CS144 Notes: AJAX

What is Web 2.0 application?

- <show examples of AJAX application>
  - Yahoo map: maps.yahoo.com
  - Google suggest: www.google.com/webhp?complete=1

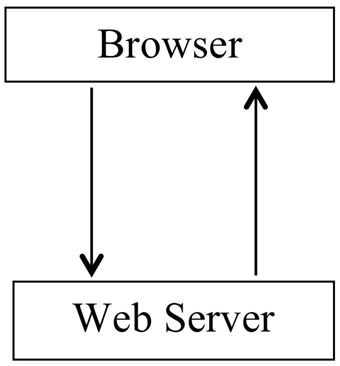
- **Q:** Web 2.0 is based on AJAX. What does AJAX mean?
  - AJAX: Asynchronous javascript and XML
    - the term first coined by Jesse James Garrett in Feb 2005
    - http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

- **Q:** AJAX vs traditional Web interface? What is new?
  - Previously, form based input
    - press "submit" button and wait until the entire page reloads
    - significant delay for interaction
  - AJAX
    - "in-place" update of page content
    - more "desktop" application like a feel

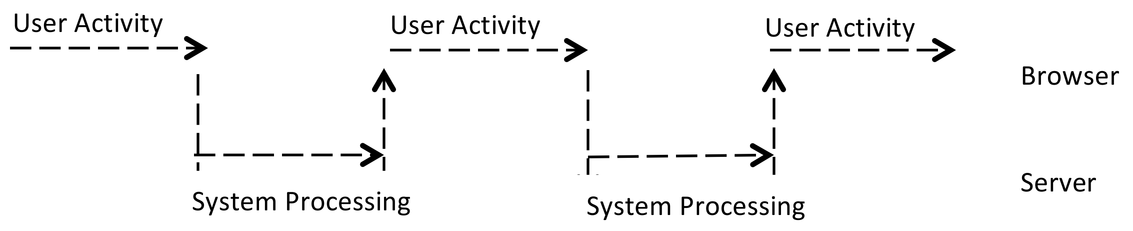
  → Started a whole bunch of companies porting existing application to AJAX style
    - mail, office applications, photos

- **Q:** How does an AJAX application work?

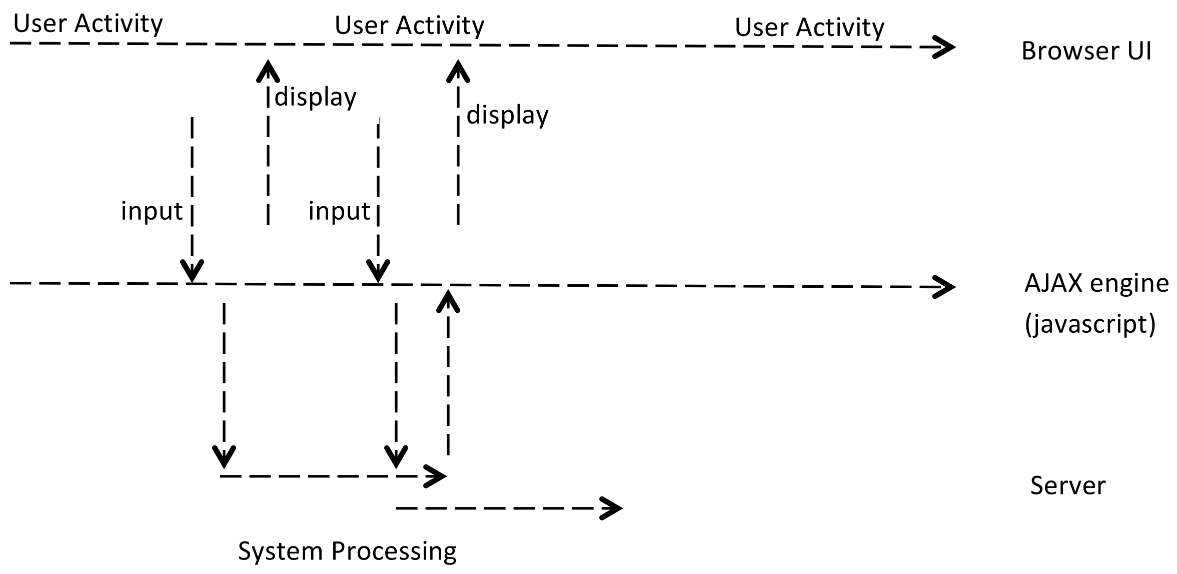
<interaction diagram of AJAX vs Web>



classic web application model (synchronous)



ajax application model (asynchronous)



- **Q:** How is the sequence of execution determined?
  - Event-Driven Programming:
    - * flow of program is driven by events
      - user interaction
      - server response

        ….

