CS1320
Creating Modern Web and Mobile Applications

Lecture 6:

Dynamic Web Pages

Mechanics

- Project preferences due
- Assignment 1 out
- PreLab for next week is non-trivial

JavaScript has its Quirks

- Procedural, Functional and Object-Oriented all at once
- Objects are very different from Java/C++
  - Newer versions have Java-like classes however
- Scoping is different
  - var versus let or const
  - Declarations can follow uses
  - Declarations are optional
- Automatic type conversion
- Strict versus non-strict equality testing
- eval function
- Semicolons are optional if unambiguous
- Read up on the language (prelab)

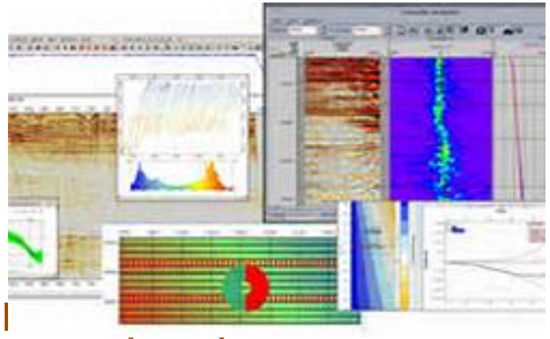
What is an Interactive Application

- How do we want to use JavaScript
- What does interactive mean
- What does it do when you interact
  - Check inputs, compute next page
  - Change the page without getting a new page

Dynamic Web Page Examples

- http://bdognom.cs.brown.edu:5000/ (spheree)
- http://conifer.cs.brown.edu/s6 (s6)
- http://conifer.cs.brown.edu:8888 (twitter)
- http://fred4.cs.brown.edu:8800/ (sign)

Interactive Applications



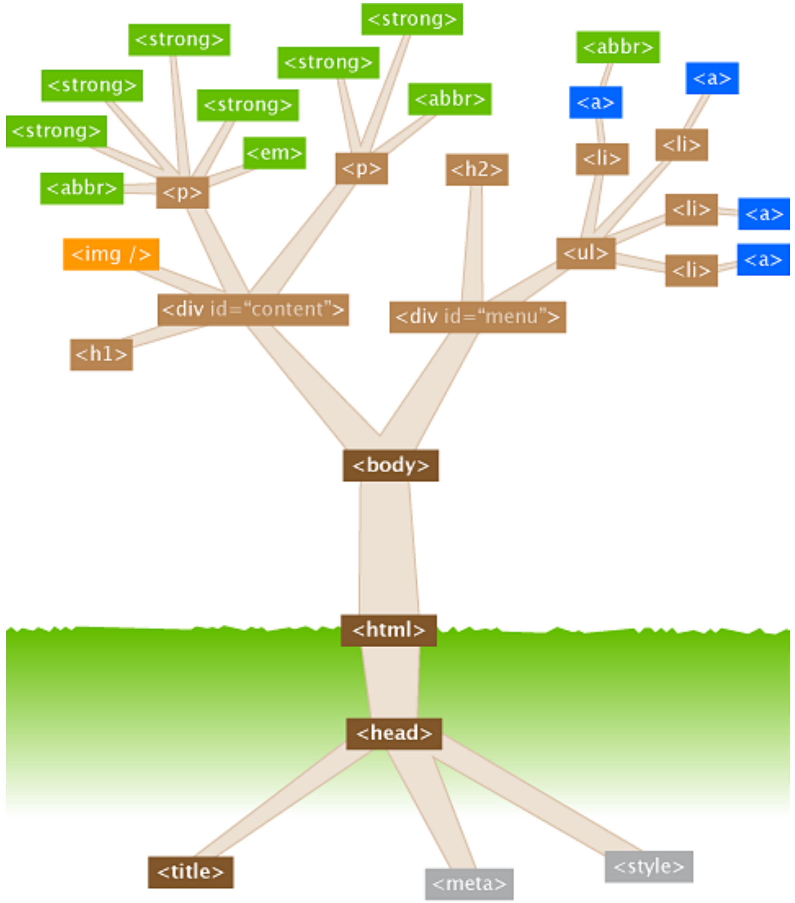
- Respond to user inputs
- Change the display (e.g. add fields, show e
- Dynamically check and verify inputs
- Allow direct manipulation (drag and drop)
- Use animation to highlight or emphasize or show things
- Display external changes in real time
- Provide input help (e.g. text completion)
- Handle dynamic resizing of the display

