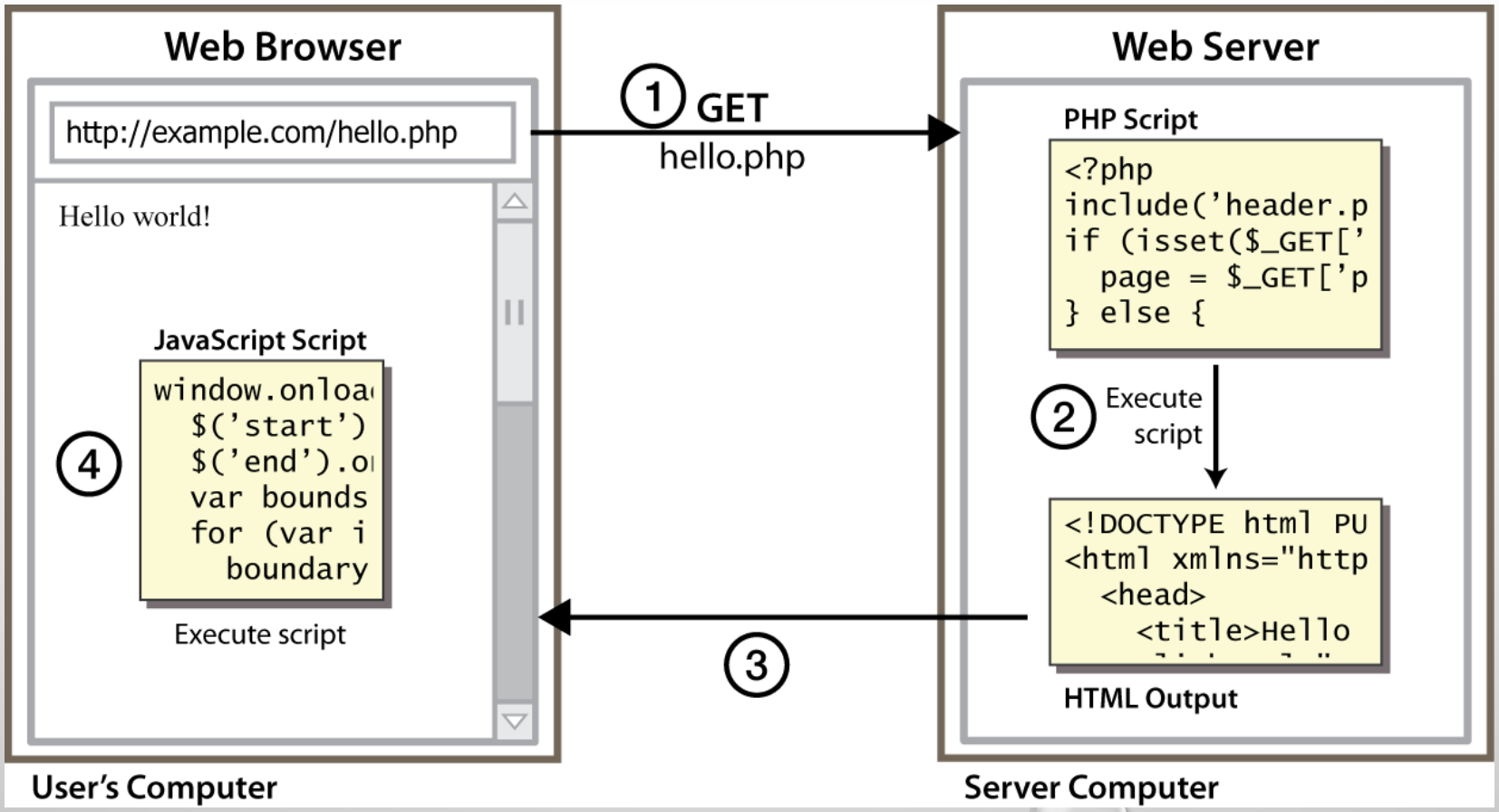
LECTURE09: JAVASCRIPT1: VARIABLES AND OPERATORS

DR. JIAN WU

Courtesy: presentation slides from Dr. Marty Stepp, Jessica Miller,. Victoria Kirst @ University of Maryland

CLIENT-SIDE SCRIPTING



WHY USING SERVER-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, ...

WHY USING CLIENT-SIDE PROGRAMMING?

