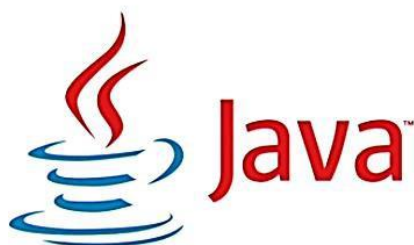# JavaScript

JavaScript is a high-level, scripting language, interpreted programming language. Alongside HTML, CSS, JavaScript is one of the three core technologies of web developing. It is used to make webpages interactive. The filename extension for JavaScript is **.js**

Java

A programming language used to write applications for computers and other devices

JavaScript

A scripting language used to interact with content in a web browser and more recently in other places

JavaScript is a scripting programming language, which is a program that is written for a special run-time, step-by-step, environment. In JavaScript, whatever happens first happens first and then the next line happens afterward.

The power of programming language is the ability to perform the same operation on many different objects, or change that object itself over and over.

How HTML, CSS, and JavaScript fit together?

We developers usually talk about three languages that are used to create web pages: HTML, CSS, and JavaScript. Where possible, aim to keep the three languages in separate files, with the HTML page linking to CSS and JavaScript files.

<html>	css { }	js ( )
.html files	**.css files**	**.js files**
This is where the content of the page lives. The HTML gives the page structure and adds semantics.	The css enhances the html page with rules that state how the html content is presented (backgrounds, borders, box,	This is where we can change how the page behaves, adding interactivity.

Example) Only html file

Queensborough Community College

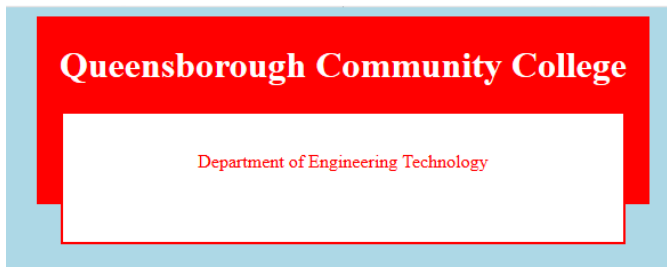Department of Engineering Technology

```
<!DOCTYPE html>
<html>
<head><title>Introduction to js</title></head>
<body align="auto">
  <div class="qcc">
  <h1 >Queensborough Community College</h1>
  <div class="et">
  Department of Engineering Technology</div>
</body>
</html>
```

Example) html + css

```
<!DOCTYPE html>
<html>
<head><title>Introduction to js</title><link href="css/intro.css" type="text/css"
rel="stylesheet"/>
</head>
<body align="auto">
  <div class="qcc">
  <h1 >Queensborough Community College</h1>
  <div class="et">
      Department of Engineering Technology
    <script src ="js/intro.js"></script>
</div>
</body>
</html>
```

Queensborough Community College

Department of Engineering Technology

```
body{background-color:
lightblue;}
div{margin:auto; border:2px
solid red}
div.qcc{
      background-color: red;
      text-align: center;
      width: 90%; height: 10em;
      color: white;
      text-align:center;}
div.et{
      background-color: white;
      width: 80%; height: 3em;
      padding: 2em}
```

Example) html + css + js

```
<!DOCTYPE html>
<html>
<head><title>Introduction to js</title>
<link href="css/intro.css" type="text/css" rel="stylesheet"/>
</head>
<body align="auto">
  <div class="qcc">
  <h1 >Queensborough Community College</h1>
  <div class="et">
      Department of Engineering Technology
</div>
    <script src ="js/intro.js"></script>
</body>
</html>
```

```
  var today=new Date();
  var hourNow = today.getHours();
  var greeting;

  if (hourNow > 18) {greeting = 'Good evening!';}
  else if (hourNow > 12) {greeting = 'Good afternoon!';}
  else if (hourNow > 0) {greeting = 'Good morning!'}
  else {greeting = 'Welcome!'}

  document.write('<h2>' + greeting + '<h2>')
```
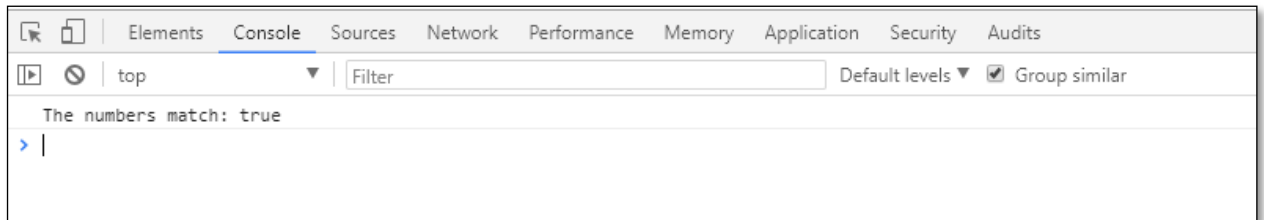
Basic JavaScript instructions

