MORE JAVASCRIPT AND FORMS

Continuing with JavaScript and Forms
Announcements

- At the demo on Monday, we will review Assignments 3, 4, and 5
Discuss questions with your Scrum Team
Quiz

1. What type of input do you use if you want to send it with the form, but don’t want the user to interact with it?

2. Should you use get or post to send an image to a server?

3. Write PHP code to get the list of files with a .txt extension from a directory, whose name is passed in as test.php?dir="dir";
And the answer is ...
Quiz (Answers)

1. Form info without user interaction?
   - Hidden

2. get or post to send an image to a server?
   - Post, urls are limited to about 1024 characters.

3. Write PHP code to get the list of files a directory, whose name is passed in as test.php?dir="dir";

```php
<?php
    $directory = $_REQUEST["dir"];  
    $reviews = glob("$directory/*.txt")
?>
```
<?php
    $movie = $_REQUEST["film"];  
    $reviews = glob("moviefiles/$movie/review*.txt")
?>
More forms
Reset Buttons

Name: <input type="text" name="name" /><br />
Food: <input type="text" name="meal" value="pizza" /><br />
<label>Meat? <input type="checkbox" name="meat" /></label><br />
<input type="reset" />

.specify custom text on the button by setting its value attribute

CSC 210
Grouping input: `<fieldset>`, `<legend>`

```html
<fieldset>
  <legend>Credit cards:</legend>
  <input type="radio" name="cc" value="visa" checked="checked" /> Visa
  <input type="radio" name="cc" value="mastercard" /> MasterCard
  <input type="radio" name="cc" value="amex" /> American Express
</fieldset>
```

- `<fieldset>` groups related input fields, adds a border; `<legend>` supplies a caption
“I changed the form's HTML code ... but when I refresh, the page doesn't update!”

By default, when you refresh a page, it leaves the previous values in all form controls.

- it does this in case you were filling out a long form and needed to refresh/return to it
- if you want it to clear out all UI controls' state and values, you must do a full refresh
  - Firefox: Shift-Ctrl-R
  - Mac: Shift-Command-R
Styling form controls

- attribute selector: matches only elements that have a particular attribute value
- useful for controls because many share the same element (input)

```css
input[type="text"] {
  background-color: yellow;
  font-weight: bold;
}
```
Hidden input parameters

- an invisible parameter that is still passed to the server when form is submitted
- useful for passing on additional state that isn't modified by the user

```html
<input type="text" name="username" /> Name
<br />
<input type="text" name="sid" /> SID
<br />
<input type="hidden" name="school" value="UW" />
<input type="hidden" name="year" value="2048" />
```
Submitting data
Problems with submitting data

- the form may look correct, but when you submit it...

- `[cc] => on, [startrek] => Jean-Luc Picard

- How can we resolve this conflict?
The **value** attribute

- value attribute sets what will be submitted if a control is selected
- `[cc] => visa, [startrek] => picard

```html
<label><input type="radio" name="cc" value="visa" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" /> MasterCard</label> <br />
Favorite Star Trek captain:
<select name="startrek">
   <option value="kirk">James T. Kirk</option>
   <option value="picard">Jean-Luc Picard</option>
<input type="submit" value="submit" />
</select> <br />
```

HTML
URL-encoding

- Certain characters are not allowed in URL query parameters:
  - Examples: " ", "/", "=" , "&"

- When passing a parameter, it is URL-encoded
  - “Xenia's cool!?" → “Xenia%27s+cool%3F%21"

- You don't usually need to worry about this:
  - The browser automatically encodes parameters before sending them
  - The PHP $_REQUEST array automatically decodes them
  - ... but occasionally the encoded version does pop up (e.g. in Firebug)
Submitting data to a web server

- though browsers mostly retrieve data, sometimes you want to submit data to a server
  - Hotmail: Send a message
  - Flickr: Upload a photo
  - Google Calendar: Create an appointment

