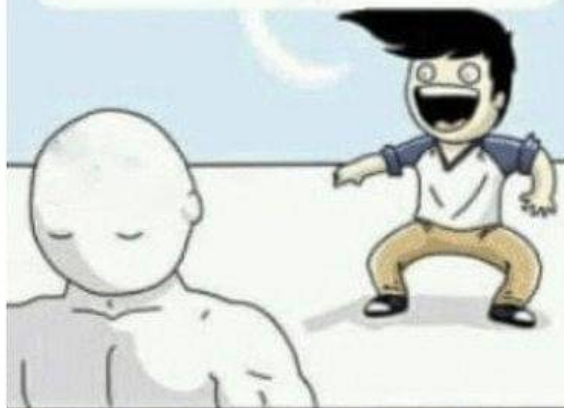


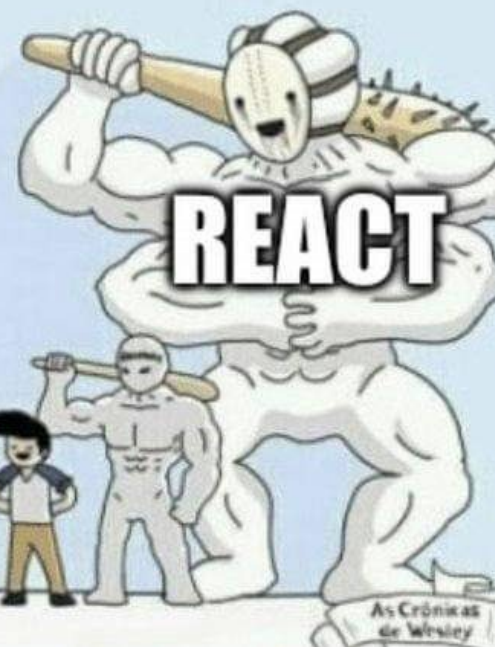
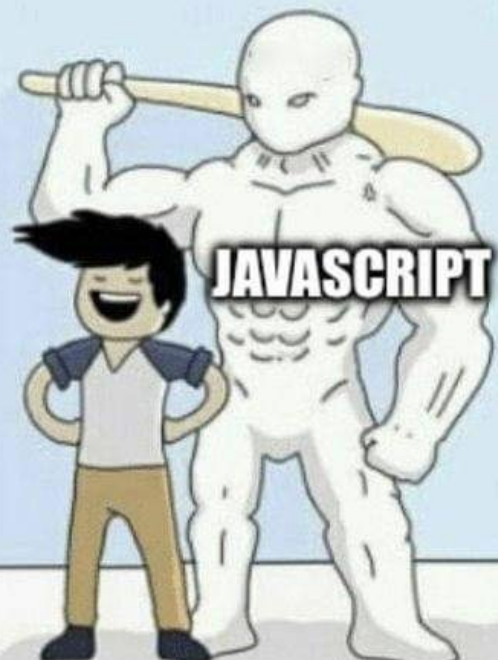
JAVASCRIPT

Intro

GOODBYE
HTML AND CSS!



I WILL NEVER
SUFFER AGAIN



What is JavaScript?

- Created by Netscape
 - Originally called LiveWire then LiveScript
- A client-side scripting language
 - Client-side refers to the fact that it is executed in the client (software) that the viewer is using. In the case of JavaScript, the client is the browser.
 - A server-side language is one that runs on the Web server. Examples: PHP, Python
- Interpreted on-the-fly by the client
 - Each line is processed as it loads in the browser

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

Why JavaScript?

- It's easier to learn than most programming languages
- It allows you to make interactive Web pages
- It can be fun!

JavaScript is not Java

- Completely different types of languages that just happen to be similarly named
 - JavaScript - programs are interpreted in the browser
 - Java - programs are compiled and can be run as stand alone applications

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

Common Uses of JavaScript

- Form validation
- Page embellishments and special effects
- Navigation systems
- Basic math calculations
- Dynamic content manipulation
- Sample applications
 - Dashboard widgets in Mac OS X, Google Maps, Philips universal remotes, Writely word processor, hundreds of others...