- **Q:** What is needed to support this interaction?

- dynamic in-place update of Web page
    - document object model (dom): what part of document
    - javascript: how to change on what event

- asynchronous interaction with the server and the user
    - XMLHttpRequest
    - Events on HTML DOM
    - event-driven callback function

<Background-color change example for Javascript and DOM>

- http://oak.cs.ucla.edu/cs144/examples/javascript.html

- **Q:** What should the browser do for this demo page?
    - monitor "clicks" on the page
    - when clicked, change the background color
    - DOM:
        - specify a particular part of the document events and properties
    - Javascript:
        - specify the actions to take

Javascript

- simple script in a Web page that is interpreted and run by the browser
  - supported by most mordern browsers
  - allows dynamic update of a web page

- Basic syntax:

```
<script type="text/javascript">
<!--
... javascript code ...
-->
</script>
```

  - <script> may appear at any point of an HTML page
  - javascript functions should be inside the <HEAD> </HEAD> tags
    - to load the functions before the page begins to display
    - <!-- --> ensures that even if a browser does not understand the <script> tag, it does not display the code on the Web page.

- basic keywords and syntax
  - almost identical to java/c
    - if (cond) { stmt; } else { stmt; }
    - for (i=0; i < 10; i++) { stmt; }

    …..
    - case is important like in java/c
  - var name=value; // variables do not have a static type
    - "var" is optional but recommended
      - Without "var", a variable becomes a global variable
    -
    ```
    function function_name(parameter1, parameter2)
    {
       ... function body ...

       return value;
    }
    ```
    - comparison operator == does automatic type conversion

=== checks for both type and value

    * inequality operator is != (like Java, but different from C)

- types and objects in javascript
  - three important primitive data type: string, number, boolean (true or false)
    * all numbers are represented as floating point
  - Javascript is loosely typed
    * variables do not have a static type. any type value can be assigned.

```
var a = 10;  // a has number type value
a = "good";  // a has string type value
```

    * typeof() returns the type of the current value
    * automatic conversion to a "reasonable" type when multiple types are used
      - surprises once in a while
        e.g., 1+"2" = "12"
      - to force numeric conversion, use Number(..), Boolean(..), String(..), parseFloat(..), parseInt(..)
  - String: one of three primitive types in javascript
    * length property returns the length of string
    * useful functions: charAt(), substring(), indexOf():

```
var a = "abcdef";
b = a.substring(1, 4);   // b = "bcd"
```

  - Array: Array(), constant - [1, 2, 3]
    * length property returns the size of the array
      - can be used to resize array as well (by setting its value)

```
var a = new Array();
a[0] = 3;
a[2] = "string";

var b = new Array(1, 2, 3);
var c = [1, 2, 3];
var sizec = c.length;  // sizec is 3
```

* array elements do not have to be uniform

```
var a = [1, "good", [2, 3] ];
```

* useful functions of Array
  - mutators: reverse, sort, push, pop, shift, unshift
  - accessors: concat, join (into a string), slice

```
var a = [1, 2, 3, 4];
b = a.slice(1, 3);   // b = [2, 3]
```

- Composite datatype: Object(), constant - { x:1, y:2 }
  * allows OOP style programming

```
var o = new Object();
o.x = 1;
o.y = "s";
 var p = { x:1, y:2, z:{ x:3, y:4 } };  // nested properties
                                        // are possible
```

  * Note: o["x"] is the same as o.x, so object properties are essentially an associative array.
  * Object assignment is *by reference* not by value
  * all non-primitive types are "objects"
    - including array for example
- two special values: undefined and null
  * two are often interchangeably used for uninitialized property value but if we really want to be precise
    - when a property has no definition or its value has not been assigned it is undefined not a null
      - undefined is a primitive value and is undefined type
      - null is an object and is null type

      e.g., document.undefinedvar == null   -> true
            document.undefinedvar === null   -> false
            document.undefinedvar == undefined -> true
            document.undefinedvar === undefined -> true

