JavaScript

[WRITING A PROGRAM: BASIC SOURCE CODE 3](#_Toc58855074)

[IDENTIFIERS (VARIABLE NAMES) 3](#_Toc58855075)

[DATA TYPES 3](#_Toc58855076)

[BASIC DATA TYPES 3](#_Toc58855077)

[FIND THE DATA TYPE OF A VARIABLE 4](#_Toc58855078)

[CONVERT DATA TYPES 4](#_Toc58855079)

[DECLARE DATA. ASSIGN VALUES 5](#_Toc58855080)

[DECLARE A VARIABLE 5](#_Toc58855081)

[LIFE OF A VARIABLE 6](#_Toc58855082)

[DECLARE AN OBJECT 6](#_Toc58855083)

[DECLARE A FUNCTION OBJECT 7](#_Toc58855084)

[CONDITIONS 7](#_Toc58855085)

[OPERATORS 7](#_Toc58855086)

[COMPARE DATA 8](#_Toc58855087)

[CONDITIONAL STATEMENTS 8](#_Toc58855088)

[LOOPS 8](#_Toc58855089)

[FOR LOOP 8](#_Toc58855090)

[WHILE LOOP 9](#_Toc58855091)

[BREAK A LOOP 10](#_Toc58855092)

[CALL A FUNCTION. FUNCTION PARAMETERS 10](#_Toc58855093)

[OUTPUT 11](#_Toc58855094)

[NUMBERS IN JAVASCRIPT 11](#_Toc58855095)

[BASIC MATH 11](#_Toc58855096)

[RANDOM NUMBERS 13](#_Toc58855097)

[TEXT PROCESSING 13](#_Toc58855098)

[COMBINE STRINGS 13](#_Toc58855099)

[ACCESS THE CHARACTERS OF A STRING. STRING LENGTH 14](#_Toc58855100)

[CONVERT STRINGS 14](#_Toc58855101)

[COMPARE STRINGS 14](#_Toc58855102)

[SEARCH FOR A SPECIFIED CHARACTER/STRING PART 15](#_Toc58855103)

[EXTRACT STRING PARTS 15](#_Toc58855104)

[CONVERT TO LOWER/UPPERCASE 16](#_Toc58855105)

[CHANGE STRINGS (RESULT MUST BE SAVED IN ANOTHER VARIABLE) 16](#_Toc58855106)

[REGULAR EXPRESSIONS 17](#_Toc58855107)

[MATCHING RULES 17](#_Toc58855108)

[PREDEFINED CLASSES 17](#_Toc58855109)

[QUANTIFIERS 18](#_Toc58855110)

[GROUP CONSTRUCTS 18](#_Toc58855111)

[FLAGS/MODIFIERS 19](#_Toc58855112)

[ANCHORS 19](#_Toc58855113)

[REGEXP IN JAVASCRIPT 19](#_Toc58855114)

[EXAMPLES 20](#_Toc58855115)

[ARRAY OBJECTS 21](#_Toc58855116)

[CREATE AN ARRAY AND ACCESS ITS ELEMENTS 21](#_Toc58855117)

[EXTRACT ARRAY PARTS. FILTER 22](#_Toc58855118)

[CHANGE/ADD/REMOVE/SWAP ARRAY ELEMENTS 22](#_Toc58855119)

[CONVERT AN ARRAY TO STRING. CONCATENATE ARRAYS AND STRINGS 23](#_Toc58855120)

[ARRANGE THE ELEMENTS OF AN ARRAY IN CERTAIN ORDER. SORT AND FILTER 23](#_Toc58855121)

[CALL A FUNCTION FOR EACH ARRAY ELEMENT 24](#_Toc58855122)

[OBJECTS 24](#_Toc58855123)

[DECLARE AN OBJECT AND ACCESS ITS VALUES 24](#_Toc58855124)

[CONVERT AN OBJECT TO AN ARRAY/A JSON STRING 25](#_Toc58855125)

[CHANGE AN OBJECT VALUE. DELETE AN OBJECT PROPERTY. SORT AN OBJECT 26](#_Toc58855126)

[DECLARE A CLASS AND CREATE INSTANCES 26](#_Toc58855127)

[MAP OBJECTS 26](#_Toc58855128)

[SET OBJECTS 27](#_Toc58855129)

[DATE OBJECTS 28](#_Toc58855130)

[CREATE A DATE OBJECT 28](#_Toc58855131)

[CONVERT A DATE TO A STRING/NUMBER. COMPARE DATES 28](#_Toc58855132)

[ACCESS ONLY PART OF THE DATE 29](#_Toc58855133)

[CHANGE PART OF THE DATE 29](#_Toc58855134)

# WRITING A PROGRAM: BASIC SOURCE CODE

|  |  |  |
| --- | --- | --- |
| File Name | only lowercase (when possible) | [Python](Python.docx#BasicSourceCode) C++ C# Java |
| Run in Command Prompt | Command Prompt > type 'nod', enter, type operation |
| Optimal Line Length in Source Code | no longer than 80 characters (break it after an operator or a comma) |
| Break a String into Multiple Lines | using `` instead of '' or "" |
| Comments (Not Executable Code) | // a single-line comment  /\* a multiline comment \*/ |

# IDENTIFIERS (VARIABLE/FUNCTION/PROPERTY NAMES)

|  |  |  |
| --- | --- | --- |
| Lower Camel Case | let firstName = 'John'; | [Python](Python.docx#Identifiers) C++ C# Java |
| Starting with a Number | no |
| Case-Sensivite | yes |
| Hyphens in Identifiers | no |

# DATA TYPES

## BASIC DATA TYPES

|  |  |  |
| --- | --- | --- |
| Data Types That Can Contain Values | *string*: a sequence of characters *number*: whole numbers or decimals *boolean*: true or false *object*: contains several properties, each of them has a value *function*: does something | [Python](Python.docx#BasicDataTypes) C++ C# Java |
| Data Types That Cannot Contain Values | *null* *undefined* |
| Primitive Data Types (Stored in the Heap and Accessed by Value, Changing a Copy Does NOT Change the Initial Variable) | *string*, *number*, *boolean*, *null*, *undefined, bigInt, symbol*  let name = 'John';  let firstName = name; firstName = 'Mark';  firstName // returns 'Mark'  name // returns 'John' |
| Reference Data Types (Stored in the Stack and Accessed by Reference, Changing a Copy DOES Change the Initial Variable) | *object*, *function*  let numbers = [1, 2, 3];  let numbers2 = numbers; numbers2 += 1;  numbers2 // returns '1,2,31'  numbers // returns [1, 2, 3] |

## FIND THE DATA TYPE OF A VARIABLE

|  |  |  |
| --- | --- | --- |
| Find the Data Type of a Variable | typeof 'John' // returns string  typeof 35 // returns number  isNaN(35) // returns false Number.isInteger(35) // returns true typeof NaN // returns number typeof Infinity // returns number typeof true // returns boolean  typeof { name: 'John',  age: 35 } // returns object typeof ['John', '35'] // returns object typeof new Date() // returns object  typeof undefined // returns undefined | [Python](Python.docx#FindTheDataTypeOfAVariable) C++ C# Java |
| Find the Constructor Function for a Variable | 'John'.constructor // returns f String() (3.14).constructor // returns f Number() false.constructor // returns f Boolean() [1, 2, 3].constructor // returns f Array() { name: 'John', age: 35 }.constructor // returns f Object() new Date().constructor // returns f Date() function () { }.constructor // returns f Function() |
| Find Out if an Object is an Array | Array.isArray([1, 2]) // returns true [1, 2].constructor == Array // returns true |
| Find Out if an Object is a Date | function isDate(myDate) {  return myDate.constructor.toString().indexOf('Date') > -1;  } // returns true or false  function isDate(myDate) {  return myDate.constructor == Date;  } // returns true or false |

## CONVERT DATA TYPES. TYPE COERCION

|  |  |  |
| --- | --- | --- |
| Convert Other Types to Strings | String(3) // returns '3' (3).toString() // returns '3' | [Python](Python.docx#ConvertDataTypes) C++ C# Java |
| Convert Other Types to Numbers | Number('3') // returns 3 parseInt('3') // returns 3 parseFloat('3.14') // returns 3.14 Number('John') // returns NaN Number('') // returns 0 Number([20]) // returns 20 Number([10, 20]) // returns NaN Number([]) // returns 0 Number({}) // returns NaN Number(false) // returns 0 Number(true) // returns 1 Number(new Date('Mar 25 2015')) // returns 1427234400000 (milliseconds counted from Jan 01 1970 00:00:00 UTC) |
| Convert Other Types to Booleans (Truthy and Falsy Values) | Boolean('3') // returns true Boolean(0) // returns false Boolean(3) // returns true  Boolean(3 / 2) // returns true Boolean(3 / 'd') // returns false Boolean('0') // returns true Boolean('') // returns false Boolean([]) // returns true Boolean({}) // returns true Boolean(null) // returns falseBoolean(undefined) // returns false |
| Type Coercion | '' + 3 // returns '3' `${3}` // returns '3' + '3' // returns 3 2 \* '10' // returns 20 4 < '14' // returns true if (3) { ... } // 3 coerced to true |  |

