ITSE412– Week 3

JavaScript Introduction

Agenda

- What is JavaScript?
- JavaScript characteristics
- Integrating JavaScript into your web documents
- Objects, Properties, and Methods

What is JavaScript?

- JavaScript is an <u>interpreted</u>, <u>object-based</u>, <u>scripting language</u> similar to C++.
- Interpreted language: an interpreter is needed to translate Javascript code into machine code.
- JavaScript Interpreter is built into the Web browser.
- Object-based language: most of client-side JavaScript objects come from Web document elements such as image, form, and table elements.

How to run Javascript code?

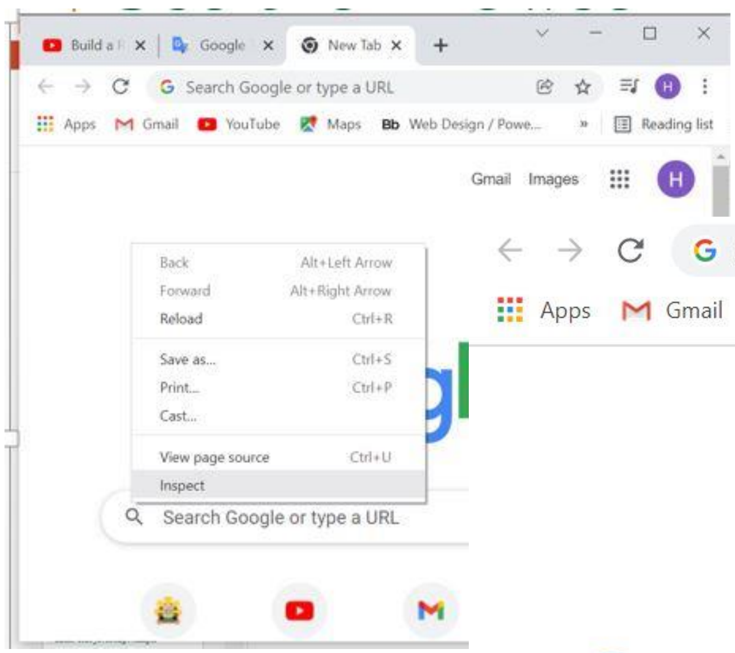
- There are three ways to run JavaScript:
  - JavaScript inside the <body> section of HTML doc.
  - Use the development tool in the browser.
  - Use Node.js