HTML DOM (Document Object Model)

- Tree-based model of HTML tag elements on a page
    - an HTML DOM object (= a node in the DOM tree) may have
        * child object
        * properties
        * methods
        * associated events
    - one HTML tag element becomes one node in the DOM
    - any text inside an HTML element creates a separate child "text node"

      e.g., <h1>Heading</h1> creates two nodes:

      "h1" element node and its child text node of "Heading"
        * note: Firefox creates a text node for empty white-space or new lines. Internet Explorer does not.
    - any attribute of an element creates an "attribute" node
        * attribute node is not a child node
    - <example>

```
<html>
<head><title>Page Title</title></head>
<body><h1>Heading</h1><a href="good/">Link</a></body>
</html>
```

* HTML DOM Tree
    - "document" becomes the root node of the HTML DOM tree in javascript
    - each node is of particular type
        - type: element, text, attribute, comment, ...
    - each node may be associated with name and value
        - name: html, head, title, body, h1, a, ...
        - value: Page Title, Heading, good/, Link, ...
            * Note: the attribute node of the "a" node is not a's child

- Manipulating an HTML DOM object (= a DOM tree node) on a page
  - common root object: document (also, window, navigator...)
  - then obtain the desired node by calling one of the following "methods" of the root object
    - document.getElementById('id')
    - document.getElementsByTagName('p')
    - document.body: special way to access the "body" element of document
      - document.forms["formname"], document.images[0], ...
  - each DOM object is associated with a set of properties and methods
    - Properties and methods can be read/written/called
      - document.body.style.background = "yellow"; // background color
      - document.body.innerHTML; // everything between <body> ... </body>
      - document.getElementById('myform1').reset(); // reset the form
  - each DOM object may be associated with a set of "events"
    - when user takes an action, an event is invoked for the relevant object
    - events are handled by an event handler of the object
      - onLoad, onUnload, onClick, onMouseOver, onMouseOut, onKeyUp
    - event handler can be set to a particular function

      e.g.)

```
onClick="ChangeColor();"            // inside element tag
document.body.onClick = ChangeColor;   // inside script
```

  - See http://www.javascriptkit.com/jsref/ for the list of DOM object properties, methods and events

<show the example code and ask them to read it>

```
      <html>
      <head>
        <script type="text/javascript">
          var colors = new Array("yellow", "blue", "red");
          var i=0;
          function ChangeColor() { document.body.style.background =
  colors[i++%3]; }
        </script>
      </head>
      <body onClick="ChangeColor();">Click on this document!</body>
      </html>
```

< explain that for dynamic update of the page >

(1) we need to set event handler for important events

(2) the event handler should take the appropriate action

- creating a new element on a page
  - createElement(), createTextNode(), appendChild(), removeChild(), replaceChild(), ...

```
var newdiv=document.createElement("div")
var newtext=document.createTextNode("A new div")
newdiv.appendChild(newtext) //append text to new div
document.getElementById("test").appendChild(newdiv) //append new div
```

  - innerHTML: allows direct manipulation of a node

```
document.body.innerHTML = "<h1>New title</h1>"
```

    * no need to call createElement("h1"), createTextNode("New title"), ...
    * non-standard, but still very popular due to its simplicity
  - Note: HTML DOM manipulation can be done only after the page has been loaded, not earlier.

XMLHttpRequest

<show google suggest example interaction>

http://oak.cs.ucla.edu/cs144/examples/google-suggest.html

- **Q:** What is going on behind the scene? What events does it monitor? What does it do when the event is detected?

- **Q:** When the "typing" event is detected, what does it have to do? How can it let users keep type while waiting for data from server?

- XMLHttpRequest: object for asynchronous communication with the server
- created differently depending on the browser
  - IE 7+, non-IE: new XMLHttpRequest()
  - IE 6+: new ActiveXObject("Msxml2.XMLHTTP")
  - IE 5.5+: new ActiveXObject("Microsoft.XMLHTTP")

  e.g., xmlHttp = new XMLHttpRequest();
- sending the request to the server: open() and send() methods

```
xmlHttp.open("GET", URL);   // method, url
xmlHttp.send(null);         // optional body of the request
```

*** Remark: same origin policy ***

* the request can be made only to the host of the web page

* cannot be used to get results from other web services

- handling server response

  Important properties of XMLHttpRequest elements:

  - onreadystatechange: event handler function for the server response

    xmlHttp.onreadystatechange = handlerfunction;

  - readyState: the status of the server's response

    0: The request is not initialized

    1: The request has been set up

    2: The request has been sent

    3: The request is in process

    4: The request is complete

  - responseText/responseXML: the data sent back from the server

    * responseText is text. responseXML is XML DOM

<show Google suggest code. ask them read it>

```
<html>
  <head>
  <script type="text/javascript">
  var xmlHttp = new XMLHttpRequest(); // works only for Firefox, Safari, ...

  // send Google suggest request based on the user input
  function sendAjaxRequest(input)
  {
    var request = "google-suggest.php?q="+encodeURI(input);

    xmlHttp.open("GET", request);
    xmlHttp.onreadystatechange = showSuggestion;
    xmlHttp.send(null);
  }

  // update Web page with the response from Google suggest
  function showSuggestion() {
    if (xmlHttp.readyState == 4) {
      response = xmlHttp.responseText;
      response = response.replace(/</g, "&lt;");
      response = response.replace(/>/g, "&gt;");
      document.getElementById("suggestion").innerHTML = response;
    }
  }
  </script>
  </head>
  <body>
    <b>Your query:</b> <input type="text" onKeyUp="sendAjaxRequest(this.value);"><br/>
    <b>Suggestion</b>: <pre id="suggestion"></pre>
  </body>
  </html>
```

<let students to read the code. ask them questions on what it does and explain relevant parts of the code>

** Note **

this: the current element

innerHTML: non-standard way of updating the text of an object

- **Q:** What events does it monitor?

- **Q:** What does it do when the event is detected? What URL does it use to send request?

- **Q:** When it receives response from the server, what does it do?

- **Q:** Could the XMLHttpRequest have been sent directly to Google?

* Note: same origin policy and the need for proxy

XML and JSON in javascript

<show google suggest v2 interaction>

http://oak.cs.ucla.edu/cs144/examples/google-suggest2.html

- * Remark: In most case, we have to process the response from the server and use part of it, instead of displaying it directly. How can we do it?

- Typeical server responses for AJAX applications:
  - The server response is often in XML, but JSON is gaining popularity
    - * responseXML is the parsed XML DOM
      - responseXML.documentElement: the root XML element
    - * JSON result should be processed from responseText

<show Google suggest v2 code>

```
function showSuggestion() {
  if (xmlHttp.readyState == 4) {
    // get the CompleteSuggestion elements from the response
    var s = xmlHttp.responseXML.getElementsByTagName('CompleteSuggestion');

    // construct a bullet list from the suggestions
    var htmlCode = "<ul>";
    for (i = 0; i < s.length; i++) {
       var text = s[i].childNodes[0].getAttribute("data");
       htmlCode += "<li><b>" + text + "</b> (" + count + " queries)";
    }
    htmlCode += "</ul>";

    // display the list on the page
    document.getElementById("suggestion").innerHTML = htmlCode;
  }
}
```

- **Q:** How does it access the relevant part of response?

- JSON (Javascript object notation)
  - The standard javascript syntax to represent "constant"

    e.g., [{ x: 3, y:"Good", z:{ a:1, b:2 } }, { x: 4, y:"Bad", z:3}]

    - Q: What does the above notation mean in javascript?

  - eval() function "evaluates" a text and return the results

    * JSON text can be "parsed" into javascript objects through eval()

```
 var x = '[ { x: 3, y:\"Good\", z:{ a:1, b:2 } },  \
            { x: 4, y:\"Bad\",  z:3} ]';
var o = eval(x);
```

    * once eval()ed, we can access its value as a standard javascript object

```
   var n = o[0].x + o[0].z.a + o[1].z;
```

      - Q: What will be the value of n?

Animation effects in AJAX

- e.g., scrolling news tickers, flying boxes, ...

  <show WSJ ticker example at the top>

  - **Q:** How can we simulate animation effect?

- Important functions/properties for animation
  - setTimeout("event_handler", interval): time-based event generator
  - element.style: allows modifying CSS styles
    - div.style.left: left margin,
    - div.style.right: top margin,
    - div.style.width: width, ...