Achieving Interactivity

- Using CSS
- Handling HTML events using JavaScript
  - Dynamically check and verify inputs
  - Handle direct manipulation
- With modern HTML features
- With animation/drawing/multimedia packages
- By talking to the server continually
- Displaying external changes in real time
- Changing styles and the content of the page
  - Change the display (e.g. add fields, show errors, …)
  - Providing input help (e.g. text completion)
  - Adding graphs, etc. to show output

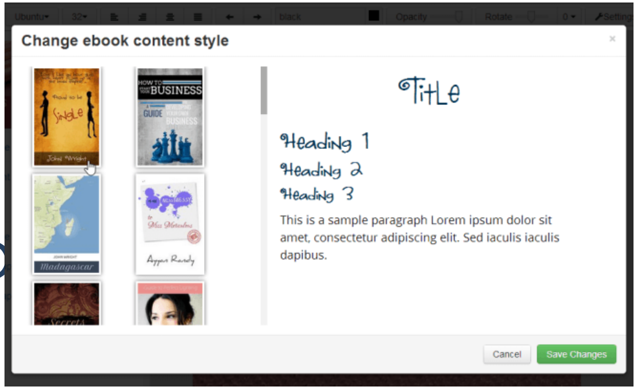
HTML is a Tree

- When it is written
- When it is displayed
- Internally



Changing the Style and Content

- Document Object Model or DOM
  - Representation of HTML in the browser
  - As a set of JavaScript objects representing the nodes
  - You can observe it using the debugger
- JavaScript on the web page can access the DO
  - Get access to DOM objects
    - Values of object attributes
    - Child objects & child text
    - Styles
  - Set values of objects
    - Setting attributes, children, style properties, etc.
    - Setting text
    - Add event handlers
  - Add/remove whole new sub-trees



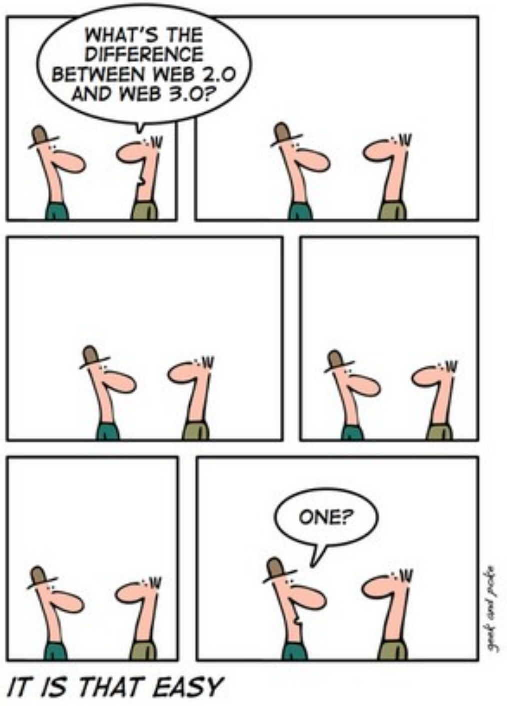
This is a sample paragraph Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed iaculis iaculis dapibus.

DOM Modifications

- Changing the DOM changes the display
  - When JavaScript returns control to the browser
  - As if the new DOM were the original HTML
- This provides most of what you need to do interaction
  - Change text and input values
  - Much can be done by changing HTML **classes**
    - Hide/show, changing styles
    - Easier than changing styles, adding or removing HTML, etc.
  - Can do limited animation as well
    - Changing position, size, etc. dynamically
    - Changing things based on timers
  - Add event handlers

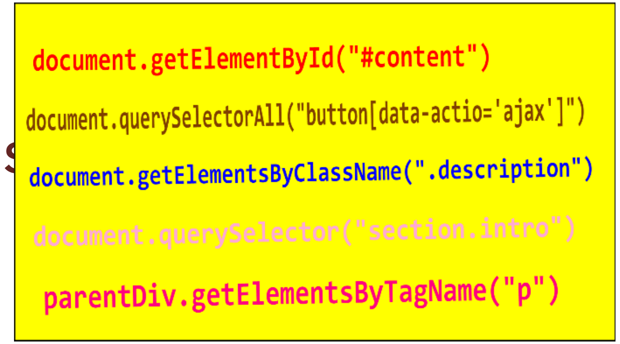
DOM Modification Syntax

- Standard notation
  - document.getElementById("id")
  - document.id1.nestedid1.nestedid2 …
  - <element>.attribute
- Want this to be easier
  - Simple element selection and setting
  - Doing it for a set of elements at once
  - Not requiring ids for all elements
- Where have you seen the definition of ele
  - CSS selectors
  - Why not use the same selectors