# DECLARE DATA. ASSIGN VALUES

## DECLARE A VARIABLE

|  |  |  |
| --- | --- | --- |
| Assignment Operators | = // assigns a value to a variable += // adds to previous value -= // subtracts from previous value \*= // multiplies previous value \*\*= // raises the value of a variable to the value of the right operand /= // divides previous value %= // assigns a remainder | [Python](Python.docx#DeclareAVariable) C++ C# Java |
| Declare a Variable without a Value | let undefinedVariable;  let undefinedVariable = undefined; // both value and type are undefined |
| Declare a String Variable | let text = 'A'; var text = 'A'; // function scope; not recommended |
| Declare a Number | let num = 6; // accurate up to 15 digits  let num = 6.25; |
| Declare a Boolean | let isFound = true; |
| Declare Multiple Variables | let person = 'John Doe', carName = 'Volvo', price = 200;  let [person, carName, price] = ['John Doe', 'Volvo', 200];  let x, y; |
| Assign the Same Value to Multiple Variables | let x = y = z = 5; |  |
| Declare a Constant | const pi = 3.14; |  |

## LIFE OF A VARIABLE

|  |  |  |
| --- | --- | --- |
| Local Variables (Inside a Function) | // code here CANNOT use firstName function functionName() {  let firstName = 'John';  // code here can use firstName  } // code here CANNOT use firstName | [Python](Python.docx#LifeOfAVariable) C++ C# Java |
| Global Variables | let firstName = 'John'; // code here can use firstName function functionName() {  // code here can use firstName  } // code here can use firstName |

## DECLARE AN OBJECT

|  |  |  |
| --- | --- | --- |
| Declare an Object {Property/Key: Value} | let person = {  name: 'John',  age: 35  }; | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare an Array Object [List of Values] | let people = ['John', 'Mary', 'George']; |
| Declare a Map Object [[Property/Key, Value]] | let person = new Map([['name', 'John'], ['age', 35]]); |
| Declare a Set Object  [List of Unique Values] | let people = new Set(['John', 'Mary', 'George']); |
| Declare a Class (a Template for Objects) | class Person {  constructor(personName, personAge) {  this.name = personName;  this.age = personAge;  } } |  |

## DECLARE A FUNCTION OBJECT

|  |  |  |
| --- | --- | --- |
| Declare a Function | function functionName() {  // content of the function  } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare an Arrow Function (Shorter Syntax, Does NOT Have 'This' Property) | hello = function() {  let name = 'John';  return 'Hello ' + name; } hello() //returns ‘Hello John’  hello = () => {  let name = 'John';  return 'Hello ' + name; } |
| Declare an Arrow Function with Only One Statement | hello = () => 'Hello!';  hello = (name) => 'Hello ' + name;  hello = name => 'Hello ' + name; |

# CONDITIONS

## TRUTHY/FALSY VALUES

|  |  |  |
| --- | --- | --- |
| Truthy Values | Values that coerce to true when evaluated in a boolean context. | [Python](Python.docx#TruthyFalsyValues) C++ C# Java |
| Falsy Values | false, null, undefined, NaN, 0, 0n, '' |

## OPERATORS

|  |  |  |
| --- | --- | --- |
| Comparison Operators | == // 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 | [Python](Python.docx#Operators) C++ C# Java |
| Logical Operators | && // and; returns the leftmost falsy value or the last truthy value if they are all true  || // or; returns the leftmost truthy value or the last falsy if they are all false  ! // not; returns false if its operand can be converted to true, otherwise true |
| Ternary Operator | *condition* ? *value1* : *value2*  age < 18 ? 'young' : 'old' // returns 'young' if the value of age is < 18, 'old' if age > 18 |

## COMPARE DATA

|  |  |  |
| --- | --- | --- |
| Comparing Numbers | 1 < 2 // returns true | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Comparing a Number to a String | 2 < '12' // returns true, numeric strings are converted to numbers  2 < 'John' // returns false 0 == '' // returns true 0 === '' // returns false |
| Comparing Strings | 'John' <= 'John' // returns true 'a' < 'b' // returns true (alphabetically ordered) '2' < '12' // returns false (alphabetically ordered) |
| Comparing Objects: Always Returns False | [1, 2, 3] == [1, 2, 3] // returns false |

## CONDITIONAL STATEMENTS

|  |  |  |
| --- | --- | --- |
| If – Else (If) | if (x > y) {  // code to be executed if x > y } else if (x < y) {  // code to be executed if x !> y and x < y } else {  // code to be executed if x !> y and x !< y } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Switch (Case) | switch (num) {  case 1: day = 'Monday'; break; // num == 1  case 2: day = 'Tuesday'; break;  case 3: day = 'Wednesday'; break;  case 4: day = 'Thursday'; break;  case 5: day = 'Friday'; break;  case 6:  case 7: day = 'Weekend'; break; // num == 6 or num == 7  default: day = 'unknown'; break; // optional } |

# LOOPS

## FOR LOOP

|  |  |  |
| --- | --- | --- |
| For Loop | for (let i = 0; i < 10; i++) {  // code block to be executed 10 times  } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| For Loop without the First Statement (Setting a Variable) | let i = 0; for (; i < 10; i++) {  // code block to be executed 10 times  } |
| For Loop without the Second Statement (Setting a Condition) | for (let i = 0; ; i++) {  // code block to be executed  if (i >= 10) {  break;  }  } |
| For Loop without the Third Statement (Changing the Value of the Variable) | for (let i = 0; i < 10;) {  // code block to be executed  i++;  } |
| For Loop with a Condition Statement Only | let cars = ['BMW', 'Audi']; let i = 0; for (; cars[i];) {  // code block to be executed  i++;  } // cars[i] will return false when i == 2 |  |
| For Loop without Any Statement (Set the Code Block in Order to Avoid an Infinite Loop) | let i = 0; for (; ;) {  // code block to be executed  if (i >= 10) {  break;  }  i++;  } |  |

## WHILE LOOP

|  |  |  |
| --- | --- | --- |
| While Loop | while (*condition*) {  // code block to be executed while the condition returns 'true' } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Do/While Loop (Executed at Least Once) | do {  // code block to be executed at least once } while (*condition*) |

## BREAK A LOOP

|  |  |  |
| --- | --- | --- |
| The Break Statement (Breaks the Loop) | for (let i = 0; i < 10; i++) {  if (i == 3) {  break;  }  // code block to be executed 2 times  } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| The Continue Statement (Breaks the Current Iteration Only) | for (let i = 0; i < 10; i++) {  if (i == 3) {  continue;  }  // code block to be executed 9 times, except for when I == 3  } |

# FUNCTIONS

## CALL A FUNCTION. FUNCTION PARAMETERS

|  |  |  |
| --- | --- | --- |
| String Parameters | function greetingByName(firstName, lastName) {  return 'Your name is ' + firstName + lastName; } greetingByName('John', 'Doe'); // returns 'Your name is John Doe' | [Python](Python.docx#Input) C++ C# Java |
| Number Parameters | function ableToVote(age) {  return 'You will be able to vote in ' + (18 – age) + ' years'; } ableToVote(15); // returns 'You will be able to vote in 3 years' |
| Multiple Inputs (the Spread Operator) | function introducing(...input) {  let name = input[0]; // 'John'  let age = input[1]; // 35  return 'My name is ' + name + 'and I am ' + age + ' years old.' } introducing('John', 35); // returns 'My name is John and I am 35 years old.' |  |
| Array as a Function Parameter | function introducing(input) {  let name = input[0]; // 'John'  let age = input[1]; // 35  return 'My name is ' + name + 'and I am ' + age + ' years old.' } introducing(['John', 35]); // returns 'My name is John and I am 35 years old.' |  |
| Default Parameters | function printStars(count = 5) {  console.log('\*'.repeat(count)); }  printStars(); // prints '\*\*\*\*\*' printStars(2); // prints '\*\*' printStars(3, 5, 8); // prints '\*\*\*' |  |

## FUNCTION CONTEXT. THIS OBJECT

|  |  |  |
| --- | --- | --- |
| Global Invoke | function myFunction() {  return this; }  myFunction(); // returns 'global' in node.js, 'window' in the browser | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Object Method | let myObj = {  name: 'Peter',  func() {  return this;  } };  myObj.func() // returns { name: 'Peter', func: *f* } |
| DOM Event | button.addEventListener('click', onClick);  function onClick(ev) {  console.log(this); } // prints |  |
| Inner Method Context | let myObj = {  name: 'Peter',  func() {  console.log(this); // { name: 'Peter', func: *f* }  function innerFunc() {  console.log(this); // global/window  }  innerFunc();  } };  myObj.func(); // prints { name: 'Peter', func: *f* } and global/window |  |
| Arrow Function Context | let myObj = {  name: 'Peter',  func() {  console.log(this); // { name: 'Peter', func: *f* }  let innerFunc = () => {  console.log(this); // { name: 'Peter', func: *f* }  }  innerFunc();  } };  myObj.func(); // prints { name: 'Peter', func: *f* } twice |  |
| Explicit Binding (Changing the Context) | let myObj = {  name: 'Peter' };  function func(a, b) {  console.log(this);  console.log(a, b); }  func.call(myObj, 2, 3); // prints { name: 'Peter' } and then 2 3 func.apply(myObj, [2, 3]); // prints { name: 'Peter' } and then 2 3 let boundFunc = func.bind(myObj); boundFunc(2, 3); // prints { name: 'Peter' } and then 2 3 |  |
|  | let myObj = {  name: 'Peter' };  function func() {  console.log(`I am ${this}`); }  func(); // prints 'I am undefined' func.call(myObj); // prints 'I am Peter' func.apply(myObj); // prints 'I am Peter' let boundFunc = func.bind(myObj); boundFunc(); // prints 'I am Peter' document.querySelector('button').addEventListener('click', myObj.func); // prints 'I am' document.querySelector('button').addEventListener('click', myObj.func.bind(myObj); // prints 'I am Peter' |  |
|  |  |  |