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

- **Usability**: can modify a page without having to post back to the server (faster UI)
- **Efficiency**: can make small, quick changes to pages without waiting for server
- **Event-driven**: can respond to user actions like clicks and key presses

WHAT IS JAVASCRIPT?

- A lightweight programming language ("scripting language")
  - Used to make web pages interactive
  - Insert dynamic text into HTML (ex: username)
  - **React to events** (ex: page load user click)
  - Get information about a users' computer (ex: browser type)
  - Perform calculations on users' 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

COMPARED WITH JAVA, JAVASCRIPT IS

- 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)
- The 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

FIRST-CLASS FUNCTION

- A programming language is said to have **First-class functions** when functions in that language are treated like any other variable.
- A first-class function can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable.
- In the first example, we assigned an **Anonymous Function** (`function()`) to a variable `foo`, then we used that variable to invoke the function by adding parentheses () at the end.
- In the second example, we defined a named function `sayHello()` and pass it to another function `greeting()` as an argument.

```
const foo = function() {
    console.log("foobar");
}
// Invoke it using the variable
foo();
```

anonymous function

```
function sayHello() {
    return "Hello, ";
}
function greeting(helloMessage, name) {
  console.log(helloMessage() + name);
}
// Pass `sayHello` as an argument to `greeting` function
greeting(sayHello, "JavaScript!");
```

named function

JAVASCRIPT VS. PHP

- Similarities:
  - Both are interpreted, not compiled
  - Both are relaxed about syntax, rules, and types
  - Both are case-sensitive (except constructs, function names, class names for PHP)
  - Both have built-in regular expressions for powerful text processing

JAVASCRIPT VS. PHP

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

LINKING AN HTML TO A JAVASCRIPT FILE: SCRIPT

test.html

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

- The script tag should be placed in an HTML page's head
- The 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)
  - Although this is bad style (should separate content, presentation, and behavior)

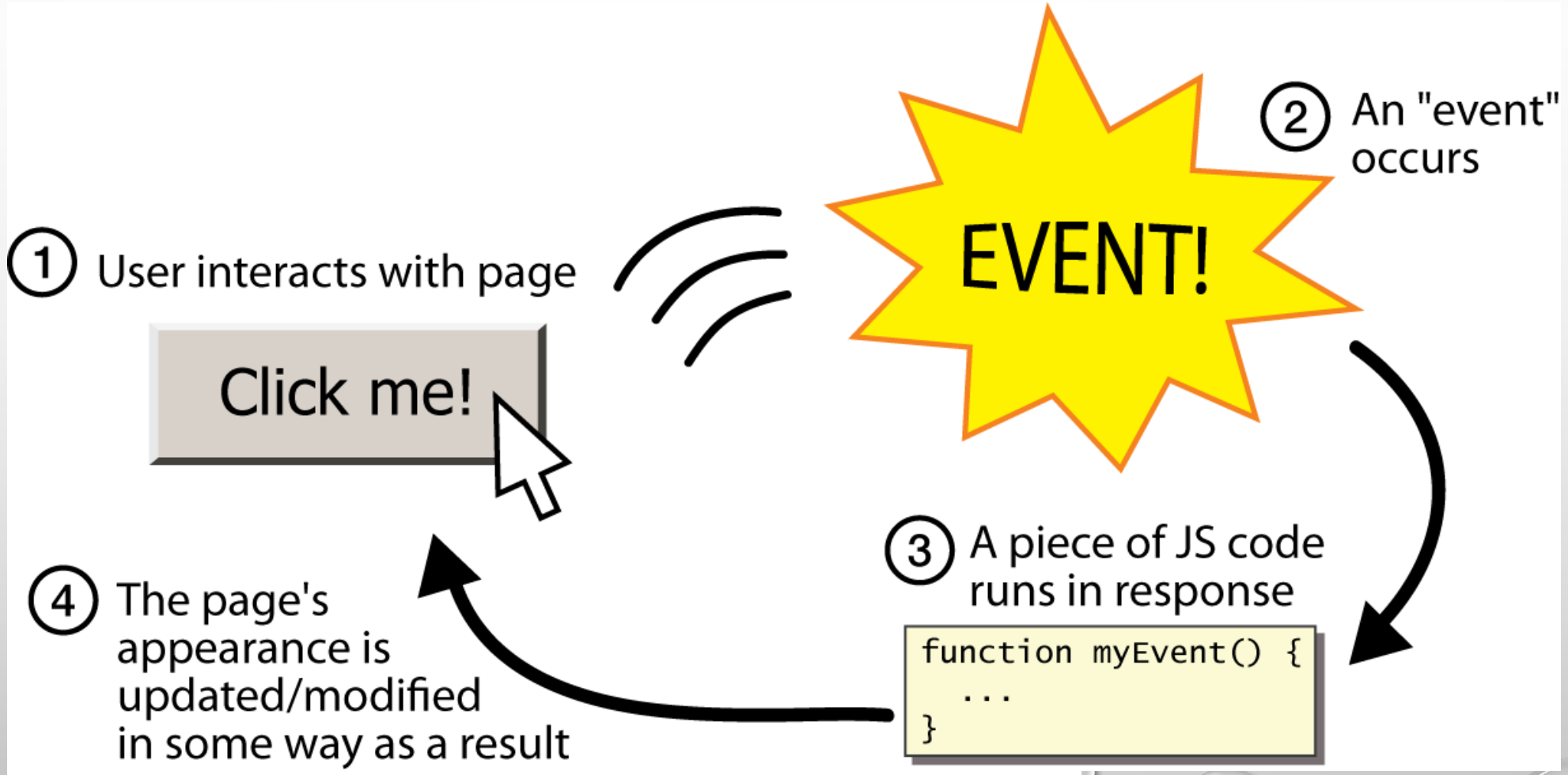
HTML file:

```
<!-- index.html -->
<html>
    <head>
        <title>My Page</title>
        <script src="my-script.js"></script>
    </head>
    <body>
        <div id="user-greeting">Welcome back, user</div>
    </body>
</html>
```