JavaScript (ES6) Selector-based Access

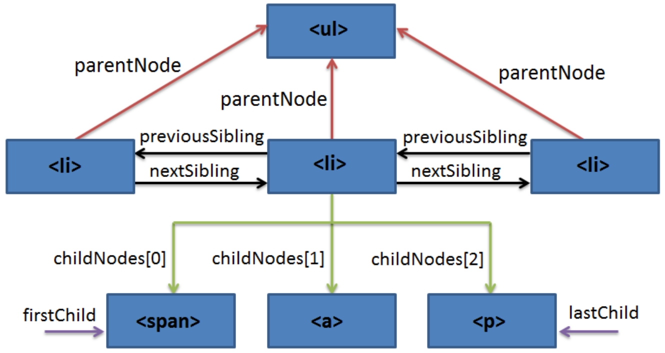
- querySelector – return first selected instance
  - let boxelt = document.querySelector(“.box”)
  - let innerelt = boxelt.querySelector(“.check”)
- querySelectorAll – return all selected ins
  - let allboxes = document.querySelectorAll(“.box”)
  - let innerelts = boxelt.querySelectorAll(“li”);
  - Returns a NodeList
    - Can iterate using **for** (**let** boxelt **of** allboxes) { … }
    - Can iterate using allboxes.**foreach**(<function (box) …>)



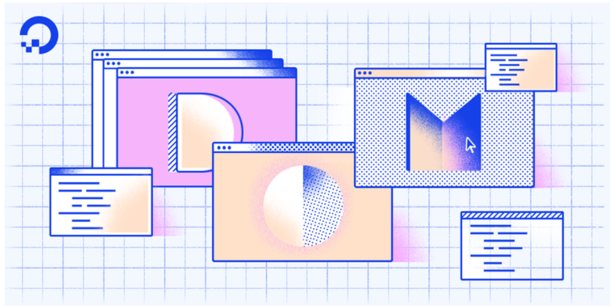
```
document.getElementById("#content")
document.querySelectorAll("button[data-actio='ajax']")
document.getElementsByClassName(".description")
document.querySelector("section.intro")
parentDiv.getElementsByTagName("p")
```

JavaScript DOM Traversal

- Can walk through the elements in tl
  - element.parentElement
  - element.nextElementSibling
  - element.previousElementSibling
  - element.childNodes (NodeList)
- Often easier (and safer) to use selectors here as well



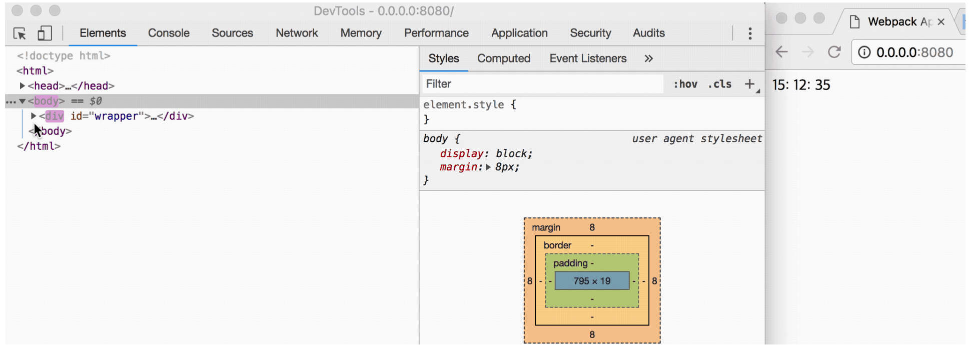
Styling DOM Objects



- Can set style properties directly
  - boxelt.style.color = “#ff00ff”;
  - boxelt.style.backgroundColor = “red”;
  - boxelt.style.cssText = “color: #ff00ff; backgroundColor: red”;
- Can hide and display nodes
  - boxelt.style.display = “none”
  - boxelt.style.display = “block”
- Can add or remove classes – preferred way of updating styles
  - boxelt.classlist.add(“hide”)
    - CSS: .hide { display : none; }
    - Can do box.classlist.add(“show”,”focus”,…)
  - boxelt.classlist.remove(“hide”)
  - boxelt.classlist.toggle(“hide”)

Creating or Modifying the HTML

- let elt = document.createElement(“div”)
  - elt.appendChild(document.createElement(“h1”)
- elt.textContent = “text”
- let textelt = document.createTextNode(“text”)
  - elt.appendChild(textelt)



```
<!doctype html>
<html>
 ▶<head>…</head>
…▼<body> == $0
  ▶<div id="wrapper">…</div>
  </body>
</html>
```

```
element.style {
}
body {                     user agent stylesheet
  display: block;
  margin: ▶ 8px;
}
```

