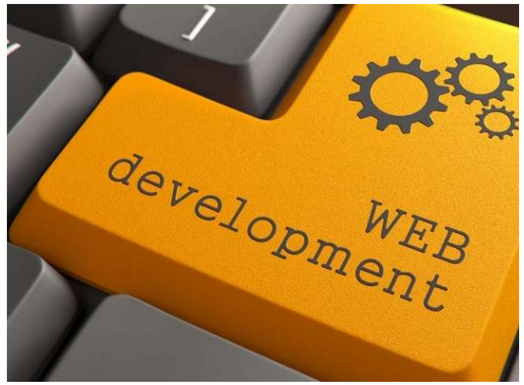
08.03.21 07:20

CSCI 165
Introduction to the Internet and the World Wide Web
Lec 1: Javascript



Jetic Gū

Overview

- Focus: Web Development
- Architecture: Internet
- Core Ideas:
    1. Adding Javascript to Your Webpage
    2. Basic Javascript Syntax

Using Javascript

In your webpage

Summary

Using Javascript

- What is Javascript?
  - Programming Language
  - Turning complete
  - Interpretive language: doesn’t need compilation, in fact all modern browsers supports it

Concept

Using Javascript

- Why Javascript?
  - HTML and CSS: static, shows webpage with still content, doesn't interact much, cannot respond to user input[1]
  - Javascript allows for dynamic interaction
    - Take user input, generate content accordingly
      - Respond to user `events` (`clicking`, `typing`, etc.)
      - Modify existing content (`add`, `remove`, `change`, etc.)
    - Security: only access permitted information[2]

1. Certain simple stuff are supported in HTML5, but in general it is still like unfolding a paper: you are not really adding new stuff
2. Achieved with other libraries and software (such as databases through SQL)

Concept

Basic Webpage (Now)

- Static Content Display
  - (`*.html`) HTML: Hypertext Markup Language
  - (`*.css`) Optional CSS: Stylish presentation of stuff
- Dynamic content
  - (`*.js`) Javascript stuff, make your webpage into a webapp

Concept

Basic Webpage (Now)

- Static Content Display
  - (`index.html`) HTML: Hypertext Markup Language
  - (`style.css`) Optional CSS: Stylish presentation of stuff
- Dynamic content
  - (`control.js`) Javascript stuff, make your webpage into a webapp

Example

Including `control.js`

- Inside `index.html`
  - In the head section, include the following line:

```
<script src="control.js"></script>
```

  - Similar to CSS isn't it

Example

Javascript stuff

- Inside `control.js`
  - Have this only line here

```
alert("I like cheese!");
```

  - In javascript, each line ends with semicolon `;`
  - `alert` is a function here that generates a pop up box, inside the parenthesis are the parameters of this function
    -- Just like $f(x)$ in math!

Example

Running the Code

- Load up `index.html` in your browser, a pop up box will show up
- Every time you load the webpage, few things are going to happen

1. The browser is going to **download** everything referenced in the `head` section
   -- CSS files, JS files, etc.

2. The browser is going to **execute** the scripts and style sheets
   Thus running all of the code, which is not what we always want (why?)

Example

User event triggered functions

- In `control.js`, change the code to

```
amPressed = function() {

  alert("I have been pressed!");

}
```

What is going on here?

- In `index.html`, add

```
<p onclick="amPressed()">Press me!</p>
```

Example

P1
Hello JS

User event triggered functions

- `control.js`

```
amPressed = function() {  // You can also add parameters in the parenthesis here

  alert("I have been pressed!");

  // This is a single instruction/statement

}
```

  - We are creating a `user-defined` function called `amPressed` here
    `alert()` is a builtin function

  - The code will **NOT** be executed unless **explicitly called**

  - The code inside the `{}` are executed sequentially when the function is called.
    No need for semicolon after `}` here

Example

User event triggered functions

- `index.html`

```
<p onclick="amPressed()">Press me!</p>
```

  - In `onclick`, here is a HTML DOM (Document Object Model) Mouse Event, it triggers whatever is in the quotation marks, in this instance the Javascript function `amPressed()`

1. Full list of Mouse Events: https://www.w3schools.com/jsref/obj_mouseevent.asp
2. Full list of DOM Events: https://www.w3schools.com/jsref/dom_obj_event.asp

Example

Exercise

- Build a new HTML webpage with a single line of text saying:
  click me!

- Use JS to create a popup window, which appears when the text above is clicked

Exercise

Javascript Syntax

Variables, Types, Conditions, Loops

Summary

P2
JS Cont.

Variables

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

- 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`

Concept

Number type

```
var enrollment = 99;

var medianGrade = 2.8;

var credits = 5 + 4 + (2 * 3);
```

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

Concept

P2
JS Cont.