Example 1: Add Two Numbers

```
<html>
  ...
  <p> ... </p>
  <script>
    var num1, num2, sum
    num1 = prompt("Enter first number")
    num2 = prompt("Enter second number")
    sum = parseInt(num1) + parseInt(num2)
    alert("Sum = " + sum)
  </script>
  ...
</html>
```

Example 2: Browser Events

Mouse event causes
page-defined function to
be called

```
<script type="text/JavaScript">
  function whichButton(event) {
    if (event.button==1) {
      alert("You clicked the left mouse button!") }
    else {
      alert("You clicked the right mouse button!")
    }
  }
</script>
...
<body onmousedown="whichButton(event)">
...
</body>
```

Other events: onLoad, onMouseMove, onKeyPress, onUnload

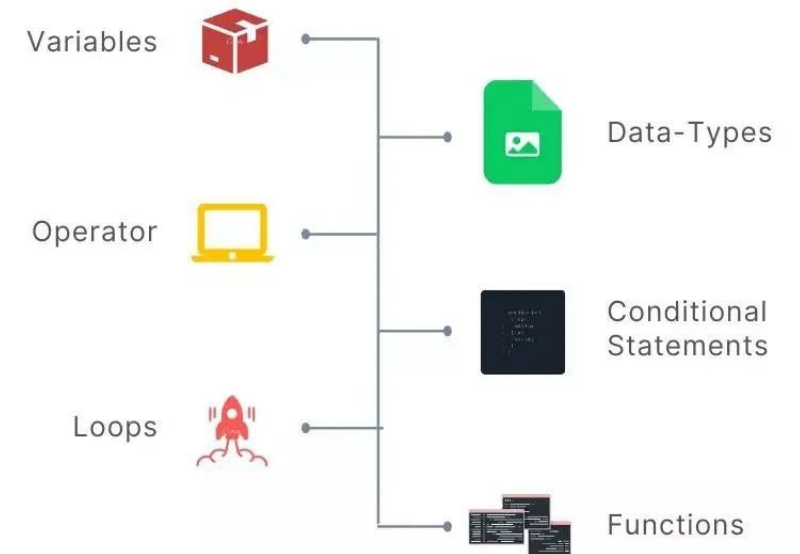
Example 3: Page Manipulation

- Some possibilities
 - createElement(elementName)
 - createTextNode(text)
 - appendChild(newChild)
 - removeChild(node)
- Example: add a new list item

```
var list = document.getElementById('t1')  
var newitem = document.createElement('li')  
var newtext = document.createTextNode(text)  
list.appendChild(newitem)  
newitem.appendChild(newtext)
```

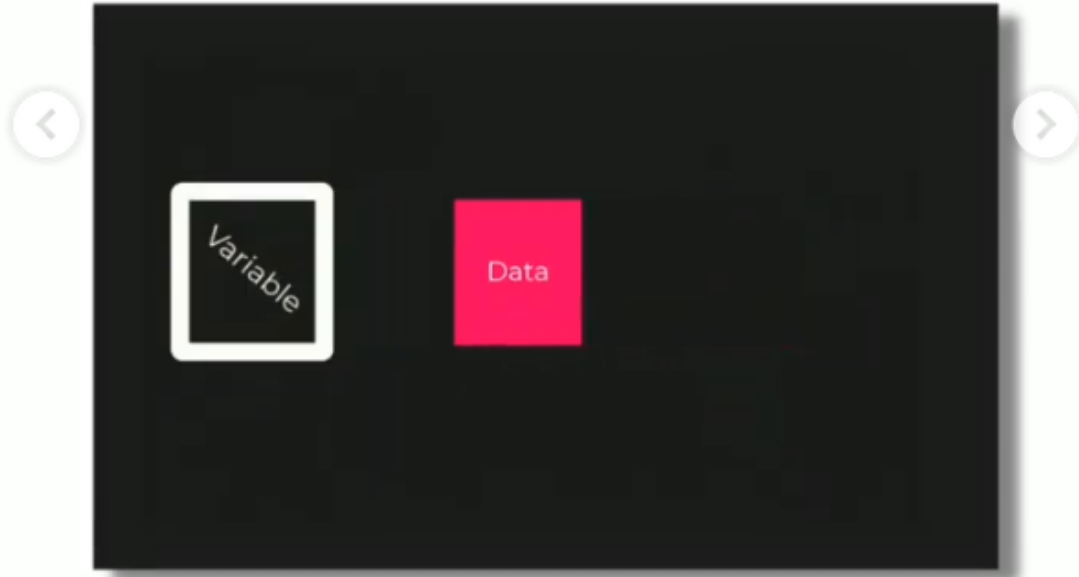
This uses the browser Document Object Model (DOM). We will focus on JavaScript as a language, not its use in the browser

