JAVASCRIPT AND FORMS

Introduction to JavaScript and Forms

Announcements

- Some of the files were missing from /home/martin/Assignment4. They are available now.
- Some of the pages have pointers to validate code on cs.washington.edu, which do not work. Change these to validate from w3.
- For Assignment 3, the files in info.txt have only three lines instead of the promised 4. The fourth line is supposed to be the number of review. Just make up a number.

Announcements (2)

- The page for the Final Project has been updated. A template for the proposal and an example proposal is available there.

Code to validate html from w3

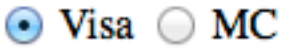

```
<p>
  <a href="http://validator.w3.org/check?uri=referer">
    <img src="http://www.w3.org/Icons/valid-xhtml10"
              alt="Valid XHTML 1.0!" height="31" width="88" />
  </a>
</p>
```

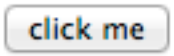
Some of the examples have pointers to cs.washington.edu, which do not work.

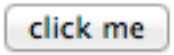
Standup

Discuss questions with your Scrum Team

Quiz

1. Write HTML that will produce: 

2. Why use client-side programming?
3. Write HTML and JavaScript to open an alert box when you click: 

4. Change the text color on 

when clicked using:

```
function changeButton() {
  var button = document.getElementById("output");
  button.style.color = "red";
}
```

And the answer is …

Quiz (Answers)

1. <label><input type="radio" name="cc" value="visa" checked="checked" /> Visa</label>

   <label><input type="radio" name="cc" value="mc" /> MC</label>

   - Slide 17

1. 1) Faster UI 2) Quick changes 3) Input actions
   - Slide 24
2. <button onclick="alert('button clicked');">click me</button>
   - Slides 36, 37, 38
3. <button id="output" onclick="changeButton();">click me</button>
   - Slide 43

Form Basics

Web Data

- Most interesting web pages revolve around data
  - examples: Google, IMDB, Digg, Facebook, YouTube, Rotten Tomatoes
  - can take many formats: text, HTML, XML, multimedia
- Many of them allow us to access their data
- Some even allow us to *submit our own new data*
- Most server-side web programs accept parameters that guide their execution

Reading/writing an entire file

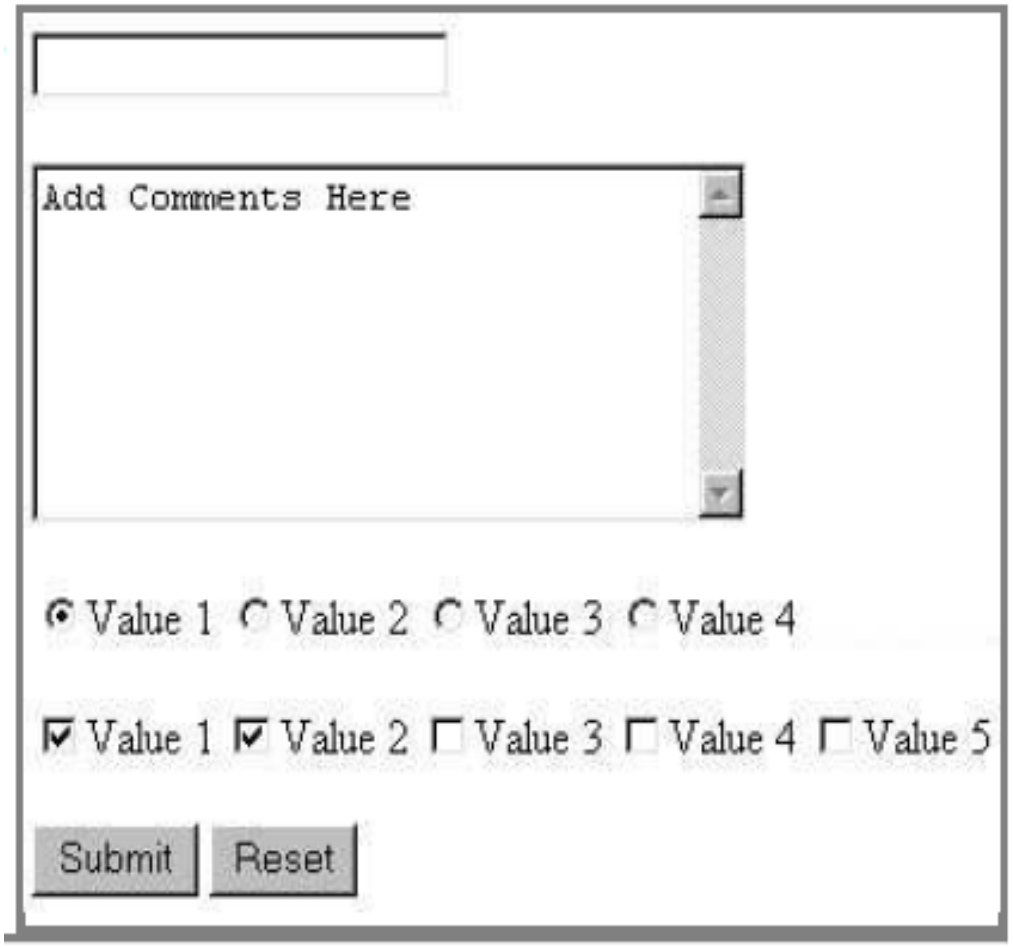

```
URL?name=value&name=value...
```

```
http://example.com/student_login.php?
username=xenia&sid=1234567
```

