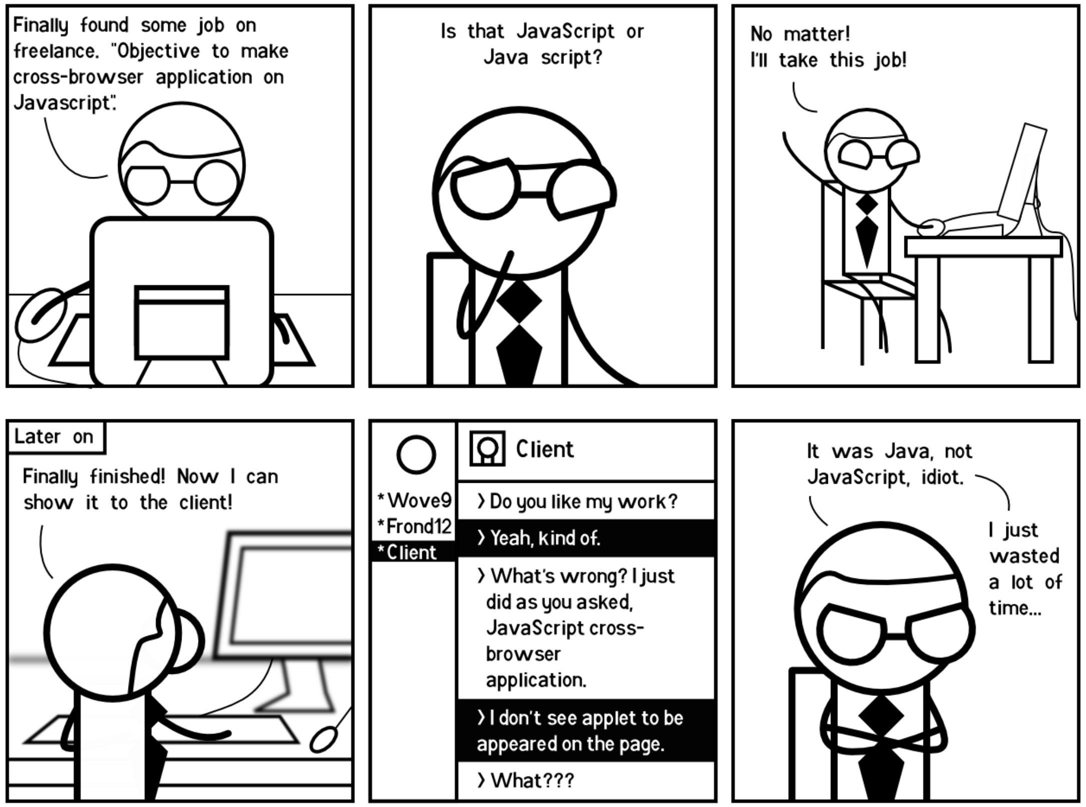
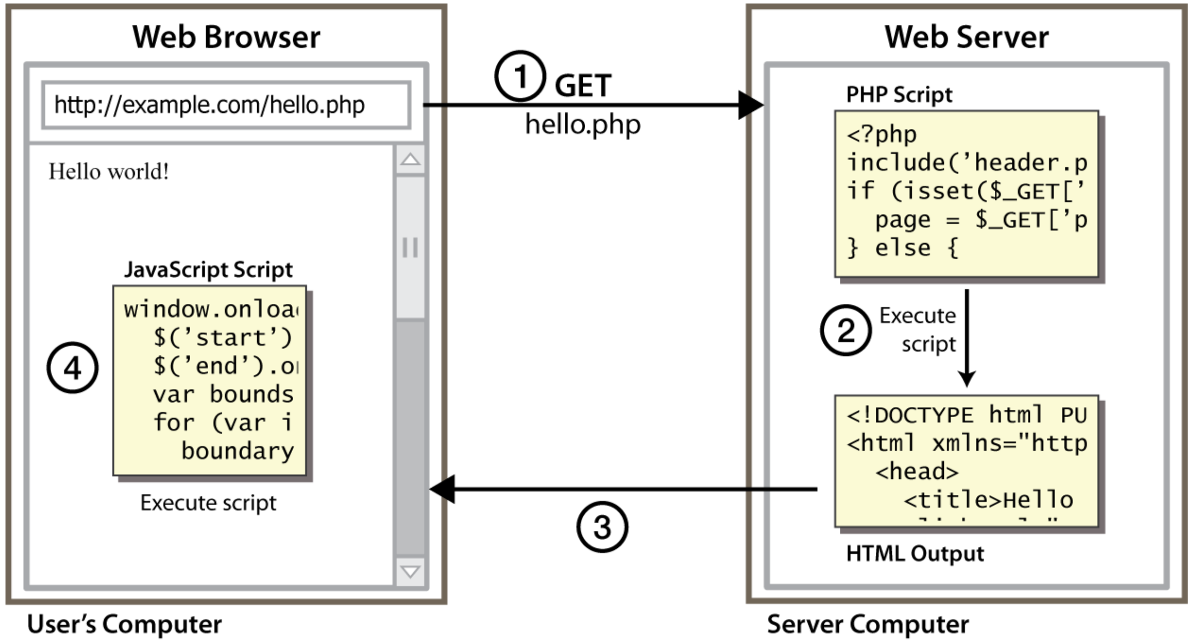
CSE 154

