext JS 4

- Ext JS is a cross-browser RIA (Rich Internet Application) framework.
- The change from Ext JS 1.x to Ext JS 2.x was a major refactoring, including the Component model creation, along with refactoring of many of the existing components.
- The Ext JS 3.x is backward-compatible with Ext JS 2.x.
- .

Getting started with Ext JS 4

- Until Ext JS 3, the layout was what expanded most of the time while rendering the application.
- Ext JS 4 has a vast improvement in this area. The generated HTML was also updated in branch 4.x.

ExtJS 4 Class System

- ExtJS 4 ships with more than 300 classes.
- **Ext** is a global singleton object which you would use it almost everywhere.
- **Ext** class has 78 methods and 59 properties.
- Some of the important methods that will be used frequently are application, apply, create, define, decode, destroy, getCmp, override etc.

ExtJS 4 Class System

- Ext
  - app
  - button
  - chart
  - container
  - data
  - dd
  - direct
  - dom
  - draw
  - enums
  - flash
  - form
  - fx
  - grid
  - layout
  - menu
  - panel
  - picker
  - resizer
  - selection
  - slider
  - state
  - tab
  - tip
  - toolbar
  - tree
  - util
  - ux
  - view

The new Ext JS 4 class system

- Class definition and creation
- Mixins
- Automatic getters and setters
- Dynamic class loading
- Statics

ExtJS 4 Class System

Class	Description
Ext	The Ext namespace (global object) encapsulates all classes, singletons, and utility methods provided by Sencha libraries.
Ext.Base	The root of all classes created with Ext.define. Ext.Base is the building block of all Ext classes. All classes in Ext inherit from Ext.Base. All prototype and static members of this class are inherited by all other classes.
Ext.Class	Handles class creation throughout the framework. This is a low level factory that is used by Ext.ClassManagerand generally should not be used directly.
Ext.ClassManager	Ext.ClassManager manages all classes and handles mapping from string class name to actual class objects throughout the whole framework.
Ext.Loader	Ext.Loader is the heart of the new dynamic dependency loading capability in Ext JS 4+. It is most commonly used via the Ext.require shorthand.

Define New Class:

- Use Ext.define() method to define new custom class in ExtJS 4.
- **Basic Syntax:**
- *Ext.define((String) className, (Object) classMembers, (Optional Function) onClassCreatedCallback);*
- *className* is the full class name in dot-namespaced format

Define New Class:

- *className* is the full class name in dot-namespaced format
- *classMembers* is an object represents a collection of class members in key-value pairs (Javascript object literals)
- *onClassCreatedCallback* is an optional function callback to be invoked when all dependencies of this class are ready, and the class itself is fully created.

Define New Class:

- Ext.define() performs the following operations:
- Detects and creates new namespaces if required.
- Extends an existing class.
- Verifies if the class being extended has been defined, if not, it will defer the creation of the new class until the class used for extending is available.
- Returns Ext.Base object.

Define New Class:

- Ext.Base is root of all classes created with Ext.define.
- All the classes in ExtJS 4 inherit fromExt.Base

Define New Class:

- Ext.define
- ('Student', { name : 'unnamed', getName : function(){ return "Student name is" + this.name; } }, function(){ alert('Student object created'); });

Create an Object of a class:

- var studentObj = Ext.create('Student');
  studentObj.getName();

- var studentObj = new Student();
  studentObj.getName();

- var studentObj = Student.create();
  studentObj.getName();

Singleton Class

- By definition a “singleton” class is the kind of class where you cannot have more than one instance of it.
- In Ext JS 4 to define a class of this kind we only have to add the property “singleton” equal to true.
- Instance doesn’t need to instantiated, because we have access to the members directly.
- If for some reason we wanted to create an instance of this class we would receive an error when trying.

Singleton Class

```
//'Bleext.desktop.Constants' is a singleton and cannot be instantiated
Ext.create("Bleext.desktop.Constants");
```

Normally the singleton class is used to handle the “Constants” of an application or any other circumstance which requires a single instance of a class.

Alias

- An alias is a short name for a class.
- The class manager maps the alias name with the actual class object.
- This feature is really useful when using the xtype property to create widgets.

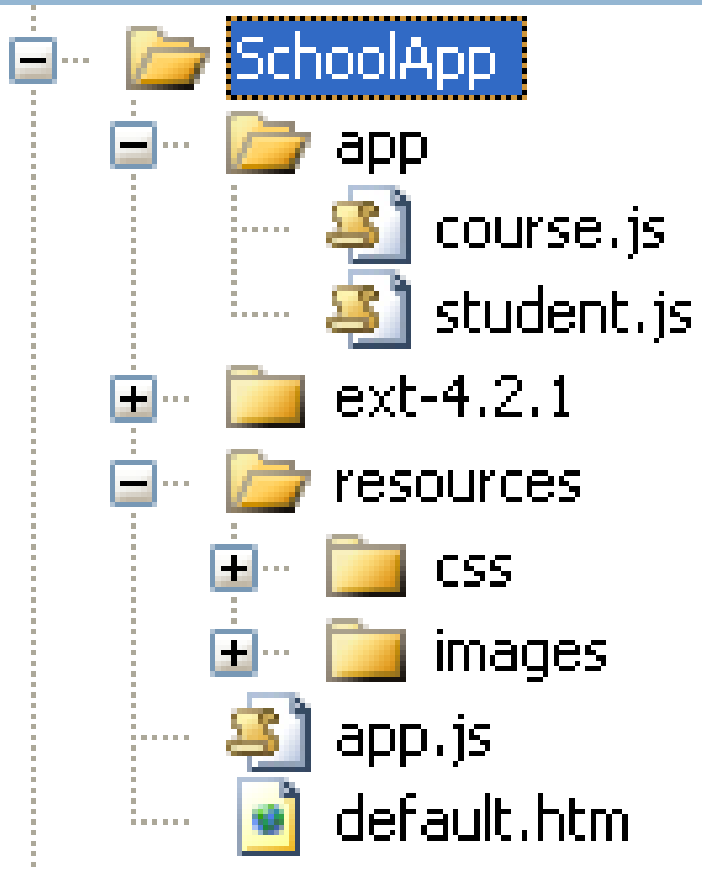
Class definition and creation

- Ext.define("MyApp.Client",{ //Step 1
- name : "John", //Step 2
- lastName : "Doe",
- constructor : function(config){ //Step 3
- Ext.apply(this,config || {}); //Step 4
- console.log("I'm a new client!");
- }
- });

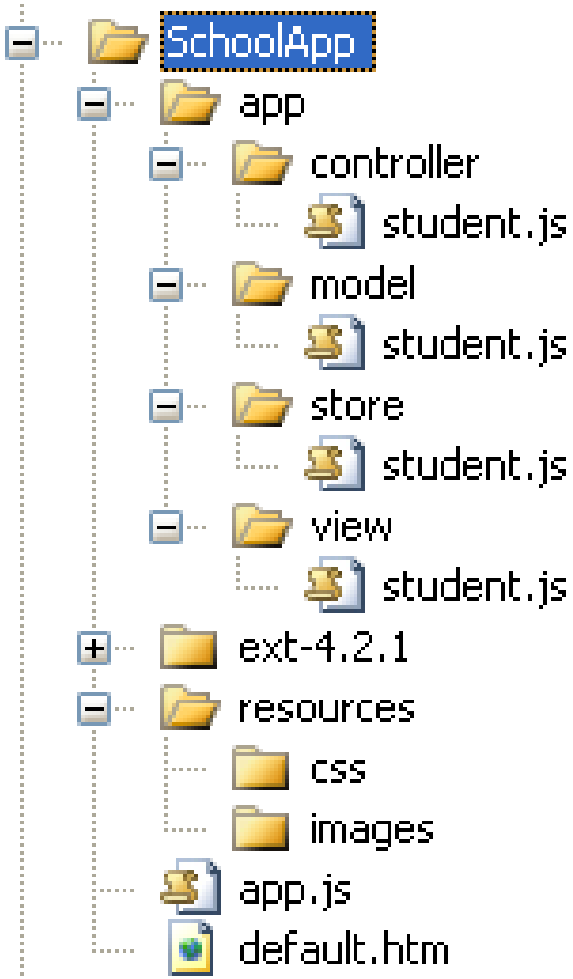
Class definition and creation

- Once we have defined our class we can create instances by using the Ext.create
- method as follows:
- //creating an instance of Client
- var client = Ext.create("MyApp.Client");
- console.log(client.name); //Print the default value: John

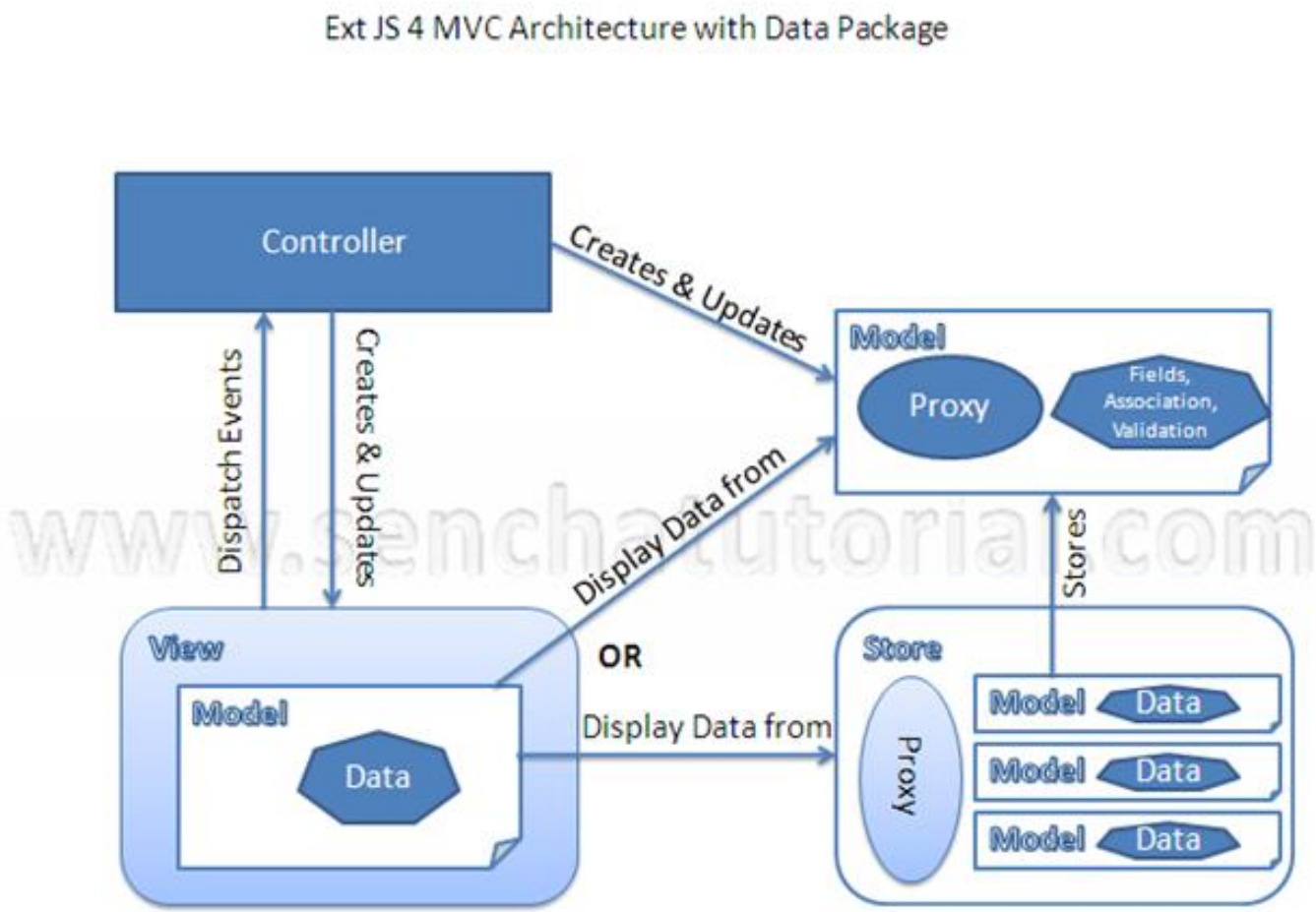
ExtJS 4 Application Structure:



ExtJS 4 MVC pattern:



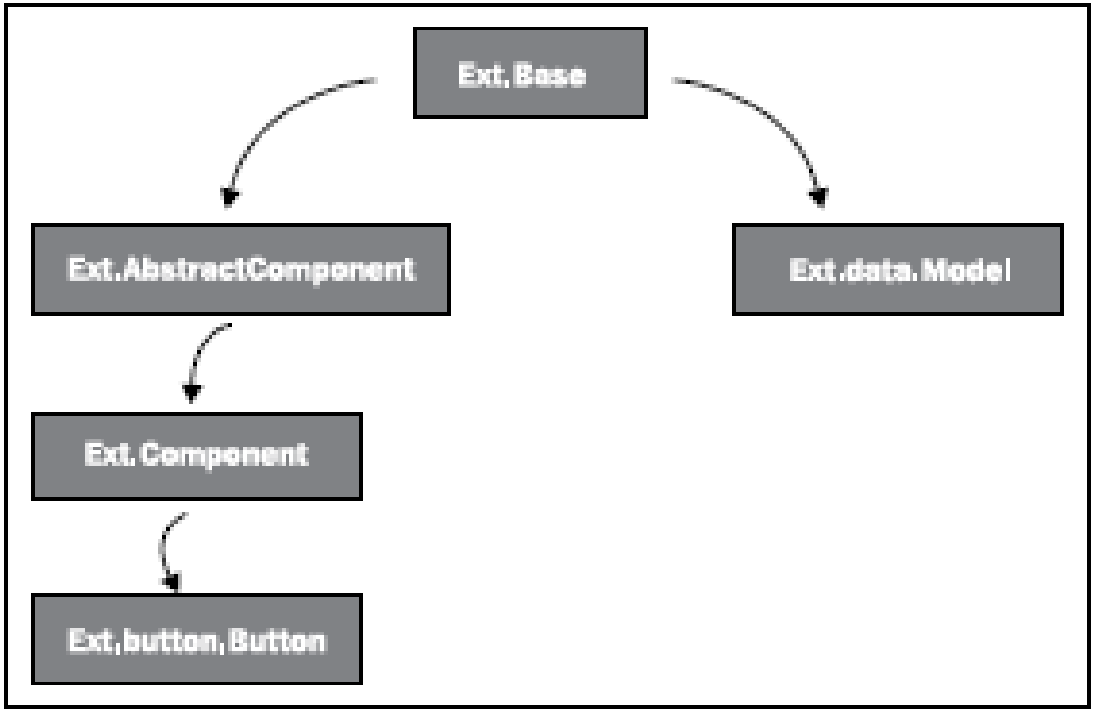
MVC Architecture



Simple inheritance

- When we create a class using the Ext.define method we're extending from the Ext.Base class.
- This class contains abstract methods that will be inherited to all the subclasses so we can use them to our convenience.