## FIRST-CLASS AND HIGHER-ORDER FUNCTIONS

|  |  |  |
| --- | --- | --- |
| First-Class Functions (Treated like Any Other Variable) | function sayHello() {  return 'Hello, '; }  const greeting = () => 'Hello!'; console.log(greeting()); // prints 'Hello!' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Higher-Order Functions (Take Other Functions as Arguments or Return a Function) | function sayHello() {  return function () {  console.log('Hello!');  } }  sayHello(); // prints 'Hello!' |
| Built-In Higher-Order Functions | array.map(x => x \* 2) array.filter(x => x > 5) array.reduce((acc, curr) => acc + curr, 0) |  |
| Predicates (Return a Boolean) | let array = [2, 4, 12, 37]; let found = array.find(isFound);  function isFound(element) {  return element > 10; }  console.log(found); // prints 12 |  |
| Pure Functions (Same Result Given Same Parameters) | let num = 1; function impure(a) {  return num += a; }  function pure(a, b) {  return a + b; } |  |
| Referential Transparency (Can Be Replaced with Its Corresponding Value) | function sum(a, b) { return a + b } function mult(a, b) { return a \* b } let x = sum(2, mult(3, 4)); // mult(3, 4) can be replaced with 12 |  |
| Closure | The scope of an inner function includes the scope of the outer function. |  |
| IIFE (Immediately Invoked Function Expression) | (function () { let name = 'Peter' })(); // name cannot be used let result = (function () {  return name = 'Peter'; })() // result = 'Peter' |  |
| Partial Application | function pow(num, pow) {  return num \*\* pow; }  function sqr(num) {  return pow(num, 2); }  sqr(3) // returns 9 |  |
| Function Decoration | function pow(pow, num) {  return num \*\* pow; }  const sqr = pow.bind(null, 2); sqr(3) // returns 9 |  |
| Currying (Function Decomposition) | function sum3(a) {  return (b) => {  return (c) => {  return a + b + c;  }  }  }  sum3(5)(6)(8) // returns 19 |  |

# OUTPUT

|  |  |  |
| --- | --- | --- |
| Print Some Data to the Console | console.log('Hi there, John!'); // prints 'Hi there, John!'  console.log(35); // prints 35 | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Print to the Console a Value Stored in a Variable | let name = 'John';  console.log(name); // prints 'John' |
| Display a Message in an Alert Window | alert('Hi there!');  window.alert('Hi there!'); |  |
| Print Page | window.print(); |  |

# NUMBERS IN JAVASCRIPT

## BASIC MATH

|  |  |  |
| --- | --- | --- |
| Operators | + // addition - // subtraction \* // multiplication / // division % // modulus: returns the division remainder (10 % 2 returns 0) ++ // increment: returns previous value + 1 -- // decrement: returns previous value – 1 \*\* // exponentiation: raises the first operand to the power of the second (2 \*\* 3 returns 8) () // parentheses: changes precedence | [Python](Python.docx#BasicMath) C++ C# Java |
| Arithmetic Precision (Accurate Up to 15 Digits) | 9999999999999999; // returns 10000000000000000 0.2 + 0.1 // returns 0.30000000004 (0.2 \* 10 + 0.1 \* 10) / 10 // returns 0.3 2 / 0 // returns Infinity -2 / 0 // returns -Infinity |
| Arithmetic Operations with Strings and Numbers | '100' / '10' // returns 10 '10' \* 10 // returns 100 100 – '10' // returns 90 '100' + 10 // returns 10010 (concatenation) 100 / 'Apple' // returns NaN (Not a Number) |
| Add a Number and a String (Concatenation) | 2 + '3' // returns 23 2 + 3 + 'A' // returns 5A 'A' + 2 + 3 // returns A23 |  |
| Absolute Value | Math.abs(-50) // returns 50 |  |
| Round | Math.round(47.54) // returns 48  47.44.toFixed(0) // returns '47' |  |
| Round Down | parseInt(47.98) // returns 47  Math.trunc(47.98) // returns 47  Math.floor(47.98) // returns 47 |  |
| Round Up | Math.ceil(47.01) // returns 48 |  |
| Round to an Exact Number of Digits after the Decimal Point | 47.445.toFixed(2) // returns '47.45' |  |
| Remove Trailing Zeroes | parseFloat(47.9000) // returns 47.9 Number(47.9000) // returns 47.9 |  |
| Find the Largest Number | Math.max(2, 5, -32) // returns 5 |  |
| Find the Smallest Number | Math.min(2, 5, -32) // returns -32 |  |
| Exponentiation | Math.pow(5, 2) // returns 25 |  |
| Square Root | Math.sqrt(36) // returns 6 Math.sqrt(-1) // returns NaN |  |
| Hexadecimal Numbers | 0xFF // returns 255 |  |
| Convert Base 10 Numbers to Another Base | (32).toString(10) // returns '32' (32).toString(16) // returns '20' (32).toString(2) // returns '100000' |  |
| Convert Binary Numbers to Decimals | let binary = parseInt('00001001', 2) // 00001001 binary.toString(10) // returns '9' |  |
| Find Number of Digits | (32).toString().length // returns 2 String(32).length // returns 2 |  |
| Extra Large/Small Numbers | let x = 123e5; // 12300000 let y = 123e-5; // 0.00123 |  |
| Pi (π) | Math.PI // returns 3.14159... |  |
| The Largest Possible Number | Number.POSITIVE\_INFINITY // returns Infinity Number.MAX\_SAFE\_INTEGER // returns 9007199254740991 Number.MAX\_VALUE // returns 1.7976931348623157e+308 |  |
| The Smallest Possible Number | Number.NEGATIVE\_INFINITY // returns -Infinity Number.MIN\_SAFE\_INTEGER // returns -9007199254740991 Number.MIN\_VALUE // returns 5e-324 |  |

## RANDOM NUMBERS

|  |  |  |
| --- | --- | --- |
| Random Numbers Lower than 1 | Math.random() // returns a random number between 0 (inclusive) and 1 (exclusive) | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Random Numbers from 0 to 9 | Math.floor(Math.random() \* 10) // returns a random number between 0 and 9 |
| Random Numbers from 0 to 10 | Math.floor(Math.random() \* 11) // returns a random number between 0 and 10 |  |
| Random Numbers from 0 to 99 | Math.floor(Math.random() \* 100) // returns a random number between 0 and 99 |  |
| Random Numbers from 0 to 100 | Math.floor(Math.random() \* 101) // returns a random number between 0 and 100 |  |
| Random Numbers from 1 to 10 | Math.floor(Math.random() \* 10) + 1 // returns a random number between 1 and 10 |  |
| Random Numbers from 1 to 100 | Math.floor(Math.random() \* 100) + 1 // returns a random number between 1 and 100 |  |
| A Function that Returns a Random Number between Min (Included) and Max (Excluded) | function getRndInteger(min, max) {  return Math.floor(Math.random() \* (max – min)) + min;  } |  |
| A Function that Returns a Random Number between Min (Included) and Max (Included) | function getRndInteger(min, max) {  return Math.floor(Math.random() \* (max – min) + 1) + min;  } |  |

# TEXT PROCESSING

## COMBINE STRINGS

|  |  |  |
| --- | --- | --- |
| Concatenate Strings | 'Hello'.concat(' John!') // returns 'Hello John!' 'Hello'.concat(' ', 'John!') // returns 'Hello John!' 'Hello' + ' ' + ' John!' // returns 'Hello John!' | [Python](Python.docx#CombineStrings) C++ C# Java |
| Concatenate Strings Using Variables (Interpolation) | let name = 'John', age = 35;  let sentence = `${name} is ${age} years old.`; // 'John is 35 years old.' |

## ACCESS THE CHARACTERS OF A STRING. STRING LENGTH

|  |  |  |
| --- | --- | --- |
| Access a Character at a Specified Position | 'abcd'[0] // returns 'a' 'abcd'.charAt() // returns 'a' 'abcd'.charAt(3) // returns 'd' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Find the Length of a String | 'abcd'.length // returns 4 |
| Loop through the Characters of a String | for (let i = 0; i < 'word'.length; i++) {  console.log('word'[i]);  } // prints 'w', 'o', 'r', 'd' on separate lines  for (let char of 'word') {  console.log(char);  } // prints 'w', 'o', 'r', 'd' on separate lines |

## CONVERT STRINGS

|  |  |  |
| --- | --- | --- |
| Convert a String to an Array | 'a b c d'.split(' ') //returns ['a', 'b', 'c', 'd'] 'a,b,c,d'.split(',') // returns ['a', 'b', 'c', 'd'] 'abcd'.split('') // returns ['a', 'b', 'c', 'd'] | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Convert a Character into the Corresponding ASCII Code | 'abcd'.charCodeAt() // returns 97 'abcd'.charCodeAt(0) // returns 97 'abcd'.charCodeAt(2) // returns 99 |
| Convert a Character Code into the Corresponding Character | String.fromCharCode(97) // returns 'a' |  |
| Convert a Special Character into a String Character | '\\\_/' // returns '\\_/' |  |