- the data is sent in HTTP requests to the server
  - with HTML forms
  - with **Ajax** (seen later)

- the data is placed into the request as parameters
HTTP GET vs. POST requests

- **GET**: asks a server for a page or data
  - if the request has parameters, they are sent in the URL as a query string

- **POST**: submits data to a web server and retrieves the server's response
  - if the request has parameters, they are embedded in the request's HTTP packet, not the URL
HTTP GET vs. POST requests

- For submitting data, a POST request is more appropriate than a GET
  - GET requests embed their parameters in their URLs
  - URLs are limited in length (~ 1024 characters)
  - URLs cannot contain special characters without encoding
  - private data in a URL can be seen or modified by users
Form **POST** example

```html
<form action="http://localhost/app.php" method="post">
  <div>
    Name: <input type="text" name="name" /> <br />
    Food: <input type="text" name="meal" /> <br />
    <label>Meat? <input type="checkbox" name="meat" /></label> <br />
  </div>
  <br />
  <input type="submit" />
</form>
```

**HTML**
GET or POST?

- some PHP pages process both GET and POST requests
- to find out which kind of request we are currently processing, look at the global `$_SERVER` array's "REQUEST_METHOD" element

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    # process a GET request
    ...
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
    # process a POST request
    ...
}
```
Uploading files

- add a file upload to your form as an input tag with type of file
- must also set the enctype attribute of the form

```html
  Upload an image as your avatar:
  <input type="file" name="avatar" />
  <input type="submit" />
</form>
```
Processing form data in PHP
### "Superglobal" arrays

<table>
<thead>
<tr>
<th>Array</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$_REQUEST</code></td>
<td>parameters passed to any type of request</td>
</tr>
<tr>
<td><code>$_GET, $_POST</code></td>
<td>parameters passed to GET and POST requests</td>
</tr>
<tr>
<td><code>$_SERVER, $_ENV</code></td>
<td>information about the web server</td>
</tr>
<tr>
<td><code>$_FILES</code></td>
<td>files uploaded with the web request</td>
</tr>
<tr>
<td><code>$_SESSION, $_COOKIE</code></td>
<td>&quot;cookies&quot; used to identify the user (seen later)</td>
</tr>
</tbody>
</table>

- PHP superglobal arrays contain information about the current request, server, etc.
- These are special kinds of arrays called associative arrays.
Associative arrays

- associative array (a.k.a. map, dictionary, hash table): uses non-integer indexes
- associates a particular index "key" with a value
  - key "xenia" maps to value "206-685-2181"

```php
$blackbook = array();
$blackbook["xenia"] = "206-685-2181";
$blackbook["anne"] = "206-685-9138";
...
print "Xenia's number is " . $blackbook["xenia"] . "\n";
```
Example: exponents

```php
<?php
    $base = $_REQUEST["base"];  
    $exp = $_REQUEST["exponent"];  
    $result = pow($base, $exp);  
?>

```

What should we do to run this with XAMPP?
Example: Print all parameters