Simple inheritance



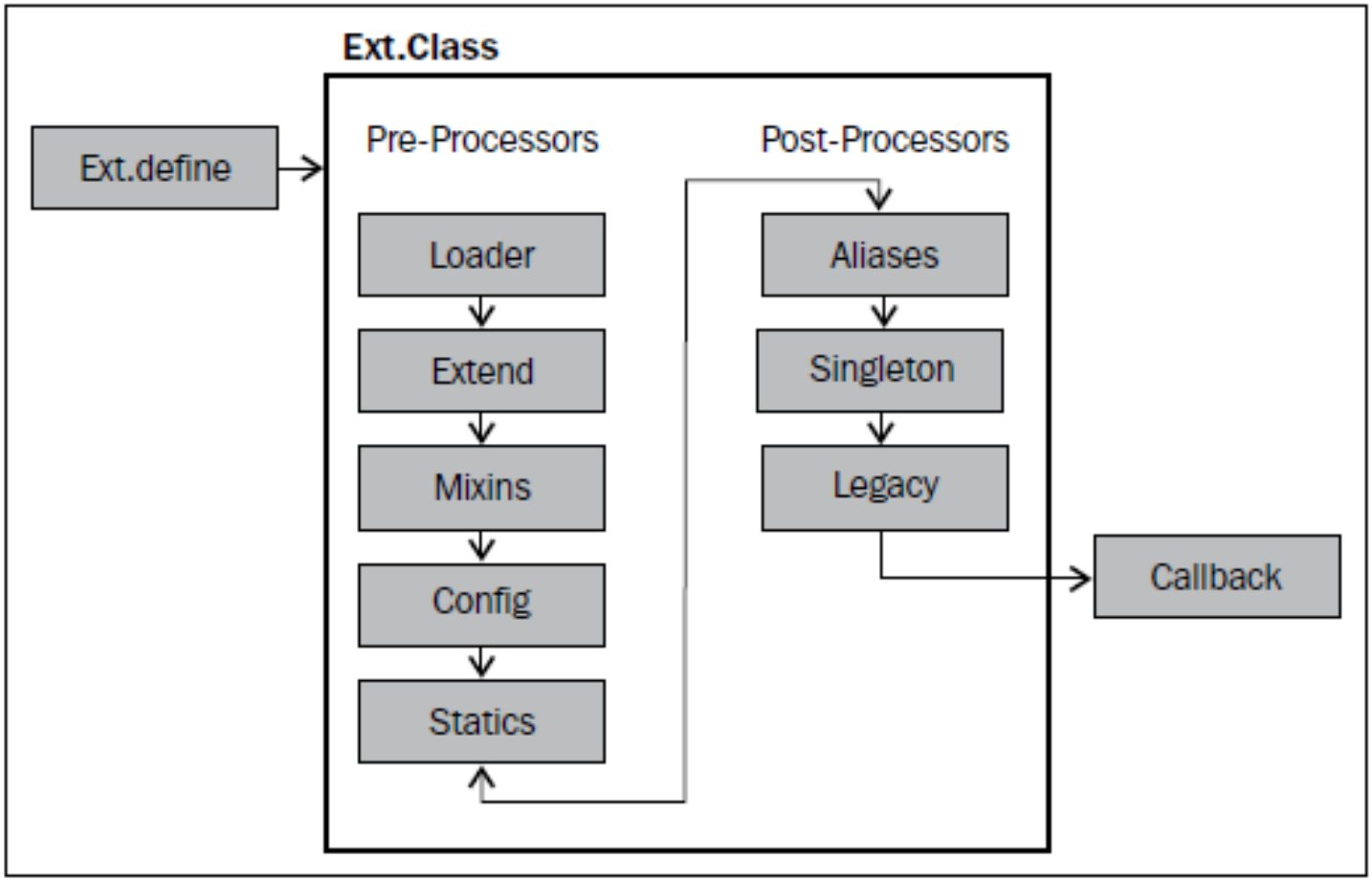
Class Processors:

- The Sencha Class System is built on top of processors, which are divided into two groups: pre-processors and post-processors.
- PreProcessors are simply a set of “hooks” that run before that class is actually created and ready to be used.
- Extend, mixins, config, requires, statics are the example of pre-processors.
- Similarly, class "post-processors" are hooks that execute as soon as the class is ready. Singleton, alias are example of post-processors.

Pre-processors and post-processors

- A **post-processor is a process that runs after our new class is created.**
- **There is a** process to make our class as singleton, to define alternative names to our class, and a few other processes.

Pre-processors and post-processors



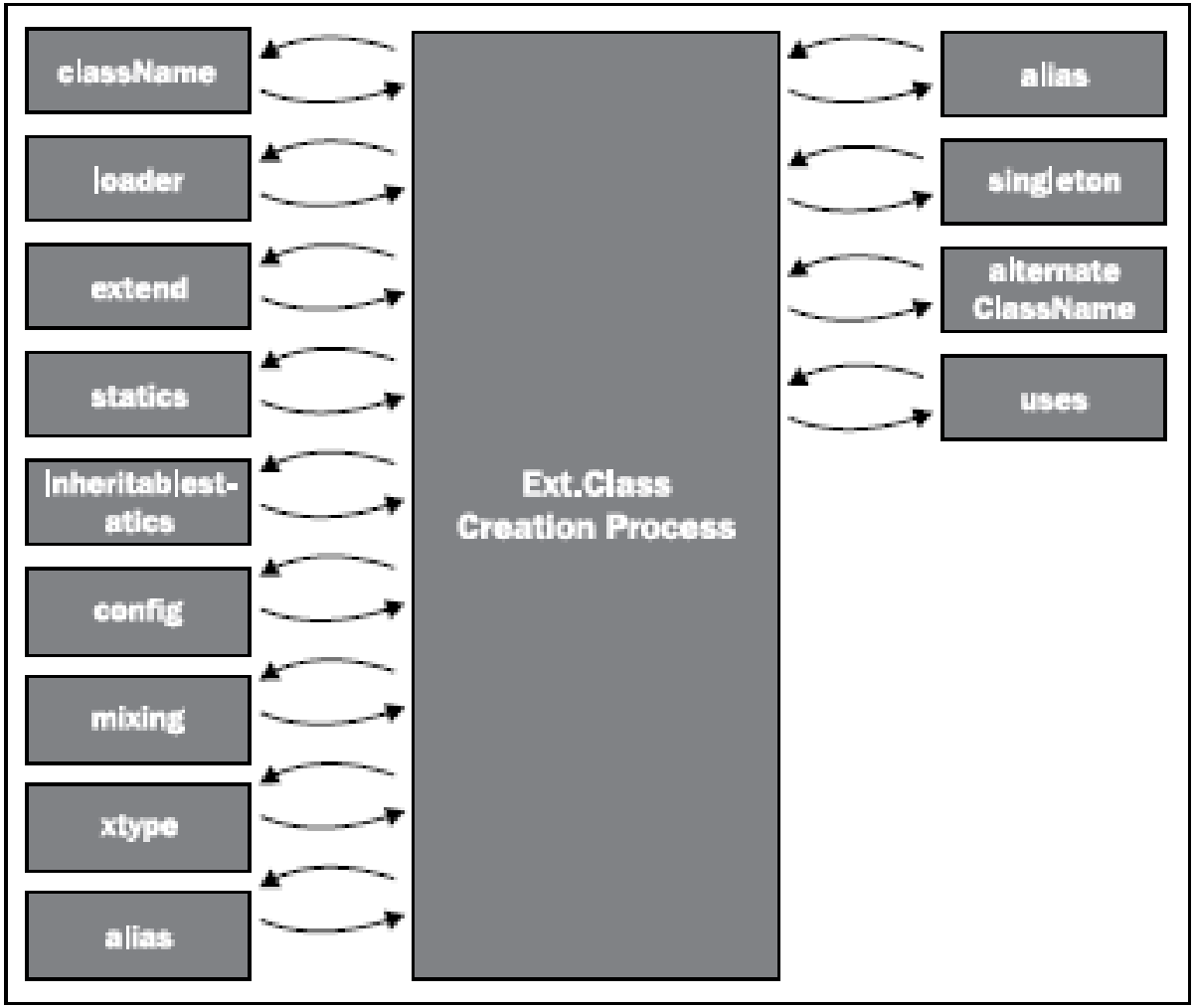
Pre-processors and post-processors

- Every class in Ext JS is an instance of the Ext.Class class.
- When we use the Ext.define method to define a class we are in fact creating an instance of the Ext.Class class.

Pre-processors and post-processors

- A **pre-processor is a process that runs before the instance of an Ext.Class class is** created, or in other words, before our new class is created.
- Each of the processes defined will change the behavior of our class, if needed.

Pre-processors and post-processors



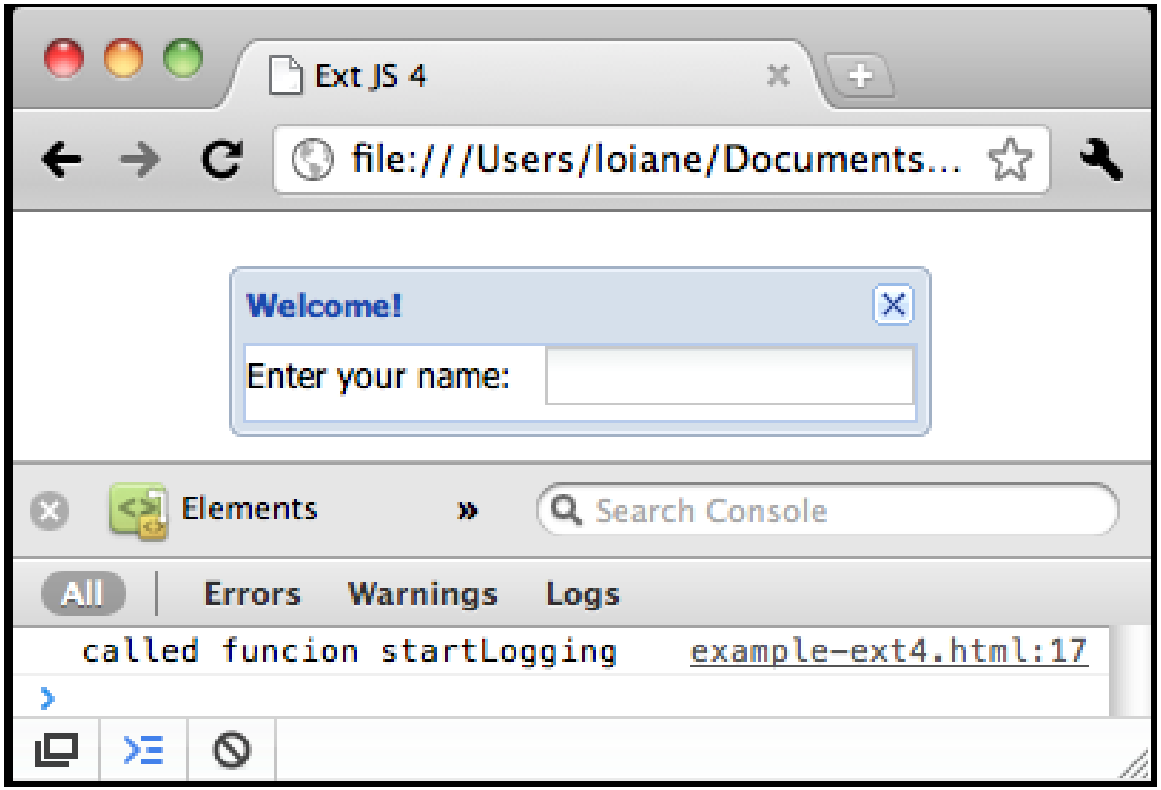
Mixins

- The mixins configuration is a brand new concept for Ext JS.
- Mixins define reusable sets of behavior and configuration that can be 'mixed in' to a class.
- In other words, it allows merging new capabilities (functions or properties) to the class prototype.

Mixins

- Ext.define('MyApp.MyWindow', {
- extend: 'Ext.Window',
- mixins: {
- console: 'MyApp.mixins.Log'
- },
- ...
- });

Mixins


Ext JS 4
file:///Users/loiane/Documents...
Welcome!
Enter your name:
Elements
Search Console
All
Errors
Warnings
Logs
called funcion startLogging
example-ext4.html:17

Config (auto setters and getters)

- When you configure properties in the config declaration, Ext JS 4 will automatically generate four methods: getter, setter, reset, and apply.

Config (auto setters and getters)

- Let's apply the config declaration on the MyApp.MyWindow class:
- Ext.define('MyApp.MyWindow', {
- extend: 'Ext.Window',
- config: {
- title: 'Welcome!'
- }
- });

Dynamic class loading

- We can ask Ext JS to load the classes that we need to use and then call a function when it is finished loading. For example:
- Ext.require('Ext.Window', function() {
- var win = new Ext.Window({
- title : 'Hello!',
- width : 100, height: 50
- });
- win.show();
- });

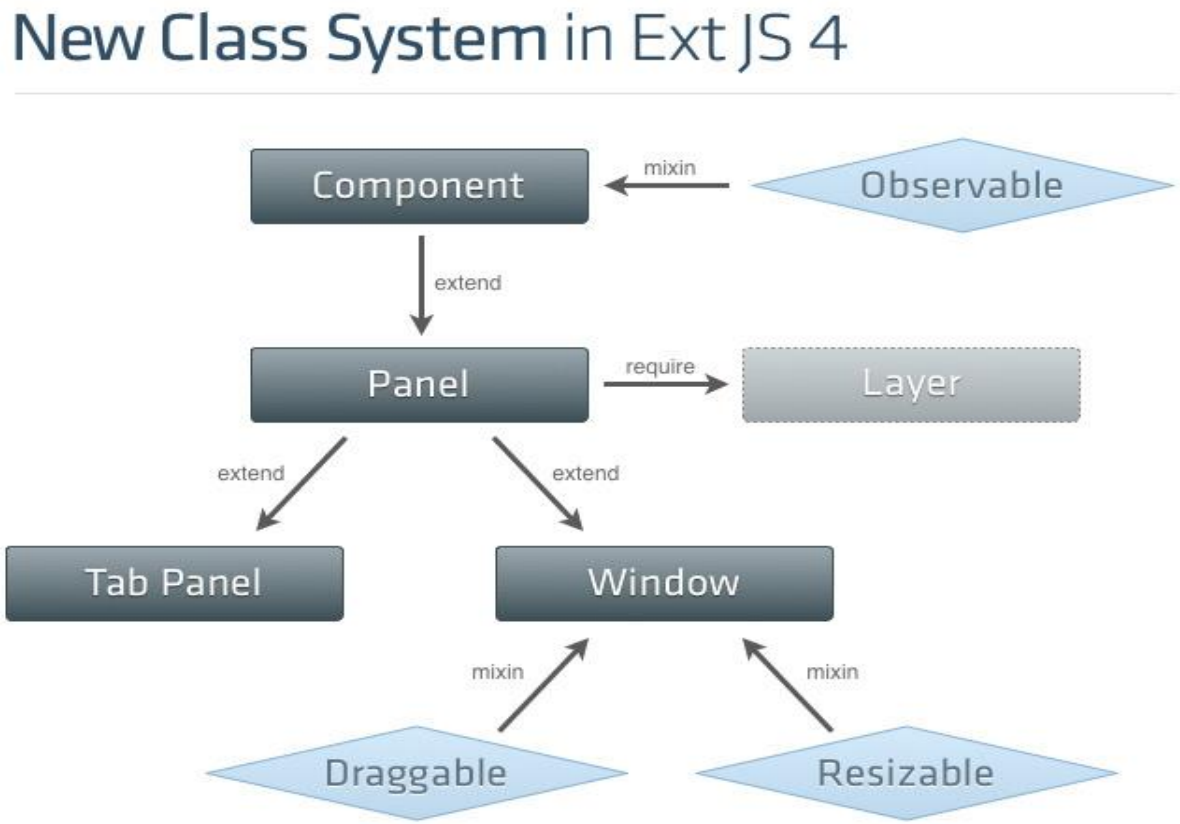
Dynamic class loading

- When we use Ext.require, we are telling Ext JS we need Ext.Window before calling the function.
- The framework will also resolve any dependencies that the loaded class has.

Dynamic class loading

- Requires: This declares the class dependencies required for a class to work. These classes are going to be loaded before the current class gets instantiated.
- Uses: This declares the optional class dependencies, but is not required. These classes do not have to be available before the current class gets instantiated.

Dynamic class loading



Statics

- In Ext JS 4, any class can define static methods, which means you do not need to instantiate the class to call the method; you can call ClassName.methodName().
- To declare a static method or property, simply define it as statics in its class property.