## COMPARE STRINGS

|  |  |  |
| --- | --- | --- |
| Compare the ASCII Code of Strings | 'a' > 'b' // returns false  'a' > 'B' // returns true | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Compare Strings Regardless of Case | 'a'.localeCompare('B') // returns -1  'B'.localeCompare('a') // returns 1  'a'.localeCompare('A') // returns -1  'a'.localeCompare('a') // returns 0 |

## SEARCH FOR A SPECIFIED CHARACTER/STRING PART

|  |  |  |
| --- | --- | --- |
| Find the Position of a Specified Text/Character in a String | 'abcd'.indexOf('c') // returns 2; CANNOT take regular expressions  'abcd'.indexOf('g') // returns -1  'Hello! Hello!'.indexOf('Hello') // returns 0  'Hello! Hello!'.indexOf('Hello', 3) // starts searching from position 3, returns 7  'Hello! Hello!'.search('Hello') // returns 0; can take regular expressions, CANNOT take a start index | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Determine whether a String Contains the Characters of a Specified String | 'abcd'.includes('a') // returns true  'abcd'.includes('g') // returns false |
| Find the Position of the Last Occurrence of a Specified Text/Character in a String | 'Hello! Hello!'.lastIndexOf('Hello') // searches backwards, returns 7  'Hello! Hello!'.lastIndexOf('John') // returns -1  'Hello! Hello!'.lastIndexOf('Hello', 3) // returns 0  'Hello! Hello!'.lastIndexOf('Hello', 7) // returns 7 |  |
| Check if a String Begins with a Specified Character/String | 'Hello'.startsWith('Hell') // returns true  'Hello'.startsWith('hell') // returns false |  |
| Check if a String Ends with a Specified Character/String | 'Hello'.endsWith('lo') // returns true 'Hello'.slice(-'lo'.length) == 'lo' // returns true |  |

## EXTRACT STRING PARTS

|  |  |  |
| --- | --- | --- |
| Extract String Parts (Start Position) | 'Hello John!'.substring(6) // returns 'John'  'Hello John!'.slice(6) // returns 'John' // 'Hello John!'.**substr**(6) returns 'John' (not recommended) | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Extract String Parts (Start Position, End Position + 1) | 'Hello John!'.substring(0, 5) // returns 'Hello'  'Hello John!'.slice(0, 5) // returns 'Hello' |
| Extract String Parts (Start Position, Length of the Extracted String) | 'Hello John!'.substring(6, 6 + 4) // returns 'John'  'Hello John!'.slice(6, 6 + 4) // returns 'John'  // 'Hello John!'.**substr**(6, 4) returns 'John' (not recommended) |  |
| Extract String Parts Counting Backwards (Start Position) | 'Hello John!'.slice(-5) // returns 'John!'  // 'Hello John!'.**substring**(-5) does not work as expected, returns 'Hello John!' // 'Hello John!'.**substr**(-5) returns 'John' (not recommended) |  |
| Extract String Parts Counting Backwards (Start Position, End Position + 1) | 'Hello John!'.slice(-5, -1) // returns 'John'  // 'Hello John!'.**substring**(-5, -1) returns an empty string |  |
| Extract String Parts (End Position + 1, Start Position) | 'Hello John!'.substring(5, 0) // returns 'Hello'  // 'Hello John!'.**slice**(5, 0) returns an empty string |  |

## CONVERT TO LOWER/UPPERCASE

|  |  |  |
| --- | --- | --- |
| Convert to Upper Case | 'John'.toLocaleUpperCase() // returns 'JOHN' (according to the language settings)  'John'.toUpperCase() // returns 'JOHN' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Convert to Lower Case | 'John'.toLocaleLowerCase() // returns 'john' (according to the language settings)  'John'.toLowerCase() // returns 'john' |

## CHANGE STRINGS (RESULT MUST BE SAVED IN ANOTHER VARIABLE)

|  |  |  |
| --- | --- | --- |
| Immutable Strings | let name = 'Mary';  name[0] = 'G';  name // returns 'Mary' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Replace a Character/String Part (Only the First Occrurence) | 'blue house, blue car'.replace('blue', 'red'); // returns 'red house, blue car' |
| Repeat a Character/String | '\*'.repeat(5) // returns '\*\*\*\*\*' |  |
| Add Characters at the Beginning of a String to Reach a Specified Length | '23'.padStart(4, '0') // returns '0023'  ('0'.repeat(4) + '23').slice(-4) // returns '0023' |  |
| Add Characters at the End of a String to Reach a Specified Length | '23'.padEnd(4, '0') // returns '2300'  ('23' + '0'.repeat(4)).slice(0, 4) // returns '2300' |  |
| Remove Whitespace from Both Sides | ' Hello '.trim() // returns 'Hello' |  |
| Remove Whitespace from the Beginning | ' Hello '.trimStart() // returns 'Hello ' |  |
| Remove Whitespace from the End | ' Hello '.trimEnd() // returns ' Hello' |  |

## REGULAR EXPRESSIONS

### MATCHING RULES

|  |  |  |
| --- | --- | --- |
| Match Any Character | **.** | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Match a Single Character of a Given Set | [abc] // matches either 'a', 'b' or 'c'  (a|b|c) // matches either 'a', 'b' or 'c' |
| Match a Single Character Except a Given Set | [^abc] // matches anything but 'a', 'b' or 'c' |  |
| Match a Character in a Range | [0-9] // matches any digit |  |
| Match a Character Not in a Range | [^A-Za-z] // matches anything but letters |  |

### PREDEFINED CLASSES

|  |  |  |
| --- | --- | --- |
| Match Any Alphanumeric Character (Letters, Digits, Underscore) | \w // equals [A-Za-z\_] | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Match Any Non-Alphanumeric Character | \W // equals [^A-Za-z\_] |
| Match Any White-Space Character | \s // equals [ \t\n\r\0] |  |
| Match Any Non-White-Space Character | \S // equals [^ \t\n\r\0] |  |
| Match Any Digit | \d // equals [0-9] |  |
| Match Any Non-Digit Character | \D // equals [^0-9] |  |

### QUANTIFIERS

|  |  |  |
| --- | --- | --- |
| Zero or One of | /a?/ // matches zero or one time 'a' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Zero or More of | /a\*/ // matches zero or more times 'a' |
| One or More of | /a+/ // matches one or more times 'a' |  |
| An Exact Number of | /a{3}/ // matches three times 'a' |  |
| An Exact Number or More of | /a{3,}/ // matches three or more times 'a' |  |
| An Exact Number in a Range of | /a{3,6}/ // matches three to six times 'a' |  |
| Greedy Quantifier | /a\*+/ // matches as many characters as possible |  |
| Lazy Quantifier | /a\*?/ // matches as few characters as possible |  |

### GROUP CONSTRUCTS

|  |  |  |
| --- | --- | --- |
| Capture Everything Enclosed and Create a Group | /(is)/ // matches ‘is’and creates a capture group 'is' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Match Everything Enclosed without Creating a Group | /(?:is)/ // matches 'is' |
| Create a Named Capture Group | /(?<tag>div)/ // matches 'div' and creates a group 'div' named 'tag' |  |
| Match the Value of a Previously Defined Capture Group | \*number* // matches the value of a numbered capture group  \k<*name*> // matches the value of a named capture group |  |
| Match One Character/Sequence or Another | /(a|b)/ // matches either 'a' or 'b' |  |
| Positive Lookahead | /a(?=b)/ // matches any 'a' followed by 'b' (without capturing the 'b') |  |
| Negative Lookahead | /a(?!b)/ // matches any 'a' NOT followed by 'b' |  |
| Positive Lookbehind | /a(?<=b)/ // matches any 'a' preceeded by 'b' (without capturing the 'b') |  |
| Negative Lookbehind | /a(?<!b)/ // matches any 'a' NOT preceeded by 'b' |  |

### FLAGS/MODIFIERS

|  |  |  |
| --- | --- | --- |
| Find All Matches in the Text | g // global | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Find Matches Regardless of Case | i // case insensitive |
| Use ^ and $ to Match at the Beginning/End of Each Line | m // multiline |  |

### ANCHORS

|  |  |  |
| --- | --- | --- |
| Match at the Beginning of a String: ^ | /^\w+/ // matches only the first word of a string | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Match at the End of a String: $ | /\w+$/ // matches only the last word of a string |
| Match at the Beginning/End of a Word: \b | /d\b/g // matches 'd' only at the end of a word, equals /d(?!\w)/ or /d(?=\W)/  /\bd/g // matches 'd' only at the beginning of a word, equals /(?<!\w)d/ or /(?<=\W)/ |  |

