INFT1206 - Web Programming

Week 1: Introduction to JavaScript

Agenda

- Introduction to JavaScript
- Basic JavaScript Rules
- JavaScript data types
- JavaScript Variable
- Expressions
- Strings concatenation
- Programming Constructs(Sequence, selection, loop)

Introduction to JavaScript

- JavaScript (sometimes shortened to JS) is a lightweight, interpreted, high-level language used along with html code.
- The language syntax is somewhat similar but not the same as the C language. Today, JavaScript is the scripting language for Web pages.
- JavaScript is not Java
- An interpreted language interprets and executes each statement - one-by-one - in the order they appear.
- JavaScript always runs inside a host environment (mostly the browser).
- The JavaScript standard is based on the European Computer Manufacturers Association (ECMAScript). As of 2012, all modern browsers fully support ECMAScript 5.1.

Introduction to JavaScript (cont'd)

- JavaScript is useful for making dynamic web pages, validating user input and changing the way the web page responds to events on the web page.
- JavaScript statements can be stored in an external file with a .js file extension or embedded within HTML code.
- JavaScript is one of the world's most popular programming languages.
  - the role as the scripting language of the WWW.
  - simple and easy to learn
- JavaScript is the world's most misunderstood programming language.
  - The name, typecasting, used by amateurs, object-oriented,…
- JavaScript may be, in the future, the most important language you will learn.

Basic JavaScript Rules

- JavaScript is **case-sensitive**
  - When writing a JavaScript script, be aware of upper and lower case characters.
  - CustomerCount is not the same as Customercount nor is it the same as customerCount
- JavaScript statement
  - A JavaScript typically consists of a series of statements.
  - A statement is a single line of instruction to the computer made up of objects, expressions, variables, and events/eventhandlers.
- Command block
  - A Command block is a group of statements that is treated as a single entity and are grouped within braces - the curly brackets - { }

Basic JavaScript Rules

- Matching Pairs
  - Opening and closing symbols need to work in pairs.
  - For example, if you use the left brace { to indicate the start of a command block, then you must use the right brace } to end the command block. The same matching pairs applies to single '.....' and double "....." quotes to designate text strings.
- The use of comments
  - Block/Multi-line comment: /* */
  - Single line comments: //
- The use of white Space
  - JavaScript ignores extras spaces however it is recommended that you use them to make your scripts easier to read.

JavaScript data types

- There are 3 main (primitive) data types:
  - string
    - must be enclosed in single or double quotes
  - number
    - can be integers or floating point
    - Special number: Infinity, NaN
  - boolean
    - values are binary, with the values (1) "true" and (0) "false" (without the quotes)
- Other types:
  - undefined, null, object, function

JavaScript data types

- JavaScript is a loosely typed language.
  - You do not have to specify the data type of a variable when you declare it.
  - Data types are converted automatically as needed during script execution.

JavaScript Variable

- Variable naming rules are: Must start with a letter, underscore (_), or dollar sign ($)
- Cannot be a reserved (key) word
- Subsequent characters can be letters
  - upper case (A...Z) or lower case (a...z),
  - numbers
  - underscores
- JavaScript reserved words
  - Similar to other programming languages, JavaScript has a list of words that are considered "reserved".

Declare and Refer Variables

- You must use the "var" keyword to precede a variable name.
- Unlike the C language, you do not need a type specifier.
  - The variable's initial value will set its initial type.
- Declaration syntax:

```
var variableName;
```

Or:

```
var variableName = "Summer";

// Referring to and using syntax:
variableName = 2015;
console.log(variableName);
```

- Dynamic typing
  - a JavaScript variable can have a different type in different parts of a program

Variables Example

Declaration	Type	Value
var identOne = "some text";	String	some text
var identOne = 'some text';	String	some text
var IdentOne = '172';	String	172
var _identOne = 25;	Number (Integer)	25
var _identTwo = 56.2564;	Number (float)	56.2564
var ident_A = true;	Boolean	true (1)
var ident_B = false;	Boolean	false (0)
var ident_C;	undefined	undefined
var ident_D="Yes", ident_E="No";	String / String	Yes / No