Statics

- Ext.define('MyApp.Math', {
- statics: {
- count: 0,
- appName: 'Math',
- sum: function(number1, number2) {
- return number1 + number2; }
- **MyApp.Math.count;**
- **MyApp.Math.appName;}**

Understanding the Ext Core

- Ext Core is a lightweight and feature rich core JavaScript library available under an MIT license.
- Ext Core is a cross-browser JavaScript library for building dynamic web pages.
- The Ext Core JavaScript library provides basic cross-browser abstractions for DOM querying, element selection and more.

Flyweight Design Pattern

- The Flyweight Design Pattern is used to minimize memory usage by creating a single global object and then re-using that object over and over again.
- Ext creates a global Ext.Element to use as the Flyweight when it starts up.
- This global flyweight instance can then be used to target any node in the Dom.
- To access this flyweight object you can use the Ext.fly method.

Browser Compatibility

- Ext Core supports all major web browsers including:
- Internet Explorer 6+
- FireFox 1.5+ (PC, Mac)
- Safari 3+
- Chrome 3+
- Opera 9+ (PC, Mac)

The Component model

- The Ext Component model is a centralized model that provides many of the essential Component-related tasks, which include a set of rules dictating how the Component instantiates, renders, and is destroyed, known as the Component lifecycle.

- All UI widgets are subclasses of Ext.Component, which means that all of the widgets conform to the rules dictated by the model.

ExtJS 4 Components:

Component Name	Class Name	xtype
TextField	Ext.form.field.Text	textfield
Label	Ext.form.Label	label
Button	Ext.button.Button	button
DateField	Ext.form.field.Date	datefield
ComboBox	Ext.form.field.ComboBox	combobox or combo
Radio Button	Ext.form.field.Radio	radio or radiofield
Checkbox	Ext.form.field.Checkbox	checkbox or checkboxfield
File Upload	Ext.form.field.File	filefield, fileuploadfield

ExtJS 4 Components:

Hidden Field	Ext.form.field.Hidden	hidden
Number Field	Ext.form.field.Number	numberfield
Spinner	Ext.form.field.Spinner	spinnerfield
Text Area	Ext.form.field.TextArea	textarea
Time Field	Ext.form.field.Time	timefield
Trigger	Ext.form.field.Trigger	triggerfield, trigger
Chart	Ext.chart.Chart	chart
Html Editor	Ext.form.field.HtmlEditor	htmleditor

ExtJS 4 Containers:

Class Name	xtype	Description
Ext.container.Viewport	viewport	ViewPort is a specialized container representing the viewable application area (the browser viewport). Generally there will be single ViewPort in an ExtJS application which will define application's main areas like North, west, south, east and center.
container	container	Container is light weight container which provides basic functionality of containing items, namely adding, inserting and removing items. So use this container where you just want to add other components and arrange them.
Ext.panel.Panel	panel	Panel is a special type of container that has specific functionality and components. Each panel has header, tool, body, toolbar components. So use Panel where you want specific user interface with header, toolbar and body part. Do not use it if you do not want these features in panel, use container instead.

ExtJS 4 Containers:

Ext.form.FieldContainer	fieldcontainer	FieldContainer implements labelable mixins so that it can be rendered with label and error message around every sub-item components. You can use fieldcontainer in the form to arrange inner fields.
Ext.form.FieldSet	fieldset	Fieldset is a container for group of fields in a Ext.form.Panel.
Ext.grid.Panel	grid	Grid displays large amount of tabular data with sorting, filtering and other functionality.
Ext.container.ButtonGroup	buttongroup	ButtonGroup provides a container for arranging a group of related Buttons in a tabular manner.
Ext.tab.Panel	tabpanel	Ext.tab.Panel is a basic tab container. Use it when you want tab or wizard in your user interface.
Ext.tree.Panel	treepanel	TreePanel provides tree-structured UI representation of tree-structured data.
Ext.menu.Menu	menu	Menu is a container to which you can add Menu Items.
Ext.toolbar.Toolbar	toolbar	Toolbar is a container for toolbar buttons, text, fill, item etc.

Layouts in ExtJS 4:

Layout	Description
Absolute	This layout inherits anchor layout and adds the ability of positioning the component at particular co-ordinates using standard x, y config options.
Accordion	This layout manages multiple panels in an expandable accordion style. By default one panel can be expanded at any given time, but you can also expand multiple panels using multi config.
Anchor	This layout enables anchoring of contained elements relative to the container's dimensions. Please visit Anchor Rules.
Auto	This layout is applicable when no layout is configured into a container. AutoLayout provides only a passthrough of any layout calls to any child containers.
Border	This is a multi-pane, application-oriented UI layout style that supports multiple nested panels, automatic bars between regions and built-in expanding and collapsing of regions.
Card	This layout manages multiple child Components, each fitted to the Container, where only a single child Component can be visible at any given time. This layout style is most commonly used for wizards and tab implementations.

Layouts in ExtJS 4:

Fit	This layout is used in a container that contains a single item that automatically expands to fill the container.
Form	This is a layout that will render form Fields, one under the other all stretched to the Container width.
HBox	A layout that arranges items horizontally across a Container. align, flex and pack are important config options for this layout.
Table	This layout allows you to easily render content into an HTML table.
VBox	A layout that arranges items vertically down a Container. align, flex and pack are important config options for this layout.

MessageBox:

- Ext.MessageBox or Ext.Msg is a singleton instance of Ext.window.MessageBox.
- You can display 4 types of message box:
- **Alert:** Displays a standard read-only message box with an OK button.
- **Confirm:** Displays a confirmation message box with Yes and No buttons.
- **Prompt:** Displays a message box with OK and Cancel buttons prompting the user to enter some text (similar comparable to JavaScript's prompt). It can be a single-line or multi-line textbox.
- **Custom:** Customized message box style.

MessageBox:

- Ext.Msg.alert('Status', 'This is First ExtJS 4 Alert.');

- //two consecutive messages, latest will be displayed
- Ext.Msg.alert('Status', 'This is First ExtJS 4 Alert.');
- Ext.Msg.alert('Status', 'This is Second ExtJS 4 Alert.');

MessageBox:

- var msg = Ext.create('Ext.window.MessageBox');
  msg.alert('Status', 'This is First ExtJS 4 Alert.');

Confirm:

- Ext.Msg.confirm("Confirmation", "Do you want to Save changes?", function(btnText){ if(btnText === "no"){ Ext.Msg.alert("Alert", "You have confirmed 'No'."); } else if(btnText === "yes"){ Ext.Msg.alert("Alert", "You have confirmed 'Yes'."); } }, this);

Prompt

- Ext.Msg.prompt("Sencha Tutorial", "Please enter your Sencha Id:", function(btnText, sInput){ if(btnText === 'ok'){ Ext.Msg.alert("Status", "You entered:" + sInput); } }, this);

Custom message

- Ext.Msg.show({ title : 'Save', msg : 'Do you want to Save the changes? ', width : 300, closable : false, buttons : Ext.Msg.YESNOCANCEL, buttonText : { yes : 'Yes & Continue', no : 'No & Continue', cancel : 'Discard' }, multiline : false, fn : function(buttonValue, inputText, showConfig){ Ext.Msg.alert('Status', buttonValue); }, icon : Ext.Msg.QUESTION });

Simple extension

```
Ext.define('MyClass', {
      extend: ’Ext.TabPanel’,

    constructor: function() {
            alert(”Look ma, I have tabs”);

        this.callParent();
      }
});
```

Class properties

The properties and methods you define for your class are added to the ***prototype*** of your class.

```
Ext.define('MyClass', {
      extend: ’Ext.TabPanel’,
    favoriteTab : 3,

    someFunction : function() { ... }
});
console.log(MyClass.prototype.favoriteTab); // => 3
```

Instantiating your class

```
var foo = Ext.create('MyClass', {
    // Config properties
});

// Or just use classic ’new’

var bar = new MyClass({
    // Config properties
});
```

Mixins

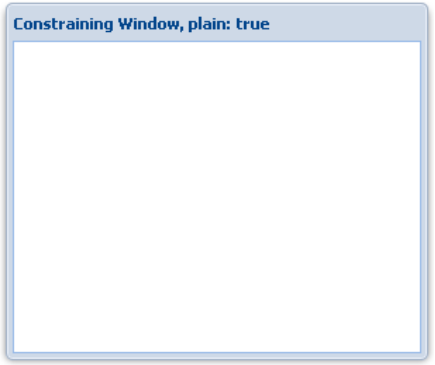
Mixins

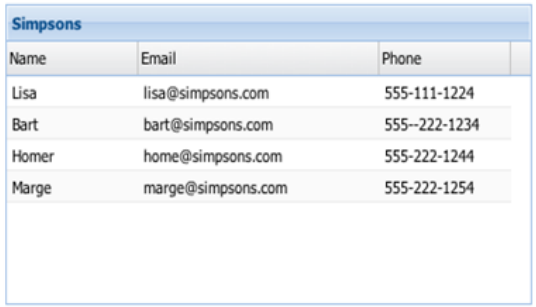
- Brand new concept when it comes to Ext JS 4. A ***mixin*** is just a set of functions (and sometimes properties) that are merged into a class.

- Mixins are really useful when a class needs to inherit multiple traits but can't do so easily using a traditional single inheritance mechanism

Mixins - example

- Ext.Window is a *draggable* component, as are Sliders, Grid headers, and many others





Mixins - example

- Because this behavior is desired in many different places it's not feasible to work the draggable behavior into a single superclass
- Creating a Draggable ***mixin*** solves this problem – anything can now be draggable very easily.

Mixins

```
// We can define some mixins at definition time
Ext.define('MyClass', {
      mixins: {
            observable: 'Ext.util.Observable'
      }
});

// It’s easy to add more later too
MyClass.mixin('draggable', 'Ext.util.Draggable');
```

Plugins

Plugins

- A ***plugin*** augments a single instance of an Ext Component. Any object with an ***init*** method.
- Used to add a feature to a component, for example adding cell-editing to a GridPanel.
- During its initialization phase, the host component calls the ***init*** method of all its plugins, and passes itself as the only argument

Defining a plugin

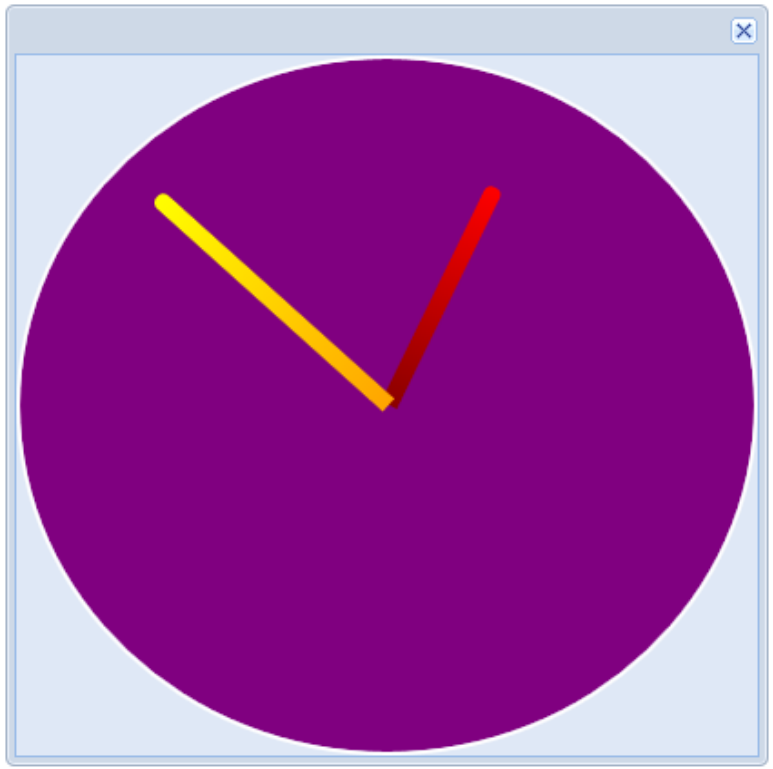
```
// Simplest possible plugin
var mySillyPlug = {
    init : function(host) { alert('Hello world'); }
};
```

Using a plugin

```
// Adding inline editing support to grid cells
Ext.create(’Ext.grid.Panel', {
      plugins: Ext.create('Ext.grid.plugin.CellEditing', {
            clicksToEdit: 1
      })
});
```

Let's create a simple extension

A simple CSS 3 clock component



Step 1. Identify suitable base class

HIERARCHY

- Ext.AbstractComponent
  - Ext.Component
    - Ext.container.AbstractContainer
      - Ext.container.Container
        - Ext.panel.AbstractPanel
          - Ext.panel.Panel
            - Ext.panel.Table
              - **Ext.grid.Panel**

Ext.Component: Base class for most popular Ext widgets

HIERARCHY

- Ext.AbstractComponent
  - Ext.Component
    - Ext.container.AbstractContainer
      - Ext.container.Container
        - Ext.panel.AbstractPanel
          - Ext.panel.Panel
            - Ext.panel.Table
              - **Ext.grid.Panel**

Simpsons

Name	Email	Phone
Lisa	lisa@simpsons.com	555-111-1224
Bart	bart@simpsons.com	555--222-1234
Homer	home@simpsons.com	555-222-1244
Marge	marge@simpsons.com	555-222-1254

HIERARCHY

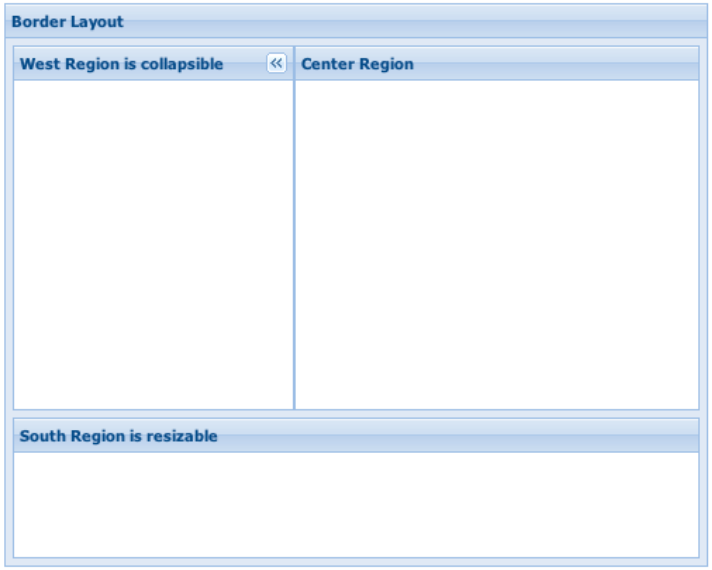
- Ext.AbstractComponent
  - Ext.Component
    - Ext.container.AbstractContainer
      - Ext.container.Container
        - Ext.panel.AbstractPanel
          - Ext.panel.Panel
            - Ext.panel.Table
              - **Ext.tree.Panel**

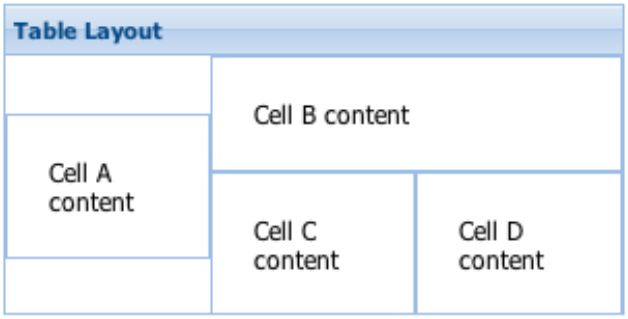
Simple Tree

- detention
- homework
  - book report
  - alegrbra
- buy lottery tickets

Ext.Component

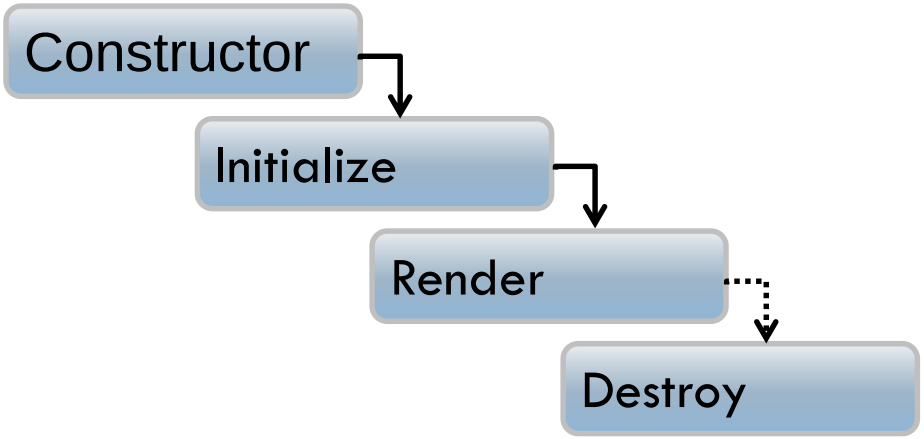
- Can be part of any layout structure as a child of a Container.





