Lecture 2: Front End

* HTML5 – SGML-based, now only declare <!DOCTYPE html> and <html lang=”en”>
* Use specific tags instead of <div>, like <header> <footer>, to represent something
* New tags <article> and <section>, use article instead of a section when the content might be reproduced in the web
* <aside> used to replace the info on the sidebar
* HTML5 Forms: <input type=”x”>, can use email, url, range, number
* Cookies - data sent from website and stored in users browser, to notify previous activity
* Local Storage – can only be read client-side
* DOM enhancements – document.getElementByClassName()
* Previous, there was no standard way of showing video on the web, had to use Flash….Now HTML5 uses the <video> element
  + <video width=”w” height=”h” controls>…’controls’ indicates we should show them
  + Number of methods, attributes and listeners like v.play(), v.pause(), and v.load()
  + <audio> tag, similar to video tag
* Geolocation – gets users position using navigator.geolocation
  + navigator.geolocation.getCurrentPosition(successCallback[, failureCallback])
  + Callback function – function that is called once a task is completed
* Canvas API – used to draw graphics dynamically within a <canvas> tag, using JS
  + canvasElement.getContext(“id”)
* Other APIs include: Drag and Drop, App Cache (controls which content is cached)
  + Cache – component that transparently stores data so that future request can be served faster.
* CSS 3 – for browser compatibility, use <http://prefixr.com/>
* Media Queries, display content differently for different resolutions
  + @media screen and (min-width: 600px) and (max-width: 900px) {
    - .class { background: #333; } }
* Transitions – effects which allow property changes in CSS values to occur smoothly over a specified duration
* Object Oriented JavaScript
* Almost everything in JS are objects except primitives
* Primitives – string, number, Boolean, undefined, null
* Reference types (objects) – object, array, function, date
  + Objects have properties, ex: “hello”.length
* The “this” inside a function refers to the object in which the function operates
* Constructor – the method used to build an object

Ex: person = function(a,b) {

Var name = a; person.score = b; //a is a private property, and b is a public

} …then call with “Var John = new person(‘John’, 25);

* There is a constructor function that exists for all objects regardless of how created
* The instanceOf operator tests which constructor function was used to create an object
  + Ex: if (a instanceOf Object) alert(‘true’);
* Data and Accessor properties
  + Data can get and set a value, containing the value and writable in their description
  + Accessor calls a user-provided function, contains a get, set or both attribute.
* Getters and Setters – instead of directly accessing properties, you can make them private
* Prototype / Inheritance – to create an object that will be used as a template for creating new objects
* Prototype Chaining – each object has an internal link to another object called its prototype….null has no prototype and is the final link in the chain.