- **query string**: a set of parameters passed from a browser to a web server
  - often passed by placing name/value pairs at the end of a URL
- PHP code on the server can examine and utilize the value of parameters

HTML forms

- **form:** a group of UI controls that accepts information from the user and sends the information to a web server
- the information is sent to the server as a query string



HTML form: <form>


```
<form action="destination URL">
  form controls
</form>
```

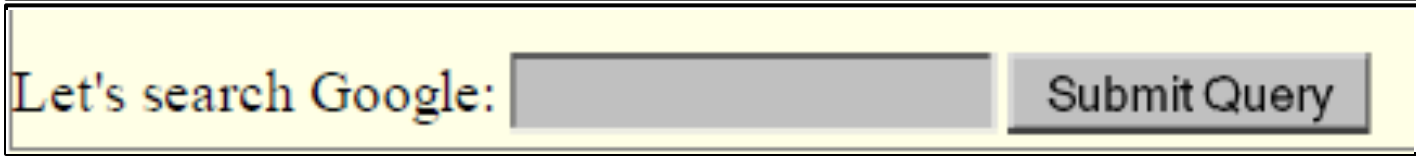
HTML

- required action attribute gives the URL of the page that will process this form's data
- when form has been filled out and **submitted**, its data will be sent to the action's URL

Form example


```
<form action="http://www.google.com/search">
  <div>
    Let's search Google:
    <input name="q" />
    <input type="submit" />
  </div>
</form>
```

HTML



- Wrap the form's controls in a block element such as div

Form controls

Form controls: <input>


```
<!-- 'q' happens to be the name of Google's required
parameter -->
<input type="text" name="q" value="Colbert Report" />
<input type="submit" value="Booyah!" />
```

- `input` element is used to create many UI controls
  - an inline element that MUST be self-closed
- `name` attribute specifies name of query parameter to pass to server

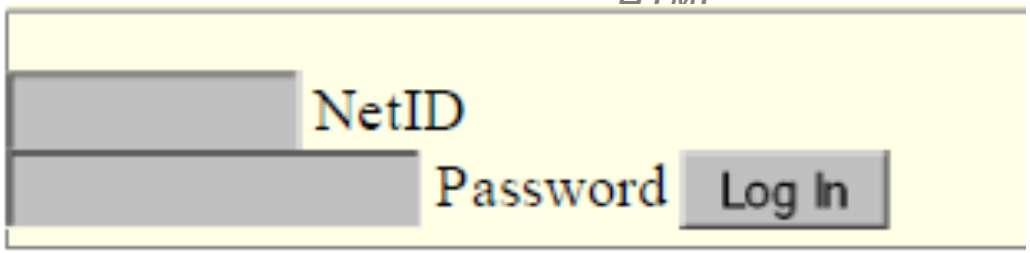
Form controls: <input> (cont.)


```
<!-- 'q' happens to be the name of Google's required
parameter -->
<input type="text" name="q" value="Colbert Report" />
<input type="submit" value="Booyah!" />
```

- `type` can be button, checkbox, file, hidden, password, radio, reset, submit, text, ...
- `value` attribute specifies control's initial text

Text fields: <input>


```
<input type="text" size="10" maxlength="8" /> NetID <br />
<input type="password" size="16" /> Password
<input type="submit" value="Log In" />
```



- input attributes: `disabled`, `maxlength`, `readonly`, `size`, `value`
- `size` attribute controls onscreen width of text field
- `maxlength` limits how many characters user is able to type into field

Text boxes: <textarea>


```
<textarea rows="4" cols="20">
Type your comments here.
</textarea>
```

HTML

- initial text is placed inside textarea tag (optional)
- required rows and cols attributes specify height/width in characters
- optional read only attribute means text cannot be modified

Check boxes: <input>


```
<input type="checkbox" name="lettuce" /> Lettuce
<input type="checkbox" name="tomato" checked="checked" />
Tomato
<input type="checkbox" name="pickles" /> Pickles
```

HTML

- none, 1, or many checkboxes can be checked at same time

Radio buttons: <input>


```
<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
```

HTML

- grouped by name attribute (only one can be checked at a time)
- must specify a value for each one or else it will be sent as value on