LECTURE 6: JAVASCRIPT


Finally found some job on freelance. "Objective to make cross-browser application on Javascript".
Is that JavaScript or Java script?
No matter! I'll take this job!
Later on
Finally finished! Now I can show it to the client!
Client
*Wove9
*Frond12
*Client
› Do you like my work?
› Yeah, kind of.
› What's wrong? I just did as you asked, JavaScript cross-browser application.
› I don't see applet to be appeared on the page.
› What???
It was Java, not JavaScript, idiot.
I just wasted a lot of time...

Client-side scripting



```
window.onloa
  $('start')
  $('end').o
  var bounds
  for (var i
    boundary
```

```
<?php
include('header.p
if (isset($_GET['
  page = $_GET['p
} else {
```

```
<!DOCTYPE html PU
<html xmlns="http
  <head>
    <title>Hello
```

- **client-side script**: code runs in browser *after* page is sent back from server
often this code manipulates the page or responds to user actions

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)
- 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

Java +

= JavaScript

Linking to a JavaScript file: script

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

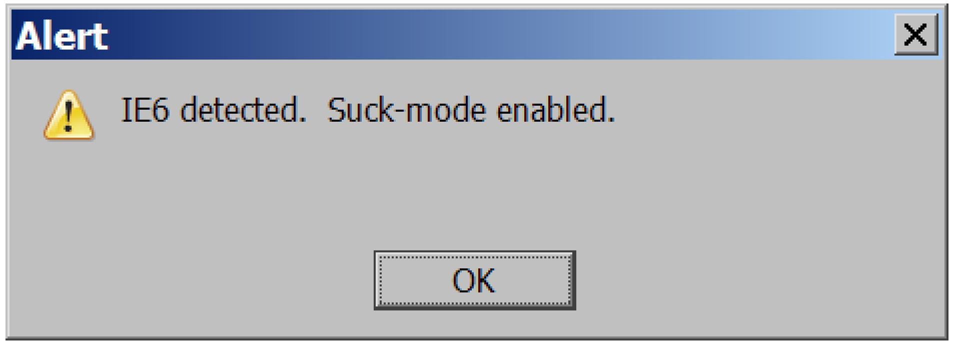
```html
<script src="example.js" type="text/javascript"></script>        HTML
```

- `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)

A JavaScript statement: alert

```
alert("message");
```
JS

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


Alert

IE6 detected. Suck-mode enabled.

output

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

Variables and types

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

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

String type

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

- 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)
- concatenation with + : 1 + 1 is 2, but "1" + 1 is "11"

More about String

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

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

- to access characters of a `String`, use [*index*] or `charAt`:

```
var firstLetter = s[0];
var firstLetter = s.charAt(0);
var lastLetter = s.charAt(s.length - 1);
```

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:`/*` *comment* `*/`
  - Java/JS:`//` *comment*

for loop (same as Java)

```
for (initialization; condition; update) {
  statements;
}                                                     JS
```

```
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[i] + s1[i];
}
// s2 stores "hheelllloo"                             JS
```

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

Logical operators

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

Boolean type

```
var iLikeJS = true;
var ieIsGood = "IE6" > 0;    // false
if ("web dev 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*`);`

Special values: null and undefined

```
var ned = null;
var benson = 9;
var caroline;

// 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?

