INTRODUCTION TO JAVASCRIPT

BY: DR. PAULINE ABESAMIS

SEPTEMBER 10, 2024

WHAT IS JAVASCRIPT?

- Is the programming language of the web used by modern web browsers (desktops and tablets)
- It can be used for front-end and back-end
- Lightweight, object-oriented programming language to build web applications, games and others.
- High-level, dynamic, interpreted programming language for object-oriented
- Syntax based on Java
- Dynamic context of website.
- JavaScript's performance has improved significantly in recent years due to advancements in browser technology and the development of Just-in-Time (JIT) compilers.

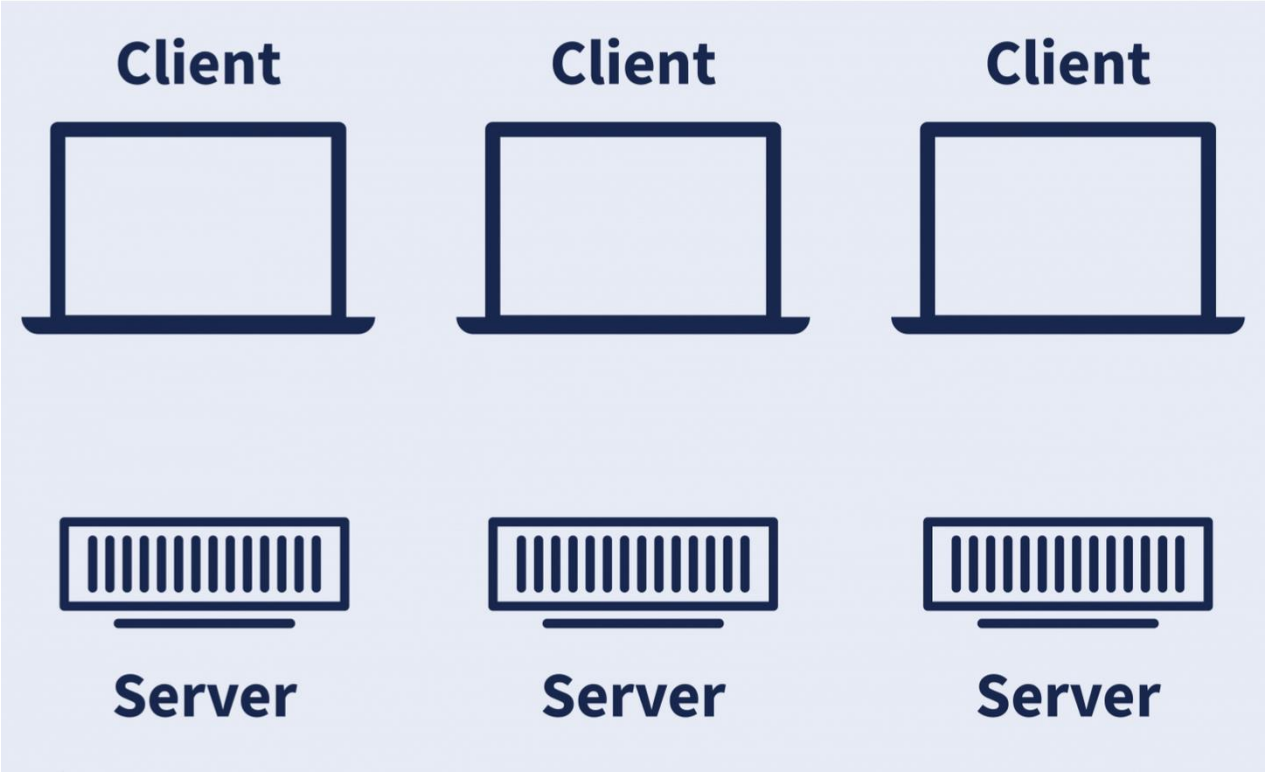
HISTORY

- The JavaScript standard was suggested for the first time as ECMAScript 1 in 1997, and is, as of late 2018, in its 9th iteration (ES 2018).
- In the late 1990s, the first version of JavaScript was developed for the Netscape Navigator Web browser.
- A Netscape developer, Brandan Eich, in September 1995, developed a new programming language.
- Initially, it was known as Mocha, but its name was changed to *LiveScript and then JavaScript*. At the time, web pages were static with minimal consumer interaction apart from loading new pages and clicking links. JavaScript, for the first time, facilitated animation, adaptive content, and form validation on the page.
- Only a small number of browsers supported JavaScript for a long time.