```php
<?php
foreach ($_REQUEST as $param => $value) {
    ?><p>Parameter <b><?</b>$param</b> has value <b><?</b>$value</b></p><p><?
}
?>
```

- What should we do to run this with xampp?
Processing an uploaded file in PHP

- uploaded files are placed into global array `$_FILES`, not `$_REQUEST`
- each element of `$_FILES` is itself an associative array, containing:
  - `name`: the local filename that the user uploaded
  - `type`: the MIME type of data that was uploaded, such as `image/jpeg`
  - `size`: file's size in bytes
  - `tmp_name`: a filename where PHP has temporarily saved the uploaded file

  - to permanently store the file, move it from this location into some other file
example: if you upload tobbay.jpg as a parameter named avatar,

- $_FILES["avatar"]['name'] will be "tobby.jpg"
- $_FILES["avatar"]['type'] will be "image/jpeg"
- $_FILES["avatar"]['tmp_name'] will be something like "/var/tmp/phpZtR4TI"
Array
(
    [file1] => Array
        (
            [name] => MyFile.txt (comes from the browser, so treat as tainted)
            [type] => text/plain (not sure where it gets this from - assume the browser, so treat as tainted)
            [tmp_name] => /tmp/php/php1h4j10 (could be anywhere on your system, depending on your config settings, but the user has no control, so this isn't tainted)
            [error] => UPLOAD_ERR_OK (= 0)
            [size] => 123 (the size in bytes)
        )
    [file2] => Array
        (
            [name] => MyFile.jpg
            [type] => image/jpeg
            [tmp_name] => /tmp/php/php6hst32
            [error] => UPLOAD_ERR_OK
            [size] => 98174
        )
)
functions for dealing with uploaded files:

- is_uploaded_file(filename)
  returns TRUE if the given filename was uploaded by the user

- move_uploaded_file(from, to)
  moves from a temporary file location to a more permanent file
Including files: **include**

- inserts the entire contents of the given file into the PHP script's output page
- encourages modularity
- useful for defining reused functions needed by multiple pages

```php
include("header.php");
```
DOM and timers
Problems with JavaScript

JavaScript is a powerful language, but it has many flaws:

- the DOM can be clunky to use
- the same code doesn't always work the same way in every browser
  - code that works great in Firefox, Safari, ... will fail in IE and vice versa
- many developers work around these problems with hacks (checking if browser is IE, etc.)
Prototype framework

- the Prototype JavaScript library adds many useful features to JavaScript:
  - many useful extensions to the DOM
  - added methods to String, Array, Date, Number, Object
  - improves event-driven programming
  - many cross-browser compatibility fixes
  - makes Ajax programming easier (seen later)

```html
<script src="https://ajax.googleapis.com/ajax/libs/prototype/1.7.0.0/prototype.js" type="text/javascript"></script>
```
The $ function

- returns the DOM object representing the element with the given id
- short for `document.getElementById("id")`
- often used to write more concise DOM code:

```javascript
$('id')

$('footer').innerHTML = $('username').value.toUpperCase();
```
DOM element objects

HTML

```html
<p>
Look at this octopus:
<img src="octopus.jpg" alt="an octopus" id="icon01" />
Cute, huh?
</p>
```

DOM Element Object

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>tagName</td>
<td>&quot;IMG&quot;</td>
</tr>
<tr>
<td>src</td>
<td>&quot;octopus.jpg&quot;</td>
</tr>
<tr>
<td>alt</td>
<td>&quot;an octopus&quot;</td>
</tr>
<tr>
<td>id</td>
<td>&quot;icon01&quot;</td>
</tr>
</tbody>
</table>

JavaScript

```javascript
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```
# DOM object properties

```html
<div id="main" class="foo bar">
<p>Hello, I am <em>very</em> happy to see you!</p>
<img id="icon" src="images/potter.jpg" alt="Potter" />
</div>
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tagName</code></td>
<td>element's HTML tag</td>
<td><code>$(&quot;main&quot;).tagName</code> is &quot;DIV&quot;</td>
</tr>
<tr>
<td><code>className</code></td>
<td>CSS classes of element</td>
<td><code>$(&quot;main&quot;).className</code> is &quot;foo bar&quot;</td>
</tr>
</tbody>
</table>
| `innerHTML`   | content inside element       | `$("main").innerHTML` is "\n<p>Hello, <em>ve..."
| `src`         | URL target of an image       | `$("icon").src` is "images/potter.jpg"                   |
## DOM properties for form controls

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>the text in an input control</td>
<td><code>$('#sid').value</code> could be &quot;1234567&quot;</td>
</tr>
<tr>
<td>checked</td>
<td>whether a box is checked</td>
<td><code>$('#frosh').checked</code> is true</td>
</tr>
<tr>
<td>disabled</td>
<td>whether a control is disabled (boolean)</td>
<td><code>$('#frosh').disabled</code> is false</td>
</tr>
<tr>
<td>readOnly</td>
<td>whether a text box is read-only</td>
<td><code>$('#sid').readOnly</code> is false</td>
</tr>
</tbody>
</table>