Text labels: <label>


```
<label><input type="radio" name="cc" value="visa"
checked="checked" /> Visa</label>
<label><input type="radio" name="cc" value="mastercard" />
MasterCard</label>
<label><input type="radio" name="cc" value="amex" />
American Express</label>
```

HTML

- associates nearby text with control, so you can click text to activate control
- can be used with checkboxes or radio buttons
- *label element can be targeted by CSS style rules*

Drop down lists: <select>, <option>


```
<select name="favoritecharacter">
  <option>Frodo</option>
  <option>Bilbo</option>
  <option selected="selected">Gandalf</option>
  <option>Galandriel</option>
</select>
```

HTML

- option element represents each choice
- select optional attributes: disabled, multiple, size
- optional selected attribute sets which one is initially chosen

Using: <select> for lists


```
<select name="favoritecharacter[]" size="3"
multiple="multiple">
  <option>Frodo</option>
  <option>Bilbo</option>
  <option>Gandalf</option>
  <option>Galandriel</option>
  <option selected="selected">Aragorn</option>
</select>
```

HTML

- optional multiple attribute allows selecting multiple items with shift- or ctrl-click
- must declare parameter's *name with []* if you allow multiple selections
- option tags can be set to be initially selected

Option groups: <optgroup>

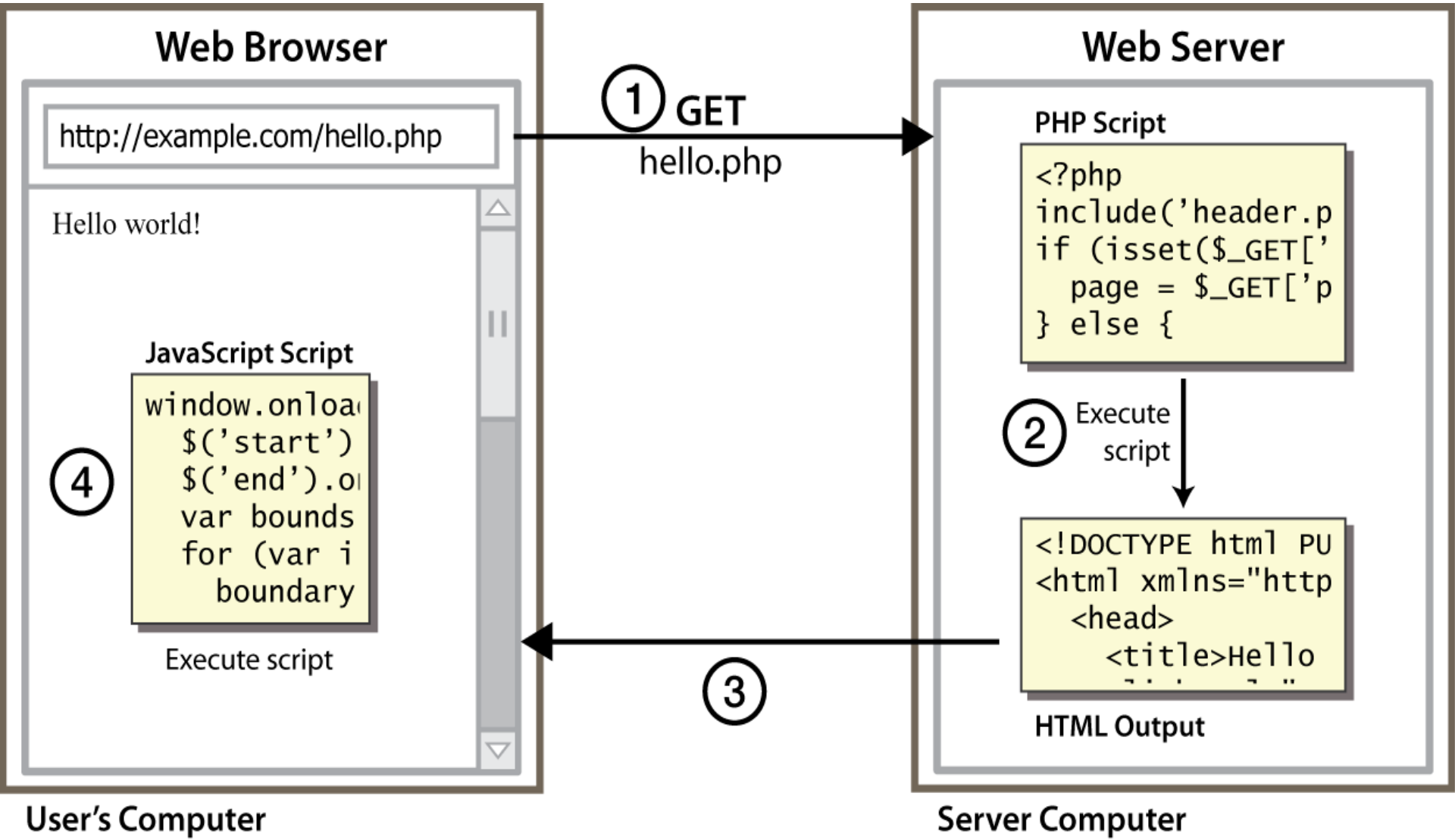

```
<select name="favoritecharacter">
  <optgroup label="Major Characters">
    <option>Frodo</option>
    <option>Sam</option>
    <option>Gandalf</option>
    <option>Aragorn</option>
  </optgroup>
  <optgroup label="Minor Characters">
    <option>Galandriel</option>
    <option>Bilbo</option>
  </optgroup>
</select>
```

HTML

- What should we do if we don't like the bold italic?