Ext.Component lifecycle

- All subclasses of Component has a managed lifecycle (*creation*, *rendering* and *destruction*) which is provided by the Container class.



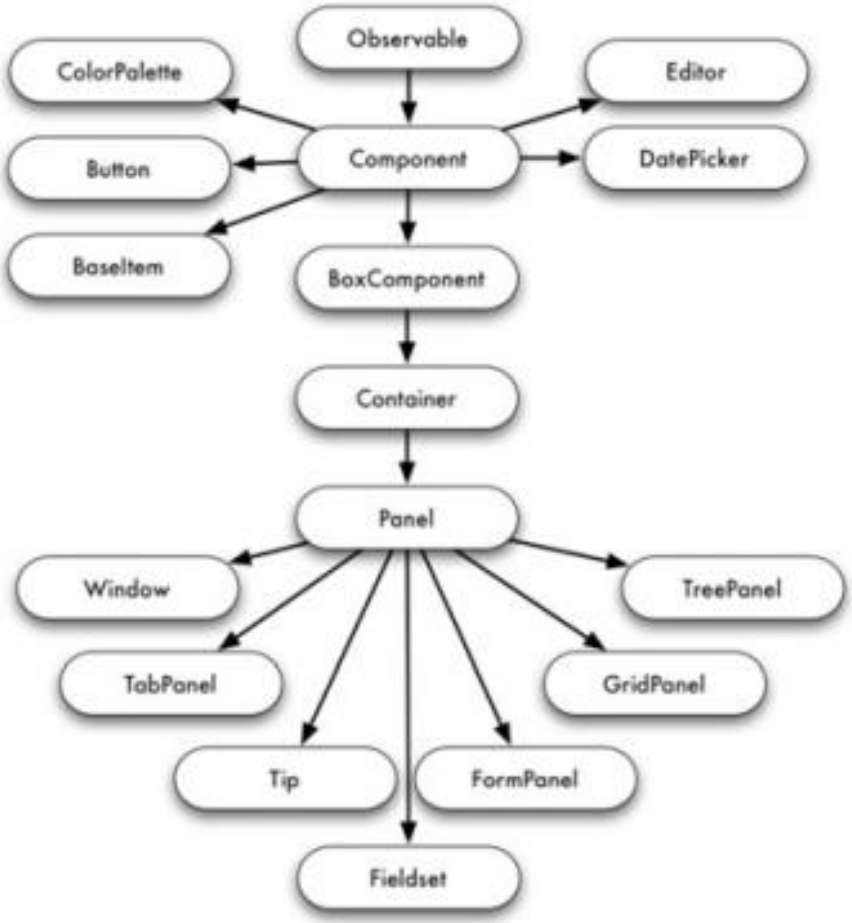
Ext.Component lifecycle and template methods

* Initialization (*constructor, initComponent*)
  - Configuration, setup etc...
* Rendering (*onRender, afterRender*)
  - Add additional elements and markup
* Layout (*afterLayout*)
  - Executed after the component has been laid out
* Destruction (*onDestroy*)
  - Clean up after yourself, destroy elements etc.

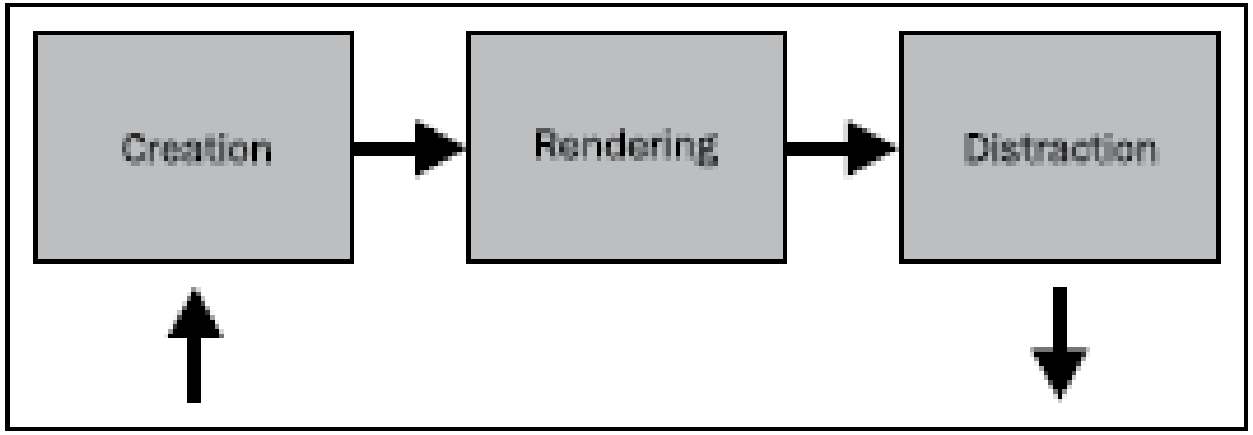
Ext.Container

- Base class for any Component that needs to contain other Components.
- Handles the basic behavior of containing items: *adding*, *inserting and removing* items.
- The most commonly used Container classes are **Panel, Window** and **TabPanel.**

