CSIS 3380
Advanced Web Programming with JavaScript & AJAX

Basic Javascript

Week 2

What is Javascript?

- A programming language that is used to make web pages interactive
- Interpreted language
- Runs in client's browser

Javascript Fundamentals

- Variables
- Arrays
- Loops
- Conditionals
- Comparsion Operators
- Objects
- Functions
- Program Flow
- Data Types
- Events

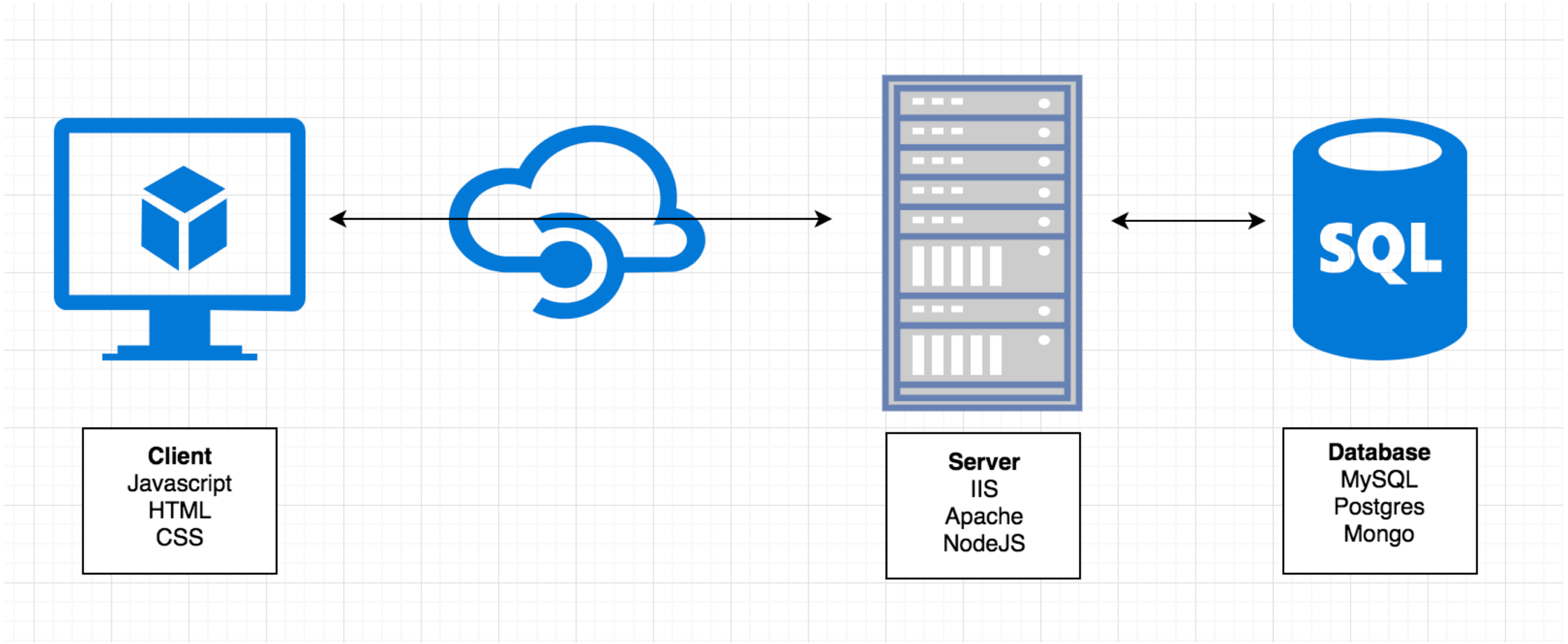
What do I need to run Javascript?

- All you need to run Javascript is a modern web browser
  - Google Chrome
  - Mozilla Firefox
  - Safari
  - Internet Explorer

Text Editor and IDE

- You can use any text editor to code in javascript. The editor has color coding and code indentation to make the program easier to read:
  - Notepad++
  - VIM
  - VS Code

Javascript and its place in a Webpage



Javascript

- The JavaScript is executed by the browser's JavaScript engine, after the HTML and CSS have been assembled and put together into a web page.
- We can use the <script> element to add javascript to a HTML page
- The javascript can part of the HTML page or in an external .js file
- Example:

```
<!-- internal javascript -->
<script>
  console.log('Hello World');
</script>
```

```
<!-- external javascript -->
<script src="script.js"></script>
```

Variable

- A variable is used to store some data
- Although JavaScript understand different data types, it doesn’t declare specific data types such as numbers, strings.
- Variable names should be consists of letters, digits, underscores, and dollar signs
- Case sensitive
- Reserved words can’t be used as var names.
- E.g. var n1 = 2; var name = ”Edmund”;

Operator

- Assignment operator
  - var x = 10;
- Arithmetic operators
  - +, -, *, /, %
  - Increment (++) or decrement (--)
- String operators
  - + (can be used to add two strings or string and numbers)
- What will be the following values of x?:
  - var x = 1 + 15;
  - var x = ’foo’ + ‘bar’;
  - var x = ‘foo’ + 1;

Operator

- Comparison operator
  - == equal to
  - === equal value and equal type
  - !=   not equal
  - !== not equal value or not equal type
  - >   greater than
  - <   less than
  - >= greater than or equal to
  - <= less than or equal to

Operator

- Logical operator

Operator	Description
Logical AND (**&&**)	Returns expr1 if it can be converted to false; otherwise, returns expr2. Thus, when used with Boolean values, && returns true if both operands are true; otherwise, returns false.
Logical OR (**||**)	Returns expr1 if it can be converted to true; otherwise, returns expr2. Thus, when used with Boolean values, || returns true if either operand is true; if both are false, returns false.
Logical NOT (**!**)	Returns false if its single operand that can be converted to true; otherwise, returns true.