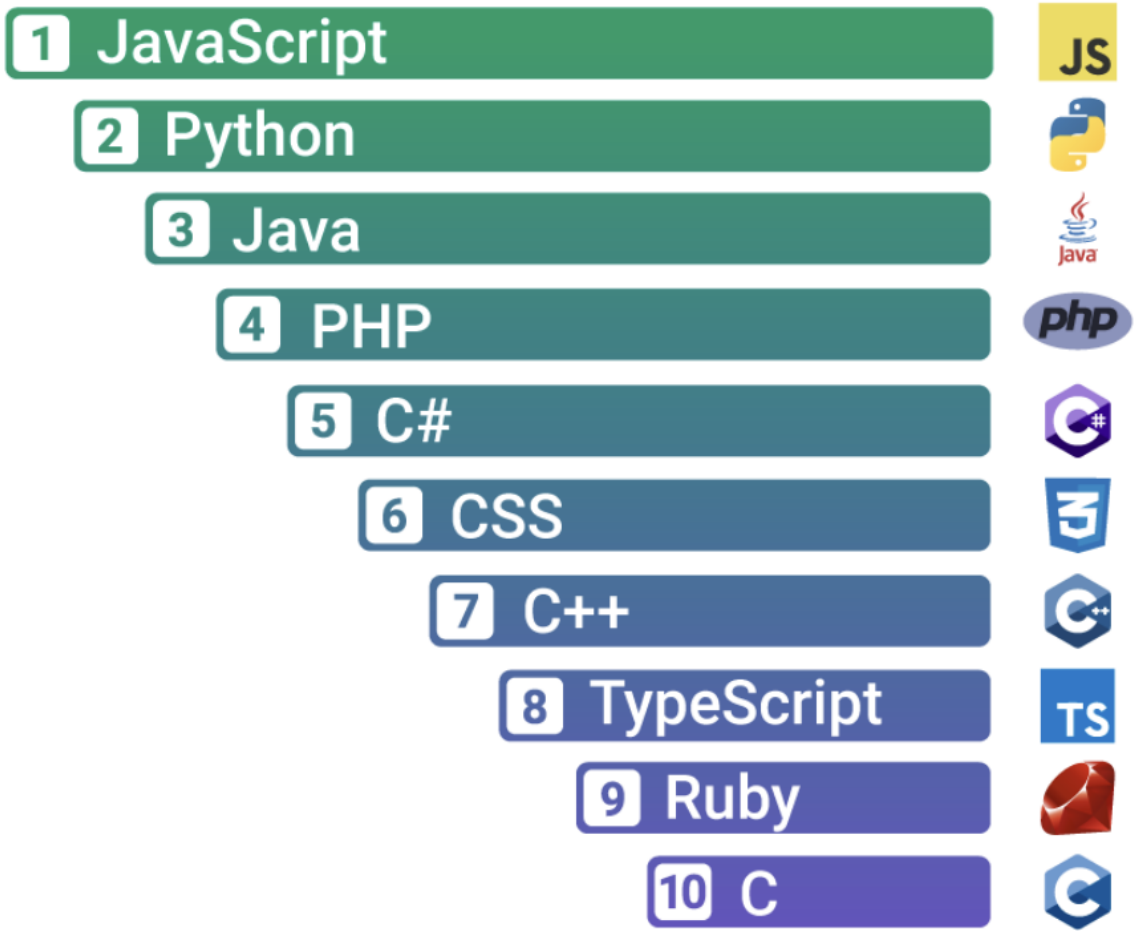
JAVASCRIPT: NAMES, VERSIONS, MODELS

- Javascript was created at Netscape in the early days of the web, trademark licensed from Sun Microsystems (Oracle) used to describe Netscape's (Mozilla)
- Netscape submitted language for standardization to ECMA – European Computer Manufacturer's Association (ECMA Script) everyone calls it "Javascipt".
- Year 2010, version 5 of the ECMAScript standard supported all web browsers.