Intro to Javascript

Client Side Scripting



Why use client-side programming?

PHP already allows us to create dynamic web pages. Why also use client-side scripting?

- client-side scripting (JavaScript) benefits:
  - **usability**: can modify a page without having to post back to the server (faster UI)
  - **efficiency**: can make small, quick changes to page without waiting for server
  - **event-driven**: can respond to user actions like clicks and key presses

Why use client-side programming?

- server-side programming (PHP) benefits:
  - **security**: has access to server's private data; client can't see source code
  - **compatibility**: not subject to browser compatibility issues
  - **power**: can write files, open connections to servers, connect to databases, ...

What is Javascript?

- a lightweight programming language ("scripting language")
  - used to make web pages interactive
  - insert dynamic text into HTML (ex: user name)
  - **react to events** (ex: page load user click)
  - get information about a user's computer (ex: browser type)
  - perform calculations on user's computer (ex: form validation)

What is Javascript?

- a web standard (but not supported identically by all browsers)
- NOT related to Java other than by name and some syntactic similarities

Javascript vs Java

- interpreted, not compiled
- more relaxed syntax and rules
  - fewer and "looser" data types
  - variables don't need to be declared
  - errors often silent (few exceptions)
- key construct is the function rather than the class
  - "first-class" functions are used in many situations
- contained within a web page and integrates with its HTML/CSS content

Javascript vs Java

+

=

JavaScript vs. PHP

- similarities:
  - both are interpreted, not compiled
  - both are relaxed about syntax, rules, and types
  - both are case-sensitive
  - both have built-in regular expressions for powerful text processing

JavaScript vs. PHP

- differences:
  - JS is more object-oriented: noun.verb(), less procedural: verb(noun)
  - JS focuses on user interfaces and interacting with a document; PHP is geared toward HTML output and file/form processing
  - JS code runs on the client's browser; PHP code runs on the web server