### REGEX IN JAVASCRIPT

|  |  |  |
| --- | --- | --- |
| Regular Expression Literal | /[A-Za-z]+/g // escape control characters (+, ^, $) using \ | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| The Constructor Function RegExp | new RegExp('[A-Za-z]+', 'g') // escape control characters (\, ') using \ |
| Check Whether There Is a Match | 'Today is 2015-05-11'.match(/\d{4}-\d{2}-\d{2}/g) // returns ['2015-05-11'] (true in an if statement)  let text = 'Today is 2015-05-11';  let pattern = new RegExp('\\d{4}-\\d{2}-\\d{2}', 'g'); text.match(pattern); // returns ['2015-05-11']  text.match(/@/); // returns null (false in an if statement) |  |
| Check Whether There Is a Match (and Advance the Last Index of the Pattern) | new RegExp('\\d{4}-\\d{2}-\\d{2}', 'g').test('Today is 2015-05-11') // returns true  let text = 'Today is 2015-05-11';  let pattern = new RegExp('\\d{4}-\\d{2}-\\d{2}', 'g'); pattern.test(text); // returns true  pattern.test(text); // next time returns false |  |
| Find the Index of a Match | 'Hello SoftUni'.search(/softuni/i) // returns 6 |  |
| Get an Array of All Matches | let text = 'Peter: 123 Mark: 456';  let pattern = /([A-Z][a-z]+): (\d+)/g; text.match(pattern); // returns ['Peter: 123', 'Mark: 456']  text.match(pattern).length; // returns 2 |  |
| Get an Array of a Match and All Groups in It | let text = 'Today is 15-Apr-2020, not 30-Nov-1988';  let pattern = /\d{2}-(?<month>[A-Za-z]{3})-(?<year>\d{4})/;  text.match(pattern); // returns ['15-Apr-2020', 'Apr', '2020', index: 9, input: 'Today is 15-Apr-2020, not 30-Nov-1988', groups: { month: 'Apr', year: '2020' }]  text.match(pattern).groups.month; // returns 'Apr'  pattern.exec(text); // returns ['2015-Apr-11', 'Apr', '2015', index: 9, input: 'Today is 2015-Apr-11, not 30-Nov-1988', groups: undefined]  pattern.exec(text).groups.month; // returns 'Apr' |  |
| Get an Array of All Matches and All Groups in Them | let text = 'Peter: 123 Mark: 456';  let pattern = /(?<name>[A-Z][a-z]+): \d+/g; pattern.exec(text); // first time returns ['Peter: 123', 'Peter', index: 0, input: 'Peter: 123 Mark: 456', groups: { name: 'Peter' }]  pattern.exec(text).groups.name; // first time returns 'Peter'  pattern.exec(text); // next time returns ['Mark: 456', 'Mark', index: 1, input: 'Peter: 123 Mark: 456', groups: { name: 'Mark' }]  pattern.exec(text).groups.name; // next time returns 'Mark'  text.matchAll(pattern); // returns a string iterator [...text.matchAll(pattern)]; // returns [['Peter: 123', 'Peter', '123'], ['Mark: 456', 'Mark', '456']] |  |

### EXAMPLES

|  |  |  |
| --- | --- | --- |
| Replace All Matches | 'Mr Blue has a blue house and a blue car'.replace(/blue/g, 'red'); // returns 'Mr Blue has a red house and a red car' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Replace All Matches (Case Insensitive) | 'Mr Blue has a blue house and a blue car'.replace(/blue/gi, 'red'); // returns 'Mr red has a red house and a red car' |
| Match All Words (Including \_) | /\w+/ |  |
| Match Dates in Format 12-Jul-1999, 3-Mar-2013 | /\d{1,2}-[A-Za-z]{3}-\d{4}/ |  |
| Match Lower Camel Case | /[a-z]+([A-Z][a-z]+)\*/ |  |
| Match Any HTML Tag | /<(.+)>.+<\/\1>/  /<(?<tag>.+)>.+<\k<tag>>/ |  |

# ARRAY OBJECTS

## JAVASCRIPT ARRAY CHARACTERISTICS

|  |  |  |
| --- | --- | --- |
| Zero-based | let numbers = [1, 2];  numbers[0] // returns 1 | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Dynamic Type of the Elements | let numbers = [1, 2];  numbers = ['one', 'two']; |
| Arrays Holding Elements of Mixed Type (NOT Recommended) | let cars = ['Audi', 2004, { model: 'BMW', year: 2010 }]; |
| Dynamic Length | let numbers = [1, 2]; numbers.length = 4; numbers // returns [1, 2, undefined, undefined] |
| NOT Guaranteed to Be Dense | let numbers = [1, 2];  numbers[4] = 3;  numbers // returns [1, 2, undefined, undefined, 3] |  |

## CREATE AN ARRAY AND ACCESS ITS ELEMENTS

|  |  |  |
| --- | --- | --- |
| Create an Empty Array Object (Declare an Array without Initializing It) | let cars = [];  let cars = new Array(); | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Create an Array Object | let cars = ['Audi', 'BMW'];  let cars = new Array('Audi', 'BMW'); |
| Destructuring Assignment | let [car1, car2] = cars; // creates two variables: car1 = 'Audi', car2 = 'BMW' |
| Destructuring with the Rest Operator | let [a, b, ...elements] = [10, 20, 30, 40, 50];  a // returns 10 b // returns 20 elements // returns [30, 40, 50] |  |
| Access the First Element of an Array | cars[0] // returns 'Audi' |  |
| Access the Last Element of an Array | cars[cars.length - 1] // returns 'BMW' |  |
| Access a Non-existing Index of an Array | cars[8] // returns undefined cars[-1] // returns undefined |  |
| Iterate through All Elements in an Array | for (let i = 0; i < cars.length; i++) {  console.log(cars[i]); // prints 'Audi' and 'BMW' on separate lines  }  for (let car of cars) {  console.log(car); // prints 'Audi' and 'BMW' on separate lines  } |  |
| Find the Number of Array Elements | cars.length // returns 2 |  |
| Check whether an Array Contains a Specified Element | cars.includes('Ford') // returns false |  |
| Find the Position of a Specified Array Element | [3, 5].indexOf(3) // returns 0 [3, 5].indexOf(5) // returns 1 [3, 5].indexOf(4) // returns -1 [3, 5, 3].indexOf(3) // returns 0 [3, 5, 3].indexOf(3, 1) // starts searching from position 1, returns 2 |  |
| Find the Position of the Last Occurrence of a Specified Array Element | [3, 5, 3].lastIndexOf(3) // returns 2  [3, 5].lastIndexOf(4) // returns -1 [3, 5, 3].lastIndexOf(3, 1) // returns 0 |  |
| Check if All Array Values Pass a Test | [2, 5, 4].every(x => x < 6); // returns true |  |
| Check if Some Array Values Pass a Test | [2, 5, 4].some(x => x > 6); // returns false |  |
| Find the Index of the First Array Element that Passes a Test | [2, 5, 4].findIndex(x => x < 6); // returns 0 |  |
| Create a Nested Array (Matrix) and Access Its Elements | let matrix = [[1, 2], [3, 4]];  matrix[0][1] // returns 2 |  |

## EXTRACT ARRAY PARTS. FILTER

|  |  |  |
| --- | --- | --- |
| Extract Array Parts | children.slice(1, 3) // returns elements 1 to 2: ['John', 'Mary'] children.slice(1) // returns elements 1 to the end | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Filter an Array (Value, Index, Array) – Must Be Saved in Another Variable | let nums = [45, 2, 4, 3];  let numsFiltered = nums.filter((v) => v > 3); // [45, 4] numsFiltered = nums.filter((v, i) => i % 2 == 1); // [2, 3] numsFiltered = nums.filter((v, i, a) => i == a.length - 1); // [3] |
| Find the Highest Number in an Array | Math.max.apply(null, [3, 5, 30]) // returns 30 |
| Find the Lowest Number in an Array | Math.min.apply(null, [2, 54, -83]) // returns -83 |  |
| Find the First Array Element that Passes a Test | [2, 5, 4].find(x => x < 6); // returns 2 |  |

## CHANGE/ADD/REMOVE/SWAP ARRAY ELEMENTS

|  |  |  |
| --- | --- | --- |
| Change an Array Element | let cars = ['Audi', 'BMW'];  cars[0] = 'Opel';  cars // returns ['Opel', 'BMW'] | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Add an Element to the End of the Array | cars.push('Fiat'); // adds 'Fiat' to the end of the array and returns 3 (new number of elements) cars[cars.length] = 'Fiat'; // adds 'Fiat' |
| Add an Element to the Beginning of the Array | cars.unshift('Fiat'); // adds 'Fiat' to the beginning of the array and returns 3 (the new number of elements) |  |
| Add Elements Anywhere into an Array | cars.splice(1, 0, 'Fiat'); // adds 'Fiat' to index 1, moves the rest of the elements to the right, removes 0 elements and returns an array with the removed elements  cars.splice(1, 0, 'Fiat', 'Ford'); // adds 'Fiat' to index 1 and 'Ford' to index 2, moves the rest of the elements to the right, removes 0 elements and returns an array with the removed elements |  |
| Fill All the Elements (between a Start Index and an End Index) with a Static Value | let numbers = [1, 2, 3, 4]; numbers.fill(0, 2, 4); // returns [1, 2, 0, 0] numbers.fill(5, 1); // returns [1, 5, 5, 5] numbers.fill(6); // returns [6, 6, 6, 6]  Array(4).fill(0); // returns [0, 0, 0, 0] |  |
| Remove the Last Element from an Array | cars.pop(); // removes the last element from the array and returns the removed element |  |
| Remove the First Element from an Array | cars.shift(); // removes the first element from the array and returns the removed element |  |
| Remove Elements from an Array | cars.splice(1, 2); // removes 2 elements starting from position 1 and returns an array with the removed elements |  |
| Swap Elements in an Array | let cars = ['Audi', 'Fiat', 'BMW'];  cars[cars.indexOf('Audi')] = cars.splice(cars.indexOf('BMW'), 1, 'Audi').join(''); // returns ['BMW', 'Fiat', 'Audi'] |  |
| Transpose a Matrix | let matrix = [[1, 2], [3, 4]]; let transpose = matrix[0].map((\_, i) => matrix.map(x => x[i])); // returns [[1, 3], [2, 4]] |  |