The Component model

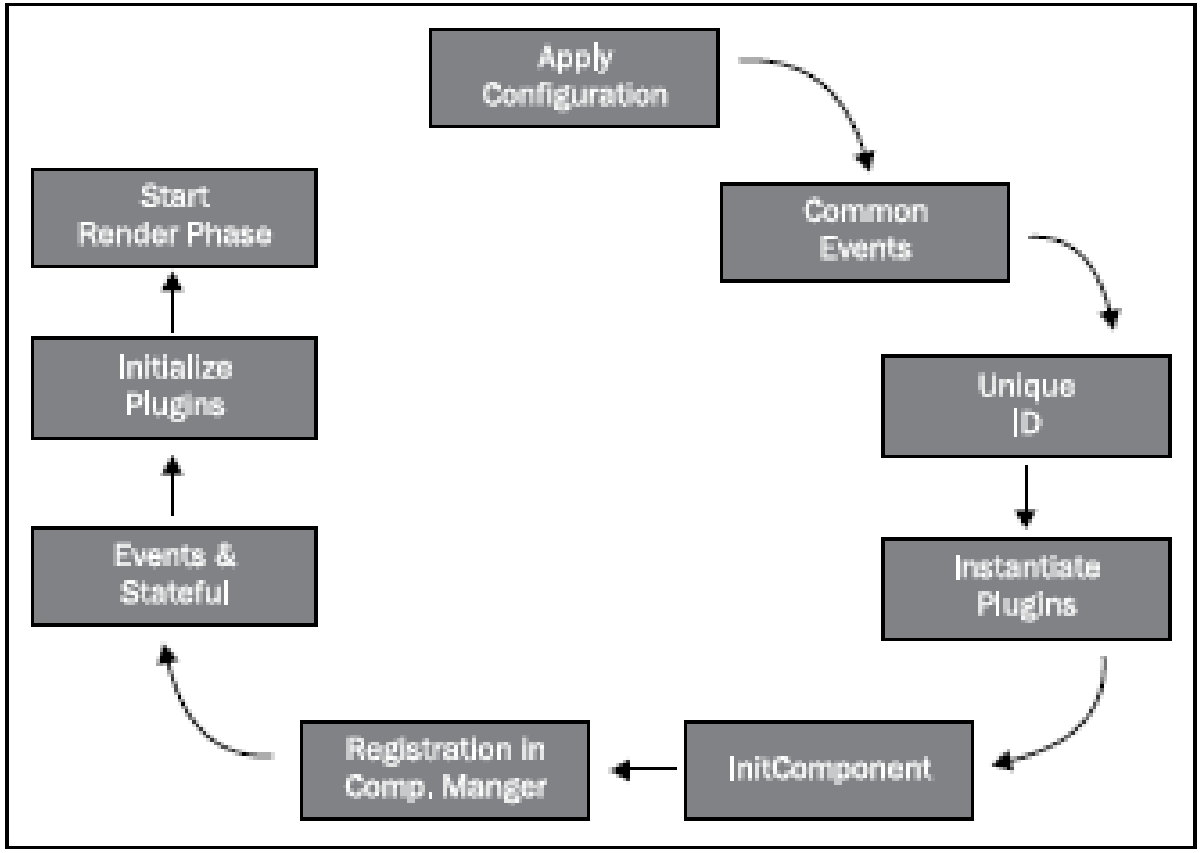


Components and Layouts

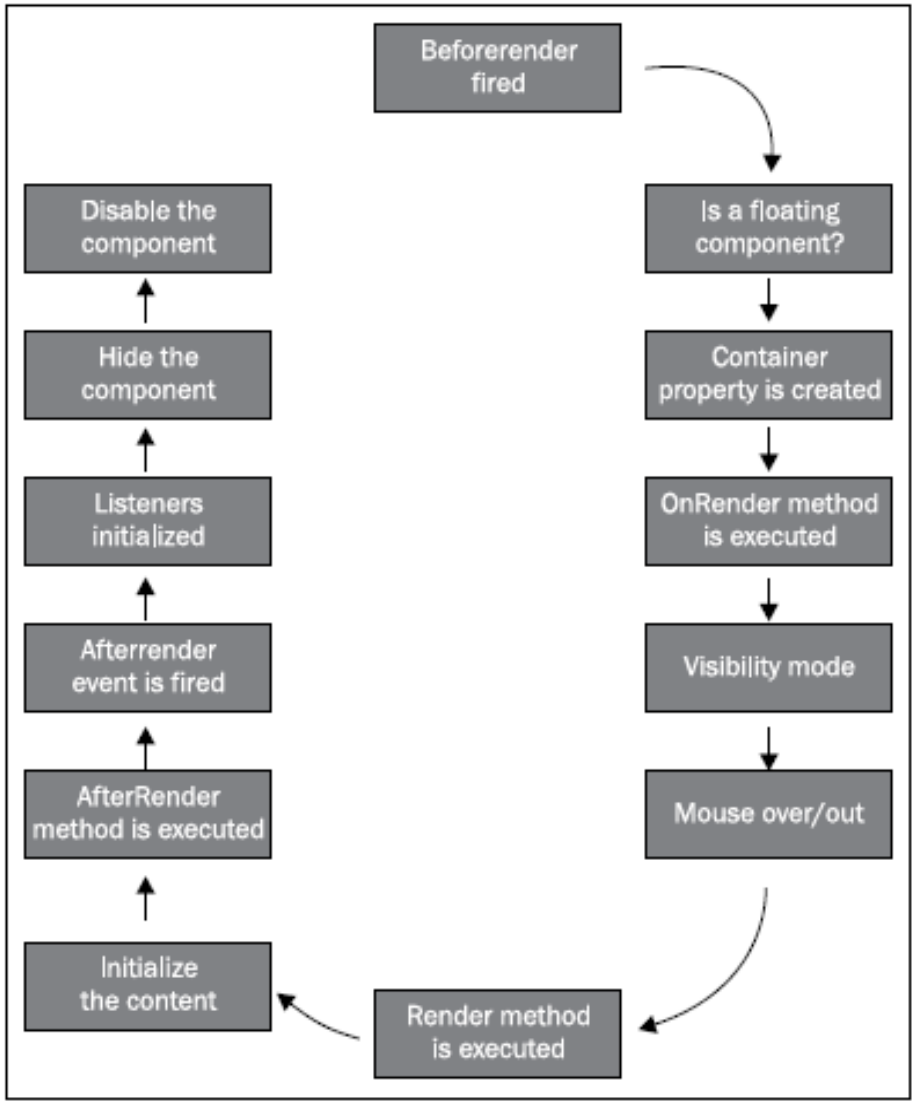


Components Life Cycle

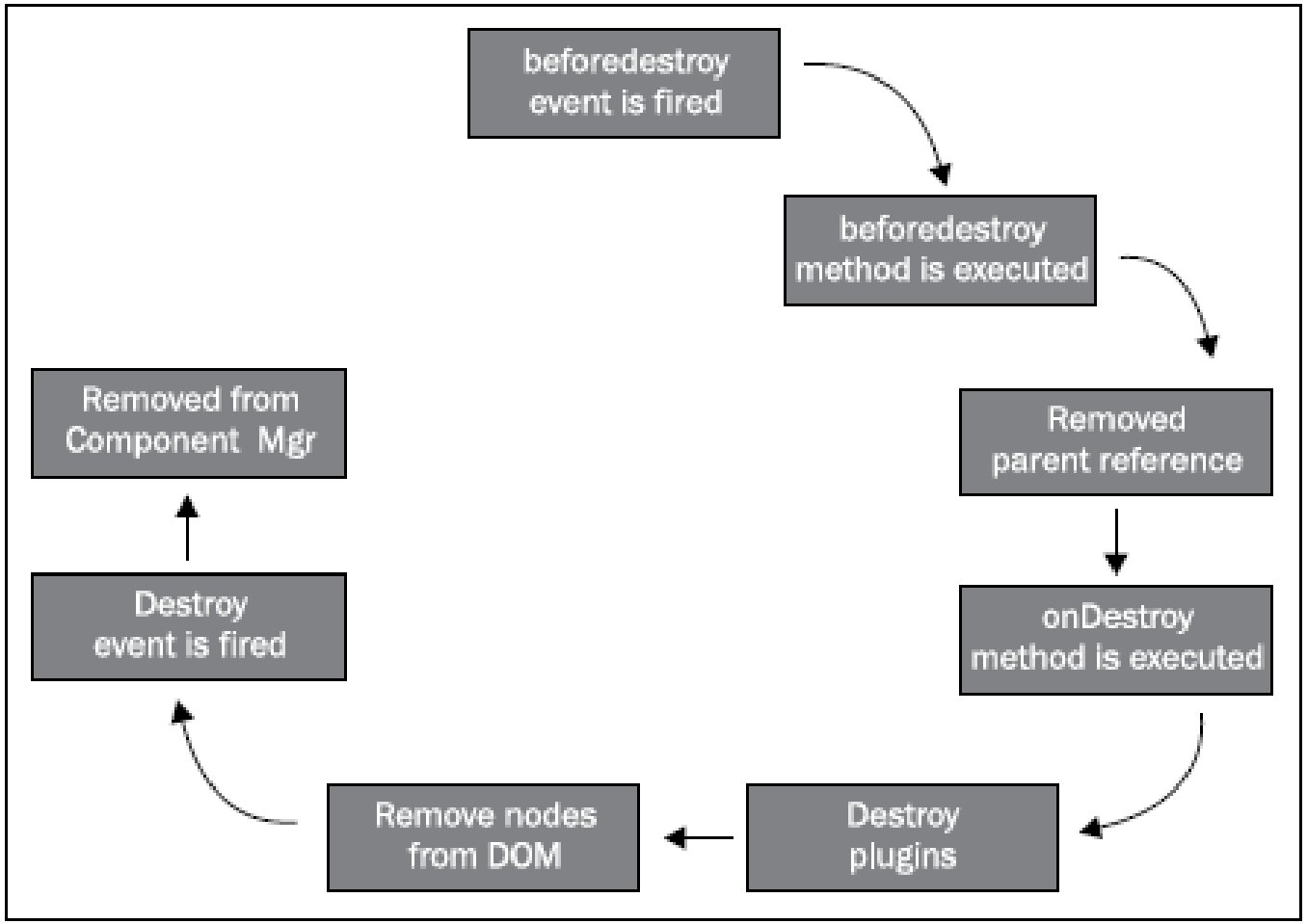
The creation phase



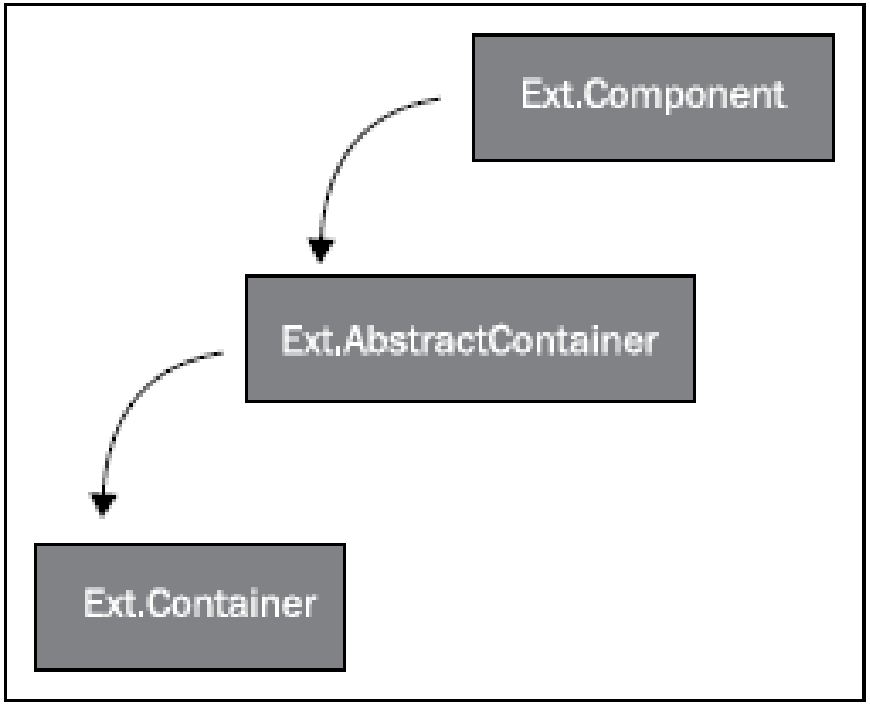
The rendering phase



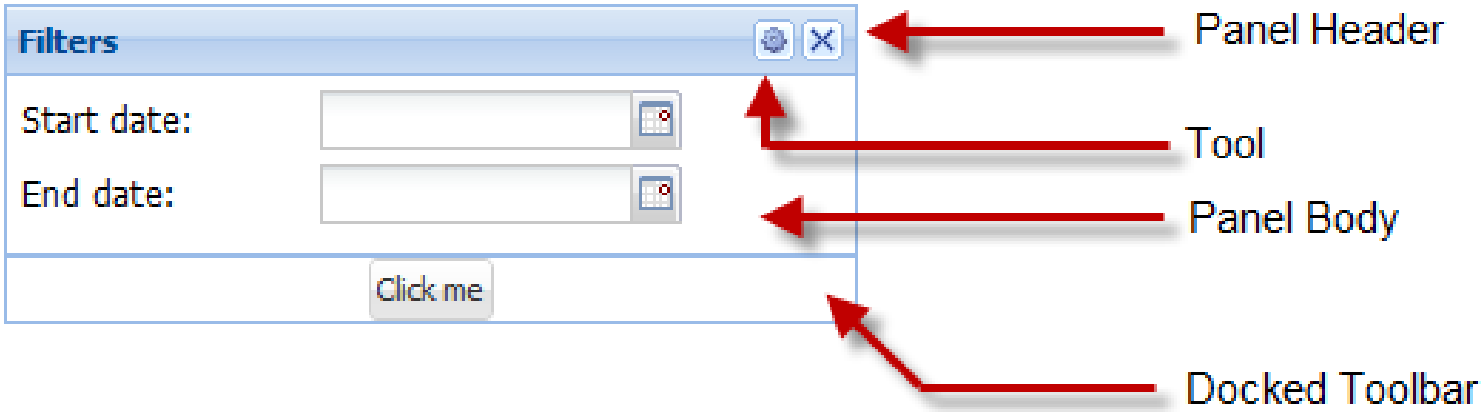
The destruction phase



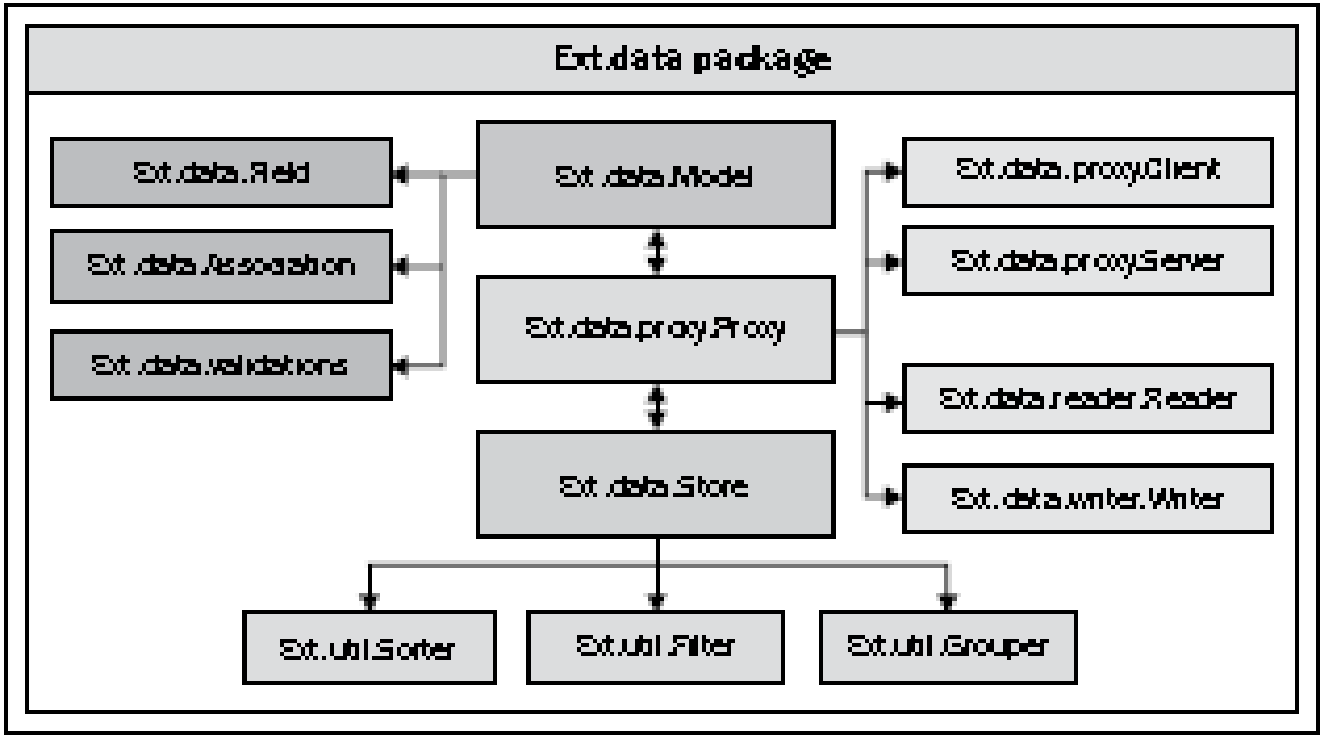
About containers



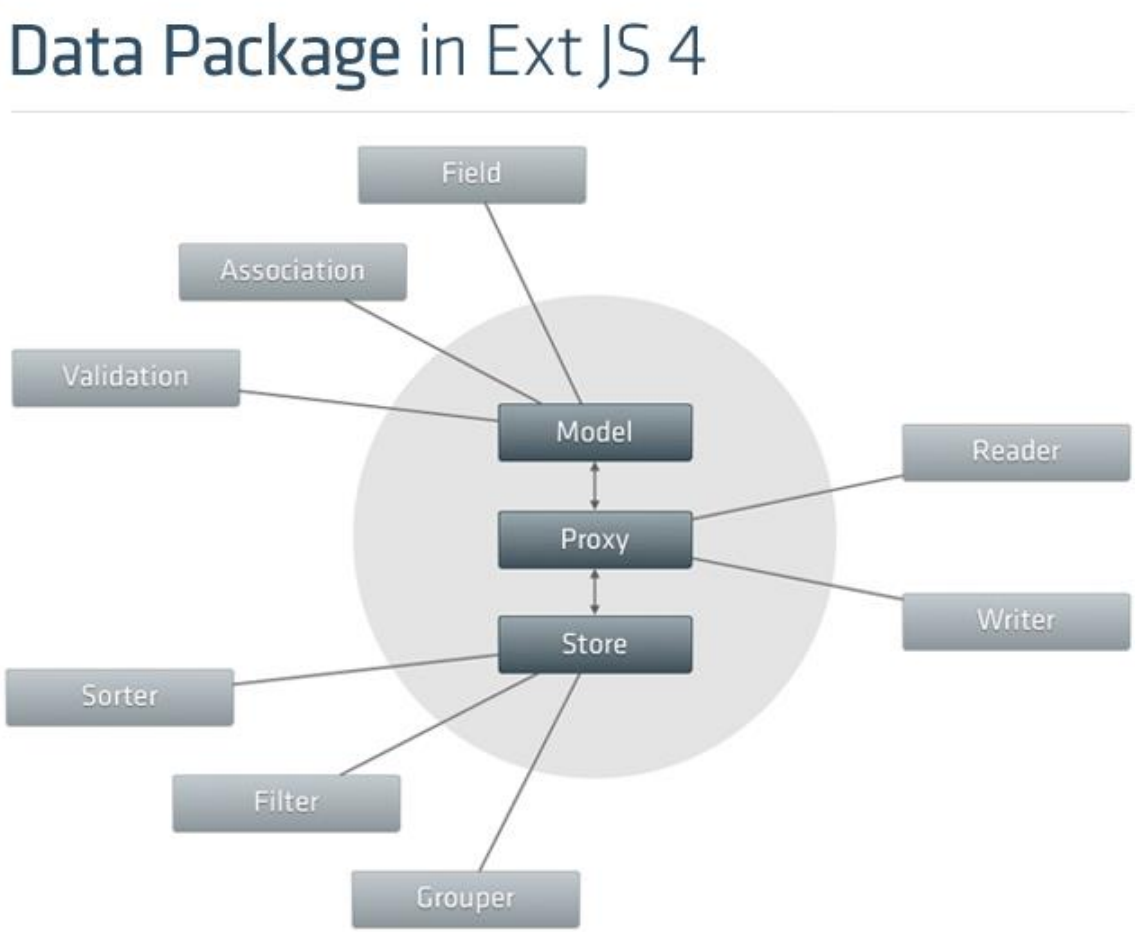
Ext.panel.Panel



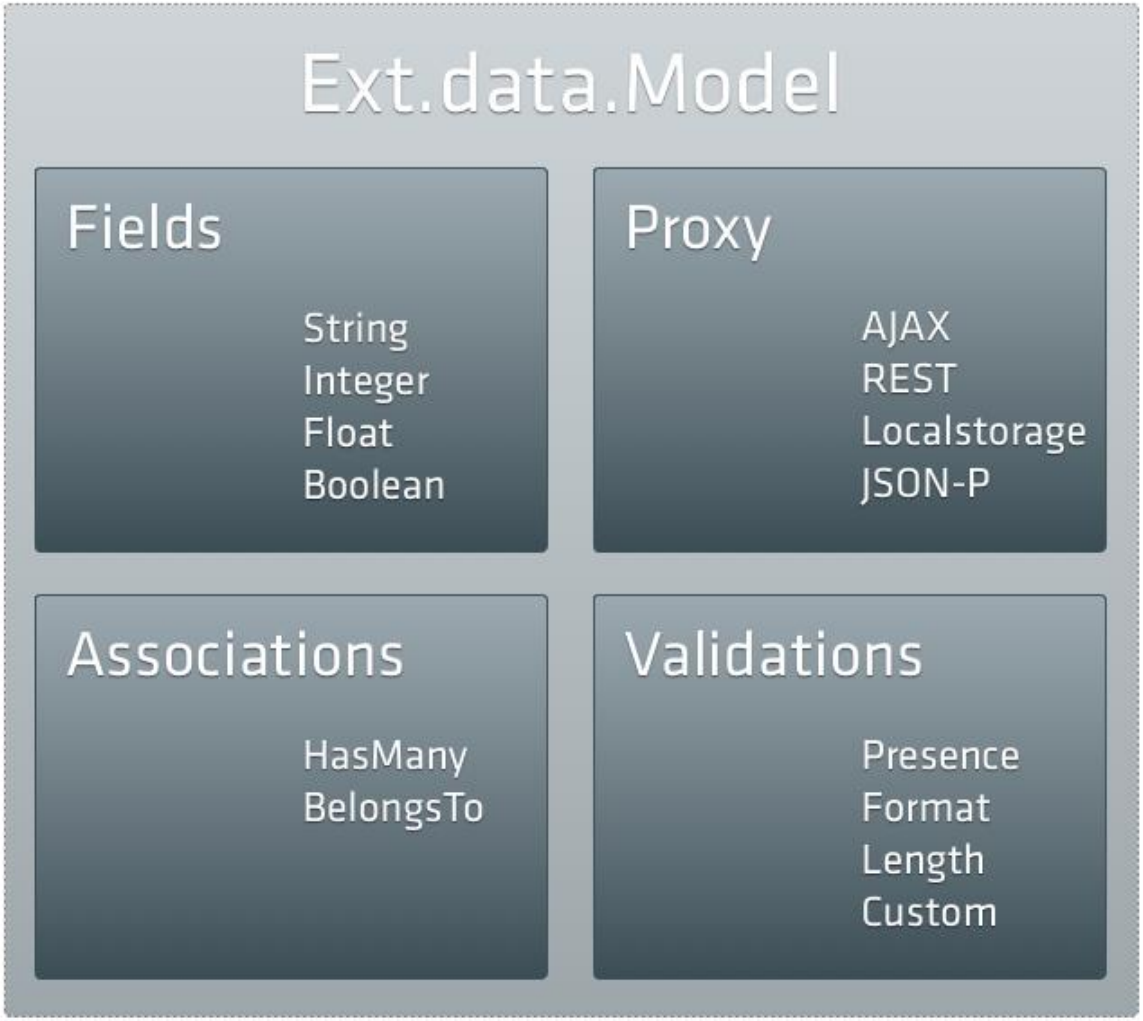
The New Data Package



The New Data Package



The new model class



Declaring fields

- Ext.define('Patient', {
- extend: 'Ext.data.Model',
- fields: [
- {name: 'name'},
- {name: 'age', type: 'int'},
- {name: 'phone', type: 'string'},
- {name: 'gender', type: 'string'},
- {name: 'birthday', type: 'date', dateFormat: 'd/m/Y'},

Declaring fields

- Ext.define('User', { extend: 'Ext.data.Model', fields: [{ name: 'id', type: 'int' }, { name: 'name', type: 'string' }] });
- //creating proxy
- Ext.create('Ext.data.Store', { model: 'User', proxy: { type: 'ajax', url : 'users.json', reader: 'json' }, autoLoad: true });

Inline Data

- Ext.create('Ext.data.Store', { model: 'User', data: [{ firstName: 'Ed', lastName: 'Spencer' }, { firstName: 'Tommy', lastName: 'Maintz' }, { firstName: 'Aaron', lastName: 'Conran' }, { firstName: 'Jamie', lastName: 'Avins' }] });

Ext.data.Store Methods

- findRecord
- getNewRccords
- getPageFromRecordIndex
- getRemovedRecords
- getUpdatedRecords
- loadRecords
- purgeRecords

Proxy

- Proxies are used by Stores to handle the loading and saving of Model data.
- Usually developers will not need to create or interact with proxies directly.
- <u>Types of Proxy</u>
- There are two main types of Proxy
  - Client
  - Server

Client Proxy

- LocalStorageProxy - saves its data to localStorage if the browser supports it.
- SessionStorageProxy - saves its data to sessionStorage if the browsers supports it.
- MemoryProxy - holds data in memory only, any data is lost when the page is refreshed.