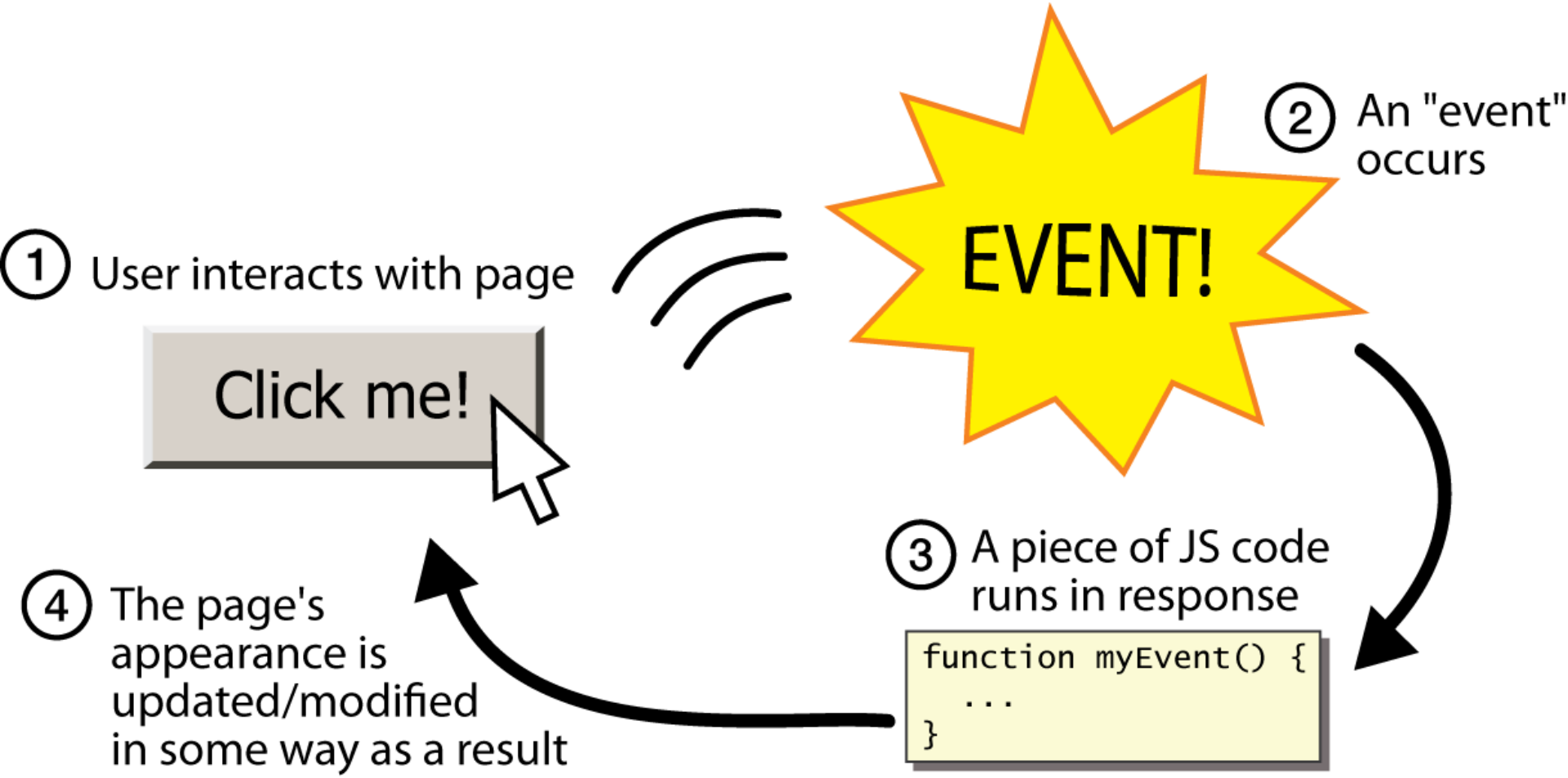
JS <3

Linking to a JavaScript file: `script`


```
<script src="filename" type="text/javascript"></script>
```
HTML

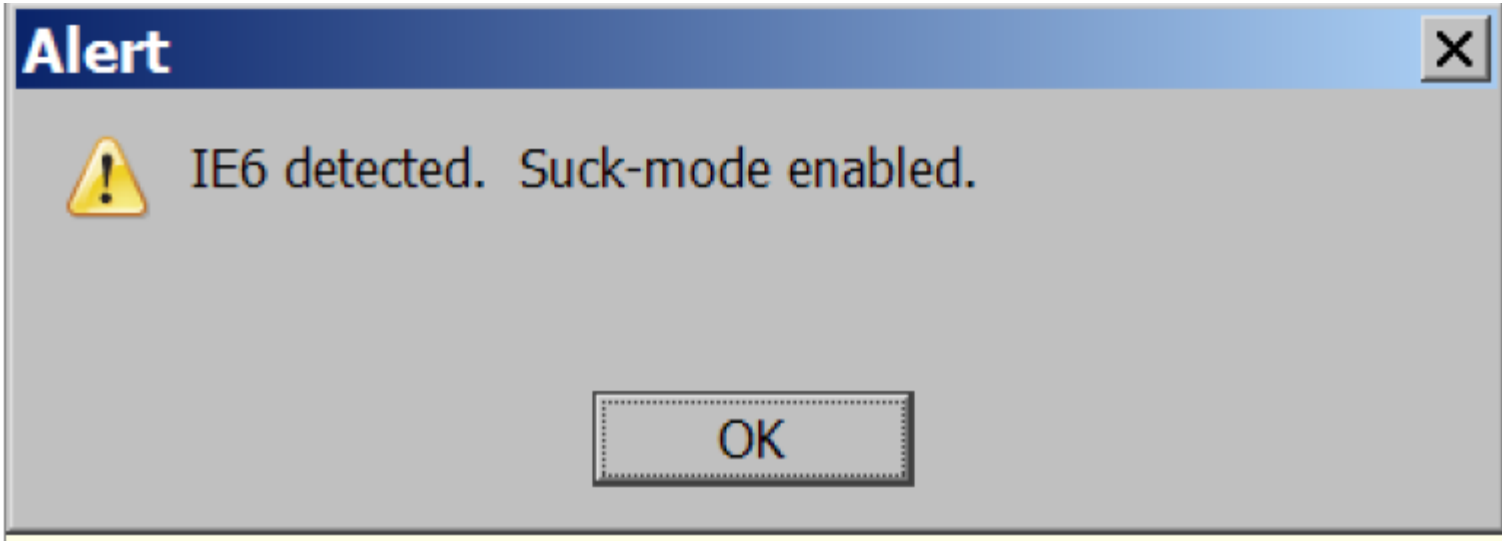
- script tag should be placed in HTML page's head
- script code is stored in a separate .js file
- JS code can be placed directly in the HTML file's body or head (like CSS)
  - but this is bad style (should separate content, presentation, and behavior

Event-driven programming



A JavaScript statement: `alert`


```
alert("IE6 detected. Suck-mode enabled.");
```
JS



- a JS command that pops up a dialog box with a message

Event-driven programming

- you are used to programs start with a main method (or implicit main like in PHP)
- JavaScript programs instead wait for user actions called *events* and respond to them
- event-driven programming: writing programs driven by user events
- Let's write a page with a clickable button that pops up a "Hello, World" window...