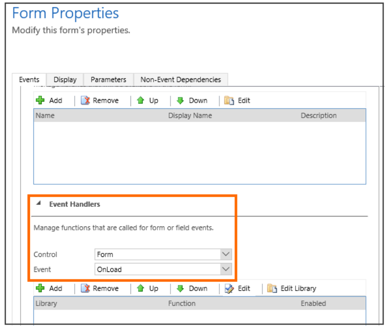
15: 12: 35

Adding Event Handlers

- Dynamically created elements might have associated events/actions
- elt.addEventListener(“click”,function)
  - click, mouseenter, keyup, keydown …
- Do this when the element is added to the DOM
- function ready(callback) {

```
        if (document.readyState != “loading”) callback();
        else document.addEventListener(“DOMContentLoaded”,callback)
    }
```

  - ready(function to be called to set things up once document is loaded)
  - Useful for adding event listeners to dynamic content


Form Properties

Modify this form's properties.

Event Handlers

Manage functions that are called for form or field events.

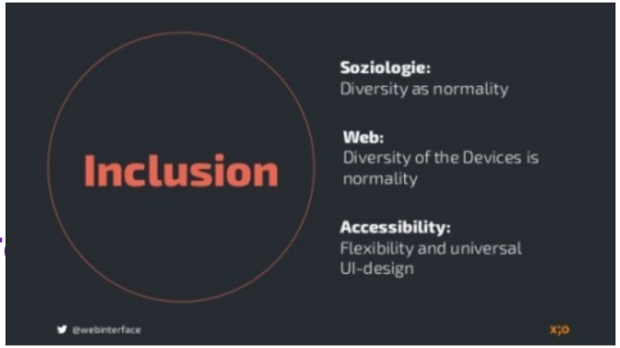
Control: Form

Event: OnLoad

Changing the DOM

- Easiest
  - Put all text on page, then hide/show as needed
  - Add or remove classes to change display properties
  - Set text or html or value for computed items
  - Moving items around within a list or table
- More difficult
  - Create new html for items in a list or table
    - Cloning original or just creating from scratch
    - Libraries exist for this
  - Actually creating new html for the page
    - Better done elsewhere
- In Between (later lectures)
  - Use a helper library such as jQuery
  - Use templates to create items as needed
  - REACT, Angular, Vue, …

DOM Update and Accessibility


Inclusion

Soziologie:
Diversity as normality

Web:
Diversity of the Devices is normality

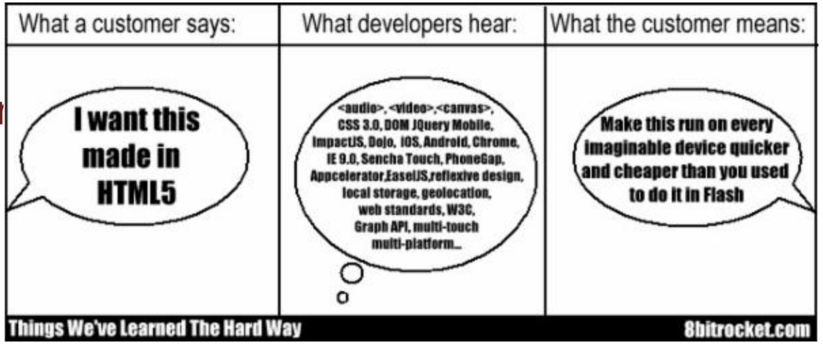
Accessibility:
Flexibility and universal UI-design

@webinterface

- Changing the DOM can cause accessibility pr
  - Updates can confuse screen readers
  - Updates might not be visible in high magnification
  - Updates might be invisible (red border for errors)
  - Updates might come too fast (before page was read)
- These should be addressed: here are some guidelines
  - If content updates for more than 5 seconds, provide the ability to pause, stop or hide the updates
  - Inform the user of changes (live region, alert, setting focus, highlight)
  - Inform user that the page is dynamic
  - Work without dynamics, provide static HTML alternative pages
- These need to be tested carefully (i.e. screen reader testing)