## CONVERT AN ARRAY TO STRING. CONCATENATE ARRAYS AND STRINGS

|  |  |  |
| --- | --- | --- |
| Convert an Array to String | ['Audi', 'BMW'].toString() // returns 'Audi,BMW' ['Audi', 'BMW'].join(' \* ') // returns 'Audi \* BMW' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Concatenate (Merge) Arrays and Strings | let boys = ['Peter', 'John']; let girls = ['Mary', 'Jill']; let children = boys.concat(girls, 'Ann'); // returns ['Peter', 'John', 'Mary', 'Jill', 'Ann'] |

## ARRANGE THE ELEMENTS OF AN ARRAY IN CERTAIN ORDER. SORT AN ARRAY

|  |  |  |
| --- | --- | --- |
| Reverse the Order of the Array Elements | ['Peter', 'John'].reverse() // returns ['John', 'Peter'] | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Sort an Array Alphabetically | ['Peter', 'John', 'ann'].sort((a, b) => a.localeCompare(b)) // returns ['ann', 'John', 'Peter'] (regardless of the case)  ['Peter', 'John', 'ann'].sort() // returns ['John', 'ann', 'Peter'] |
| Sort Numbers in an Array (Ascending) | [40, 2, 179].sort(function (a, b) { return a – b }) // returns [2, 40, 179] (if result is > 0, a is sorted after b) |  |
| Sort Numbers in an Array (Descending) | [40, 2, 179].sort(function (a, b) { return b – a }) // returns [179, 40, 2] |  |
| Sort an Array in Random Order | numbers.sort(function (a, b) { return 0.5 – Math.random() }) // not accurate  for (let i = numbers.length - 1; i > 0; i--) {  let j = Math.floor(Math.random() \* i);  let k = numbers[i];  numbers[i] = numbers[j];  numbers[j] = k; } // accurate, the Fisher Yates shuffle |  |
| Sort Objects in an Array | let cars = [  { type: 'Volvo', year: 2016 },  { type: 'Saab', year: 2010 },  { type: 'BMW', year: 2010 } ];  cars.sort(function (a, b) { return a.year – b.year || a.type.localeCompare(b.type) }) // sorts by year (ascending), then alphabetically by type |  |

## CALL A FUNCTION FOR EACH ARRAY ELEMENT

|  |  |  |
| --- | --- | --- |
| Call a Function for Each Array Element | ['John', 'Mark'].forEach((v, i, a) => console.log(`${i}: ${v}`)); // prints '0: John' and '1: Mark' on separate lines | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a New Array by Performing a Function on Each Element | let nums = [2, 5]; let numbersBy2 = nums.map(x => x \* 2); // [4, 10] let objNumsBy2 = nums.map(x => ({ num: x, numBy2: x \* 2 }));  // [{ num: 2, numBy2: 4 }, { num: 5, numBy2: 10 }] |
| Run a Function on Each Array Element (Left to Right) to Produce a Single Value | [2, 5, 4].reduce((a, b) => a + b); // returns 11  [2, 5, 4].reduce((a, b) => a + b, 10); // returns 21 (initial value 10)  let average = [2, 5, 4].reduce((acc, curr, i, array) => a + b / array.length, 0); // the average of the array equals 3.(6); the reducer function takes 4 arguments: accumulator, current value, (current index, source array) |  |
| Run a Function on Each Array Element (Right to Left) to Produce a Single Value | [2, 5, 4].reduceRight((a, b) => a + b); // returns 11  [2, 5, 4].reduceRight((a, b) => a + b, 10); // returns 21 (initial value 10)  [2, 5, 4].reduceRight((a, b) => a > b ? a : b); // returns 5 (the biggest number) |  |

# OBJECTS

## DECLARE AN OBJECT AND ACCESS ITS VALUES

|  |  |  |
| --- | --- | --- |
| Declare an Object with a Literal  {Key: Value} – Properties (Elements) with Number Keys First, Ascending, Then the Other Properties in Insertion Order | let townOrCity = 'town';  let person = {  name: 'John',  age: 35,  [townOrCity]: 'London'  }; | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Constructed | function Person(name, age) {  this.name = name;  this.age = age }  let myPerson = new Person('John', 35); |  |
|  | let person = {  name: 'John',  age: 35  };  let myPerson = Object.create(person); myPerson.name = 'Peter'; myPerson.name // returns 'Peter' |  |
| Declare an Empty Object and Then Add Properties | let person = {};  person.name = 'John'; // { name: 'John' }  person.age = 35; // { name: 'John', age: 35 } |  |
| Access the Values in an Object | person.name // returns 'John' person['name'] // returns 'John' (required in case of special symbols)  person.town // returns 'London'  person[townOrCity] // returns 'London' |  |
| Declare an Object with Properties that Already Exist as Variables | let name = 'John';  let age = 35;  let person = { name, age }; // returns { name: 'John', age: 35 } |  |
| Bind an Object Property to a Function (Getter) | let person = {  sex: 'M',  get occupation() { return this.sex == 'M' ? 'actor' : 'actress' }  };  person.occupation // returns 'actor' |  |
| Declare an Associative Array (a Collection of Values of the Same Type) | let contacts = {  'John': 00112345678,  'Peter': 00148716666 } |  |
| Factory Functions (Create an Object with Given Properties; No Need for 'This') | function createRect(width, height) {  let rect = { width, height };  rect.getArea = () => rect.width \* rect.height;  return rect; }  createRect(2, 7).getArea() // returns 14 |  |
| Decorator Functions (Add New Data and Behavior to Objects) | function canPrint(device) {  device.print = () => {  console.log(`${device.name} is printing a page.`);  } }  let printer = { name: 'ACME Printer' }; canPrint(printer); printer.print(); // prints 'ACME Printer is printing a page.' |  |
| Copy All Properties from One or More Source Objects to a Target Object | let person = { name: 'John', age: 35 }; let anotherPerson = Object.assign({}, person); // { name: 'John', age: 35 } |  |
| Loop through the Properties of an Object | let person = { name: 'John', age: 35 }; for (let [key, value] in person) {  console.log(key); // prints 'name', 'age'  console.log(value); // prints 'John', 35  } |  |
| Check if an Object Has a Specific Property | person.hasOwnProperty('age') // returns true  person[age] // returns true in an if statement |  |

## INTERNAL PROPERTIES

|  |  |  |
| --- | --- | --- |
| Get the Internal Properties of an Object Property | let person = { name: 'John', age: 35 };  Object.getOwnPropertyDescriptor(person, 'name'); // returns { value: 'John', writable: true, enumerable: true, configurable: true } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Create an Non-Enumerable Object Property | Object.defineProperty(person, 'town', { value: 'Sofia', enumerable: false, writable: true }); // default value: false  person.town // returns 'Sofia' person // returns { name: 'John', age: 35 } Object.keys(person).join(' ') // returns 'name age' |
|  | Object.defineProperty(person, '\_town', { value: 'Sofia', enumerable: false, writable: true });  Object.defineProperty(person, 'town', {  get() { return this.\_town; },  set(value) { this.\_town = value; },  enumerable: true }); |  |
| Create an Non-Configurable Object Property | Object.defineProperty(person, 'town', { value: 'Sofia', configurable: false });  delete person.town; // throws error in strict mode person // returns { name: 'John', age: 35, town: 'Sofia' } |  |
| Freeze (Sets All properties to Non-Writable and Non-Configurable) | Object.freeze(person); |  |
| Seal (Sets All properties to Non-Configurable) |  |  |

## METHODS

|  |  |  |
| --- | --- | --- |
| Declare an Object with a Method (a Function as an Object Property) | let person = {  firstName: 'John',  lastName: 'Doe',  fullName() { return this.firstName + ' ' + this.lastName } }; // this = the owner of the function  person.fullName(); // returns 'John Doe' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Call the Method with Another Object | let person2 = { firstName: 'Don', lastName: 'Johnson' }; person2.fullName(); // returns 'Don Johnson' |
| Declare a Method Outside of the Object | function print() {  return `${this.name} is printing a page.` }  let printer = {  name: 'ACME Printer',  print };  printer.print() // returns 'ACME Printer is printing a page.' |  |
| Override Built-in Methods | let person = {  name: 'Peter',  age: 35,  toString() {  return `${this.name} is ${this.age} years old.`;  } }  person.toString() // returns 'Peter is 35 years old.' console.log(`${person.toString()}`); // prints 'Peter is 35 years old.' console.log('' + person.toString()); // prints 'Peter is 35 years old.' |  |

## DESTRUCTURING SYNTAX

|  |  |  |
| --- | --- | --- |
| Destructuring Syntax | let person = { name: 'John', age: 35 }; let { age, name } = person; // 35, 'John'  let { age: myAge, name } = person; myAge // returns 35 | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Nested Destructuring | let department = {  name: 'Engeneering',  data: {  director: {  name: 'John',  position: 'Engeneering Director'  },  employees: []  } };  let {data: { director } } = department; // returns { name: 'John', position: 'Engeneering Director' } |
| Object and Array Destructuring: an Array of Objects | let employees = [  { name: 'John', position: 'worker' },  { name: 'Jane', position: 'secretary' } ];  let [{ name }] = employees; // returns 'John' |  |
| Object and Array Destructuring: an Object Containing an Array | let company = {  employees: ['John', 'Jane', 'Peter'],  name: 'Quick Build'  };  let { employees: [name] } = company; // returns 'John' |  |