Comments (same as Java/C/C++)

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

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

Concept

Math object

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

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

Concept

P2
JS Cont.

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
```

- `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?

Concept

Logical operators

- Similar to Java/C/C++
- `>  <  >=  <=  &&  ||  !  ==  !=  ===  !==`
- 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

Concept

P2
JS Cont.

`if/else` statement (same as Java/C/C++)

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

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

Concept

P2
JS Cont.

Boolean type (same as Java/C/C++)

```
var iLike190M = true;

var ieIsGood = "IE6" > 0; // false

if ("web devevelopment is great") { /* true */ }

if (0) { /* false */ }
```

- 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);
```

Concept

P2
JS Cont.

`for` loop (same as Java/C/C++)

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

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

Concept

`while` loops (same as Java/C/C++)

```
while (condition) {
       statements;
}

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

- break and continue keywords also behave as in Java/C/C++

Concept

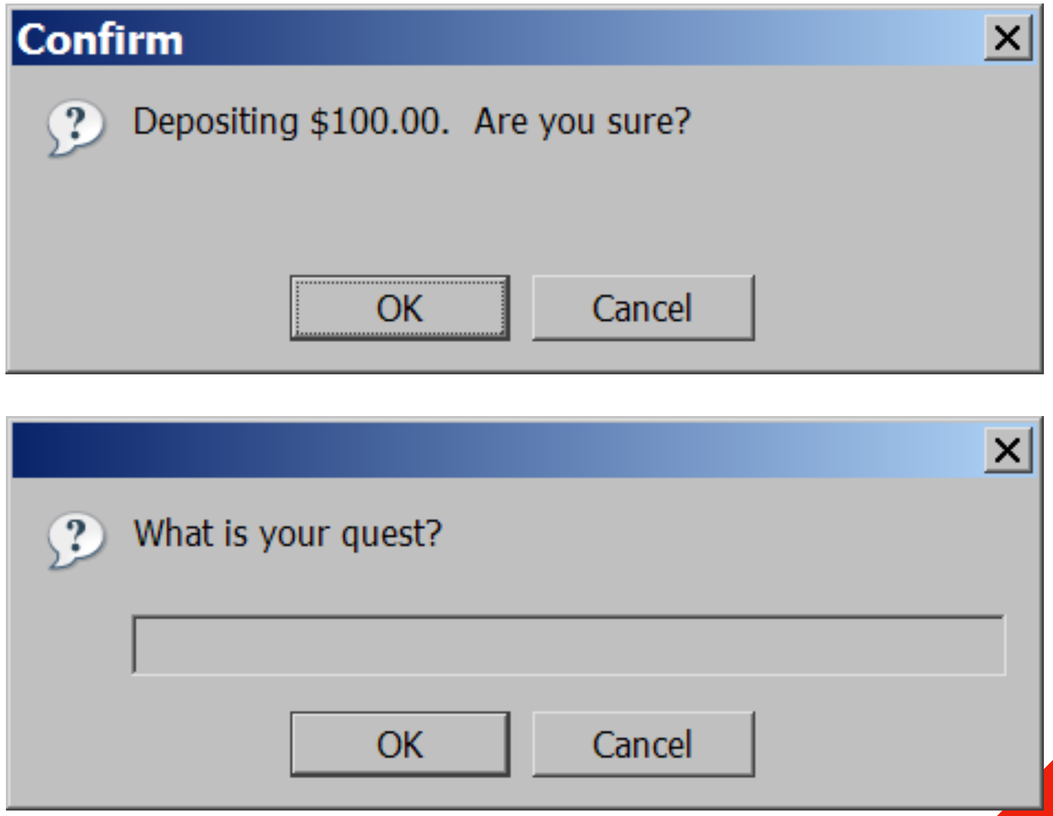
P2
JS Cont.

Popup boxes

```
alert("message"); // message

confirm("message"); // returns true or
false

prompt("message"); // returns user
input string
```



Concept

P2
JS Cont.

Arrays

```
var name = []; // empty array

var name = [value, value, ..., value]; // pre-filled

name[index] = value; // store element


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
```

Concept

P2
JS Cont.

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
```

- 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

Concept

P2
JS Cont.

`String` type

- methods: `charAt, charCodeAt, fromCharCode, indexOf, lastIndexOf, replace, split, substring, toLowerCase, toUpperCase, length`
  - `charAt` returns a one-letter String (there is no char type)
- Strings can be specified with "" or ''
- concatenation with + :
  - 1 + 1 is 2, but `"1" + 1` is `"11"`

```
var s = "Connie Client";

var fName = s.substring(0, s.indexOf(" ")); // "Connie"

var len = s.length; // 13

var s2 = 'Melvin Merchant';
```

Concept

P2
JS Cont.

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
```

- 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);
```

Concept

P2
JS Cont.

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"
```

- 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

Concept