CLIENT-SIDE WEB APPLICATIONS OF JAVASCRIPT



JavaScript Ranked Number One Programming Language by RedMonk in 2022



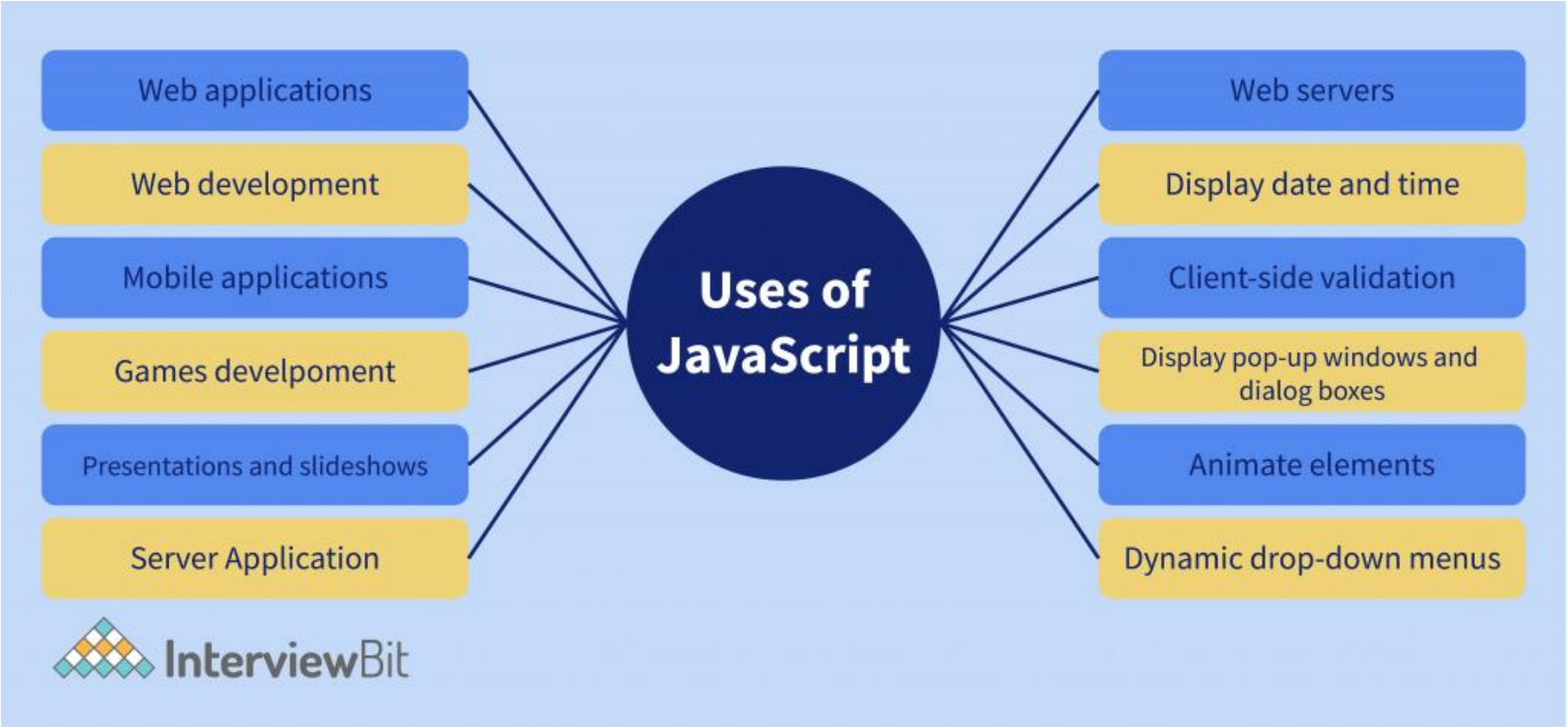
Advantages of JavaScript

- Client-Side Security
- Less Overhead
- Inherently Fast
- Easy to Implement
- Popular
- Reduces Server Load
- Versatile
- Rich Interfaces