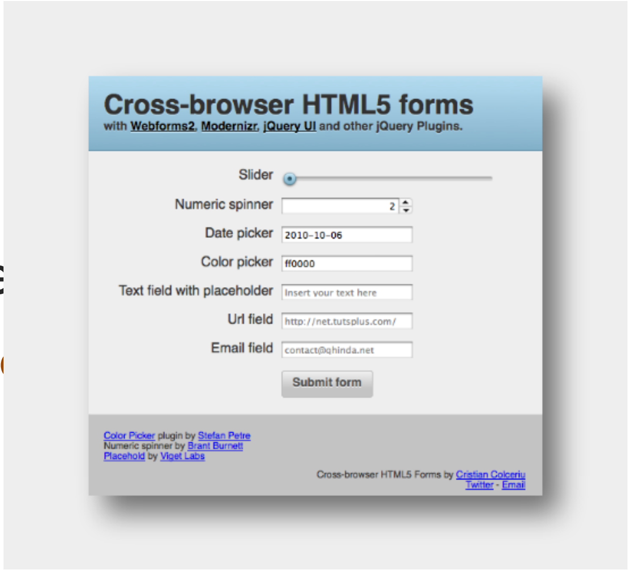
HTML = HTML5

- HTML5 is designed to support modern web apps
  - More interaction
  - More devices
- Multimedia and animations are more comm
  - A large fraction of web sites are using them
  - They shouldn't require plugins to be usable
  - These should be standard in all browsers
- Other features have similar properties
  - Simple databases, cookie management, …
- Basic HTML doesn't provide enough context information
  - About the page (for search, readers, …)
  - About forms (numbers, dates, …)


What a customer says:
I want this made in HTML5
What developers hear:
<audio>, <video>,<canvas>, CSS 3.0, DOM JQuery Mobile, ImpactJS, Dojo, IOS, Android, Chrome, IE 9.0, Sencha Touch, PhoneGap, Appcelerator,EaselJS,reflexive design, local storage, geolocation, web standards, W3C, Graph API, multi-touch multi-platform...
What the customer means:
Make this run on every imaginable device quicker and cheaper than you used to do it in Flash
Things We've Learned The Hard Way
8bitrocket.com

HTML5 Forms

- Do forms work on your smart phone/table
- Forms are the basis for much HTML intera
  - But they are quite limiting
  - And not well-oriented to tablets / smart phones
  - And require JavaScript to validate
- HTML5 significantly expands the input types in forms
  - Text, password, submit, radio, checkbox, button
  - Color, date, datetime, email, month, number, range, search, tel, time, url, week
  - With built-in validation
  - Generic regular-expression based validation



Cross-browser HTML5 forms

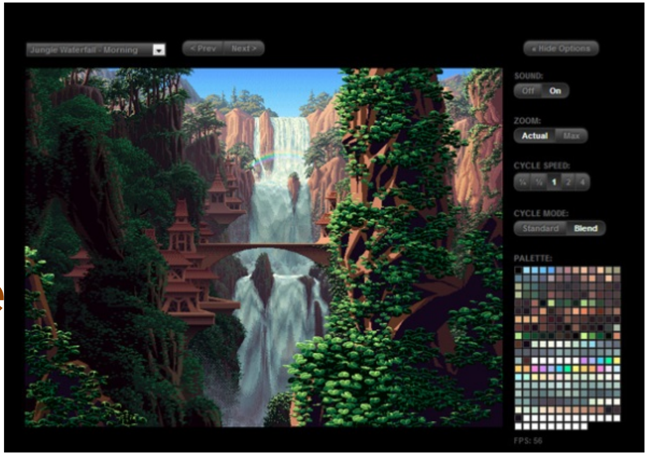
with Webforms2, Modernizr, jQuery UI and other jQuery Plugins.

Slider	
Numeric spinner	2
Date picker	2010-10-06
Color picker	ff0000
Text field with placeholder	Insert your text here
Url field	http://net.tutsplus.com/
Email field	contact@ghinda.net

Color Picker plugin by Stefan Petre
Numeric spinner by Brant Burnett
Placehold by Viget Labs

Cross-browser HTML5 Forms by Cristian Colceriu
Twitter - Email

HTML5 Canvas



- A canvas is a drawing area on the page
  - Use JavaScript to draw on that canvas
  - Drawing is similar to Java2D drawing
    - Similar primitives, transformations, coordinates, etc.
    - Rectangles, paths, arcs, text
    - Java Graphics2D maps to HTML5 Context
  - Can be used for static graphics and animations
- http://www.youtube.com/watch?v=xnAiEJEBLJg
- http://www.youtube.com/watch?v=oZInfZ0wecw

SVG Graphics

- Different approaches to graphics
  - Procedural calls to draw everything
  - Structure representing what should be drawn
- SVG takes the second approach
  - The structure is part of the DOM
    - Can be manipulated by JavaScript
  - Objects correspond to various primitives
  - Often easier than functional drawing
    - Refresh handled automatically
- http://www.youtube.com/watch?v=6SDKN-Amlyo