Javascript:

```
// my-script.js
document.addEventListener("DOMContentLoaded", function() {
    // this function runs when the DOM is ready, i.e. when the document has be
    document.getElementById("user-greeting").textContent = "Welcome back, Bart
});
```

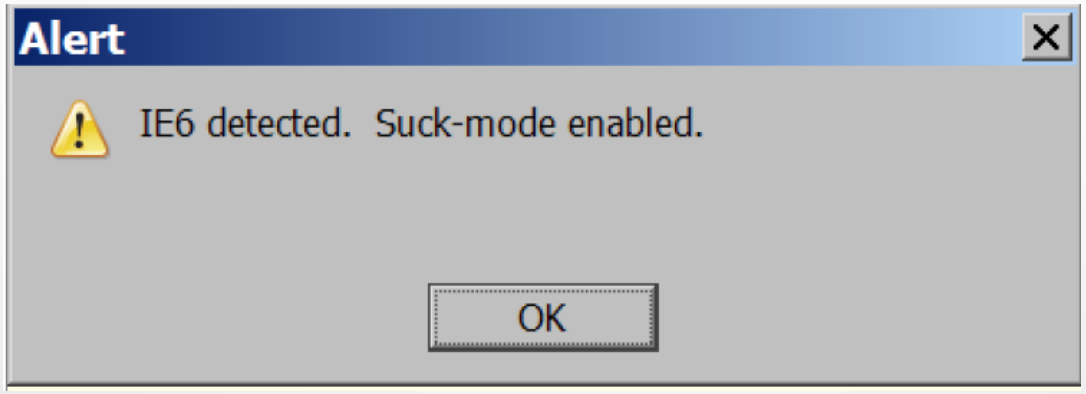
EVENT-DRIVEN PROGRAMMING



Examples: the voting buttons on YouTube and Stackoverflow.

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 starting with a main method (or an implicit main method 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

BUTTONS

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

- Button's text appears inside the 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 executed in response to user events

EVENT HANDLERS

General:

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

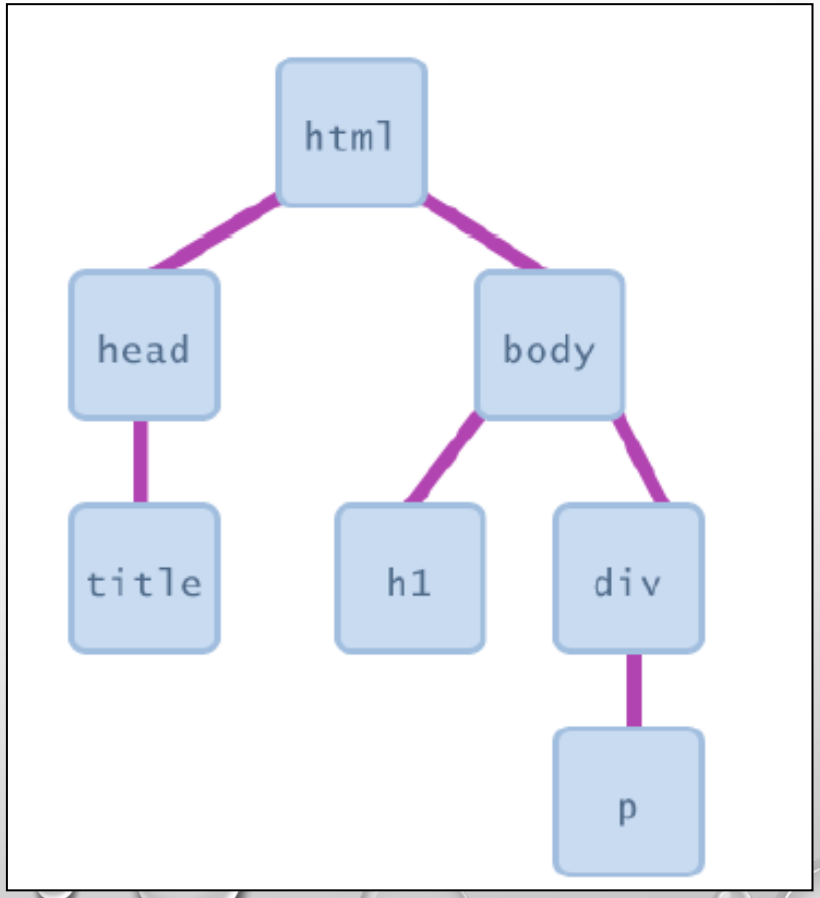
Example:

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

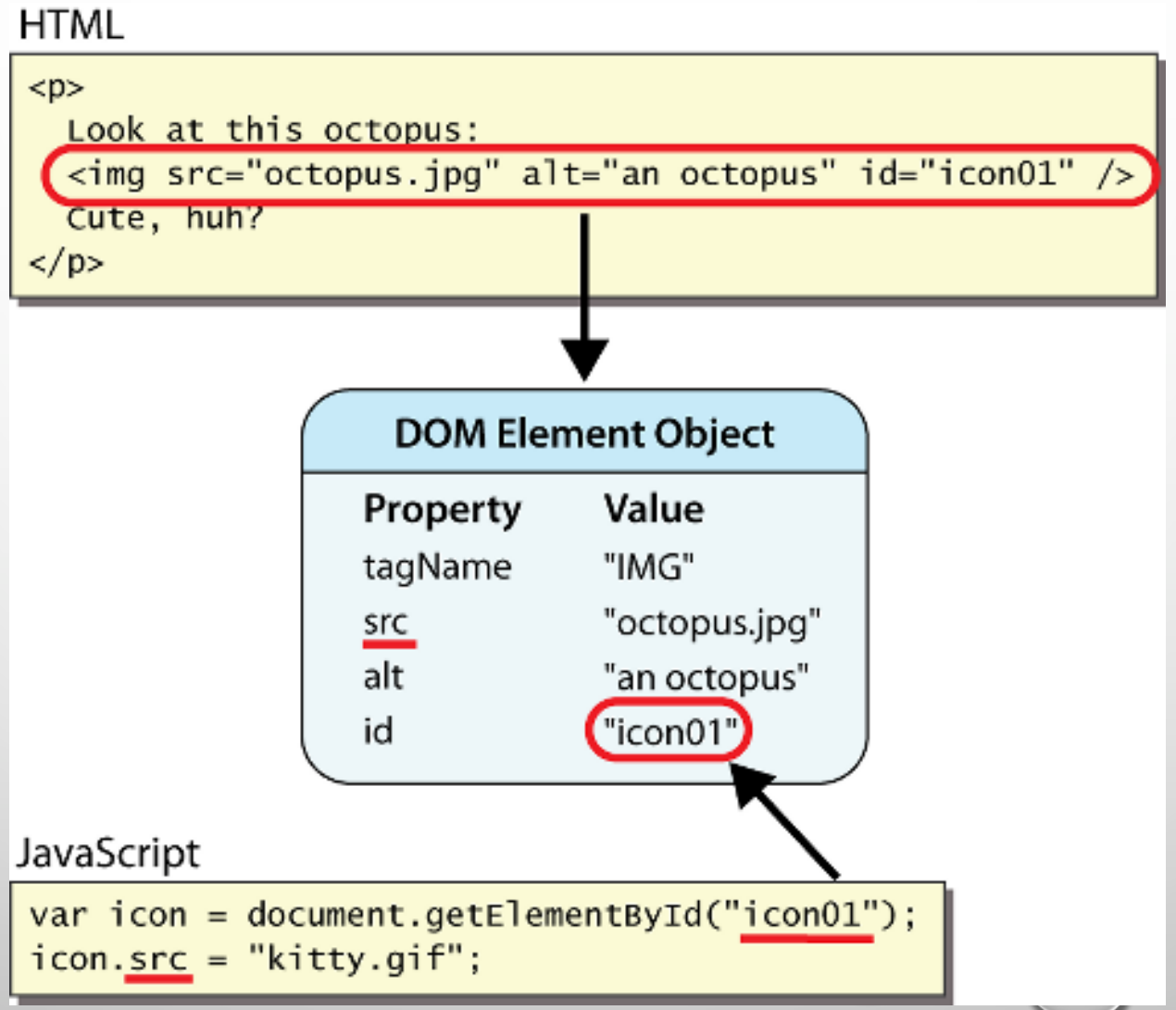
- JavaScript functions can be set as event handlers
  - When you interact with the element, the function will execute
- Onclick is just one of many event HTML attributes
- BUT, popping up an alert window is disruptive and annoying
  - A better user experience would be to have the message appear on the page…