## CONVERT AN OBJECT TO AN ARRAY/A JSON STRING

|  |  |  |
| --- | --- | --- |
| Get an Array with All the Object Keys | let person = { name: 'John', age: 35 };  Object.keys(person); // returns ['name', 'age'] | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Get an Array with All the Object Values | let person = { name: 'John', greeting() { return 'Hi, ' + this.name } };  };  Object.values(person); // returns ['John', f] |
| Get an Array of Tuples (Arrays of Two Elements: [Key, Value]) | let person = { name: 'John', age: 35 };  Object.entries(person); // returns [['name', 'John'], ['age', 35]] |  |
| Convert an Object to a JSON String | JSON.stringify(person) // returns '{"name":"John","age":35}' JSON.stringify(person, null, 2) // formatted with indentation |  |
| Convert a JSON String to Object | JSON.parse('{"name":"John","age":3}') // returns { name: 'John', age: 3 } |  |

## CHANGE AN OBJECT VALUE. DELETE AN OBJECT PROPERTY. SORT AN OBJECT

|  |  |  |
| --- | --- | --- |
| Change an Object Property | let person = { name: 'John', age: 35 };  person.name = 'George'; // { name: 'George', age: 35 } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Delete a Property from an Object | delete person.age; delete person['age']; |
| Sort an Object | let grades = { 'Tim': 4, 'Bill': 6 };  Object.entries(grades).sort((a, b) => a[0].localeCompare(b[0]));  // sorts alphabetically by name and returns [['Bill', 6], ['Tim', 4]] |  |

## DECLARE A CLASS AND CREATE INSTANCES

|  |  |  |
| --- | --- | --- |
| Declare a Class (a Template for Creating Objects) | class Rectangle {  constructor(height, width) {  this.height = height;  this.width = width;  }  } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Create an Instance | let square = new Rectangle(3, 3); // { height: 3, width: 3 } |  |
| Create a Static Method for a Class (Part of the Class, NOT of the Instances) | class MyClass {  static myStaticMethod() {  console.log('from static method');  } }  MyClass.myStaticMethod(); // prints 'from static method' |  |
| Check if an Object is an Instance of a Specified Class | square instanceof Rectangle // returns true square instanceof Object // returns true |  |
| Accessor Properties (Getter and Setter) | class Circle {  constructor(r) {  this.r = r;  }    get diameter() {  return this.r \* 2;  }  set diameter(value) {  this.r = value / 2;  } }  const myCircle = new Circle(3); myCircle.diameter // returns 6 myCircle.diameter = 10; myCircle.r // returns 5 |  |
| Prototype | Object.getPrototypeOf(myObj)  Person.prototype.nationality = 'Bulgarian'; |  |
|  | function Person(name) {  this.name = name; }  Person.prototype.sayHi = function() {  console.log(`${this.name} says hi!`); }  function Employee(name, salary) {  Person.call(this, name);  this.salary = salary; }  Employee.prototype = Object.create(Person.prototype); Employee.prototype.collectSalary = function() {  console.log(`${this.name} collected ${this.salary}`); }  const myEmployee = new Employee('Peter', 60000);  myEmployee // returns Person { name: 'Peter', salary: 60000 } myEmployee.collectSalary() // prints 'Peter collected 60000' myEmployee.sayHi() // prints 'Peter says hi!' |  |

# MAP OBJECTS

|  |  |  |
| --- | --- | --- |
| Declare a Map | let numbers = new Map([[1, 'one'], [0, 'zero']]);  let books = new Map([['title', 'Harry Potter'], ['author', 'J.K. Rowling']]); | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Add Element (Property/Key, Value) | numbers.set(2, 'two'); |
| Element Order (Always in Insertion Order) | numbers // returns {1 => 'one', 0 => 'zero', 2 => 'two'} |  |
| Access Value | numbers.get(1) // returns 'one' |  |
| Find Number of Entries | numbers.size // returns 3 |  |
| Check if a Map Has a Specified Key | numbers.has(0) // returns true numbers.has(4) // returns false |  |
| Change a Value | numbers.set(1, 'two');  numbers // returns {1 => 'two', 0 => 'zero', 2 => 'two'} |  |
| Delete an Element | numbers.delete(0);  numbers.delete(2);  numbers // returns {1 => 'one'} |  |
| Delete All Elements | numbers.clear(); |  |
| Loop through Elements | for (let [number, word] of numbers) {  console.log(`${number} - ${word}`);  } // prints '1 – one', '0 – zero' |  |
| Create a Map Iterator Holding All Entries | numbers // the map is an iterator holding all key => value pairs  numbers.entries() // returns an iterator holding all key => value pairs of the map |  |
| Create a Map Iterator Holding All Keys | numbers.keys() // returns an iterator holding all keys of the map |  |
| Create a Map Iterator Holding All Values | numbers.values() // returns an iterator holding all values of the map |  |
| Convert a Map Iterator to an Array | Array.from(numbers.keys()) // returns [1, 0]  [...numbers.keys()] // returns [1, 0]  [...numbers] // returns [[1, 'one'], [0, 'zero']] |  |
| WeakMap |  |  |

# SET OBJECTS

|  |  |  |
| --- | --- | --- |
| Declare a Set (a List of Unique Values) | let numbers = new Set([1, 2, 2]);  numbers // returns {1, 2} | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Add a Value | numbers.add(6); // adds 6 and returns the new set |
| Check if a Set Has a Specified Value | numbers.has(6) // returns true |  |
| Delete a Value | numbers.delete(6) // deletes 6 and returns true |  |
| Loop through Elements | for (let number of numbers) {  console.log(number);  } // prints 1, 2, 6 |  |
| WeakSet |  |  |

# DATE OBJECTS

## CREATE A DATE OBJECT

|  |  |  |
| --- | --- | --- |
| Create a Date Object | newDate() // returns current date and time | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Date Object with a Specified Date and Time (7 Parameters) | newDate(2020, 6, 1, 9, 36, 0, 0) //returns 'Wed Jul 01 2020 09:36:00 GMT+0300 (Eastern European Summer Time)': year (one and two digit years are interpreted as 19xx), month (counted from 0), day, hour, minute, second, millisecond |
| Use only 2 Parameters | newDate(2020, 6) // returns 'Wed Jul 01 2020 00:00:00 GMT+0300 (Eastern European Summer Time)' |  |
| Use only 1 Parameter | newDate(2020) // returns 'Thu Jan 01 1970 02:00:02 GMT+0200 (Eastern European Standard Time)': only 1 parameter is treated as milliseconds counted from Jan 01 1970 00:00:00 UTC |  |
| Use a String as a Parameter | newDate('2015-03-25T12:00:00Z') // returns 'Wed Mar 25 2015 14:00:00 GMT+0200 (Eastern European Standard Time)': Z for UTC time |  |
| JS Short Dates | newDate('03/25/2015') // returns 'Wed Mar 25 2015 00:00:00 GMT+0200 (Eastern European Standard Time)' |  |
| JS Long Dates | newDate('Mar 25 2015') // returns 'Wed Mar 25 2015 00:00:00 GMT+0200 (Eastern European Standard Time)': MMM DD YYYY or DD MMM YYYY |  |
| JS Long Dates with the Month Written in Full | newDate('October 13, 2014 11:13:00') // returns 'Mon Oct 13 2014 11:13:00 GMT+0300 (Eastern European Summer Time)': case insensitive, commas are ignored |  |

## CONVERT A DATE TO A STRING/NUMBER. COMPARE DATES

|  |  |  |
| --- | --- | --- |
| Convert a Date to a String (UTC, Date and Time) | newDate().toUTCString() | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Convert a Date to a String (without Time) | newDate().toDateString() |
| Convert a Date to a Number (Milliseconds from January 1st, 1970) | Date.now() // returns 1427234400000 (milliseconds from Jan 01 1970 00:00:00 UTC) Date.parse('Mar 25 2015') // returns 1427234400000  Number(new Date('Mar 25 2015')) // returns 1427234400000  new Date('Mar 25 2015').getTime() // returns 1427234400000 |  |
| Compare Dates | new Date('Mar 25 2015') < new Date('Mar 30 2015') // returns true |  |

## ACCESS ONLY PART OF THE DATE

|  |  |  |
| --- | --- | --- |
| Get the Year from a Date Object | new Date('Mar 25 2015').getFullYear() // returns 2015 | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Get the Month from a Date Object (0 to 11) | new Date('Mar 25 2015').getMonth() // returns 2 |
| Get the Month from a Date Object (As a Name) | let months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'];  months[new Date('Mar 25 2015').getMonth()] // returns 'March' |  |
| Get the Day from a Date Object (1 to 31) | new Date('Mar 25 2015').getDate() // returns 25 |  |
| Get the Hours from a Date Object (0 to 23) | new Date('Mar 25 2015').getHours() // returns 0 |  |
| Get the Minutes from a Date Object (0 to 59) | new Date('Mar 25 2015').getMinutes() // returns 0 |  |
| Get the Seconds from a Date Object (0 to 59) | new Date('Mar 25 2015').getSeconds() // returns 0 |  |
| Get the Milliseconds from a Date Object (0 to 999) | new Date('Mar 25 2015').getMilliseconds() // returns 0 |  |
| Get the Weekday from a Date Object (0 to 6) | new Date('Mar 25 2015').getDay() // returns 3 (Wednesday; Sunday is counted as 0) |  |