BASICS OF PROGRAMMING



Variables

Variables are like a container where we can store the data



Data Type

A classification that specifies what type of value a variable can hold



Operator

An operator is a symbol that operates on a value or a variable

```
int a = 5 + 5
```



Operator

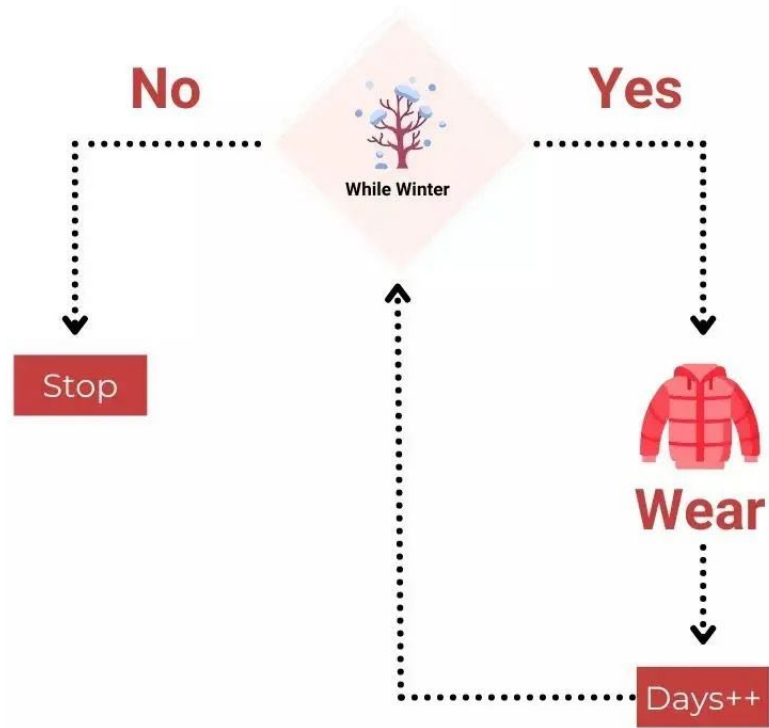


Conditional-Statements



Loops

Loops are used to execute the block of code several times



Functions

Think of a function like **machine**
Which perform specific task

Like making a cup of coffee



Big Picture

Application contains multiple programs and one program consists of multiple functions