Writing and debugging

JavaScript console is a very handy tool for learning basic JavaScript in general, and with the debugger, it can give you great insight into your code. You can do this either by using the View, Developer, JavaScript Console menu here in **Chrome**, on the **Mac,** you can type **cmd + opt + i** or on Windows, **ctrl + shift + i** or just the function key **F12** to open it.

```
 ☐ ☐ | Elements   Console   Sources   Network   Performance   Memory   Application   Security   Audits

 ▷ ⊘ | top                    ▼ | Filter                                    Default levels ▼  ☑ Group similar

   The numbers match: true
 ❯ |
```

To test a JavaScript code, you can create a html file link to a JavaScript file. You can open the html file and open the web developer console, F12, to check your JavaScript

```
<!DOCTYPE html>
<html>
<head><title>JavaScript</title></head>
<body>
</body>
<script src="js/intro.js"></script>
</html>
```

A script is a series of instructions that a computer can follow one-by-one. Each individual instruction or step is known as **statement.** Statements should end with a semicolon **;**

JavaScript is case sensitive so a variable named **hourNow** is not the same as variable **HOURNOW** or **HourNow**.

For example, the following .js file

```
/*This script will display the greeting
message according to the system time obtain
using the command today.getHours() */

  var today=new Date();
  var hourNow = today.getHours();
  var greeting;

  if (hourNow > 18)
        { greeting = 'Good evening!'; }
  else if (hourNow > 12)
        { greeting = 'Good afternoon!'; }
  else if (hourNow > 0)
        { greeting = 'Good morning!'; }
  else
         { greeting = 'Welcome!'; }

document.write('<h2>' + greeting + '<h2>')
```

Each of the line of code in **purple** are multi-line comments.

The line in **green** is a statement.

The **red** curly braces indicates that start and end of a code block. Each code block could contain many more statements.

The code in **blue** is a javascript decision statement that indicates which code should run.

You can also check on the demo sample at https://codepen.io/ste-vg/details/GRooLza

Where and how to read and run a JavaScript source code?