## CHANGE PART OF THE DATE

|  |  |  |
| --- | --- | --- |
| Change the Year (and Optionally the Month and Date) in a Date Object | new Date('Mar 25 2015').setFullYear(2020) // returns 'Wed Mar 25 2020 00:00:00 GMT+0200 (Eastern European Standard Time)'  new Date('Mar 25 2015').setFullYear(2020, 11, 3) // returns 'Thu Dec 03 2020 00:00:00 GMT+0200 (Eastern European Standard Time)' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Change the Month in a Date Object (0 to 11) | new Date('Mar 25 2015').setMonth(11) // returns 'Fri Dec 25 2015 00:00:00 GMT+0200 (Eastern European Standard Time)' |
| Change the Date in a Date Object (1 to 31) | new Date('Mar 25 2015').setDate(15) // returns 'Sun Mar 15 2015 00:00:00 GMT+0200 (Eastern European Standard Time)'  let date = new Date('Mar 25 2015')  date.setDate(date.getDate() + 50) // returns 'Thu May 14 2015 00:00:00 GMT+0300 (Eastern European Summer Time)' |  |
| Change the Hours in a Date Object (0 to 23) | new Date('Mar 25 2015').setHours(22) // returns 'Wed Mar 25 2015 22:00:00 GMT+0200 (Eastern European Standard Time)' |  |
| Change the Minutes in a Date Object (0 to 59) | new Date('Mar 25 2015').setMinutes(22) // returns 'Wed Mar 25 2015 00:22:00 GMT+0200 (Eastern European Standard Time)' |  |
| Change the Seconds in a Date Object (0 to 59) | new Date('Mar 25 2015').setSeconds(22) // returns 'Wed Mar 25 2015 00:00:22 GMT+0200 (Eastern European Standard Time)' |  |

# HTML DOM (DOCUMENT OBJECT MODEL)

## SELECT HTML ELEMENTS

|  |  |  |
| --- | --- | --- |
| Select an HTML Element by Its ID | document.getElementById('demo') // returns the HTML element with ID 'demo' | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Select HTML Elements by Their Class Name | document.getElementsByClassName('demo') // returns a live HTML collection of all elements having a class named 'demo' |
| Select HTML Elements by Their Tag Name | document.getElementsByTagName('p') // returns a live HTML collection of all <p> elements |  |
| Select HTML Elements by Their CSS Selector | document.querySelector('#title') // returns the first HTML element with ID 'title' |  |
| Select All HTML Elements That Share a CSS Selector | document.querySelectorAll('article.list') // returns a static NodeList with all <article> elements having a class named 'list' |  |
|  | liElement.classList; liElement.classList.add('myClass'); liElement.classList.remove('myClass'); |  |
|  | .dataset // obtain DOMStringMap of custom data attributes (case-insensitive) |  |
|  |  |  |

## SORT HTML COLLECTIONS

|  |  |  |
| --- | --- | --- |
|  | Array  .from(collection.children)  .sort((a, b) => a.textContent.localeCompare(b.textContent))  .forEach(g => collection.appendChild(g));  Array  .from(collection.children)  .sort((a, b) => a.childNodes[0].textContent.localeCompare(b.childNodes[0].textContent))  .forEach(g => collection.appendChild(g)); | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

## MODIFY HTML ELEMENTS

|  |  |  |
| --- | --- | --- |
|  | .setAttribute() .removeAttribute('spellCheck') | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## CREATE/DELETE DOM ELEMENTS

|  |  |  |
| --- | --- | --- |
|  | let paragraph = document.createElement('p'); paragraph.textContent = 'This is a paragraph.'; | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Copy/Clone DOM Element |  |
| Add a New Child (at the End) | document.body.appendChild(paragraph);  parent.insertBefore(newEl, referenceEl) |  |
| Add a New Child (at the Beginning) | document.body.prependChild(paragraph); |  |
| Delete DOM Elements from Parent Element | let ulElement = document.querySelector('ul'); let liElement = document.querySelector('li'); ulElement.removeChild(liElement); |  |
| Delete DOM Elements | liElement.remove(); |  |
|  | liElement.replaceWith(newElement); |  |
|  | liElement.before(newElement); |  |
|  | liElement.after(newElement); |  |
|  |  |  |

## EVENTS

|  |  |  |
| --- | --- | --- |
| Mouse Events | click, mouseover, mouseout, mouseup, mousedown | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  | ev.preventDefault() |
|  | element.addEventListener('click', (event) => console.log(event)); event.target // returns element ev.target.tagName |  |

## GOOGLE APIS

|  |  |  |
| --- | --- | --- |
| Google Maps |  | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Google Fonts |  |
| Google Charts |  |  |

# ERRORS IN JS

## THROWING ERRORS (EXCEPTIONS)

|  |  |  |
| --- | --- | --- |
| General Error | throw new Error('Invalid state'); | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Range Error | throw new RangeError('Invalid index'); |
| Type Error | throw new TypeError('String expected'); |  |
| Reference Error | throw new ReferenceError('Missing age'); |  |

## EXCEPTION PROPERTIES

|  |  |  |
| --- | --- | --- |
|  | ex.name | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  | ex.message |
|  | ex.stack |  |

## TRY – CATCH

|  |  |  |
| --- | --- | --- |
|  | try {  new Array(-1) } catch (err) {  console.log(err); // prints 'RangeError: invalid array length'  console.log(err.message); // prints 'invalid array length' } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |
|  |  |  |

# MODULES

## EXPORT MODULES

|  |  |  |
| --- | --- | --- |
| In the Source File (myModule.js) | function sum(a, b) {  return a + b; }  module.exports = sum; | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| In Another File | const sum = require('./myModule');  sum(4, 6); // returns 10 |
| Exporting Several Functions | function sum(a, b) {  return a + b; }  function mult(a, b) {  return a \* b; }  module.exports = {  sum,  mult }; |  |
|  | const { mult } = require('./myModule');  mult(4, 6); // returns 24 |  |

# MOCHA AND CHAI

## INSTALLATION

|  |  |  |
| --- | --- | --- |
| Type in the VSC Terminal (or in CMD), Then Press Enter | npm install -g mocha --save  npm install -g chai --save | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Check if It Worked | mocha --version |  |
| Usage | const { mult } = require('./myModule'); const { expect } = require('chai'); // const { assert } = require('chai');  describe('Mult function', () => {  beforeEach()  it('works', () => {  expect(mult(2, 3)).to.equal(6);  });  );  deepEqual  // in the terminal: mocha myModule.test.js // returns Mult function, works, 1 passing |  |

# AJAX

## SUBTITLE

|  |  |  |
| --- | --- | --- |
|  | setTimeout(function () {  console.log('Hi!'); }, 2000); // if 0, once again after everything else | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  | fetch(url).then(); |  |
| fetch(resource, Options object) | fetch(resource, Options) |  |
|  | const [response1, response2] = Promise.All({ fetch(url1), fetch(url2) }); |  |

# REMOTE DATA AND AUTHENTICATION

## SUBTITLE

|  |  |  |
| --- | --- | --- |
| Register | document.querySelector('form').addEventListener('submit', onRegisterSubmit);  async function onRegisterSubmit(event) {  event.preventDefault();  const formData = new FormData(event.target);  // [...formData.entries()] returns an array of all form field names + values   const email = formData.get('email');  const password = formData.get('password');  const rePass = formData.get('rePass');  if (email == '' || password == '') {  return alert('All fields are required!')  } else if (password != rePass) {  return alert('Passwords don\'t match!');  }  const response = await fetch('http://localhost:3030/users/register', {  method: 'post',  headers: { 'Content-Type': 'application/json' },  body: JSON.stringify({ email, password })  });  if (response.ok == false) {  const error = await response.json();  return alert(error.message);  }  const data = await response.json();  sessionStorage.setItem('userToken', data.accessToken);  window.location.pathname = 'index.html'; } | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
| Login | document.querySelector('form').addEventListener('submit', onLoginSubmit);  async function onLoginSubmit(event) {  event.preventDefault();  const formData = new FormData(event.target);  const email = formData.get('email');  const password = formData.get('password');  const response = await fetch('http://localhost:3030/users/login', {  method: 'post',  headers: { 'Content-Type': 'application/json' },  body: JSON.stringify({ email, password })  });  if (response.ok == false) {  const error = await response.json();  return alert(error.message);  }  const data = await response.json();  sessionStorage.setItem('userToken', data.accessToken);  window.location.pathname = 'index.html'; } |  |
| Logout | const token = sessionStorage.getItem('userToken');    if (token != null) {  document.getElementById('user').style.display = 'inline-block';  document.getElementById('logoutBtn').addEventListener('click', logout);  } else {  document.getElementById('guest').style.display = 'inline-block';  }  async function logout() {  const token = sessionStorage.getItem('userToken');  const response = await fetch('http://localhost:3030/users/logout', {  method: 'get',  headers: { 'X-Authorization': token }  });  if (response.ok == false) {  const error = await response.json();  return alert(error.message);  }  sessionStorage.removeItem('userToken');  window.location.pathname = 'index.html';  } |  |
|  |  |  |

# SINGLE PAGE APPLICATION (SPA)

## SUBTITLE

|  |  |  |
| --- | --- | --- |
|  | <script type="module" src="/src/app.js"></script>  dom.js // export function e() {...}  home.js:  let main;  let section;  export function setupHome(mainTarget, sectionTarget) {  main = mainTarget;  section = sectionTarget;  }  export async function showHome() {  main.innerHTML = '';  main.appendChild(section);  }  app.js:  import { setupHome } from './home';  import { setupDetails } from './details';  import { setupLogin } from './login';  import { setupRegister } from './register';  import { setupCreate } from './create';  import { setupEdit } from './edit'; | [Python](Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |  |