HTML5 Multimedia

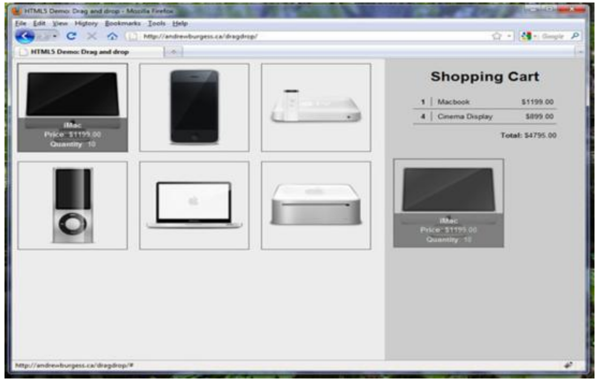
- <audio> and <video> tags
  - Controls
  - Multiple formats can (and have to) be provided
- Examples

```
<video width="320" height="240" controls="controls">
        <source src="movie.mp4" type="video/mp4" />
        <source src="movie.ogg" type="video/ogg" />
        Your browser does not support the video tag.
</video>
<audio controls="controls“>
       <source src="song.ogg" type="audio/ogg" />
       <source src="song.mp3" type="audio/mpeg" />
       Your browser does not support the audio element.
</audio>
```

HTML5 Drag and Drop

- Direct manipulation interfaces are sometimes based on drag and drop
  - That's what users have come to expect
- HTML5 lets any element be dragged
  - And any element can be a drop target
- HTML5 also provides JavaScript events to support
  - On drag start (set the content of the drag)
  - On drag over (allow/disallow drop)
  - On drop (use the contents)
- Much simpler to use than Java drag and drop

Drag and Drop Example


HTML5 Demo: Drag and drop - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://andrewburgess.ca/dragdrop/
HTML5 Demo: Drag and drop
iMac
Price: $1199.00
Quantity: 10
Shopping Cart
iMac
Price: $1199.00
Quantity: 10
http://andrewburgess.ca/dragdrop/#

1	Macbook	$1199.00
4	Cinema Display	$899.00

Total: $4795.00

```
<!DOCTYPE HTML>
<html> <head> <script type="text/javascript">
function allowDrop(ev) { ev.preventDefault(); }
function drag(ev)   { ev.dataTransfer.setData("Text",ev.target.id); }
function drop(ev)
{
        var data=ev.dataTransfer.getData("Text");
        ev.target.appendChild(document.getElementById(data));
        ev.preventDefault();
}
</script> </head> <body>
<div id="div1" ondrop="drop(event)“ ondragover="allowDrop(event)"></div>
<img id="drag1" src="img_logo.gif" draggable="true“ ondragstart="drag(event)" width="336" height="69" />
</body> </html>
```

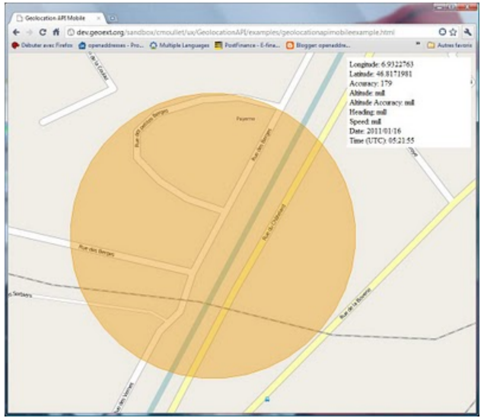
HTML5 Web Storage

- Cookies are not efficient or secure
  - Have to be sent with each HTTP request
- HTML5 offers several new facilities
  - Local storage (name-value) of arbitrary data
    - Permanent, fixed time, or session-based
  - Generation of public-private keys
    - Offers secure communication
    - Rarely used – use HTTPS instead

"THIS WEBSITE USES PIES INSTEAD OF COOKIES."

HTML5 Geolocation

- HTML5 enables using the current location
  - Accurate from a device with GPS
  - Approximate from other computers
- Can use this with JavaScript
  - Locally (place on a map)
  - Globally (send to server)
- Can also get automatic updates
  - JavaScript code that is invoked as the position changes
  - There's an event for that



```
Longitude: 6.9322763
Latitude: 46.8171981
Accuracy: 179
Altitude: null
Altitude Accuracy: null
Heading: null
Speed: null
Date: 2011/01/16
Time (UTC): 05:21:55
```

Geolocation Example

```
<script>
var x=document.getElementByName(“demo”);
function getLocation()
{
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);   Fix
    }
    else { x.html("Geolocation is not supported.”);}
}
function showPosition(position)
{
    x.html("Latitude: " + position.coords.latitude +
        "<br />Longitude: " + position.coords.longitude);
}
</script>
```

728 morons within a radius of 400 feet
TWEET TWEET
phew...
27 bitches within a radius of 400 paws
wow!