Special values

- Infinity
  - Number data type
  - e.g. console.log(12/0);
- NaN
  - means "Not a Number"; Number data type
- null
  - both a value and a data type
- undefined
  - both a value and a data type
  - e.g. var x;
    console.log(x);

console.log()

- The console.log function to show a messages in a web console.
- Example code:

```
var name = "Some text;
console.log(name);
```

- open web console and run the codes

prompt()

- Prompt box, displays a dialog box (Modal window) that prompts the user for input.
- Returns a string entered by the user input; or return the value null.
- Example code:

```
var cl = prompt("Enter your favorite color", "green");

if (cl) { // if cl is not null
    console.log (cl);
} else {  // cl is null
    console.log ("No color entered.");
}
```

- Note: you should not use this function in a web app.

Expressions

- An expression is any valid set of literals, variables, operators, and expressions that evaluates to a single value.
- The value may be a number, a string, or a logical value.
- Two types of expressions:
  1. those that assign a value to a variable, e.g. x = 7 .
  2. those that simply have a value, e.g., 3 + 4 simply evaluates to 7; it does not perform an assignment.
- JavaScript has the following kinds of expressions:
  1. Arithmetic - evaluates to a number
  2. String - evaluates to a character string
  3. Logical - evaluates to true or false

Expressions - Ternary Operator

- A conditional expression can have one of two values based on a condition. The syntax:

(condition) ? val1 : val2;

- If the condition is true, the expression has the value of val1, Otherwise it has the value of val2.

condition	When True	When False

var status = (age >= 18) ? "adult" : "minor";

Arithmetic Operators

Operator	Operation	Example
+	addition of numbers Concatenation of strings	y + x; "INT" + "222"
-	subtraction	x - y;
*	multiplication	x * y;
/	division	x / y;
%	modulo	x % y; // remainder of x divided by y

++	post/pre -increment	x = y++; // assign y to x, then increment y (y+=1)
		x = ++y; //increment y < (y+=1), then assign y to x

--	post/pre decrement	x = y--; // assign y to x, then decrement y (y-=1)
		x = --y; // decrement y < (y-=1), then assign y to x

Arithmetic Operators - Assigning Values

Operator	Example	Equivalent	For
=	a = b;	a = b;	Numbers, strings, …
+=	a += b;	a = (a + b);	numbers or strings
-=	a -= b;	a = (a - b);	numbers
*=	a *= b;	a = (a * b);	numbers
/=	a /= b;	a = (a / b);	numbers
%=	a %= b;	a = (a % b); // divide a by b, // assign remainder to a	numbers

Logical Operators

- Given x = 2;

Operator	Operation	Example
&&	Logical AND	(x > 3 && x == 2) is false
||	Logical OR	(true || x > 10) is true
!	Logical NOT	!(x ==2) is false

Comparison Operators

Operator	Description	Example
==	Equal (The operands are converted to the same type before being compared.)	1 == 1 is true 1 == "1" is true 1 == true is true 0 == false is true
===	Strictly equal (There is no type conversion.)	1 === 1 is true 1 === "1" is false 1 === true is false 0 === false is false
!=	Not equal (with type conversion)	1 != 1 is false 1 != '1' is false
!==	Not equal (without type conversion)	1 !== 1 is false 1 !== '1' is true
>	Greater than	expr1 > expr2
>=	greater than or equal to	expr1 >= expr2
<	Less than	expr1 < expr2
<=	less than or equal to	expr1 <= expr2

Other Operators

- The typeof operator (for variable or values):
  - possible return values:

```
typeof "John"                 // Returns string
typeof 3.14                   // Returns number
typeof false                  // Returns boolean
typeof [1,2,3,4]              // Returns object
typeof {name:'John', age:34}  // Returns object
```

- The instanceof operator
  - Used for objects

Strings and Quotation Marks

- Literal strings can be denoted by either single or double quotes, which gives you some flexibility about how to handle awkward situations such as quotation marks inside a string:

Expression	Values
"Let's start with JavaScript"	Let's start with JavaScript
'Not "it"!'	Not "it"!