Disadvantages of JavaScript

- Security Risks
- Interoperable
- Increased Server Load (at times)
- Lack of Browser Support
- Debugging Tools aren't Advanced
- Single Inheritance
- Slow Rendering

TOP 10 APPLICATIONS OF JAVASCRIPT



MOBILE APPLICATIONS OF JAVASCRIPT

https://www.interviewbit.com/blog/javascript-applications/

JAVASCRIPT EXTRAS:

- **Artificial Intelligence**: You can utilize JavaScript to operate on AI-related projects. With the ***Tensorflow.js machine learning library***, you can accomplish AI stuff with the help of JavaScript.
- **Embedded Systems**: Node.js is renowned for creating server-side web applications.
- **Virtual Reality** has garnered attention for pretty much 4-5 years now and is that it is shifting to web browsers..
- **WebAR is just like augmented reality one might know it, just that it can be accessed instantly from a mobile browser**.

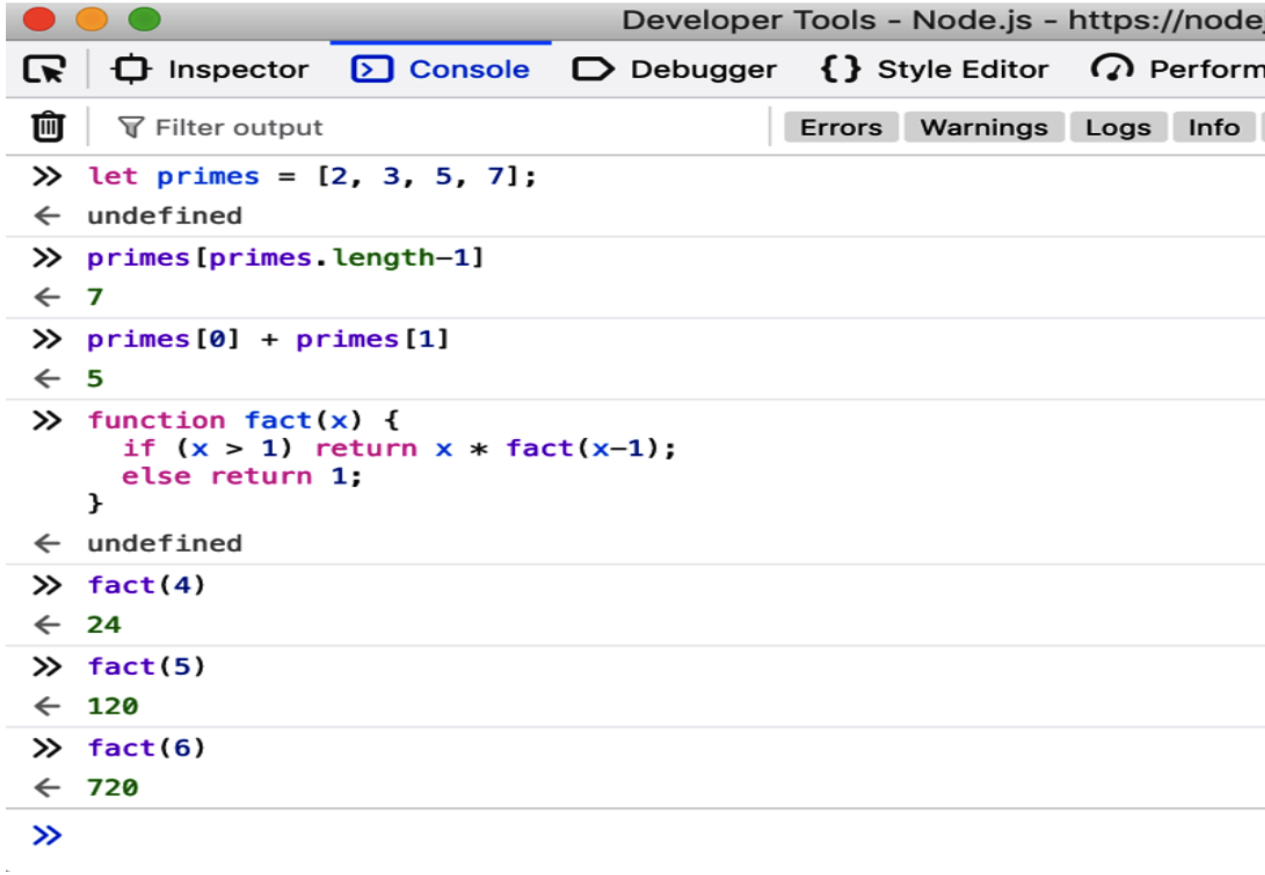
EXERCISE #3 EXPLORING JAVASCRIPT

1) Follow the steps:

The easiest way to try out a few lines of JavaScript is to open up the web developer tools in your web browser (with F12, Ctrl-Shift-I, or Command-Option-I) and select the Console tab.

You can then type code at the prompt and see the results as you type.

Browser developer tools often appear as panes at the bottom or right of the browser window, but you can usually detach them as separate windows.



EXERCISE: EXPLORING JAVASCRIPT

Inspector | Console | Debugger | Network | Style Editor | Performance | Memory | Storage | Accessibility | Application

Filter Output | Errors | Warnings | Logs | Info | Debug | CSS | XHR | Requests

```
⚠ This error page has no error code in its security info                    aboutNetError.mjs:865:13
» $ node
⊗ Uncaught SyntaxError: unexpected token: identifier [Learn More]           debugger eval code:1:3
» let primes = [2,3,5,7];
← undefined
» primes [primes.length-1]
← 7
» primes [0] + primes [1]
← 5
» function fact (x) { if (x>1) return x * fact (x-1); else return 1;}
← undefined
» fact (4)
← 24
» fact (5)
← 120
» fact (6)
← 720