Logical operator

- Example:

```
var a1 = true && true;       // t && t returns true
var a2 = true && false;      // t && f returns false
```

```
var o1 = true || true;        // t || t returns true
var o2 = false || true;       // f || t returns true
```

```
var n1 = !true;               // !t returns false
var n2 = !false;              // !f returns true
```

Conditional

- if ... else statements

```
if (condition) {
  //  code to run if condition is true
} else if (other condition) {
  // code to run if other condition is true
} else {
  //  run some other code instead
}
```

```
function weather(choice) {
  if(choice==='sunny') {
    console.log(‘It is nice and sunny outside today’);
  } else if(choice==='rainy') {
    console.log('Rain is falling outside’);
  } else {
    console.log(’no info’);
  }
}
```

- What will the following code prints out?

```
if(1 === 2 || 'Cat' === 'Cat') {
    console.log('OR operator works in if statement');
} else {
    console.log('Something is wrong');
}
```

Conditional

- switch statement

```
switch (expression) {
    case choice1:
         run this code
         break;
    case choice2:
         run this code instead
         break;
    // include as many cases as you like
    default:
        actually, just run this code
}
```

```
function weather(choice) {
  switch (choice) {
    case 'sunny':
       console.log(‘It is nice and sunny outside today’);
       break;
    case 'rainy':
       console.log('Rain is falling outside’);
       break;
    case 'snowing':
       console.log(‘The snow is coming down’);
       break;
    case 'overcast':
       console.log(‘The sky is grey and gloomy’);
       break;
    default:
      console.log(’no info’);
  }
}
```

Function

- A function allows you to store a piece of code that does a single task inside a defined block
- A function is executed when it is invoked (called)
- Some functions require **parameters** to be specified when you are invoking them — these are values that need to be included inside the function parentheses, which it needs to do its job properly
- Example:

```
function add(p1, p2) {
   return result;
}
var result = add(12,20);
```

Function

- Different ways to write a function:
  - Function expression

```
var myGreeting = function() {
  console.log('hello');
};
```

  - Function declaration

```
function myGreeting() {
  console.log('hello');
}
```

  - Anonymous function

```
function() {
  console.log('hello');
}
```

- Just stick to one way if you find it too confusing

Comment

- it is possible to write comments into your JavaScript code that will be ignored by the browser
- A single line comment is written after a double forward slash (//)

```
//   I am a comment
```

- A multi-line comment is written between the strings /* and */

```
/*
    I am also
    a comment
*/
```

var or no var

- Is there any difference?

```
/*** any difference? ***/
var a = 2;
a = 2;
```

- If you are in a function, var will create a local variable.
- The absence of a var will cause the program to look up the scope chain until it finds the variable or hits the global scope (at which point it will create the variable)

Scope chain

- Run the below program. Observe how the values for *foo* and *bar* changes after the invocation of the function

```
var foo = 1;
var bar = 2;

function myFunction() {
  var foo = 3; // Local
  bar = 3;     // Global
}

console.log('foo before myFunction invocation = ', foo);
console.log('bar before myFunction invocation = ', bar);
myFunction();
console.log('foo after myFunction invocation = ', foo);
console.log('bar after myFunction invocation = ', bar);
```

Troubleshooting

- The console.log() is a really useful debugging function that prints a value to the console.
- The Chrome Developer Tools (DevTools for short), are a set of web authoring and debugging tools built into Google Chrome.
- To open the DevTools, in the Chrome browser, right-click on any page element and select Inspect
- You can view the HTML in the **Element** panel and the output of the console.log in the **Console** panel

Running javascript

- Instead of relying on the browser to execute the javascript, you can run the javascript using **NodeJS**
- NodeJS is already installed on the lab PC
- Create a javascript that prints 'Hello' and save it as **hello.js**
- Open a command prompt, navigate to the location of the file and execute the hello.js with the command: ***node hello.js***
- Use the Nodejs for lab exercise this week

Lab

- Write a javascript function named **fahrenheitToCelsius** that converts Fahrenheit to Celsius.
- Invoke the function and store the result in a variable
- Print the result to the console
- The formula to convert °F to °C is:

$$T_{(°C)} = (T_{(°F)} - 32) \times 5/9$$

Lab

- Create a global variable representing a bank account balance, initialized to zero
- Write the function **deposit(amount)**
  - Should check that amount being deposited is positive before adding it to the bank account balance
  - If the amount being deposited is not positive, then you should produce an error message
- Write the function **withdrawal(amount)**
  - Should check that the amount being withdrawn is positive, and that there is enough money in the bank account balance for the withdrawal to occur
  - If there isn’t enough money for the withdrawal, produce an insufficient funds error
  - If the amount being deposited is not positive, then you should produce an error message
- Now make the function calls to withdraw $100, deposit $2500, deposit $55 and withdraw $1000