A JS source code can be inspected through a console on the developer tools that comes with the web browser. Usually, when we create a .js file, we link it to the html file, and then we open the html file through a web browser. Another alternative is to add the JS code in the HTML file using a <script> element. Once the html file is running in the browser, we can open the developer tools of the web browser by right clicking on any space in the website. Once a menu opens, we can click on **Inspect** to open a developer tools. There is also a shorthand to open the developer tools in a browser, which is by using the function key F12 in Chrome or Firefox browsers.
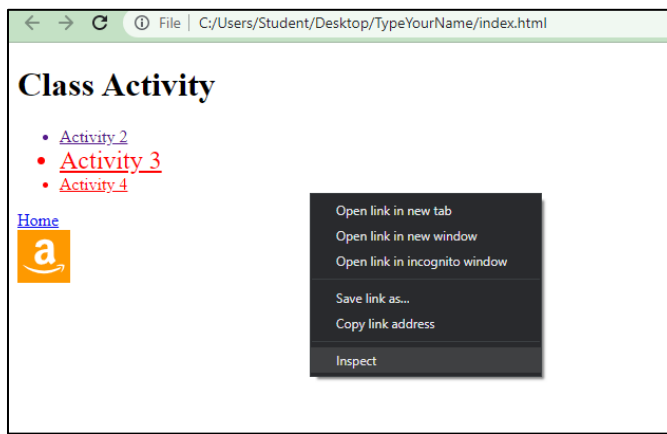
Once the developer tools is open, on the top right of the tools bar, we can see the **Console** tab.
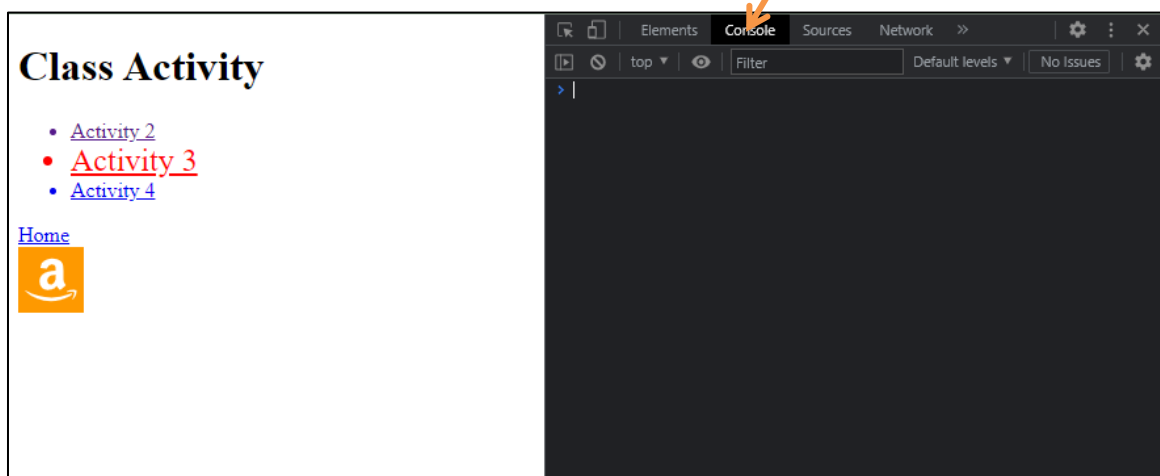
We can also use the console to have a quick check, to troubleshoot, or to test a JS code, but never to write a complete JS code.

Variables

Variables in JavaScript are containers for temporary storing data values. JavaScript traditionally uses the keyword **var** to declare a variable. But in the latest years, JS came out with a new method to declare variables known as **let**

<h1 style="text-align:center">let   number;</h1>

<div style="text-align:center">Variable key   Variable Name</div>

Rules for naming variables:

1. The name must begin with a letter, an underscore_, or money sign $

```
let x1;
let _value;
let $;
```

2. Variable names CAN'T BE keyword or reserved words.

   let *var*; ➔ *var* is not a valid name for a variable because **var** is a keyword

Reserved Keywords				
break	default	function	return	var
case	delete	if	switch	void
catch	do	in	this	while
const	else	instanceof	throw	with
continue	finally	let	try	debugger
for	new	typeof		

3. Variable names CAN'T start with a number, any other symbols or characters, and no space.

```
let 8value;
let $99;
let white-car;
let first name;
let switch;
```

4. All variables are case sensitive, so the variable **color** is different than variable **Color** or **COLOR**, but it is bad practice to create two variables that have the same name using different cases.

```
let color = 'beige';
let Color = 'red';
let COLOR = 'blue';
```

5. Use name that describes the kind of information that the variable stores. For example, **firstName** might be used to store a person's first name, and **id** for their id number. Note: try to avoid long words or names. For example, instead of naming a variable **programming1,** you can name it **prog1**

Data types

Numeric data

Numeric data handles numbers. To store a numeric data type in a variable, simply assign a number to it, could be positive, negative, or fraction.

```
let x = 100; let y = -100;
```

String data

The string data type handles strings or letters and symbols, what we commonly refer to as words and sentences. To store a string data type in a variable, wrap your string of text in straight quotation marks. You can use single or double quotes. Just make sure you use the same one at the beginning and at the end.

```
let color1 = 'beige';
let string1 = "Strings are typically words and sentences";
let string2 = 'I want to keep the "quotation" mark';
let string3 = "I want to keep the \"quotation\"";
```

Boolean data

The Boolean data type handles the binary true/false. To store a Boolean data type in a variable, type in true or false without quotation marks. The Boolean data type handles the binary true/false. To store a Boolean data type in a variable, type in true or false without quotation marks.

```
let boolean1 = true; let boolean2 = false;
```

The null data

The null data type is the intentional absence of an object value. So if we want a variable to be empty, but not undefined, you set its value to null, without quotation marks as well.

```
let emptyVar = null;
```