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*

while loops (same as Java)

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

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

- `break` and `continue` keywords also behave as in Java but do not use them in this class!

Arrays

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

```
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                PHP
```

- two ways to initialize an array
- `length` property (grows as needed when elements are added)

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

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 surrounded by /:

    ```
    var a = s.split(/[ \t]+/);
    ```
- join merges an array into a single string, placing a delimiter between them

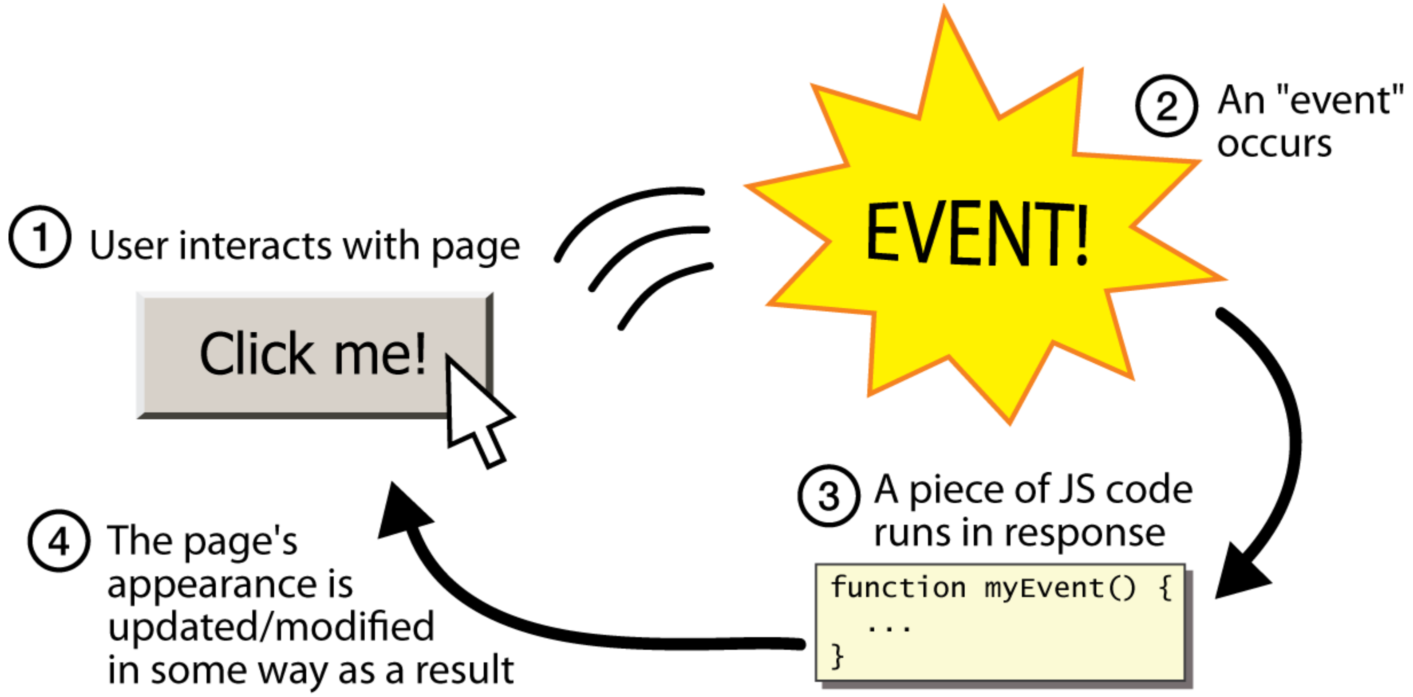
Defining functions

```
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-driven programming



1 User interacts with page
Click me!
EVENT!
2 An "event" occurs
3 A piece of JS code runs in response
4 The page's appearance is updated/modified in some way as a result