Buttons


```
<button>Click me!</button>
```
HTML

- button's text appears inside tag; can also contain images
- To make a responsive button or other UI control:
  1. choose the control (e.g. button) and event (e.g. mouse 1. click) of interest
  2. write a JavaScript function to run when the event occurs
  3. attach the function to the event on the control

JavaScript functions


```
function name() {
statement ;
statement ;
...
statement ;
}
```
JS

```
function myFunction() {
      alert("Hello!");
      alert("How are you?");
}
```
JS

- the above could be the contents of example.js linked to our HTML page
- statements placed into functions can be evaluated in response to user events

Event handlers


```
<element attributes onclick="function();">...
```

HTML

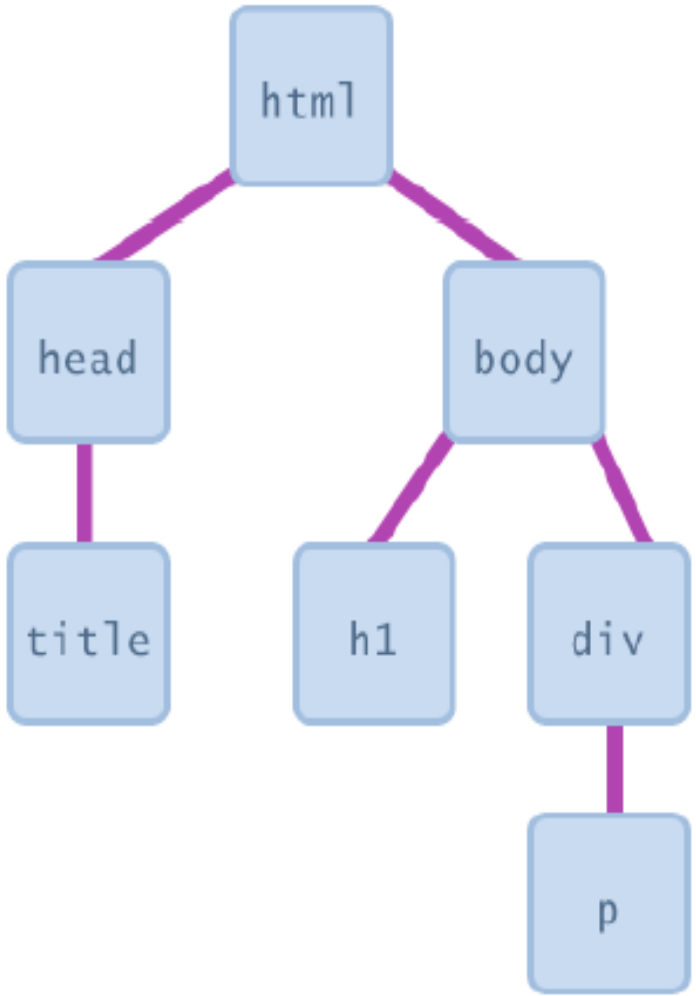
```
<button onclick="myFunction();">Click me!</button>
```

HTML

- JavaScript functions can be set as event handlers
  - when you interact with the element, the function will execute
- onclick is just one of many event HTML attributes we'll use
- but popping up an alert window is disruptive and annoying
  - A better user experience would be to have the message appear on the page...

Document Object Model (DOM)

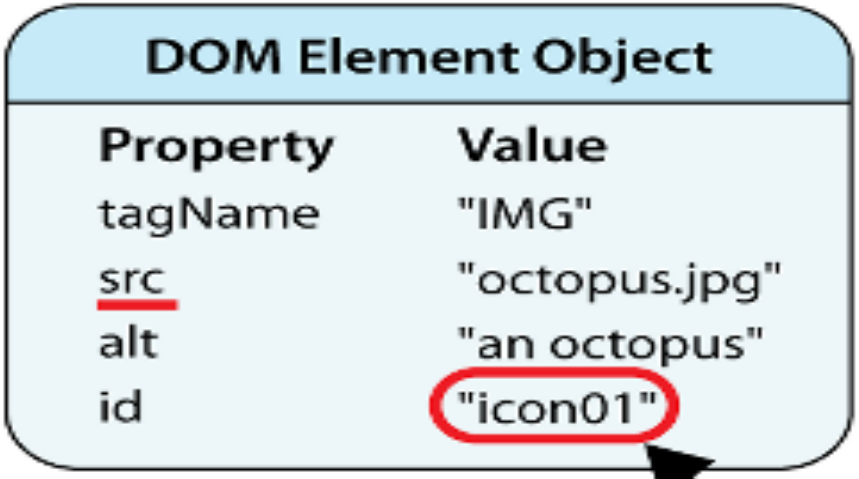
- most JS code manipulates elements on an HTML page
- we can examine elements' state
  - e.g. see whether a box is checked
- we can change state
  - e.g. insert some new text into a div
- we can change styles
  - e.g. make a paragraph red