⚠ Scam Warning: Take care when pasting things you don't understand. This could allow attackers to steal your identity or take control of your computer. Please type 'allow pasting' below (no need to press enter) to allow pasting.
»
```

JAVASCRIPT CODE

```javascript
// Anything following double slashes is an English-language comment.
// Read the comments carefully: they explain the JavaScript code.

// A variable is a symbolic name for a value.
// Variables are declared with the let keyword:
let x;                          // Declare a variable named x.

// Values can be assigned to variables with an = sign
x = 0;                          // Now the variable x has the value 0
x                               // => 0: A variable evaluates to its value.

// JavaScript supports several types of values
x = 1;                          // Numbers.
x = 0.01;                       // Numbers can be integers or reals.
x = "hello world";              // Strings of text in quotation marks.
x = 'JavaScript';               // Single quote marks also delimit strings.
x = true;                       // A Boolean value.
x = false;                      // The other Boolean value.
x = null;                       // Null is a special value that means "no value."
x = undefined;                  // Undefined is another special value like null.
```

1) JAVASCRIPT OBJECT

```
// JavaScript's most important datatype is the object.
// An object is a collection of name/value pairs, or a string to value map.
let book = {                // Objects are enclosed in curly braces.
    topic: "JavaScript",    // The property "topic" has value "JavaScript."
    edition: 7              // The property "edition" has value 7
};                          // The curly brace marks the end of the object.

// Access the properties of an object with . or []:
book.topic                  // => "JavaScript"
book["edition"]             // => 7: another way to access property values.
book.author = "Flanagan";   // Create new properties by assignment.
book.contents = {};         // {} is an empty object with no properties.

// Conditionally access properties with ?. (ES2020):
book.contents?.ch01?.sect1 // => undefined: book.contents has no ch01 property.
```