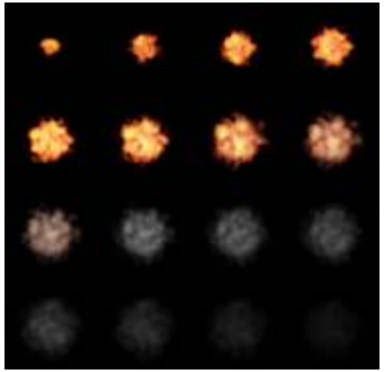
HTML5 Messaging

- Mashups
  - Web pages composed of information from multiple sources
  - Browsers limit where requests can be sent based on URLs
    - Make mash-ups difficult to implement
  - Messaging allows this to be bypassed in a selective manner
  - Usually embedded in libraries – not something you do directly
    - Maps: Google maps, leaflet.js
    - Payments: Stripe, Paypal
    - Other: address decoding, weather, …
- Web Sockets
  - Continuous communication with your server
  - Easy to set up and use (callback functions on both ends)

Animation on Web Pages

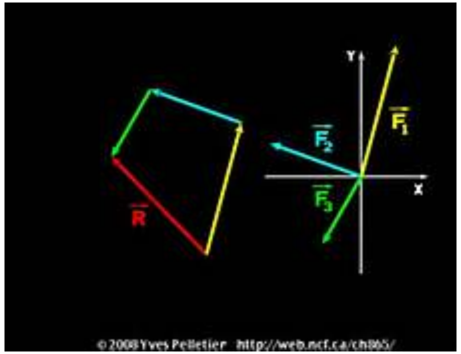
- Is animation a good idea in a web application?
- Something moving (changing) on the screen
- Properties
  - Can be one-time or continuous
  - Can be smooth or jerky
    - All animation is jerky, why does it appear smooth
    - Persistence of vision, frames per second
- Types of animation
  - Movies
  - Sound
  - Bitmap animation (canvas)
  - Vector animation (svg, flash)

Bitmap Animation



- Animation as a sequence of bitmaps
- Techniques
  - Animated GIFs
    - All images stored in a single GIF file
    - Browser takes care of the animation
  - JavaScript
    - Change the image associated with a region using timers
    - Let the browser then redraw the image
- Pros/cons
  - Simple to use, built into browser, tool support
  - Limited in what it can do

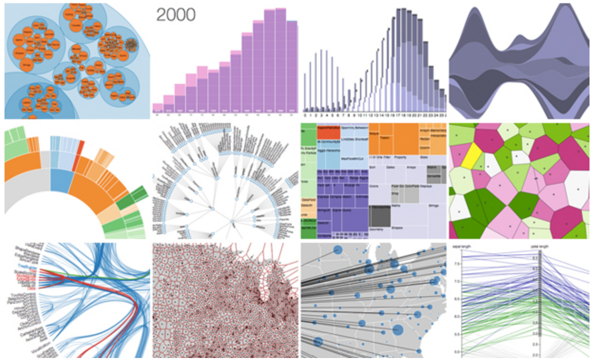
Vector Animation



- Consider a drawing program
  - You place objects on the screen
    - Circles, rectangles, lines, text
    - Each of these is parameterized
      - Position, size, angle, sting, color, line style, fill style, …
  - The result is an image
- Can create a sequence of images drawn this way
  - Next image is going to be similar to the previous
    - What is going to change
  - Change expressed in terms of changes to parameters
    - This makes it easy to define such a sequence
- Show the images at the appropriate speed
- Can be done using canvas/SVG

Data Visualization

- Canvas/SVG
- D3
  - http://www.youtube.com/watch?v=0oOC2FYNo1M



Next Class

- More Dynamic Web Pages

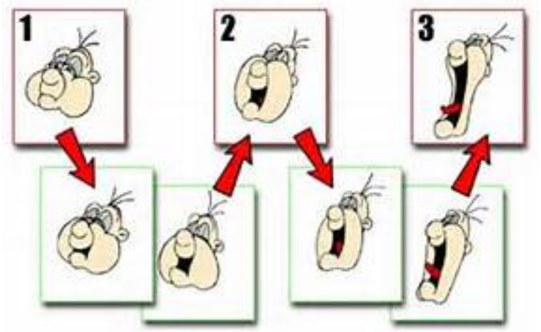
Videos

- Full bitmap images with automatic sequencing
- Typically encoded to save space
  - Data doesn’t change much from one image to another
  - Code key frames completely, otherwise just differences
  - Decoding should be fast and cheap