Server Proxy

- Ajax - sends requests to a server on the same domain
- JsonP - uses JSON-P to send requests to a server on a different domain
- Direct - uses Ext.direct to send requests

Who

- ExtJS is provided by Sencha (www.sencha.com)
  - Sencha Touch
  - GWT
  - CSS Animator
  - IO (Cloud Data Management)

What

- ExtJS is a MVC Framework
  - Model-View-Control
  - Unlike JQuery and other major JS library providers

When

- ExtJS 4.0 was released around April 2011

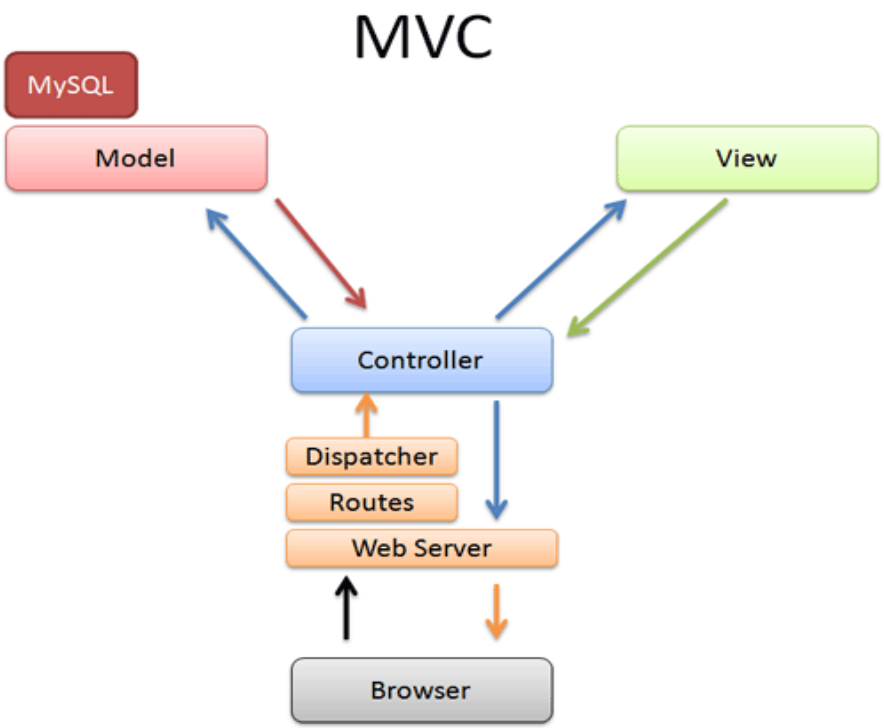
Why

- This is really what matters (MVC)
  - Easy Client-side data modeling
    - Relational Models
  - Simple to use GUI widgets
  - Full robustness of EcmaScript control

MVC

- Why is MVC so important?
  - In this case, it is because it is 100%, agent-based, client side code
  - This means typical MVC on the server is not needed
    - Good or Bad? Design decision

Server Side MVC



Server Side Models

- Server Side Models are simple classes that house an 'instantiated' version of a resource record
  - Resource Records can be a row from a MySql Table, required parameters from another server public api, web service, etc
- These models should be transparent to the controller on how the raw data is represented, but rather be in a specified format for the controller

Server Side Models

- To facilitate how the model instantiates data, usually a map is present
- The Map is capable of understanding how to speak with the resource
  - "Select `id`, `first`, `last` from `names`.....
- The model would then have member variables:
  - $model->id
  - $model->first
  - $model->last
  - ....

Server Side Models

- All of my models have key features
  - 1-to-1 resource mapping
  - $model->save()
  - $model->find()
  - $model->delete()
- Similar to CRUD operations except I leave save() to determine wether it is Create or Update
  - CRUD === 'Create Read Update Destroy'

Server Side Views

- Sever Side View classes, for most frameworks, take the model data and return the requested type of view
  - echo($view->buildTable(records));
- This buildTable() function is called by a controller, who then echo()'s the html generated by the view
- Has one major fault
  - What happens when I want to use this server side stack for mobile apps?
- Are there any performance flaws?

Server Side Control

- We have seen that how models and views work
  - These require some sort of delegation
- Controllers will receive the request from the client (old view), do any preprocessing, call the model (CRUD), use the model data, call the view, and return the html
- Within this return, we usually find JavaScript embedded as a code agent to 'enchance' our viewing pleasure.
- What if we mixed this up a bit and used JavaScript as our primary source of control?

Client Side JS with ExtJS

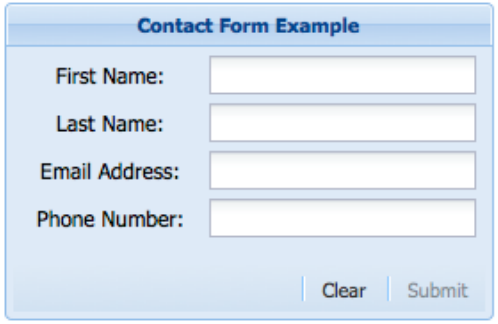
- MVC for JavaScript
- Exactly same process for server side stack, except we now try to use the server as little as possible
  - This allows for powerful client machines to do most of our processing and rendering
  - Only allow the client to manipulate data that can be considered hostile!

ExtJS Models

- The most important feature of ExtJS
  - Can have relational models!!!!!!!
  - Example:
    - Orders (a model) can have many Order Items (another model)
  - Each record of orders is stored in store
  - Each record of orders points to another store that has its Order items
  - This allows us to select an order, and then immediately have access to all its order items

ExtJS View

- Since this is JavaScript, we immediately expect robust GUI widgets



- Of course, you can add CSS and style them

ExtJS Control

- JavaScript is a functional language
  - This allows for very strong and very easy control logic

```
//create record and submit to store
function submitButtonHandler(){

    //first get the form values
    var vals = form.getForm().getValues();

    //now create a record
    var rec = Ext.ModelMgr.create(vals, 'Contacts');

    //insert it at the beginning
    contactStore.insert(0, rec);

    //clear the form
    form.getForm().reset();
}
```

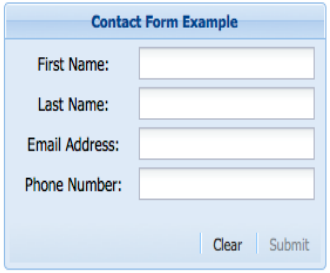
  - Of course, you can still use OOP style if desired

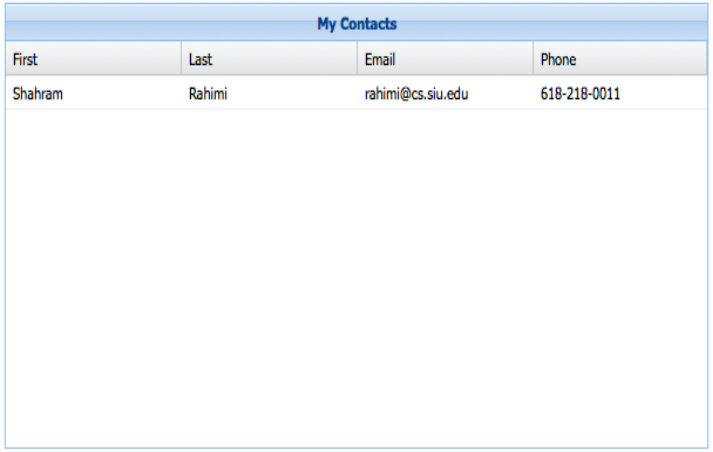
So how does this all work?

- By using MVC on the client side:
  - We only need to contact the server when using CRUD operations
  - By control logic when otherwise needed
- Lets go through an example

Simple Form with Grid

- Our goal will be to make a form, that upon submit, updates a local store
- This store feeds a grid and will automatically update when a new record is inserted





Our data we will work with

- JSON (JavaScript Object Notation)

```
{"contacts": [
    {"id": 1, "first_name": "Shahram", "last_name": "Rahimi", "email": "rahimi@cs.siu.edu", "phone": "618-218-0011"}
]}
```

Configure Script

- We first must wrap all of our JavaScript within an onLoad() function

```
Ext.onReady(function() {

});
```

Create the Model

- Typically, we will always represent modeled data, so it seems wise to start there
- Similar to the server stack, we use the model to communicate with the resource
  - In this case, a flat file named contacts.json

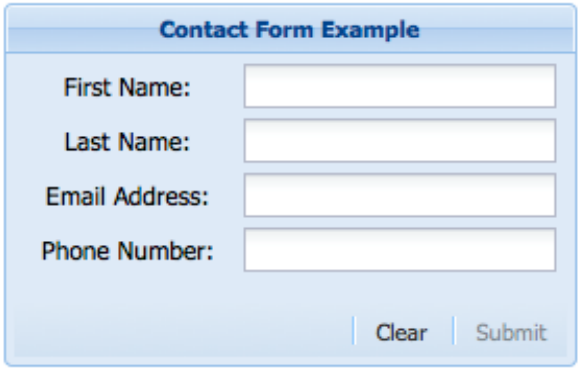
```
// Model to represent presidential terms
Ext.define('Contacts', {
    extend: 'Ext.data.Model',
    fields: [
        'id', 'first_name', 'last_name', 'email', 'phone'
    ],

    proxy: {
        type: 'ajax',
        url: 'contacts.json',
        reader: {
            type: 'json',
            root: 'contacts'
        }

    }

});

// Local store to house the contact records
var contactStore = Ext.create('Ext.data.Store', {

    model: 'Contacts',
    autoLoad: true

});
```

Create the View

```
//first name
var firstNameTF = Ext.create('Ext.form.Text', {"name": 'first_name'...});

//last name
var lastNameTF = Ext.create('Ext.form.Text', {"name": 'last_name'...});

//email
var emailTF = Ext.create('Ext.form.Text', {"name": 'email'...});

//phone
var phoneTF = Ext.create('Ext.form.Text', {"name": 'phone'...});

//clear
var clearButton = Ext.create('Ext.Button', {"text": 'Clear'...});

//submit to local store
var submitButton = Ext.create('Ext.Button', {"formBind": true...});

//tb to hold buttons
var toolbar = Ext.create('Ext.toolbar.Toolbar', {"dock": 'bottom'...});

//basic form
var form = Ext.create('Ext.form.Panel', {"title": 'Contact Form Example'...});

//view grid (shows the contents of store)
var grid = Ext.create('Ext.grid.Panel', {"renderTo": 'grid'...});
```

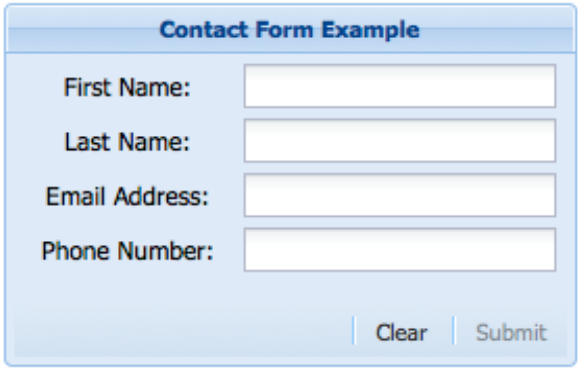
Create the View

```
//first name
var firstNameTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'First Name',
    name: 'first_name',
    validator: nameValidator

});

//last name
var lastNameTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'Last Name',
    name: 'last_name',
    validator: nameValidator

});
```



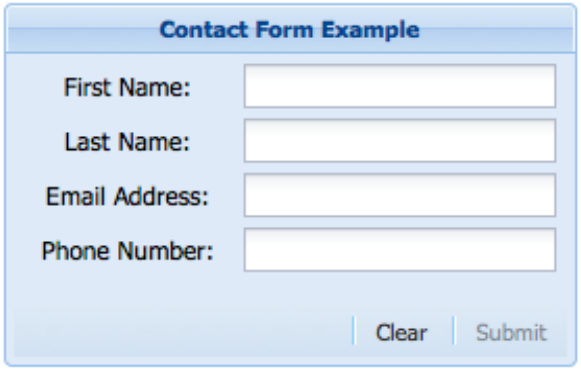
Create the View

```
//email
var emailTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'Email Address',
    name: 'email',
    validator: emailValidator

});


//phone
var phoneTF = Ext.create('Ext.form.Text', {

    fieldLabel: 'Phone Number',
    name: 'phone',
    validator: phoneValidator

});
```



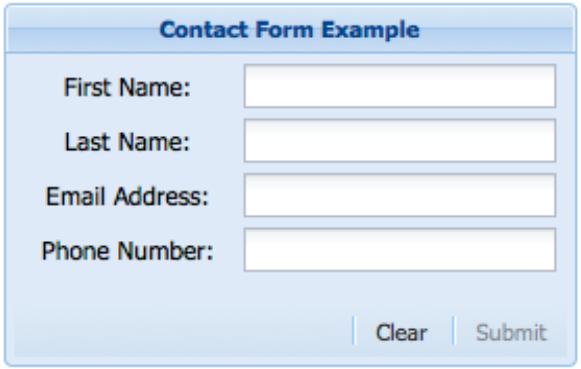
Create the View

```
...

//clear
var clearButton = Ext.create('Ext.Button', {

    text: 'Clear',
    handler: clearButtonHandler
});

//submit to local store
var submitButton = Ext.create('Ext.Button', {

    formBind: true,
    text: 'Submit',
    handler: submitButtonHandler
});
```



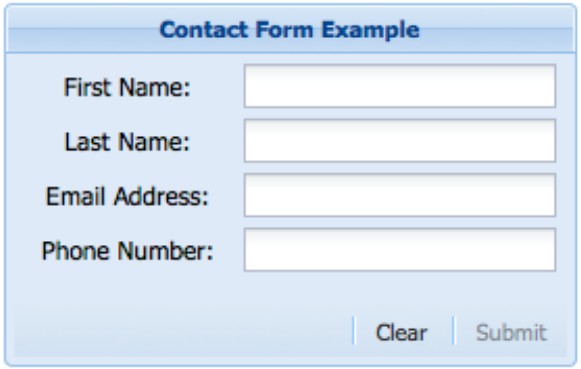
Create the View

```
//tb to hold buttons
var toolbar = Ext.create('Ext.toolbar.Toolbar', {

    dock: 'bottom',
    items: ['->', '-', clearButton, '-', submitButton]
});
```