```html
<input id="sid" type="text" size="7" maxlength="7" />
<input id="frosh" type="checkbox" checked="checked" />
```

<html>HTML</html>
Abuse of `innerHTML`

- `innerHTML` can inject arbitrary HTML content into the page
- However, this is prone to bugs and errors and is considered poor style
Adjusting styles with the DOM

- contains same properties as in CSS, but with camelCasedNames
  - examples: backgroundColor, borderLeftWidth, fontFamily
Common DOM styling errors

- Forgetting to write `.style` when setting styles:

```javascript
var clickMe = document.getElementById("clickme");
clickMe.color = "red";
clickMe.style.color = "red";
```

- Style properties are capitalized like `this`, not like `this`:

```javascript
clickMe.style.fontSize = "14pt";
clickMe.style.fontSize = "14pt";
```

- Style properties must be set as strings, often with units at the end:

```javascript
clickMe.style.width = 200;
clickMe.style.width = "200px";
clickMe.style.padding = "0.5em";
```
Unobtrusive styling

```javascript
function okayClick() {
  this.style.color = "red";
  this.className = "highlighted";
}
```

```css
.highlighted { color: red; }
```

- well-written JavaScript code should contain as little CSS as possible
- use JS to set CSS classes/IDs on elements
- define the styles of those classes/IDs in your CSS file
# Timer events

<table>
<thead>
<tr>
<th>method</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>setTimeout(function, delayMS);</code></td>
<td>安排给定函数在给定延迟后ms调用。</td>
</tr>
<tr>
<td><code>setInterval(function, delayMS);</code></td>
<td>重复每隔delayMS ms调用函数。</td>
</tr>
<tr>
<td><code>clearTimeout(timerID);</code></td>
<td>停止给定计时器，使其不再调用其函数。</td>
</tr>
<tr>
<td><code>clearInterval(timerID);</code></td>
<td>停止给定计时器，使其不再调用其函数。</td>
</tr>
</tbody>
</table>

- **both** `setTimeout` and `setInterval` **return an ID representing the timer**
- **this ID can be passed** to `clearTimeout/Interval` **later to stop the timer**
setTimeout example

```html
<button onclick="delayMsg();">Click me!</button>
<span id="output"></span>
```

```js
function delayMsg() {
    setTimeout(booyah, 5000);
    $('#output').innerHTML = "Wait for it...";
}
function booyah() { // called when the timer goes off
    $('#output').innerHTML = "BOOYAH!";
}
```
setInterval example

```html
<button onclick="delayMsg2();">Click me!</button>
<span id="output"></span>
```

```js
var timer = null; // stores ID of interval timer
function delayMsg2() {
    if (timer == null) {
        timer = setInterval(rudy, 1000);
    } else {
        clearInterval(timer);
        timer = null;
    }
}
function rudy() { // called each time the timer goes off
    $\"output\".innerHTML += \" Rudy!\";
}
```
Passing parameters to timers

```js
function delayedMultiply() {
  // 6 and 7 are passed to multiply when timer goes off
  setTimeout(multiply, 2000, 6, 7);
}

function multiply(a, b) {
  alert(a * b);
}
```

- any parameters after the delay are eventually passed to the timer function
- why not just write this?

```js
setTimeout(multiply(6 * 7), 2000);
```
Common timer errors

```javascript
setTimeout(booyah(), 2000);
setTimeout(booyah, 2000);
setTimeout(multiply(num1 * num2), 2000);
setTimeout(multiply, 2000, num1, num2);  // JS
```

- what does it actually do if you have the ()?
  - it calls the function immediately, rather than waiting the 2000ms!