2) JAVASCRIPT ARRAYS

```
// JavaScript also supports arrays (numerically indexed lists) of values.
let primes = [2, 3, 5, 7]; // An array of 4 values, delimited with [ and
primes[0]                  // => 2: the first element (index 0) of the ar
primes.length              // => 4: how many elements in the array.
primes[primes.length-1]    // => 7: the last element of the array.
primes[4] = 9;             // Add a new element by assignment.
primes[4] = 11;            // Or alter an existing element by assignment.
let empty = [];            // [] is an empty array with no elements.
empty.length               // => 0
```

JAVASCRIPT ARRAYS AND OBJECTS

```
// Arrays and objects can hold other arrays and objects:
let points = [                // An array with 2 elements.
    {x: 0, y: 0},             // Each element is an object.
    {x: 1, y: 1}
];
let data = {                    // An object with 2 properties
    trial1: [[1,2], [3,4]],  // The value of each property is an array.


    trial2: [[2,3], [4,5]]   // The elements of the arrays are arrays.
};
```

JAVASCRIPT OPERATORS

```
// Operators act on values (the operands) to produce a new value.
// Arithmetic operators are some of the simplest:
3 + 2                       // => 5: addition
3 - 2                       // => 1: subtraction
3 * 2                       // => 6: multiplication
3 / 2                       // => 1.5: division
points[1].x - points[0].x   // => 1: more complicated operands also work
"3" + "2"                   // => "32": + adds numbers, concatenates strings

// JavaScript defines some shorthand arithmetic operators
let count = 0;              // Define a variable
count++;                    // Increment the variable
count--;                    // Decrement the variable
count += 2;                 // Add 2: same as count = count + 2;
count *= 3;                 // Multiply by 3: same as count = count * 3;
count                       // => 6: variable names are expressions, too.

// Equality and relational operators test whether two values are equal,
```

JAVASCRIPT OPERATORS

```
// unequal, less than, greater than, and so on. They evaluate to true or false.
let x = 2, y = 3;           // These = signs are assignment, not equality tests
x === y                     // => false: equality
x !== y                     // => true: inequality
x < y                       // => true: less-than
x <= y                      // => true: less-than or equal
x > y                       // => false: greater-than
x >= y                      // => false: greater-than or equal
"two" === "three"           // => false: the two strings are different
"two" > "three"             // => true: "tw" is alphabetically greater than "th"
false === (x > y)           // => true: false is equal to false

// Logical operators combine or invert boolean values
(x === 2) && (y === 3)      // => true: both comparisons are true. && is AND
(x > 3) || (y < 3)          // => false: neither comparison is true. || is OR
!(x === y)                  // => true: ! inverts a boolean value
```

LEXICAL STRUCTURE OF JAVASCRIPT

- Case sensitivity, spaces, and line breaks
- Comments
- Literals
- Identifiers and reserved words
- Unicode
- Optional semicolons

```
12                  // The number twelve
1.2                 // The number one point two
"hello world"       // A string of text
'Hi'                // Another string
true                // A Boolean value
false               // The other Boolean value
null                // Absence of an object
```

RESERVED WORDS

- reserved keywords that must not be used as the names of constants, variables, functions, or classes (though they can all be used as the names of properties within an object).

```
as      const      export     get         null     target   void
async   continue   extends    if          of       this     while
await   debugger   false      import      return   throw    with
break   default    finally    in          set      true     yield
case    delete     for        instanceof  static   try
catch   do         from       let         super    typeof
class   else       function   new         switch   var
```

```
enum  implements  interface  package  private  protected  public
```

STEPS TO SET-UP JAVASCRIPT

SETTING UP YOUR ENVIRONMENT

1. Download and install Visual Studio Code from the official website.
2. Open VSCode.
3. Install the "Node.js Extension Pack" from the extensions marketplace. To do this, click on the Extensions icon in the sidebar (it looks like a square with an arrow coming out of the top left corner), search for "Node.js Extension Pack," and click "Install." This pack includes a collection of extensions that will be helpful when working with JavaScript and Node.js.
4. Install Node.js, which is a JavaScript runtime that we'll use to run our JavaScript code. You can download it from the official Node.js website. Choose the LTS (Long Term Support) version, which is the most stable and recommended for most users.