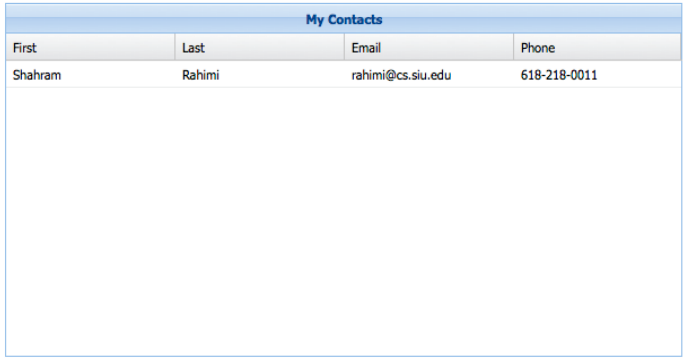
Create the View

```
//basic form
var form = Ext.create('Ext.form.Panel', {

    title: 'Contact Form Example',
    frame: true,
    width: 275,
    height: 180,
    renderTo: 'form',
    dockedItems: toolbar,
    items: [firstNameTF, lastNameTF, emailTF, phoneTF]
});
```



Create the View

```
//view grid (shows the contents of store)
var grid = Ext.create('Ext.grid.Panel', {

    store: contactStore,
    renderTo: 'grid',

    width: 600,
    height: 300,
    autoScroll: true,

    title: 'My Contacts',
    columns:[
        {header: 'First', dataIndex: 'first_name', width: 150},
        {header: 'Last', dataIndex: 'last_name', width: 150},
        {header: 'Email', dataIndex: 'email', width: 150},
        {header: 'Phone', dataIndex: 'phone', flex: 1}
    ]
});
```



Tie in the Control

```
//validate names
function nameValidator(){

    if(!/[a-zA-Z]/.test(arguments[0])) return 'Name may only contain letters';
    return true;
}

//validate emails
function emailValidator(){

    if(!/^[a-z A-Z 0-9]{3,9}@[a-z A-Z 0-9]{3,10}.[ceo][odr][mug]$/.test(arguments[0]))
        return 'Email must contain @ and . and must end in com, edu, or org';
    return true;
}

//validate phones
function phoneValidator(){

    if(!/^[\d]{3}-[\d]{3}-[\d]{4}$/.test(arguments[0]))
        return 'Phone must only contain digits and be in the format of 555-555-5555';
    return true;
}
```

Tie in the Control

```
//clear the form
function clearButtonHandler(){ form.getForm().reset(); }

//create record and submit to store
function submitButtonHandler(){


    //first get the form values
    var vals = form.getForm().getValues();

    //now create a record
    var rec = Ext.ModelMgr.create(vals, 'Contacts');

    //insert it at the beginning
    contactStore.insert(0, rec);

    //clear the form
    form.getForm().reset();
}
```

Final Product

Contact Form Example

First Name:

Last Name:

Email Address:

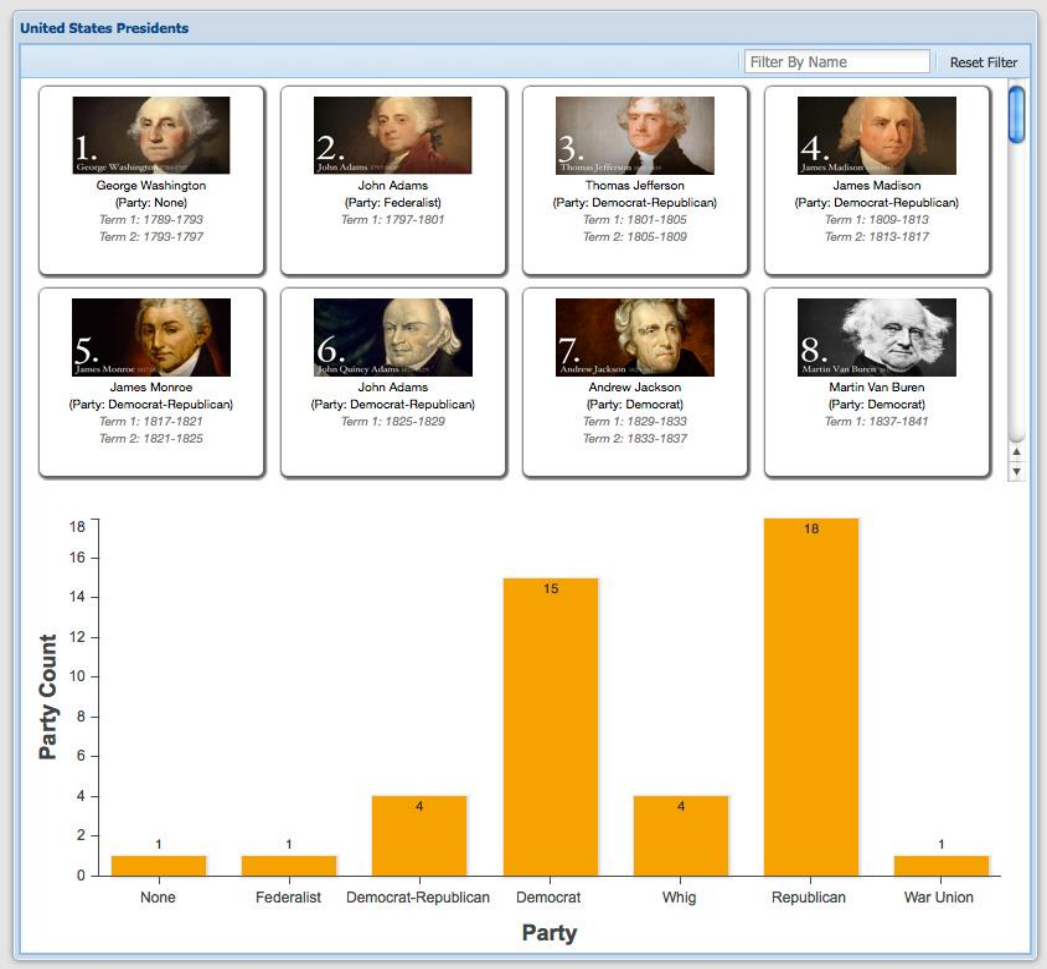
Phone Number:

Clear | Submit

My Contacts

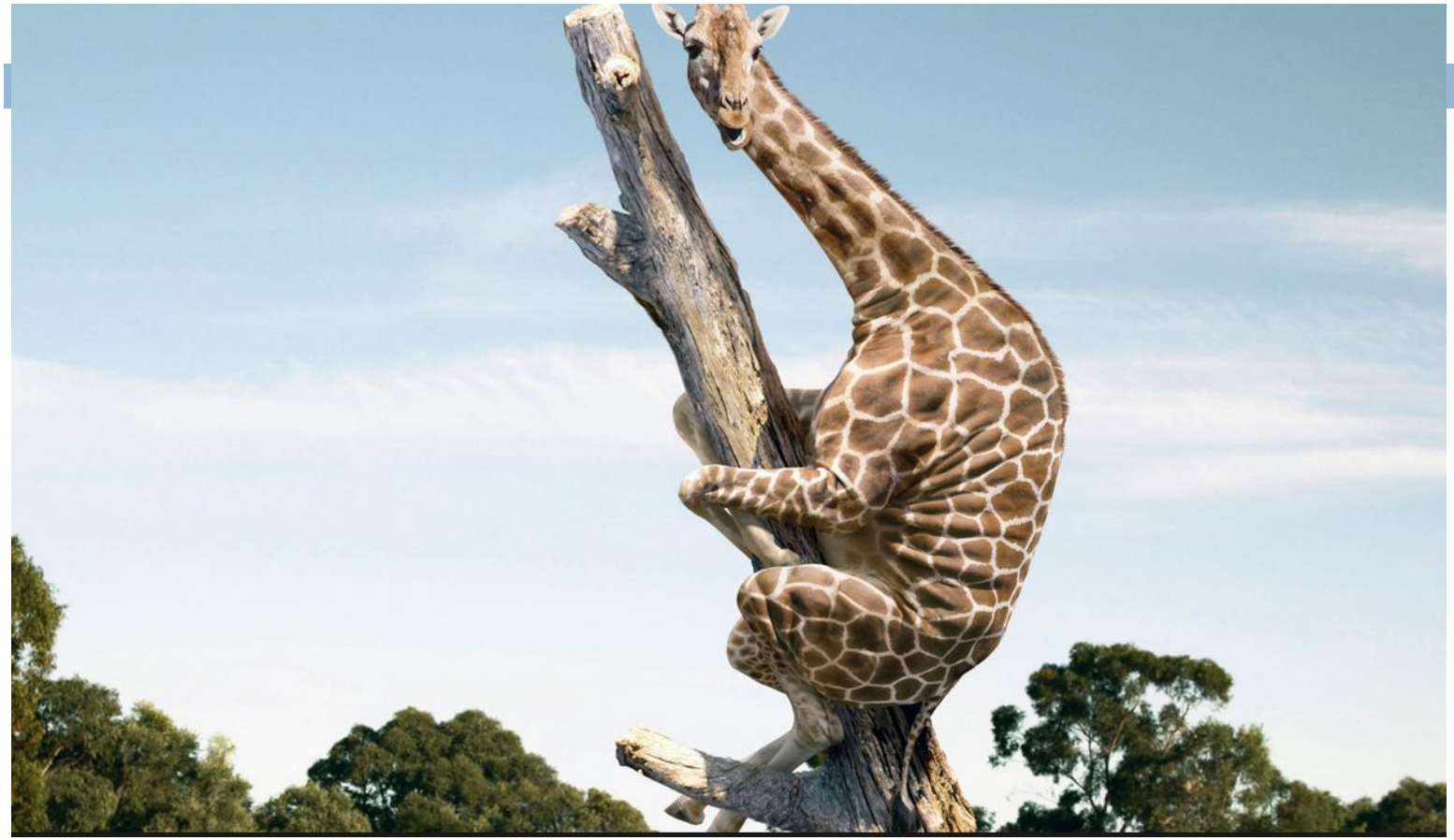
First	Last	Email	Phone
Shahram	Rahimi	rahimi@cs.siu.edu	618-218-0011

Now lets create this



ANGULARJS JUMPSTART

http://angularjs.org



Transclusion

Restriction

Scope

Directive

Linking

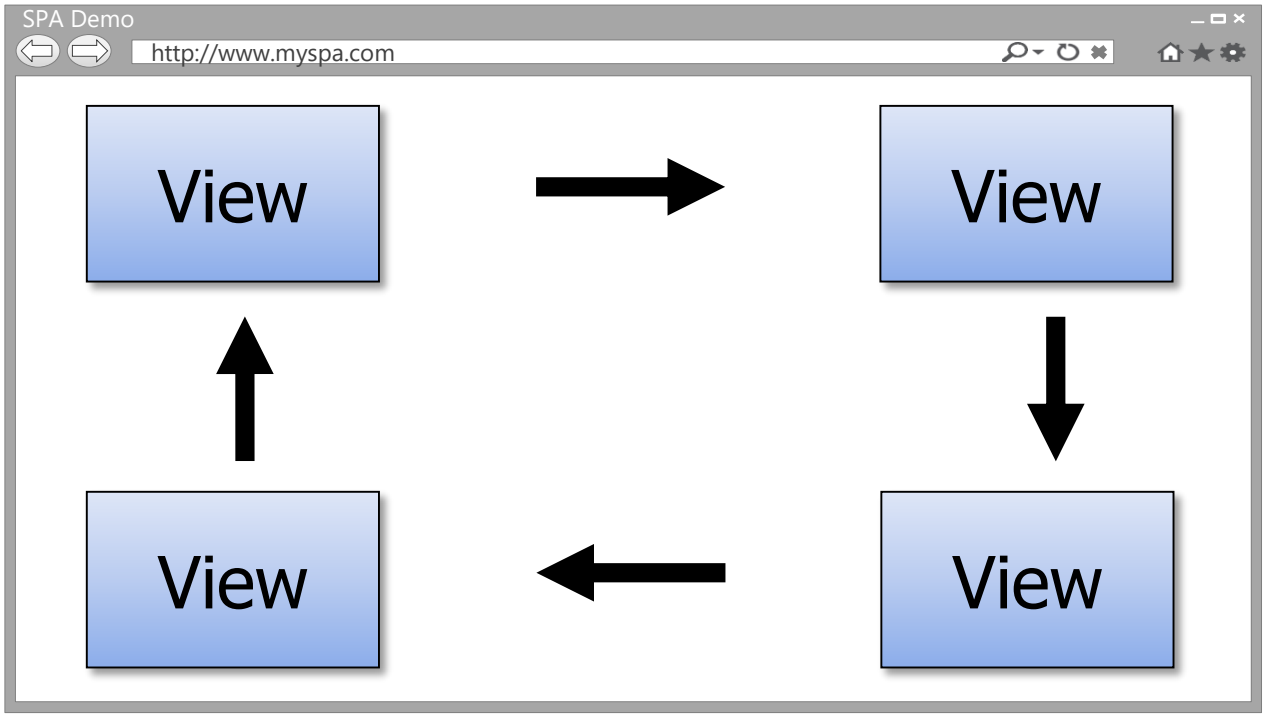
ANGULARJS

Agenda

- AngularJS Features
- Getting Started
- Directives, Filters and Data Binding
- Views, Controllers and Scope
- Modules, Routes and Factories

Getting Started

Single Page Application (SPA)



The Challenge with SPAs

DOM Manipulation

History

Module Loading

Routing

Caching

Object Modeling

Data Binding

Ajax/Promises

View Loading

ANGULARJS

Data Binding | MVC | Routing | Testing

jqLite | Templates | History | Factories

AngularJS is a full-featured SPA framework

ViewModel | Controllers | Views | Directives

Controllers | Dependency Injection | Validation

Directives, Filters and Data Binding

What are Directives?

They teach HTML new tricks!

Using Directives and Data Binding

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title></title>
</head>
<body>
    <div class="container">
        Name: <input type="text" ng-model="name" /> {{ name }}
    </div>

    <script src="Scripts/angular.js"></script>
</body>
</html>
```

Directive

Directive

Data Binding Expression

Iterating with the ng-repeat Directive

```
<html data-ng-app="">
...