Undefined data

The undefined data type is what you get when you create a variable but do not set it to anything. As you've learned, JavaScript is a weakly typed language, so the data types are applied when you set the content in a variable.

```
let undefined1;
```

Arithmetic operators

JavaScript contains the following mathematical operators

Name	JavaScript Operator	Purpose and notes	Example	Results var x
Addition	+	Adds two value to another	`let x = 10 + 5`	15
Subtraction	-	Subtracts one value from another	`let x = 10 – 5`	5
Division	/	Divides two values	`let x = 10 / 5`	2
Multiplication	*	Multiplies two values using an asterisk (this is not the letter x)	`let x = 10*5`	50
Increment	++	Adds one to the current number. If ++ is placed before the variable, one is added instantly to the value.	`let x = 10;` `++x;`	11
		If ++ is placed after the variable, one is added in the next execution of the variable.	`x++;` `x;`	11
Decrement	- -	Subtracts one from the current number. If -- is placed before the variable, one is subtracted instantly to the value.	`let x = 10;` `--x;`	9
		If -- is placed after the variable, one is subtracted in the next execution of the variable.	`x--;` `x;`	9
Modulus	%	Divides two values and returns the remainder	`let x = 10 % 3`	1

Example)

```
let a = 3; let b = 2; let c = 10;
let math1 = a+c;   ➔ 13
let math2 = c/b;   ➔5
let math3 = c%3;   ➔ 1
c++;               ➔11
```

The plus operator: combining number and string
The plus operator is unique in this regard**. It's the only one of the arithmetic operators** that is also a string operator, meaning you have to be careful about what type of data your variables contain when you use it.

Material prepared by professor H. Wu

```
let a = '3', b = 2, c = 10;
let math1 = a+b;            (output) ➔ '32'
typeof(math1)              (output) ➔ string
```

If we use one of the other operators, say **a** minus **b** instead, the math works even though we have a number and a string. JavaScript reads it as: you are trying to do math here and even though **a** is a string.

```
let math2 = a - b;(output) ➔  1
typeof(math2)      (output) ➔ number

let math3 = a * b;(output) ➔  6
typeof(math3)      (output) ➔ number

let math4 = a / b;(output) ➔1.5
typeof(math4)      (output) ➔ number
```

However, if you put anything other than a number in the string you're trying to subtract, multiply, or divide, bad things happen. If a is **a** number and **b** is a string and I try to say sum **a** minus **b**, then it returns **NaN**, or **Not a Number**.

```
let a = 'cats';
let b = 2;


let math1 = a + b    (output) ➔ cats2
let math2 = a – b    (output) ➔ NaN
let math2 = a * b    (output) ➔ NaN
let math2 = a / b    (output) ➔ NaN
```

How to check a type of value in a variable?
In JavaScript, to check the type of value in a variable, we can use the **typeof** operator. The typeof operator returns a string indicating the type of the unevaluated operand.

```
let nothing= null;
typeof(nothing)     (output) ➔ 'object'
let emptyVar="";
typeof(emptyVar)    (output) ➔ string
```

Material prepared by professor H. Wu

Conversion Between Data Type

From string to numerical

In JavaScript **parseInt()** function is used to convert the string to an integer. This function returns an integer of base which is specified in second argument of **parseInt()** function. **parseInt()** function returns NAN (Not A Number) when the string does not contain number.

```
let num1 = parseInt(prompt ("Enter number 1", ""));
let num2 = parseInt(prompt ("Enter number 2", ""));
let num3 = num1+num2;
prompt(`The sum of ${num1} and ${num2} is: `, num3);
```

From numerical value to string

The **toString()** returns a number as a string. The **toString()** method is used internally by JavaScript when an object needs to be displayed as a text (like in HTML), or when an object needs to be used as a string.

prompt() method

The **prompt()** method in JavaScript is used to display a prompt box that prompts the user for the input. It is generally used to take the input from the user before entering the page.

Syntax:
```
prompt(text, defaultText);
```

Example
```
prompt ("Message on top of the text field", "Text field");
```

Template literals

Using backticks `` `` `` in JavaScript along with **${}** around the expressions allows to embed information in the **${}** into the string.