- Example: http://oak.cs.ucla.edu/cs144/examples/ticker.html

  (show what page does, let students read the code)

```
<html>
<head>
  <script type="text/javascript">
    var ticker;
    var tickerText = "Hello, there...";

    function tickerStart() {
      ticker = document.getElementById("ticker");
      ticker.innerHTML = tickerText;
      setTimeout("tickerSlide(10)", 100);
    }

    function tickerSlide(x) {
      var newLeft = parseInt(ticker.style.left) + parseInt(x);
      if (newLeft > 300) newLeft = 0;
      ticker.style.left = String(newLeft) + "px";
      setTimeout("tickerSlide(10)", 100);
    }
  </script>
</head>
<body onLoad="tickerStart();">
  <div id="ticker" style="position: absolute; left: 0px;"></div>
</body>
</html>
```

  - Note: "position" property allows setting an element location

* fixed: element location cannot be set. only default location
* absolute: element location is set by absolute coordinate
* relative: element location is set relative to the default location
    - Q: How is the text "Hello, there..." assigned to ticker div?
      What sequence of function calls?
    - Q: Why does the text move? What sequence of function calls?

    - Q: What if we set ticker variable when we declare it first?
      Is it necessary to set the variable inside startTicker?

- Q: http://oak.cs.ucla.edu/cs144/examples/box.html
  What will the following page do?

```
<html>
<head>
  <script type="text/javascript">
    var box;

    function boxStart() {
      box = document.getElementById("box");
      box.style.width = "200px";
      box.style.height = "200px";
      box.style.border = "solid 5px black";
      setTimeout("shrinkBox(5)", 80);
    }

    function shrinkBox(x) {
      var newSize = parseInt(box.style.width) - parseInt(x);
      if (newSize < 0) newSize = 200;
      box.style.width = String(newSize) + "px";
      box.style.height = String(newSize) + "px";
      setTimeout("shrinkBox(5)", 80);
    }
  </script>
</head>
<body onLoad="boxStart();">
  <div id="box"></div>
</body>
</html>
```

- CSS3 animation: @keyframes rules and animation property

```
@keyframes css3animation
{
   0%    { background: red; }
   50%   { background: blue; }
   100%  { background: yellow; }
}

div
{
   animation: css3animation 5s;  /* apply css3animation over 5 seconds
*/
}
```

  - other relevant CSS3 properties
    * animation-delay: when the animation will start
    * animation-play-state: whether the animation is running or paused
    * animation-iteration-count: # of times animation is played (or “infinite”)
  - For WebKit based browsers (Chrome, Safari, Opera) prefix all names with “`-webkit-`”, such as “`@-webkit-keyframes`”
  - Example: http://oak.cs.ucla.edu/cs144/examples/css-animation.html

HTML5

- Provide well-defined logic to translate "ill-defined" documents into compliant documents
  - more consistent rendering between browsers
- Standardize what is often done in an "ad-hoc" manner or what is critically needed to build full-blown Web apps
  - Videos (video element)
  - Offline storage (localStorage and sessionStorage)
  - Dynamic graphics (canvas element)
  - doucment editing and drag and drop (designMode and contentEditable attributes)
  - and many more

    e.g., Video

```
<video src="cs144.mp4" width="320" height="240"></video>
```

  - Video becomes a first-class citizen like an image.
  - HTML5 is codec agnostic, but browsers are expected to support "popular" codecs like JPG, PNG for images
    * contraversy on the licensing issue for H.264 due to past experience from GIF and MP3
  - e.g., Persistent offline storage
    * localStorage vs sessionStorage (per domain vs per page)
      - e.g., localStorage["location"] = "UCLA";
  - e.g., Dynamic graphics

```
<canvas width="100" height="200"></canvas>
```

    * We can draw on canvas elements using javascript using functions like rectFill(10, 20, 50, 20)
    * Canvas avoids performance problem of SVG (scalable vector graphics)
      - no need to maintain dom structure for each vector element
      - more suitable for applications like games

References

- Tutorials
  - Javascript: http://en.wikipedia.org/wiki/JavaScript_syntax
  - DOM: http://www.w3schools.com/htmldom/
  - XMLHttpRequest: http://en.wikipedia.org/wiki/XMLHttpRequest
- References
  - Javascript and HTML DOM: http://www.javascriptkit.com/jsref/
  - DOM: http://www.w3schools.com/DOM/default.asp
- Popular javascript libraries
  - jQuery, Scriptaculous, Dojo, GWT (Google Web Toolkit), YUI (Yahoo User Interface library), ...