```
function myEvent() {
   ...
}
```

- JS programs have no `main`; they respond to user actions called **events**
- **event-driven programming**: writing programs driven by user events

Event handlers

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

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

```
Click me!                                                          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

Buttons: <button>

the canonical clickable UI control (inline)

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

output

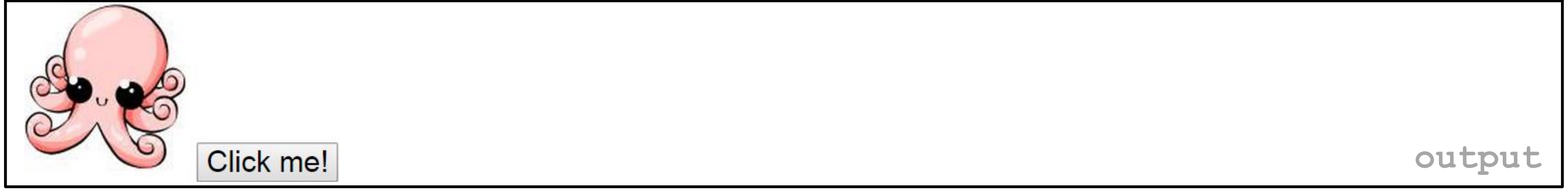
- button's text appears inside tag; can also contain images
- To make a responsive button or other UI control:
    1. choose the control (e.g. button) and event (e.g. mouse click) of interest
    2. write a JavaScript function to run when the event occurs
    3. attach the function to the event on the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

Accessing an element: document.getElementById

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

```
<img id="icon01" src="images/octopus.jpg" alt="an animal" />
<button onclick="changeImage();">Click me!</button>            HTML
```

```
function changeImage() {
    var octopusImage = document.getElementById("icon01");
    octopusImage.src = "images/kitty.gif";
}                                                               JS
```

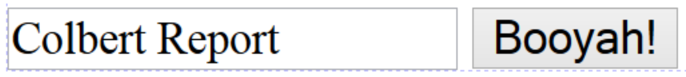


- `document.getElementById` returns the DOM object for an element with a given `id`

<input>

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

HTML

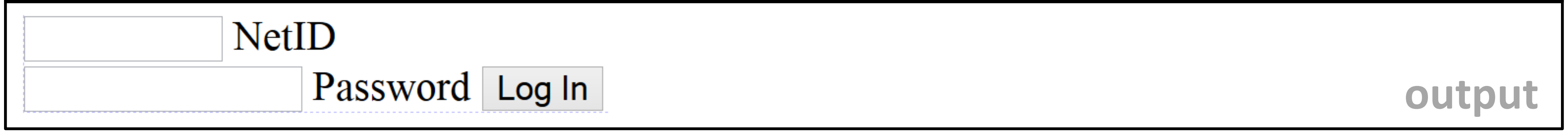


output

- 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
- 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" />                        HTML
```

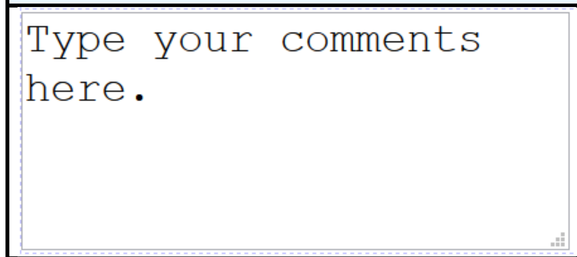


- input attributes: disabled, maxlength, readonly, size, value
- size attribute controls onscreen width of text field
- maxlength limits how many characters user is able to type into field

Text boxes: <textarea>

a multi-line text input area (inline)

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



```
Type your comments
here.
```
output

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