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
- Example on the next slide



DOM ELEMENT OBJECTS



This JavaScript changes the picture from octopus.jpg to kitty.gif

ACCESSING ELEMENTS

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

HTML

```js
function changeText() {
        var textbox = document.getElementById("textbox");

        textbox.style.fontSize = "33pt";
        textbox.style.color = "green";
}
```

JS

Effect: js/accesselement.html

COMBINE JS WITH PHP: EXAMPLE ON INPUT VALIDATION

```
<!DOCTYPE html>
<html>
<head>
<script>
function validateForm() {
  let x = document.forms["myForm"]["fname"].value;
  if (x == "") {
    alert("Name must be filled out");
    return false;
  }
}
</script>
</head>
<body>

<h2>JavaScript Validation</h2>

<form name="myForm" action="action_page.php" onsubmit="return validateForm()"
method="post">
  Name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>

</body>
</html>
```

Note: action_page.php can be created separately.

ACCESSING ELEMENTS

- `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

EXAMPLE: CHANGE TEXT USING innerHTML

```
<!DOCTYPE html>
<html>
<body>
<h1>The Element Object</h1>
<h2>The innerHTML Property</h2>

<p id="myP">I am a paragraph.</p>

<p>The content of "myP" is:</p>
<p id="demo"></p>

<script>
let html = document.getElementById("myP").innerHTML;
document.getElementById("demo").innerHTML = html;
</script>

</body>
</html>
```

Example: getelementinnerhtml.html
The change happens at runtime

```
<!DOCTYPE html>
<html>
<body>
<h1>The Element Object</h1>
<h2>The innerHTML Property</h2>

<p id="demo" onclick="myFunction()">Click me to change my HTML content (innerHTML).</p>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "I have changed!";
}
</script>

</body>
</html>
```

Example: changeelementinnerhtml.html
The change happens when the user clicks the text.

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 family	fontFamily

4 WAYS TO DISPLAY DATA BY JAVASCRIPT

- Writing into an HTML element, using `innerHTML`.
- Writing into the HTML output using `document.write()`.
  - Using `document.write()` after an HTML document is loaded, will delete all existing HTML
  - The document.write() method should only be used for testing.
- Writing into an alert box, using `window.alert()`.
  - The window object is the global scope object, that means that variables, properties, and methods by default belong to the window object.
  - Therefore, you can skip the window keyword.
- Writing into the browser console, using `console.log()`.
  - Press F12 to view the browser console

MORE JAVASCRIPT SYNTAX

CS418/518

VARIABLES

- 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

- Integers and real numbers are the same type (no int or double) called Number
- Same operators: + - * / % ++ -- = += -= *= /= %=
- Similar precedence to Java
- Many operators auto-convert types: "2" * 3 is 6

COMMENTS (SAME AS JAVA)