Concatenating strings
```
let a = 5;
let b = 10;
console.log('Fifteen is ' + (a + b) + ' and\nnot ' + (2 * a + b) + '.');
```

Using backticks
```
let a = 5;
let b = 10;
console.log(`Fifteen is ${a + b} and not ${2 * a + b}.`);
```

Null and undefined variables

Null
- Intentional absence of any value

```
let name = null;
```

```
name
null
```

- Value must be assigned later

```
name="Will Smith"
'Will Smith'
```

Undefined
- Variable that do not have an assigned value are undefined
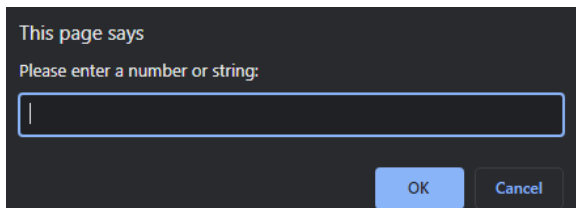
```
let x;
undefined
```

The `prompt()` method

The **`prompt()`** method displays a dialog box that prompts the user for input.
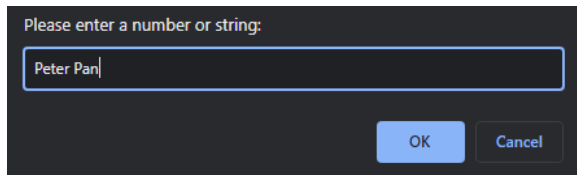
The **`prompt()`** method returns the input value if the user clicks "OK", otherwise it returns null.

Example) use `prompt()` method to collect a name from a keyboard. Then display the name in the console.

```
let name= prompt('Please enter a name: ');
console.log(`The entered name was: ${name}`);
```

When the program is ran, a dialog box appears

This page says
Please enter a number or string:

|

OK   Cancel

Please enter a number or string:

Peter Pan

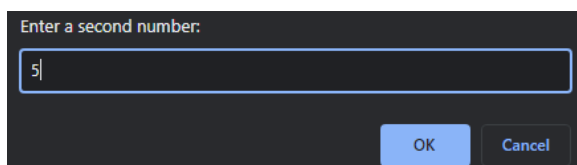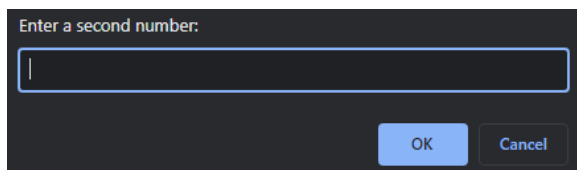OK   Cancel

If the user type *Peter Pan* and click OK
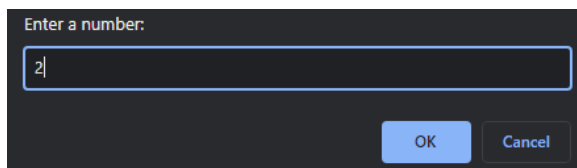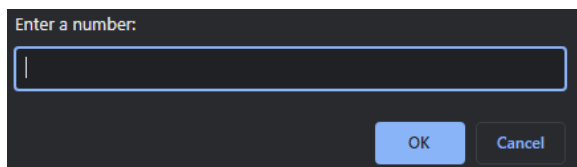
The entered name was: Peter Pan

Convert from string to integer

The value collected from the `prompt()` dialog box is a string. If we want to convert the value into an integer, we need use method `parseInt()` to convert the string into an integer.

For example, if we collect two numbers from the `prompt()` dialog box, the sum of the two numbers will be concatenated:

```
let number1 = prompt("Enter a number: ");
let number2 = prompt("Enter a second number: ");
let sum = number1 + number2;
console.log(`${number1} + ${number2} = ${sum}`);
```

Enter a number:

|

OK   Cancel

Enter a number:

2

OK   Cancel

Enter a second number:

|

OK   Cancel

Enter a second number:

5

OK   Cancel

2 + 5 = 25

Using the previous code, if we convert **number1** and **number2** to integers first before adding them, we will have:

```javascript
let number1 = prompt("Enter a number: ");
number1 = parseInt(number1);
let number2 = parseInt(prompt("Enter a second number: "));
let sum = number1 + number2;
console.log(`${number1} + ${number2} = ${sum}`);
```

```
2 + 5 = 7
```