Application

More Javascript BASIC

Variables

```
var name = expression;
```

JS

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

JS

- A variable is a label that references a value.
 - Use the let or var keyword to declare a variable.
 - Use the const keyword to define a readonly reference to a value.
 - 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);
```

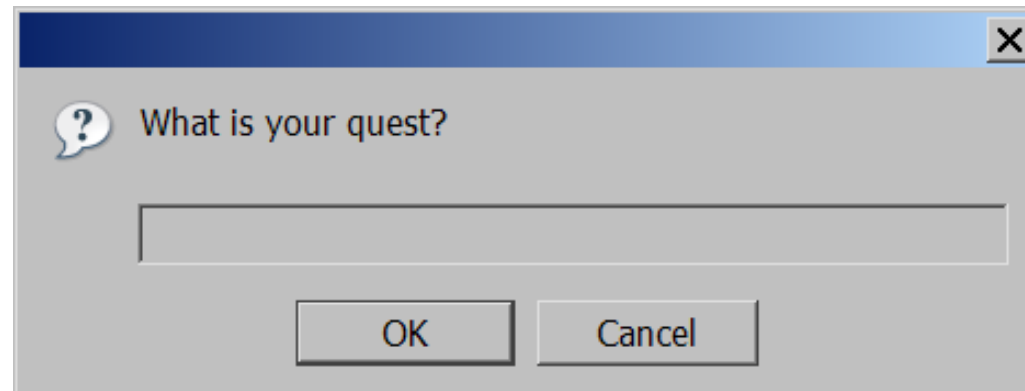
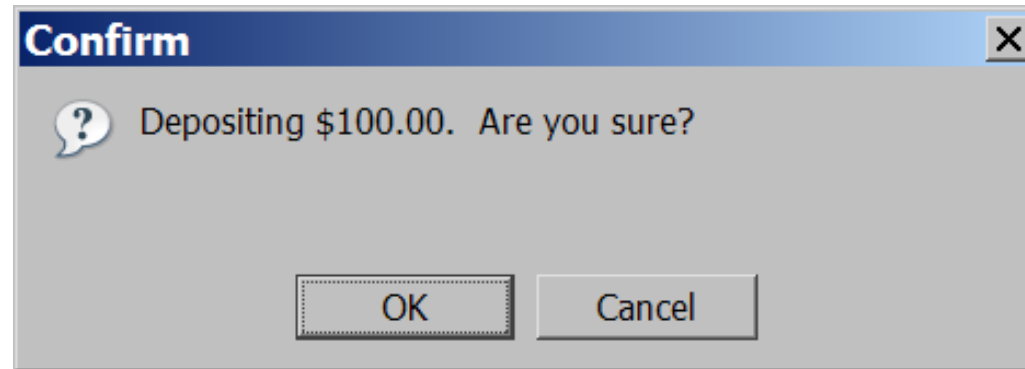
JS

- 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

- 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

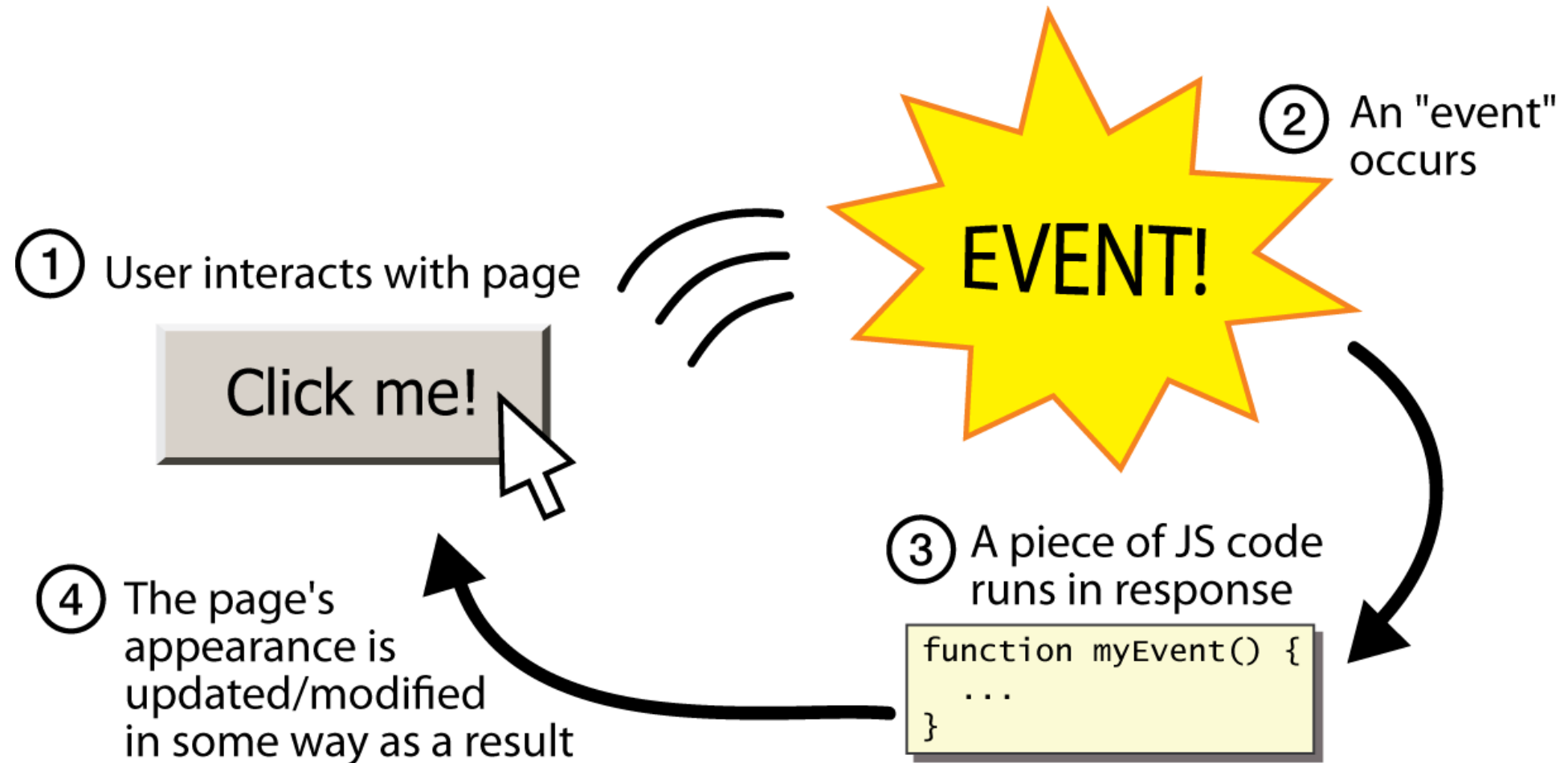
Linking to a JavaScript file: `script`