DOM element objects

HTML

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



JavaScript

```
var icon = document.getElementById("icon01");
icon.src = "kitty.gif";
```

Accessing elements: document.getElementById


```
var name = document.getElementById("id");
```
JS

```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />
```
HTML

```
function changeText() {
        var span = document.getElementById("output");
        var textBox = document.getElementById("textbox");

         textbox.style.color = "red";

}
```
JS

Accessing elements: `document.getElementById`

- document.getElementById returns the DOM object for an element with a given id
- can change the text inside most elements by setting the innerHTML property
- can change the text in form controls by setting the value property

Changing element style: `element.style`

Attribute	Property or style object
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

Preetify


```JS
function changeText() {
	//grab or initialize text here

	// font styles added by JS:
	text.style.fontSize = "13pt";
	text.style.fontFamily = "Comic Sans MS";
	text.style.color = "red"; // or pink?
}
```

More Javascript Syntax

Variables


```
var name = expression;                                    JS
```

```
var clientName = "Connie Client";
var age = 32;
var weight = 127.4;                                       JS
```

- variables are declared with the var keyword (case sensitive)
- types are not specified, but JS does have types ("loosely typed")
  - `Number, Boolean, String, Array, Object, Function, Null, Undefined`
  - can find out a variable's type by calling `typeof`

Number type


```
var enrollment = 99;
var medianGrade = 2.8;
var credits = 5 + 4 + (2 * 3);
```
JS

- integers and real numbers are the same type (no int vs. double)
- same operators: + - * / % ++ -- = += -= *= /= %=
- similar precedence to Java
- many operators auto-convert types: "2" * 3 is 6

Comments (same as Java)


```
// single-line comment
/* multi-line comment */
```
JS

- identical to Java's comment syntax
- recall: 4 comment syntaxes
  - HTML: <!-- comment -->
  - CSS/JS/PHP: /* comment */
  - Java/JS/PHP: // comment
  - PHP: # comment

Math object


```
var rand1to10 = Math.floor(Math.random() * 10 + 1);
var three = Math.floor(Math.PI);
```
JS

- **methods:** `abs, ceil, cos, floor, log, max, min, pow, random, round, sin, sqrt, tan`
- **properties:** `E, PI`

Special values: null and undefined


```
var ned = null;
var benson = 9;
// at this point in the code,
// ned is null
// benson's 9
// caroline is undefined
```

JS

- `undefined` : has not been declared, does not exist
- `null` : exists, but was specifically assigned an empty or null value
- Why does JavaScript have both of these?

Logical operators

- > < >= <= && || ! == != === !==
- most logical operators automatically convert types:
  - 5 < "7" is true
  - 42 == 42.0 is true
  - "5.0" == 5 is true
- === and !== are strict equality tests; checks both type and value
  - "5.0" === 5 is false

if/else statement (same as Java)


```
if (condition) {
        statements;
} else if (condition) {
        statements;
} else {
        statements;
}
```

JS

- identical structure to Java's if/else statement
- JavaScript allows almost anything as a condition

Boolean type


```
var iLike190M = true;
var ieIsGood = "IE6" > 0; // false
if ("web devevelopment is great") { /* true */ }
if (0) { /* false */ }
```

JS

- any value can be used as a Boolean
  - "falsey" values: 0, 0.0, NaN, "", null, and undefined
  - "truthy" values: anything else
- converting a value into a Boolean explicitly:
  - `var boolValue = Boolean(otherValue);`
  - `var boolValue = !!(otherValue);`

for loop (same as Java)


```
var sum = 0;
for (var i = 0; i < 100; i++) {
        sum = sum + i;
}
```
JS

```
var s1 = "hello";
var s2 = "";
for (var i = 0; i < s.length; i++) {
        s2 += s1.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```
JS

while loops (same as Java)