Sound

- Sound is nearly continuous
  - Time between values depends on frequency
    - To get a frequency response of 20K, need 40K sample/second
    - This means every 25 microseconds
    - But this is only one value, not a whole image
    - 8, 16, or 32 bits of data
- Again a variety of encodings are possible and used
- Synchronizing sound and image can be tricky
  - Handled by movie players
  - Handled by multimedia languages
- Sound in web applications can be annoying
- HTML5 <audio> tag

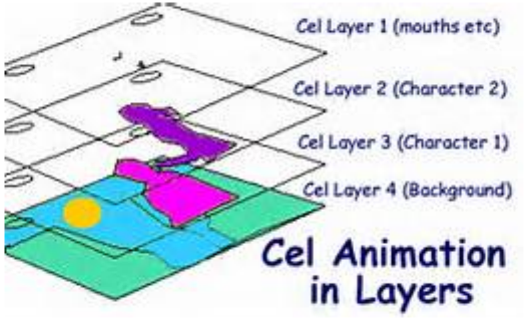
Making This Easier



- Key Frames
  - Suppose one takes 2 images K-frames apart
    - Parameters/positions change from one to the next
  - Have the computer construct the intermediate frames
    - Parameter values need to change from one value to the other
    - Interpolate values based on the end points
  - Types of interpolation
    - Linear: simplest to do
    - Path-based: user specifies a path (virtual or real)
- Example
  - https://www.youtube.com/watch?v=jJlAcTc5HUw

Making This Easier

- Layers
  - Split the drawing into different layers
  - Layers can be animated independently
    - Bottom layer(s) might be fixed (scenery/background)
    - Top layer(s) might be animated (person)
  - Can reuse the top layer or change its properties easily



Languages for Vector Animations

- Flash is the most widely known
- Microsoft Silverlight, Adobe AIR, JavaFX are alternatives
- These are implemented as browser plugins
  - Pretty much trusted
  - Provide a sandbox for executing programs
  - Include a scripting language for writing animations
  - Generally provide lots of other multimedia features
  - But not always available
- Java via Applets
- JavaScript implementations of these
- JavaScript using SVG

Question

What is a good rule of thumb for the minimum number of frames per second for a video or animation to look smooth to the human eye?

A. 10

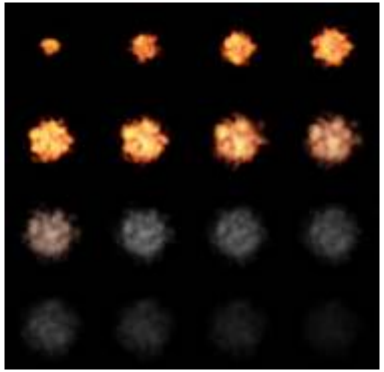
B. 30

C. 80

D. 240

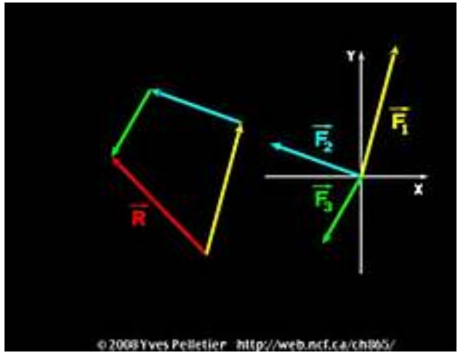
E. 1080

Bitmap Animation



- Animation as a sequence of bitmaps
- Techniques
  - Animated GIFs
    - All images stored in a single GIF file
    - Browser takes care of the animation
  - JavaScript
    - Change the image associated with a region using timers
    - Let the browser then redraw the image
- Pros/cons
  - Simple to use, built into browser, tool support
  - Limited in what it can do

Vector Animation



- Consider a drawing program
  - You place objects on the screen
    - Circles, rectangles, lines, text
    - Each of these is parameterized
      - Position, size, angle, sting, color, line style, fill style, …
  - The result is an image
- Can create a sequence of images drawn this way
  - Next image is going to be similar to the previous
    - What is going to change
  - Change expressed in terms of changes to parameters
    - This makes it easy to define such a sequence
- Show the images at the appropriate speed
- Can be done using canvas/SVG

Question

Which of the following are not built-in to HTML(5)?

A. 2D and 3D drawing areas
B. Input from a web or phone camera
C. Drag and drop
D. Geolocation with arbitrary updating
E. All are built into HTML5

Resize Experience

- What pages did you try?
  - What happened when you resized them
  - When did this "work"
  - When did it "fail"
- Do web sites use the same pages for phone & browser?
  - Why or why not?
- How might this be done?
  - Responsive applications