```
<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)

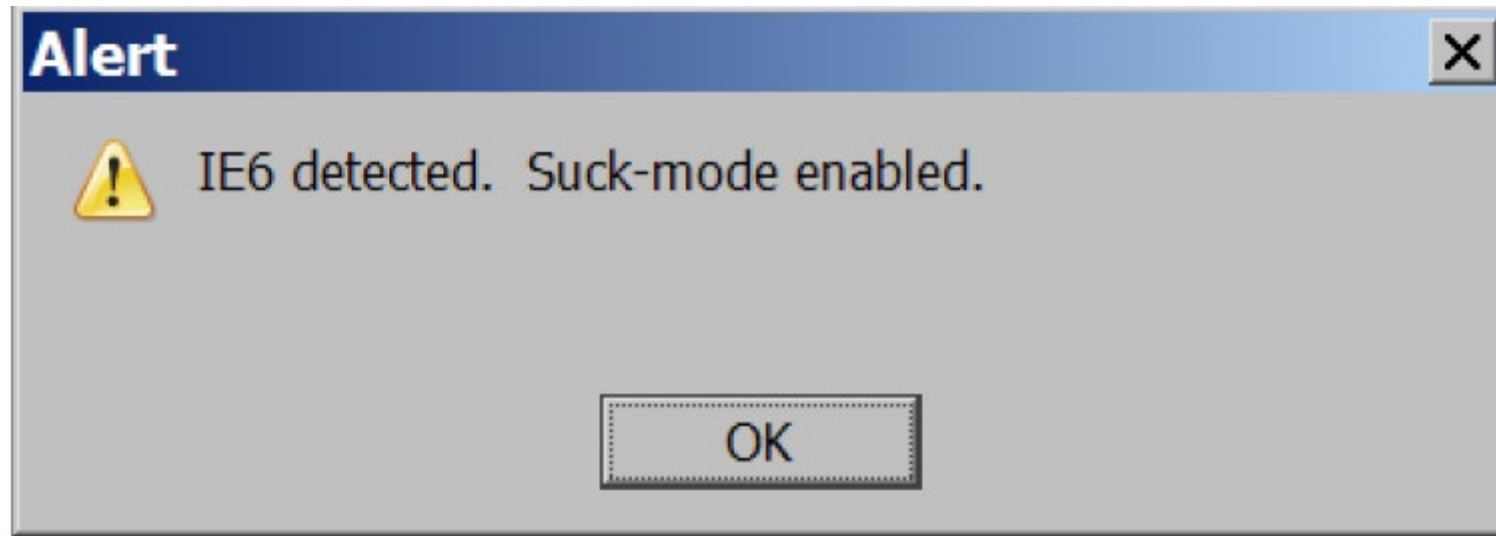
Event-driven programming



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

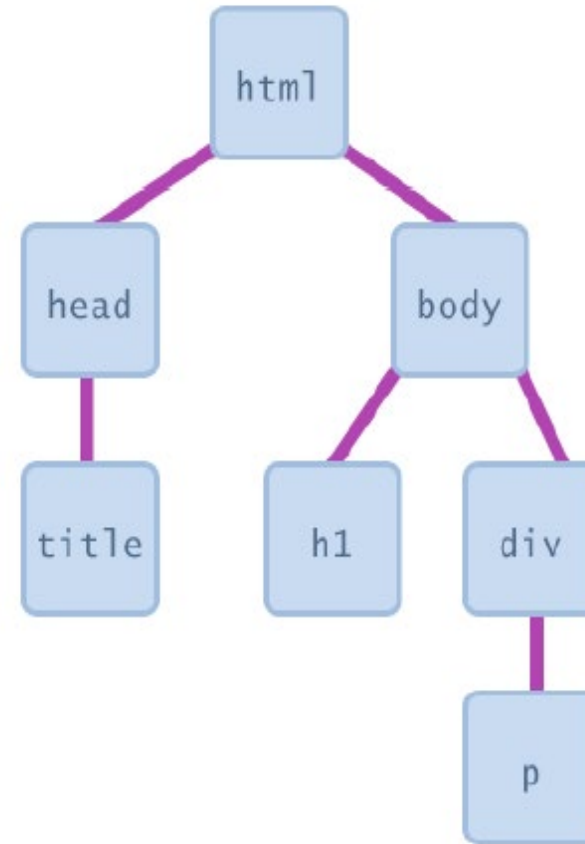
- JavaScript functions can be set as event handlers
 - when you interact with the element, the function will execute
- onclick is just one of many event HTML attributes we'll use
- but popping up an alert window is disruptive and annoying
 - A better user experience would be to have the message appear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

Document Object Model (DOM)

- HTML page is structured data
- DOM provides representation of this hierarchy
- Examples
 - Properties: `document.alinkColor`, `document.URL`,
`document.forms[]`, `document.links[]`, `document.anchors[]`,
...
 - Methods: `document.write(document.referrer)`
 - These change the content of the page!
- Also Browser Object Model (BOM)
 - Window, Document, Frames[], History, Location, Navigator
(type and version of browser)

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:  
    
  Cute, huh?  
</p>
```

DOM Element Object	
Property	Value
tagName	"IMG"
<u>src</u>	"octopus.jpg"
alt	"an octopus"
id	"icon01"

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

Some DOM Object Methods

Method	Description
getElementsByTagName()	Returns a node list (collection/array of nodes) containing all elements with a specified tag name
getElementsByClassName()	Returns a node list containing all elements with a specified class
appendChild()	Adds a new child node to a specified node
removeChild()	Removes a child node
replaceChild()	Replaces a child node
insertBefore()	Inserts a new child node before a specified child node
createAttribute()	Creates an Attribute node
createElement()	Creates an Element node
getAttribute()	Returns the specified attribute value
setAttribute()	Sets or changes the specified attribute, to the specified value

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

```
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?  
}
```

JS