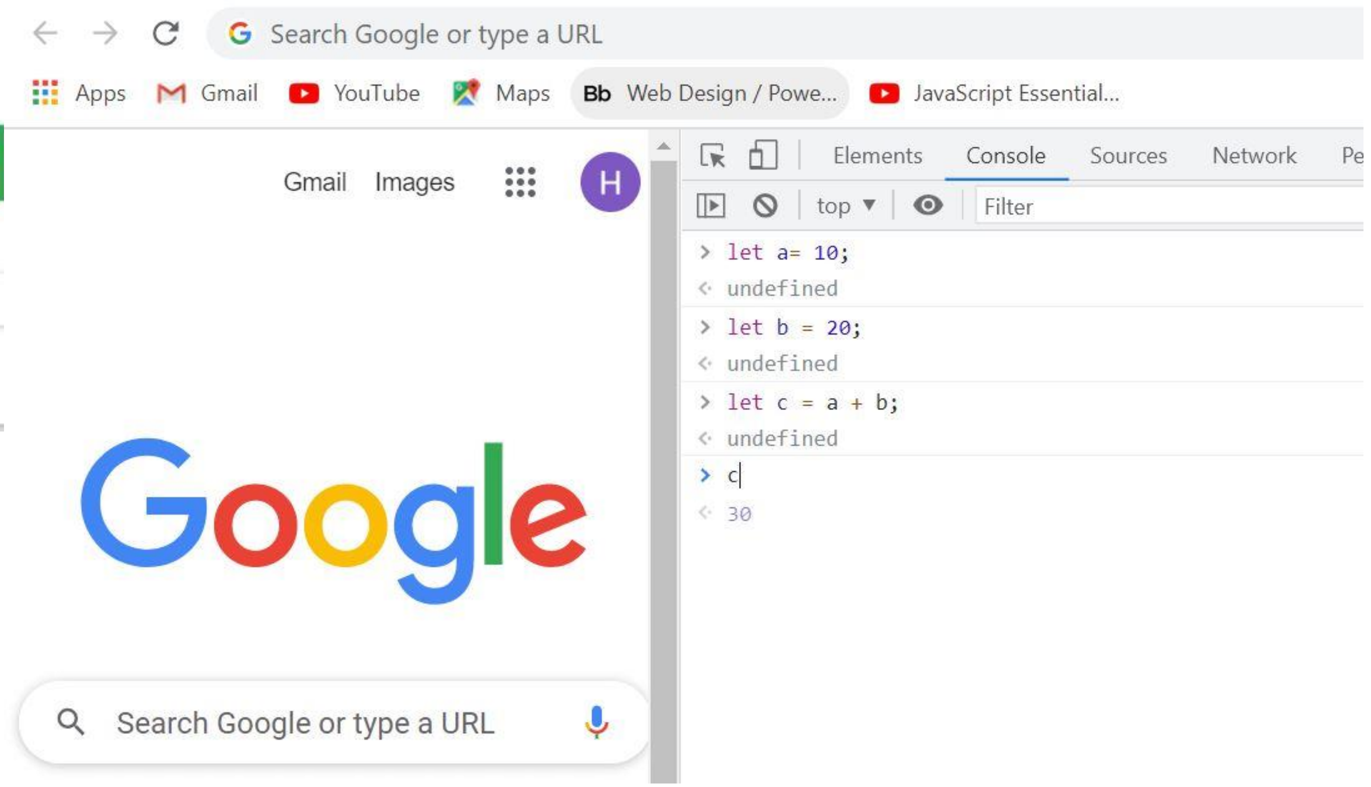
JavaScript inside the <body> section

- Directly code the JS inside the body element.
- Use the document.write() or console.log() functions to display the output.
- Inside the <body> element:

```
<script>
var a = 10;
document.write("Resut is: " + a);
</script>
```

Use the Browser Inspect ->console





```
> let a= 10;
< undefined
> let b = 20;
< undefined
> let c = a + b;
< undefined
> c
< 30
```

Use Node.js

- Download and run Node.js
- Use c:\> node or
- Write code into filename.js and run it using c:\>node filename.js

Client-Side JavaScript sample

```
<html>
<head><title>JavaScript sample</title>
</head>
<body>
<h1>Client-side JavaScript sample</h1>
<script language="javaScript" type="text/javascript">
/* use the document object to write “Hello World” to the
    html page /*
document.write("Hello World! ");
</script>
</body>
</html>
```

JavaScript is Case Sensitive

- JavaScript is a case-sensitive language:

```
Document.Write("Hello!");     // NOT OK
document.write("Hello!");     // OK
```

- JavaScript ignores "Whitespaces" (spaces, tabs, and newlines) that appear between token in programs.
- JavaScript statements end with semicolon (;). (but still optional)
- JavaScript automatically inserts semicolons before a line break.

```
var mytext = "abc";     // OK
var mytext = "abcd
          efgh "; //NOT OK – a line break inserted into a
string
```

Literals/Identifiers

- A **literal** is a data value that appears directly in a program.

```
12                 //the number twelve (number literal)
1.2                //the number one point two (number literal)
"hello world"      //a string of text (string literal)
'Hi'               //another string (string literal)
True/false         //a Boolean value (Boolean literal)
null               //absence of an object (special value)
```

- **Identifiers** are used to name variables and functions.
  - Identifier rules:
    - ✓ The first character must be a letter or an underscore (_).
    - ✓ Subsequent characters can be a letter, a digit, or an underscore.

      Ex: a, abc, _abc , ab1, aName, aName123 firstName, first_name …….