Concatenation of Strings

- The main operation on strings is the concatenation operator, +:

Expression	Value
`"WEB" + "222"`	`WEB222`
`"John" + " Smith"`	`John Smith`

Adding Strings and Numbers

- x =5+5;
  console.log(x); // Output: 10
- x="5"+"5";
  console.log(x); // Output: 55
- x=5+"5";
  console.log(x); // Output: 55
- x="5"+5;
  console.log(x); // Output: 55

Example - Evaluating Expressions

```
var x = 5;
x = x + "2";
console.log("The value of x is " + x);

x = 2 * x;
console.log("The new value of x is now " + x);

x = x + 1;
console.log("x is now " + x);
console.log("x divided by 3 is: " + x/3);
```

Programming Constructs

- JavaScript execution flow is determined using the following four (4) basic control structures:
  1. Sequential:
     an instruction is executed when the previous one is finished.
  2. Conditional
     a logical condition is used to determine which instruction will be executed next - similar to the "if" and "switch" statements in C.
  3. Looping
     a series of instructions are repeatedly executed until some condition is satisfied - similar to the "for" and "while" statements in C.
  4. Transfer
     jump to a different part of the code - similar to calling a function in C.

Construct (1) - Sequence

- Tasks are executed one after another in "sequence" – e.g.

```
var a = 3;
var b = 6;
var c = a + b;

console.log(c);
```

Construct (2) - Selection

- Make decisions and perform single or multiple tasks based on the outcome of the decision (true or false).
- Types of conditional statements :
  - if
  - if / else
  - switch / case

if-else Example

```
var grade, mark = 86;

if (mark >= 90)
    grade='A+';
else if (mark >= 80)
    grade='A';
else if (mark >= 70)
    grade='B';
else if (mark >= 60)
    grade='C';
else if (mark >= 50)
    grade='D';
else
    grade="F";

console.log( "Your grade: " + grade);
```

Switch-case Example

```
var semester = 3;

switch (semester) {
        case '1':   console.log("IPC144, ULI101");
                    break;
        case '2':   console.log("OOP244, WEB222");
                    break;
        case '3':   console.log("OOP344, INT322");
                    break;
        case '4':   console.log("JAC444, INT422");
                    break;
        default:
                    console.log("You may have graduated from CPD");
}
```

Construct (3) - Iteration

- Loop - an action that occurs again and again until a certain condition is met.
- Continuously check a condition and based on the outcome, either terminate the loop or repeat a set of statements.
- Three basic types of loop structures:
  - The for loop
  - The for / in loop
  - The while loop
  - The do-while loop

for loop Example

```
var days = "The days in september: \n";

for (var ident = 1 ; ident <= 30 ; ident++) {
        days += ident + "\n";
}
console.log(days);
```

for in loop Example

- Iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

```
var student = {name:"John", program:"CPD", semester:2};
var str = "Student info:\n\n";

for (var x in student) { // x is the current property ('key') – ie: name, program, or semester
    str += x + ": " + student[x] + "\n";
}
console.log(str);
```

while & do…while loop Examples

```
var text = "";
var i = 0;
while (i < 10) {
    text += "\nThe number is " + i++;
}
console.log(text);
```

```
var i=10;
do {
    console.log("week " + i++);
} while (i<15)
```

break and continue Statements

- break: breaks the loop and continue executing the code that follows after the loop (if any).

```
var i=1;
while (i<5)    {
  console.log("week "+i);
  if (i==3)
    break;
  else
    i++;
}
```

- continue: breaks one iteration (in the loop), and continues with the next iteration in the loop.

```
var i=1, j=1;
while (i<5) {
   console.log('week: ' + i );
   for (j=1; j<=7; j++){

      if (j==3) continue;
            console.log('day:'+ j +'of
week:'+ i);
   } // for
   i++;
```

Thank you!

Any Questions?