```
while (condition) {
       statements;
}
```
JS

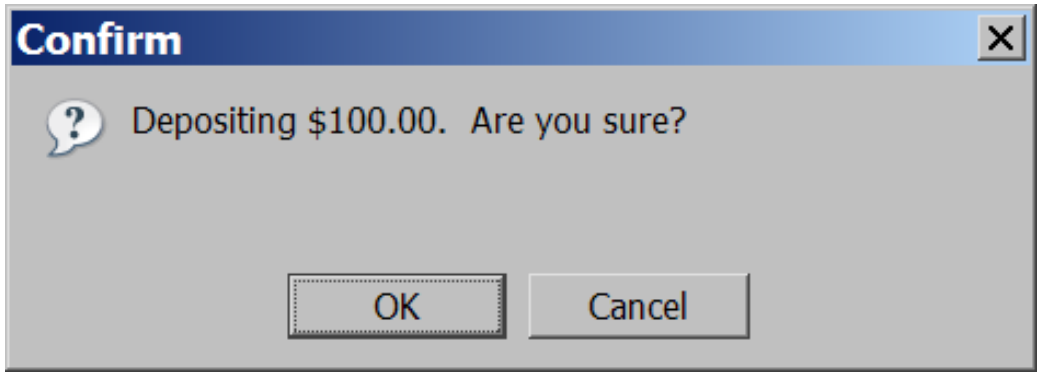
```
do {
    statements;
} while (condition);
```
JS

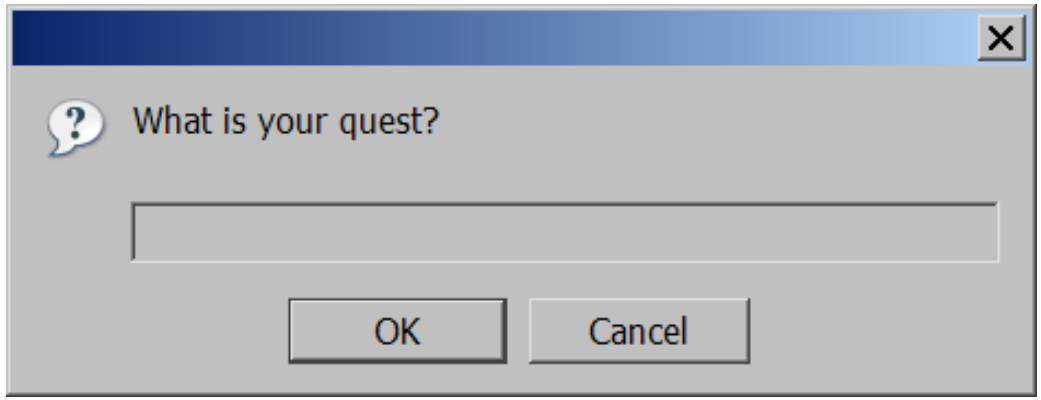
- break and continue keywords also behave as in Java

Popup boxes


```
alert("message"); // message
confirm("message"); // returns true or false
prompt("message"); // returns user input string
```

JS





Arrays


```
var name = []; // empty array
var name = [value, value, ..., value]; // pre-filled
name[index] = value; // store element
```

JS

```
var ducks = ["Huey", "Dewey", "Louie"];
var stooges = []; // stooges.length is 0
stooges[0] = "Larry"; // stooges.length is 1
stooges[1] = "Moe"; // stooges.length is 2
stooges[4] = "Curly"; // stooges.length is 5
stooges[4] = "Shemp"; // stooges.length is 5
```

JS

Array methods


```
var a = ["Stef", "Jason"]; // Stef, Jason
a.push("Brian"); // Stef, Jason, Brian
a.unshift("Kelly"); // Kelly, Stef, Jason, Brian
a.pop(); // Kelly, Stef, Jason
a.shift(); // Stef, Jason
a.sort(); // Jason, Stef
```

JS

- array serves as many data structures: list, queue, stack, ...
- methods: `concat, join, pop, push, reverse, shift, slice, sort, splice, toString, unshift`
  - push and pop add / remove from back
  - unshift and shift add / remove from front
  - shift and pop return the element that is removed

String type


```
var s = "Connie Client";
var fName = s.substring(0, s.indexOf(" ")); // "Connie"
var len = s.length; // 13
var s2 = 'Melvin Merchant';
```
JS

- **methods:** `charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase`
  - charAt returns a one-letter String (there is no char type)
- length property (not a method as in Java)
- Strings can be specified with "" or ''
- concatenation with + :
  - 1 + 1 is 2, but "1" + 1 is "11"

More about `String`

- escape sequences behave as in Java: \' \" \& \n \t \\
- converting between numbers and Strings:

```
var count = 10;
var s1 = "" + count; // "10"
var s2 = count + " bananas, ah ah ah!"; // "10 bananas, ah ah ah!"
var n1 = parseInt("42 is the answer"); // 42
var n2 = parseFloat("booyah"); // NaN
```
JS

- accessing the letters of a String:

```
var firstLetter = s[0]; // fails in IE
var firstLetter = s.charAt(0); // does work in IE
var lastLetter = s.charAt(s.length - 1);
```
JS

Splitting strings: split and join


```
var s = "the quick brown fox";
var a = s.split(" "); // ["the", "quick", "brown", "fox"]
a.reverse(); // ["fox", "brown", "quick", "the"]
s = a.join("!"); // "fox!brown!quick!the"
```

JS

- split breaks apart a string into an array using a delimiter
  - can also be used with regular expressions (seen later)
- join merges an array into a single string, placing a delimiter between them