The <script> tag

- The <script> tag has two important attributes : language and type.

  ```
  <script language="JavaScript"
     type="text/javascript">
    code…..
  </script>
  ```

- The src attribute: you only need to set this attribute when you attach an external JavaScript file.
  - An external JS file is a text file with a .js extension (filename.js)
  - Only contain JS statements
  - <script> tags are unnecessary in the js file

How to integrate JS code into your web page?

- In the <head> of an HTML document
- In the <body> of an HTML document
- Inline with HTML as an event handler
- In an external JavaScript file

Placing JS statements in the HTML <head> section

- Code is executed before the contents of your Web document (in the <body> tag) load.
- Good place to declare user-defined functions
- Good place to declare global variables
- Should never place statements that “write” Web page content in here.

Placing JS statements in the <body> section

- This is the best, and only, place to write statements that actually produce content for the inclusion in an HTML document.
- Calls to functions that declared in the <head>

JS Example: Custom Greeting

```
<html>
<head><title>Custom Greeting</title>
<script language="JavaScript" type="text/javascript">
/* Global variable */
var visitor = prompt("What is your name?", "");
</script>
</head>
<body>
<h1>Custom Greeting</h1>
<script type="text/javascript">
document.write("<h1>Welcome, ",  visitor,  "</h1>");
</script>
</body>
</html>
```

Writing JS statements inline as event handlers

```
<html>
<head><title>JS inline as event handler</title>
</head>
<body onload="alert('Welcome!');">
<h1>JS inline with event handler</h1>
</body>
</html>
```

Placing JS statements in an external JS file

- An external JS file is a simple text file containing only JS statements whose name has a .js extension.
- Used to declare functions, especially functions you plan to use again and again.
- By using an external JS file, you can reduce the overall loading time of your Web site.
- The external JS file will be loaded once, the first time the visitor requests a page that uses it. Any future pages that use that file can access it from the cache.
- Using an external JS file, you can begin building a library of frequently used functions and routines. Such as formValidation.js

Object Oriented Concepts

- Object: is an item that has: *attributes/properties* which describe it; and *methods* which are actions that you can perform with the object.
- JavaScript uses *dot* notation to refer to an object and its associated properties and methods.
- For example, if pen is an object which has a property (inkColor) and a method (write)
- To change the value of the property inkColor to blue:

```
Pen.inkColor = "blue";
```

- To use the object's method, we call the write method:

```
pen.write("Hello");
```

Using the write method

```
<html>
<head><title>Using the write method</title>
</head>
<body>
<h1>Using the write method</h1>
<script language="JavaScript"
   type="text/javascript">
document.write("Hello World!!");
</script>
</body>
</html>
```

Using the write method to write HTML data

```
<html>
<head><title>Using the write method</title>
</head>
<body>
<h1>Using the write method to write HTML data</h1>
<script language="JavaScript" type="text/javascript">
document.write("<h1 style='text-align: center;'>Hello
    World!!</h1>");
</script>
</body>
</html>
```

Changing the background and foreground colors

```
<html>
<head><title>change the background and foreground colors</title>
</head>
<body>
<h1>Changing the background and foreground colors</h1>
<script language="JavaScript" type="text/javascript">
document.write("<h1 style='text-align: center;'>Hello World!!</h1>");
document.bgColor = "blue";
document.fgColor = "white";
</script>
</body>
</html>
```

Where does JS Objects come from

- Built into the language, like Math, String, Date and Array (Core JS)
- Come from Web documents and are made available to Client-Side JS via the Document Object Model (DOM)
- Come from the browser, such as navigator, location, and history objects also made available to Client-Side JavaScript by the DOM
- Programmers create our own custom objects

Summary of Object Oriented Concepts

	Description	Real-world example	JavaScript example
Object	An item or thing	pen	document
Properties	An attribute that describes an object	pen.inkColor	document.bgColor
Method	An action that can be performed with or on an object	pen.write()	document.write()