CREATING A JAVASCRIPT FILE

1. In VSCode, go to the "File" menu, then select "New File" (or press Ctrl+N). This will open a new, empty file.
2. Save the file with a .js extension to indicate that it's a JavaScript file. For example, you could name it hello-world.js. To save the file, go to the "File" menu, then select "Save As" (or press Ctrl+Shift+S), and choose a location on your computer.
3. In the newly created file, type the following code:

- console.log("Hello, world!");

https://www.altcademy.com/blog/how-to-run-a-javascript-file-in-visual-studio-code/

JAVASCRIPT EXERCISES

BY: DR. PAULINE ABESAMIS

FUNCTION

```javascript
/// Previously
// function expression
let product = function(x, y) {
    return x * y;
};

result = product(5, 10);

console.log(result);  // 50
```

```
[Running] node "c:\Users\reyes\Documents\Javascript Lessons\Javascript\function.js"
50

[Done] exited with code=0 in 0.167 seconds
```

JAVASCRIPT DATETIME

1. Write a JavaScript function to get the number of days in a month.

- *Test Data*:
  console.log(getDaysInMonth(1, 2012));
  console.log(getDaysInMonth(2, 2012));
  console.log(getDaysInMonth(9, 2012));
  console.log(getDaysInMonth(12, 2012));

Output :
31
29
30
31

```javascript
// Define a JavaScript function called getDaysInMonth with parameters month and year
var getDaysInMonth = function(month, year) {
    // Get the number of days in the specified month and year
    return new Date(year, month, 0).getDate();
    // Here January is 0 based
    // return new Date(year, month+1, 0).getDate();
  };

  // Output the number of days in January 2012
  console.log(getDaysInMonth(1, 2012));
  // Output the number of days in February 2012
  console.log(getDaysInMonth(2, 2012));
  // Output the number of days in September 2012
  console.log(getDaysInMonth(9, 2012));
  // Output the number of days in December 2012
  console.log(getDaysInMonth(12, 2012));
```

JAVASCRIPT DATETIME: MONTHS

2. Write a JavaScript function to get the month name from a particular date.

- *Test Data*:
  console.log(month_name(new Date("10/11/2009")));
  console.log(month_name(new Date("11/13/2014")));
  Output :
  "October"
  "November"

```javascript
// Define a JavaScript function called month_name with parameter dt
var month_name = function(dt){
    // Define an array containing names of months
    mlist = [ "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December" ];
    // Return the name of the month corresponding to the month index of the provided date
    return mlist[dt.getMonth()];
  };

  // Output the name of the month for the provided date "10/11/2009"
  console.log(month_name(new Date("10/11/2009")));
  // Output the name of the month for the provided date "11/13/2014"
  console.log(month_name(new Date("11/13/2014")));
```

JAVASCRIPT ARRAY

Write a JavaScript function to check whether an 'input' is an array or not.

- *Test Data*:
  console.log(is_array('w3resource'));
  console.log(is_array([1, 2, 4, 0]));
  false
  true

```
// Function to check if the input is an array
var is_array = function(input) {
    // Using toString method to get the class of the input and checking if it is "[object Array]"
    if (toString.call(input) === "[object Array]")
      // Return true if the input is an array
      return true;
    // Return false if the input is not an array
    return false;
  };

  // Testing the function with a string
  console.log(is_array('w3resource'));

  // Testing the function with an array
  console.log(is_array([1, 2, 4, 0]));
```

ROUND A NUMBER TO SPECIFID AMOUNT DIGITS

- Write a JavaScript program to round a number to a specified amount of digits.
- Use Math.round() and template literals to round the number to the specified number of digits.
- Omit the second argument, decimals, to round to an integer.

```
const round = (n, decimals = 0) =>
    // Convert 'n' to a string in scientific notation with the desired number of decimals
    Number(`${Math.round(`${n}e${decimals}`)}e-${decimals}`);

  // Test the 'round' function with different inputs
  console.log(round(1.005, 2)); // Output: 1.01
  console.log(round(1.05, 2));  // Output: 1.05
  console.log(round(1.0005, 2)); // Output: 1
```