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

MATH OBJECT

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

SPECIAL VALUES: NULL AND UNDEFINED

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

- `null` : exists, but was specifically assigned an empty or null value
- `undefined` : has not been declared, does not exist

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 () { /* 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);`

Javascript operator's precedence:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

FOR LOOP (SAME AS JAVA)

```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.charAt(i) + s1.charAt(i);
}
// s2 stores "hheellllloo"
```

WHILE LOOPS (SAME AS JAVA)

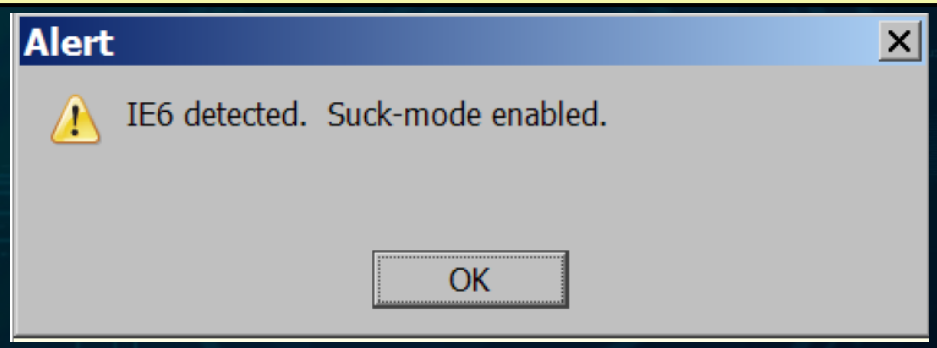
```JS
while (condition) {
	statements;
}
```

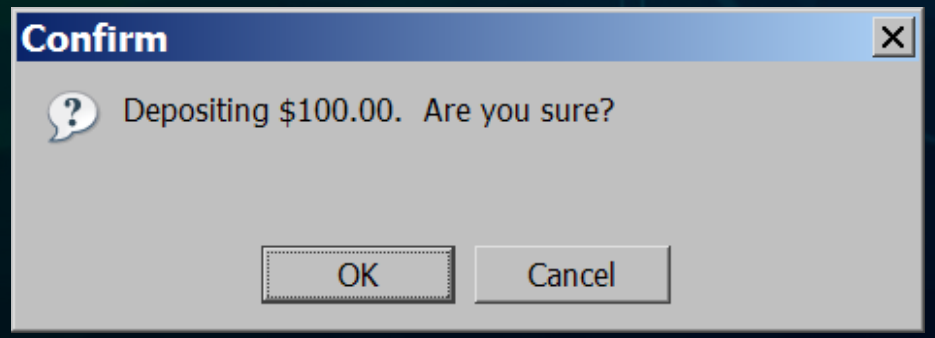
```JS
do {
	statements;
} while (condition);
```

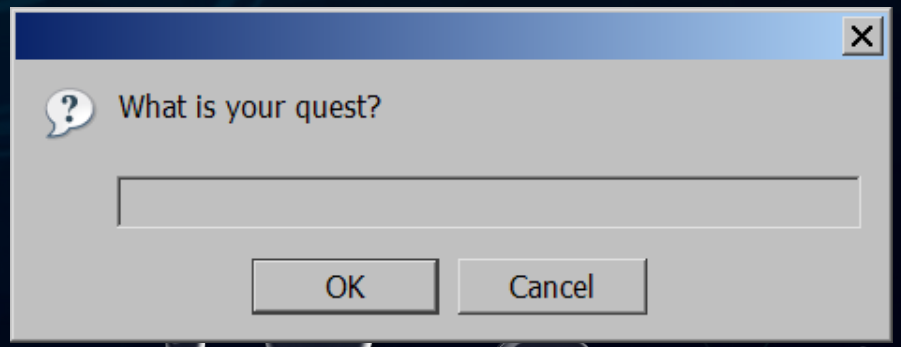
- Break and continue keywords also behave as in Java

POPUP BOXES

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







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 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)
- 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.charAt(0);
var lastLetter = s.charAt(s.length - 1);
```
JS

SPLITTING STRINGS: SPLIT AND JOIN

```JS
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"
```

- Split breaks apart a string into an array using a delimiter
  - Can also be used with regular expressions
- Join merges an array into a single string, placing a delimiter between them

BACKUP SLIDES BEYOND THIS POINT

CS418/518