   <div class="container"
      data-ng-init="names=['Dave','Napur','Heedy','Shriva']">

       <h3>Looping with the ng-repeat Directive</h3>
       <ul>
           <li data-ng-repeat="name in names">{{ name }}</li>
       </ul>
   </div>

...
</html>
```

Iterate through names

AngularJS Help for Directives

Using Filters

```
<ul>
    <li data-ng-repeat="cust in customers | orderBy:'name'">
        {{ cust.name | uppercase }}
    </li>
</ul>
```

Order customers by name property

Filter customers by model value

```
<input type="text" data-ng-model="nameText" />
<ul>
    <li data-ng-repeat="cust in customers | filter:nameText | orderBy:'name'">
      {{ cust.name }} - {{ cust.city }}</li>
</ul>
```

AngularJS Help for Filters

Demo

Views, Controllers and Scope

View, Controllers and Scope

$scope is the "glue" (ViewModel) between a controller and a view

Creating a View and Controller

```
<div class="container" data-ng-controller="SimpleController">
    <h3>Adding a Simple Controller</h3>
    <ul>
        <li data-ng-repeat="cust in customers">
           {{ cust.name }} - {{ cust.city }}
        </li>
    </ul>
</div>

<script>
    function SimpleController($scope) {
        $scope.customers = [
           { name: 'Dave Jones', city: 'Phoenix' },
           { name: 'Jamie Riley', city: 'Atlanta' },
           { name: 'Heedy Wahlin', city: 'Chandler' },
           { name: 'Thomas Winter', city: 'Seattle' }
        ];
    }
</script>
```

Define the controller to use

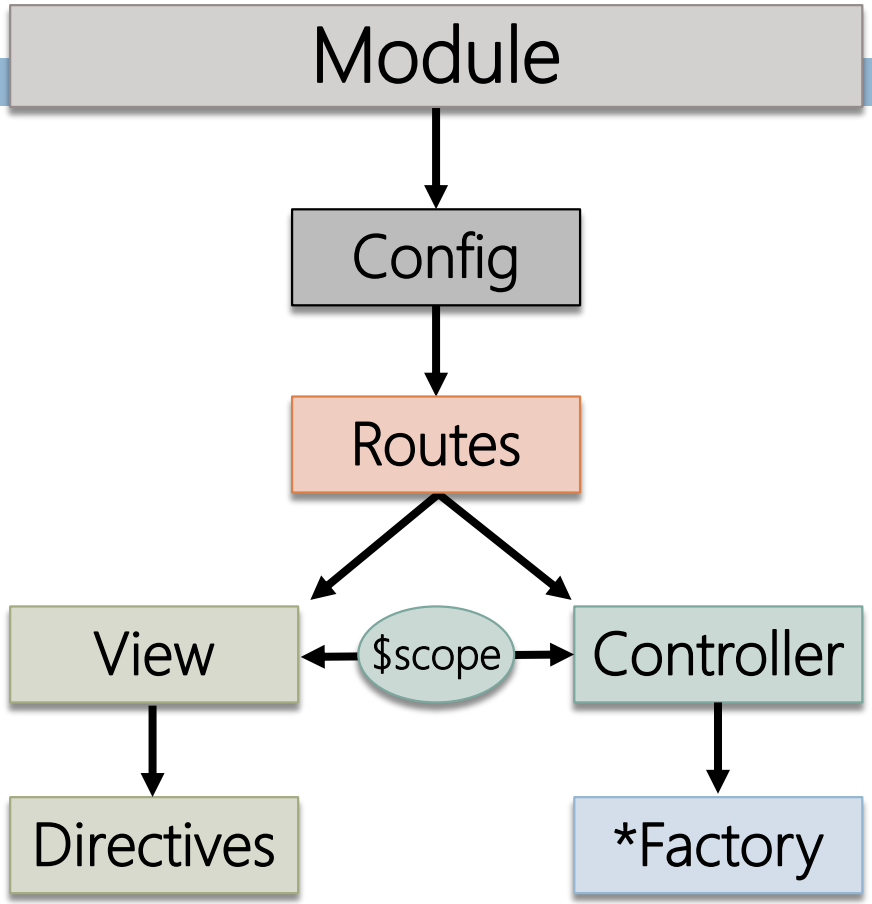
Access $scope

$scope injected dynamically

Basic controller

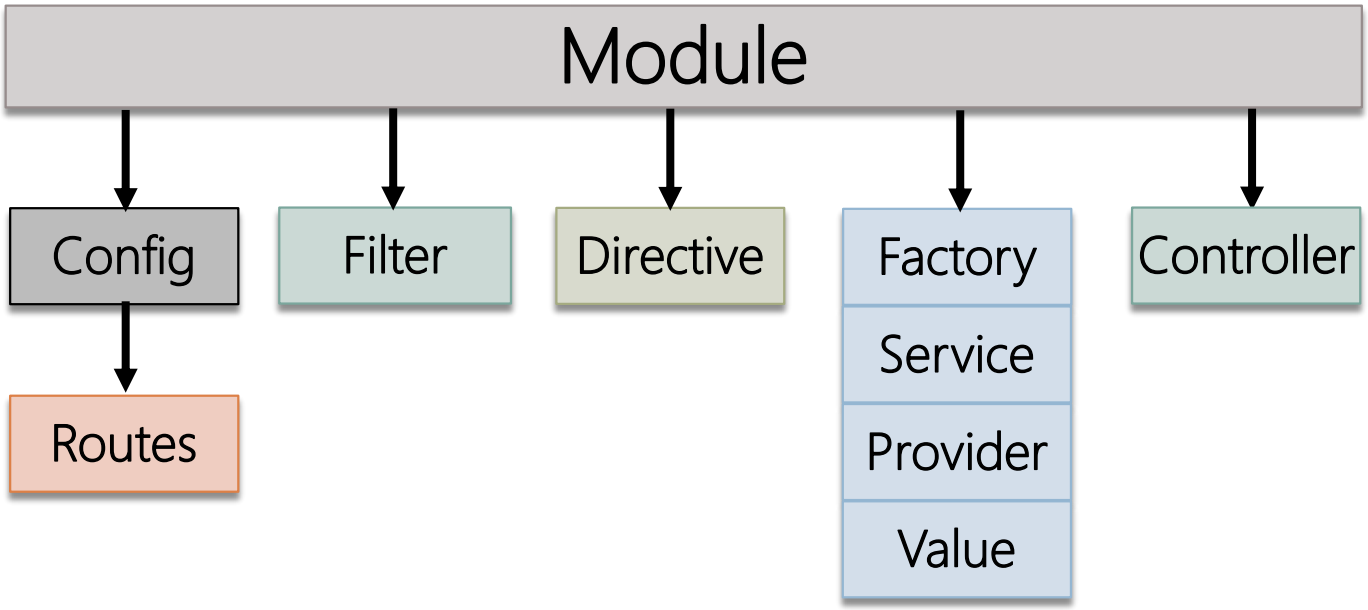
Demo

Modules, Routes and Factories


Module
Config
Routes
View
$scope
Controller
Directives
*Factory

Modules are Containers

```
<html ng-app="moduleName">
```



Creating a Module

What's the Array for?

```
var demoApp = angular.module('demoApp', []);
```

```
var demoApp = angular.module('demoApp',
  ['helperModule']);
```

Module that demoApp depends on

Creating a Controller in a Module

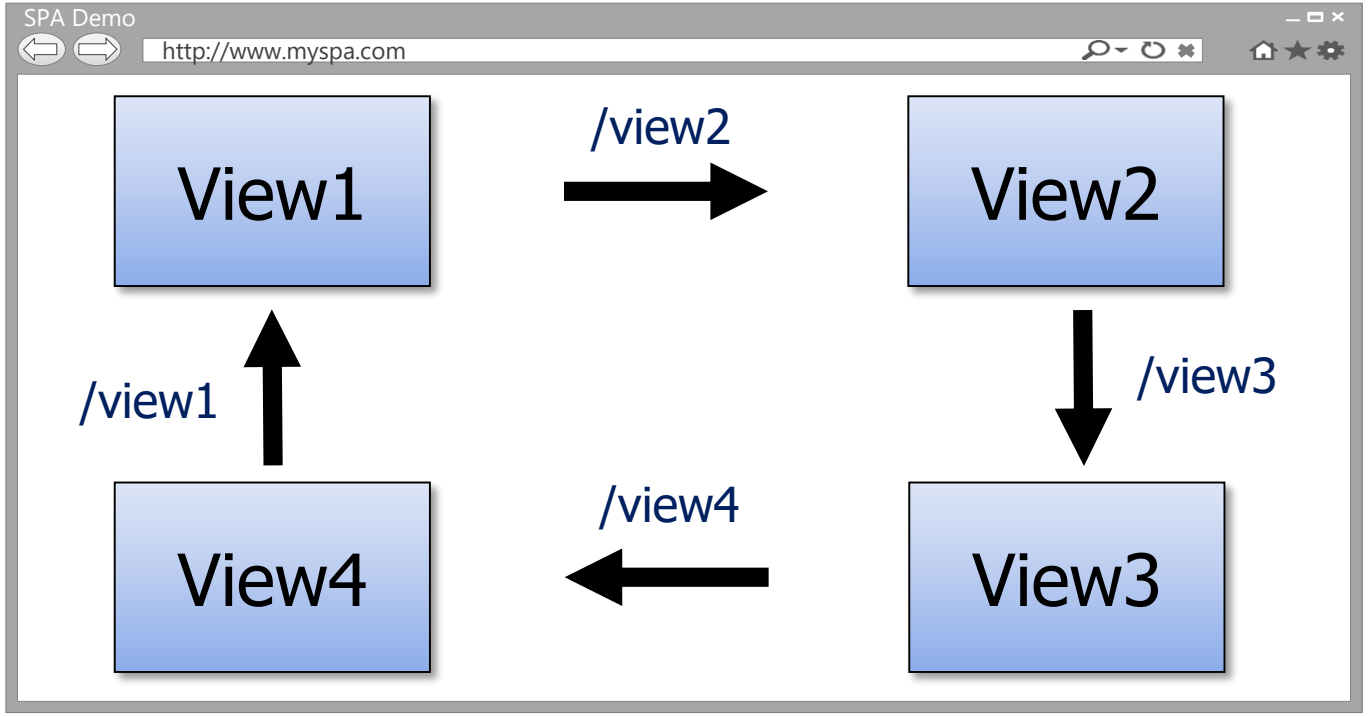
```
var demoApp = angular.module('demoApp', []);
```

Define a Module

Define a Controller

```
demoApp.controller('SimpleController', function ($scope) {
    $scope.customers = [
       { name: 'Dave Jones', city: 'Phoenix' },
       { name: 'Jamie Riley', city: 'Atlanta' },
       { name: 'Heedy Wahlin', city: 'Chandler' },
       { name: 'Thomas Winter', city: 'Seattle' }
    ];

});
```

The Role of Routes



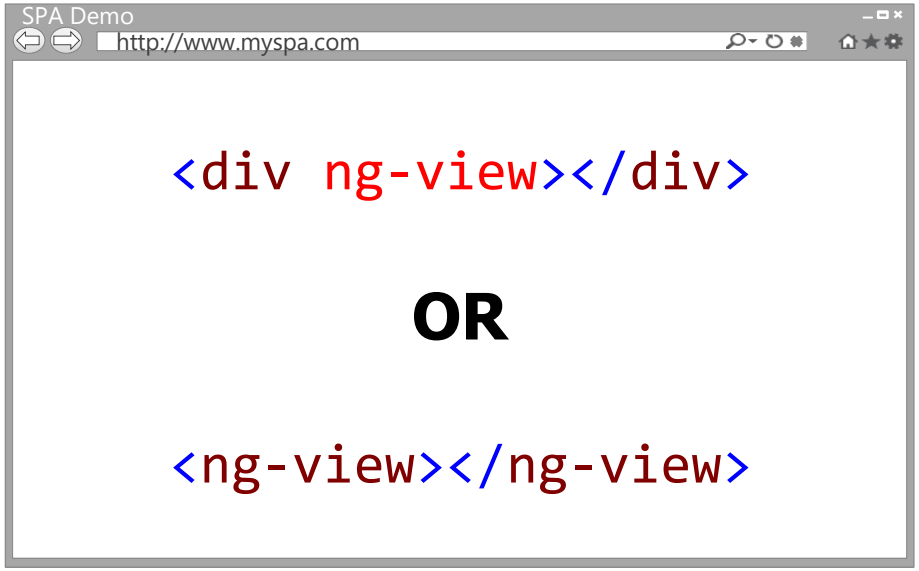
Defining Routes

```
var demoApp = angular.module('demoApp', ['ngRoute']);

demoApp.config(function ($routeProvider) {
    $routeProvider
        .when('/',
            {
                controller: 'SimpleController',
                templateUrl:'View1.html'
            })
        .when('/view2',
            {
                controller: 'SimpleController',
                templateUrl:'View2.html'
            })
        .otherwise({ redirectTo: '/' });
});
```

Define Module Routes

Where do Views Go in a Page?

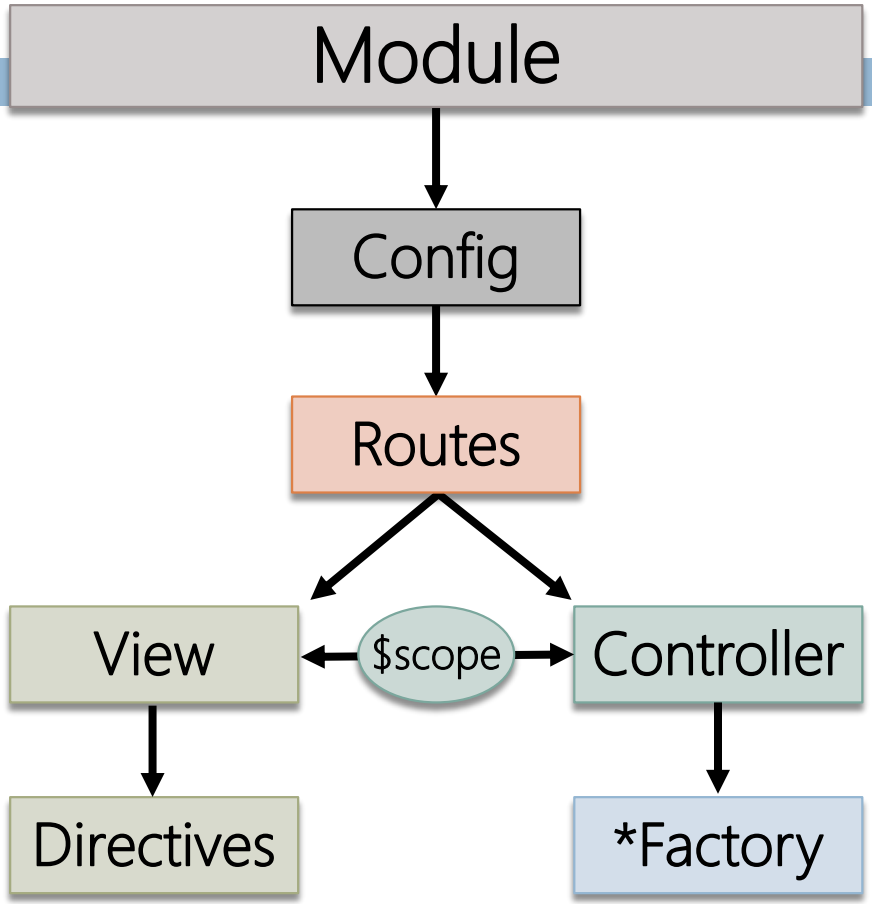
Dynamically loaded views are injected into the shell page as a module loads:



The Role of Factories

```
var demoApp = angular.module('demoApp', [])
 .factory('simpleFactory', function () {
    var factory = {};
    var customers = [ ... ];
    factory.getCustomers = function () {
        return customers;
    };
    return factory;
 })
 .controller('SimpleController', function ($scope,
    simpleFactory) {
     $scope.customers = simpleFactory.getCustomers();
});
```

Factory injected into controller at runtime

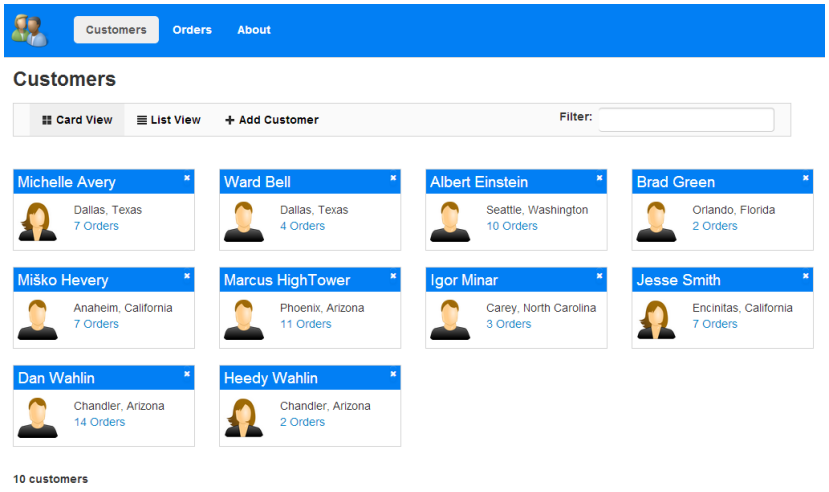

Module
Config
Routes
View
$scope
Controller
Directives
*Factory

Summary

- AngularJS provides a robust "SPA" framework for building robust client-centric applications

- Key features:
  - Directives and filters
  - Two-way data binding
  - Views, Controllers, Scope
  - Modules and Routes

Sample Code

http://tinyurl.com/AngularJSDemos

http://tinyurl.com/CustomerMgr