From pure server to client/server computation

- So far we have seen pure server-side programming

Next
- Enrich user experience, interactivity with client-side computations (JavaScript)
  - For example, validate that the user typed a number in a textbox
- Combine the best of both worlds with Ajax technologies
  - Assignment focus: live views

Architecturally speaking: Pure server-side programming
Architecturally speaking: Client-side programming w Javascript

User events may lead to activation of Javascript code, evaluated by the browser. They lead either to alerts, prompts, etc or to modification of part of the page.

Architecturally speaking: Ajax programming (@10 miles high)

Event leads to Javascript activation, leading to http request

Server responds with data that are enough for the page update.

Preview: From basics to higher-level programming

1. First examples will demonstrate the essentials:
   - Directly accessing and manipulating the DOM object representation of the data shown on the browser
   - Packing & unpacking response data via XML
   - Not part of assignment – just broad knowledge
2. Making Javascript/DOM programming easier with utilities
   - Transferring data via JSON
   - Jquery
3. (much later) High level frameworks where you can altogether avoid Javascript
   - FORWARD
   - Ruby-on-Rails partials
   - GWT
JavaScript

- Programming language embedded in HTML
  - Directly or indirectly
- Evaluated by the browser, interpreted
- Triggered on page load and on certain programmer-defined events
- While OO, it allows weak typing and many oddities
  - Great opportunities for making a coding mess

JavaScript Example 1

```html
<html>
  <body>
    <script type="text/javascript">
      document.write("Hello World!");
    </script>
  </body>
</html>
```

DOM object, standing for entirety of displayed HTML

JavaScript Example 2

```html
<html>
  <head>
    <script type="text/javascript">
      function showMessage() { alert("Hello!"); }
    </script>
  </head>
  <body>
    <form>
      <input type="button" value="Click me!" onclick="showMessage()" />
    </form>
  </body>
</html>
```
Basics

• Incorporate code in `<script>` element
• Code in `<body>` part evaluates on page load
• Code in `<head>` part are typically functions waiting for events triggered by the user’s activity on the browser

• Typical control structures
  – Statements, conditionals, loops, functions...
• Typical expressions
• JavaScript can access and modify the HTML document and its parts (HTML elements) currently displayed

Specific available objects

• Predefined JavaScript objects:
  – **Window**: Represents a browser window
  – **Navigator**: Contains browser info
  – **Screen**: Contains client screen info
  – **History**: Visited URLs within a browser window (tricky)
  – **Location**: Info about the current URL

• The displayed HTML’s DOM tree
  – **Document**: Top of navigation
  – **Area**: Areas you may have defined inside maps
  – **Form
  – **Option
  – …

JavaScript Example 3

```html
<html>
<body>
<script type="text/javascript">
// Write "Good Evening" if time >16 and <21
var d = new Date();
var time = d.getHours();
if (time < 21 && time > 16)
document.write("<b>Good Evening</b>");
else
document.write("<b>Hello</b>");
</script>
</body>
</html>`
**Interaction Basics: Popup Boxes**

- Alerts
  - Make sure the user saw something
- Confirmations
  - Click either "OK" or "Cancel" to proceed
- Prompts

**JavaScript Example 4**

```html
<html>
<body>
  ...
  <script type="text/javascript">
    response = confirm("If you proceed we'll charge your card");
    document.write(response);
  </script>
</body>
</html>
```

**JavaScript Example 5**

```html
<html>
<body>
  ...
  <script type="text/javascript">
    response = prompt("The page will be whatever you type here", "default");
    document.write(response);
  </script>
</body>
</html>
```
Events

- Elements of a page have associated events
  - Mouse click on a button
  - Mouse over the element's area
  - Start typing in (selecting) an input box
- Trigger function upon event

JavaScript Example 6

```html
<html>
  <head>
    <script type="text/javascript">
      function displayMsg() { alert("This is Mars!"); }
    </script>
  </head>
  <body>
    <img src="earth.jpg">
    <br />
    <img onmouseover="displayMsg()" src="mars.jpg">
  </body>
</html>
```

When Should You Use JavaScript?

- Client-side form validation
  - Avoid roundtrips to the server for simple validation cases
- Form dependencies
  - Particular forms become irrelevant in light of answers typed in other forms
- Fancy stuff popping up
  - But avoid hiding important information in various forms of popups
- Client side computing of cookie-related niceties
  - We'll see along with HTML5
- Browser environment issues
Invoke Function Upon Event – Example 8

```html
<head>
  <script type="text/javascript" src="javascript/example08.js"></script>
</head>
<body>
  <form action="nowhere" onsubmit="return validate()">
    Name (max 10 characters):
    <input type="text" id="fname" name="fname" size="20">
    Age (from 1 to 100):
    <input type="text" id="age" name="age" size="20">
    E-mail:
    <input type="text" id="email" name="email" size="20">
    <input type="submit" value="Submit">
  </form>
</body>
```

In Ajax, we will get rid of the form element. We'll just have a button element.

... and Validate Values – Example 8

```javascript
function validate() {
  var at=document.getElementById("email").value.indexOf("@");
  var age=document.getElementById("age").value;
  var fname=document.getElementById("fname").value;
  var submitOK="true";
  if (fname.length > 10) {
    alert("The name may have no more than 10 characters");
    submitOK="false"; }
  if (isNaN(age) || age < 1 || age > 100) {
    alert("The age must be a number between 1 and 100");
    submitOK="false"; }
  if (at == -1) {
    alert("Not a valid e-mail!");
    submitOK="false"; }
  if (submitOK=="false") { return false; }
}
```

How To Access?

- Navigation from the top
- Search for elements using any of multiple possible ways
- Access by **ID** – my preferred technique, definitely so when jQuery is not used
  - But be disciplined about creating IDs
- Typically associate HTML elements that will be modified by JavaScript with IDs
  - You can use a `<span>` element if you want to associate an area with an ID
Dependencies – Example 9

```html
<body>
  <form>
    <h4>Questionaire:</h4>
    <fieldset>
      <legend>Gender: </legend>
      <select id="gender" onchange="enableDisable()">
        <option value="">Female</option>
        <option value="">Male</option>
      </select>
      <br>
      Are you pregnant? <select id="pregnant">
        <option value="">No</option>
        <option value="">Yes</option>
      </select>
    </fieldset>
  </form>
</body>
```

Dependencies – Example 9 (cont’d)

```javascript
function enableDisable() {
  if (document.getElementById("gender").selectedIndex == 1) {
    document.getElementById("pregnant").disabled = true
  } else {
    document.getElementById("pregnant").disabled = false
  }
}
```

JavaScript Example 10 (dismiss, we will do the same with HTML5, cleaner)

```javascript
function getCookie(c_name) {
  if (document.cookie.length > 0) {
    c_start = document.cookie.indexOf(c_name + "=");
    c_start = c_start + c_name.length + 1;
    c_end = document.cookie.indexOf(";",c_start);
    if (c_end == -1) {c_end = document.cookie.length;
      return unescape(document.cookie.substring(c_start,c_end));
    }
    return ""
  }
}
function setCookie(c_name, value, expdays) {
    var exp = new Date();
    exp.setDate(exp.getDate() + expdays);
    document.cookie = c_name + "=" + escape(value) + ((expdays==null) ? "" : "; expires=" + exp.toGMTString());
}

function checkCookie() {
    username = getCookie('username');
    if (username != null && username != "")
        alert('Welcome again ' + username + '!');
    else {
        username = prompt('Please enter your name:','');
        if (username != null && username! = "")
            setCookie('username', username, 365);
    }
}