JavaScript

[WRITING A PROGRAM: BASIC SOURCE CODE 5](#_Toc74153807)

[IDENTIFIERS (VARIABLE/FUNCTION/PROPERTY NAMES) 5](#_Toc74153808)

[DATA TYPES 5](#_Toc74153809)

[BASIC DATA TYPES 5](#_Toc74153810)

[FIND THE DATA TYPE OF A VARIABLE 6](#_Toc74153811)

[CONVERT DATA TYPES. TYPE COERCION 7](#_Toc74153812)

[DECLARE DATA. ASSIGN VALUES 8](#_Toc74153813)

[DECLARE A VARIABLE 8](#_Toc74153814)

[LIFE OF A VARIABLE 8](#_Toc74153815)

[CONSTANTS IN JAVASCRIPT 9](#_Toc74153816)

[DECLARE AN OBJECT (CONST RECOMMENDED) 9](#_Toc74153817)

[DECLARE A FUNCTION OBJECT 9](#_Toc74153818)

[CONDITIONS 10](#_Toc74153819)

[TRUTHY/FALSY VALUES 10](#_Toc74153820)

[OPERATORS 10](#_Toc74153821)

[COMPARE DATA 11](#_Toc74153822)

[CONDITIONAL STATEMENTS 11](#_Toc74153823)

[LOOPS 11](#_Toc74153824)

[FOR LOOP 11](#_Toc74153825)

[WHILE LOOP 12](#_Toc74153826)

[BREAK A LOOP 13](#_Toc74153827)

[FUNCTIONS 13](#_Toc74153828)

[CALL A FUNCTION. FUNCTION PARAMETERS 13](#_Toc74153829)

[FUNCTION CONTEXT. THIS OBJECT 14](#_Toc74153830)

[FIRST-CLASS AND HIGHER-ORDER FUNCTIONS 16](#_Toc74153831)

[OUTPUT 17](#_Toc74153832)

[NUMBERS IN JAVASCRIPT 18](#_Toc74153833)

[BASIC MATH 18](#_Toc74153834)

[RANDOM NUMBERS 20](#_Toc74153835)

[TEXT PROCESSING 21](#_Toc74153836)

[COMBINE STRINGS 21](#_Toc74153837)

[ACCESS THE CHARACTERS OF A STRING. STRING LENGTH 21](#_Toc74153838)

[CONVERT STRINGS 21](#_Toc74153839)

[COMPARE STRINGS 22](#_Toc74153840)

[SEARCH FOR A SPECIFIED CHARACTER/STRING PART 22](#_Toc74153841)

[EXTRACT STRING PARTS 23](#_Toc74153842)

[CONVERT TO LOWER/UPPERCASE 23](#_Toc74153843)

[CHANGE STRINGS (RESULT MUST BE SAVED IN ANOTHER VARIABLE) 23](#_Toc74153844)

[REGULAR EXPRESSIONS 24](#_Toc74153845)

[MATCHING RULES 24](#_Toc74153846)

[PREDEFINED CLASSES 25](#_Toc74153847)

[QUANTIFIERS 25](#_Toc74153848)

[GROUP CONSTRUCTS 25](#_Toc74153849)

[FLAGS/MODIFIERS 26](#_Toc74153850)

[ANCHORS 26](#_Toc74153851)

[REGEX IN JAVASCRIPT 27](#_Toc74153852)

[EXAMPLES 28](#_Toc74153853)

[ARRAY OBJECTS 28](#_Toc74153854)

[JAVASCRIPT ARRAY CHARACTERISTICS 28](#_Toc74153855)

[CREATE AN ARRAY AND ACCESS ITS ELEMENTS 29](#_Toc74153856)

[INSPECT AN ARRAY 30](#_Toc74153857)

[EXTRACT ARRAY PARTS. FILTER 30](#_Toc74153858)

[CHANGE/ADD/REMOVE/SWAP ARRAY ELEMENTS 31](#_Toc74153859)

[CONVERT AN ARRAY TO STRING. CONCATENATE ARRAYS AND STRINGS 32](#_Toc74153860)

[CONVERT A NESTED ARRAY TO AN OBJECT 32](#_Toc74153861)

[ARRANGE THE ELEMENTS OF AN ARRAY IN CERTAIN ORDER. SORT AN ARRAY 32](#_Toc74153862)

[CALL A FUNCTION FOR EACH ARRAY ELEMENT 33](#_Toc74153863)

[OBJECTS 33](#_Toc74153864)

[DECLARE AN OBJECT AND ACCESS ITS VALUES 33](#_Toc74153865)

[INTERNAL PROPERTIES 35](#_Toc74153866)

[METHODS 37](#_Toc74153867)

[DESTRUCTURING SYNTAX 37](#_Toc74153868)

[CONVERT AN OBJECT TO AN ARRAY/A JSON STRING 38](#_Toc74153869)

[CHANGE AN OBJECT VALUE. DELETE AN OBJECT PROPERTY. SORT AN OBJECT 39](#_Toc74153870)

[PROTOTYPE- AND CLASS-BASED INHERITANCE 39](#_Toc74153871)

[DECLARE A CLASS AND CREATE INSTANCES 40](#_Toc74153872)

[THE FOUR PILLARS OF OBJECT-ORIENTED PROGRAMMING 43](#_Toc74153873)

[MAP OBJECTS 43](#_Toc74153874)

[SET OBJECTS 44](#_Toc74153875)

[DATE OBJECTS 45](#_Toc74153876)

[CREATE A DATE OBJECT 45](#_Toc74153877)

[CONVERT A DATE TO A STRING/NUMBER. COMPARE DATES 45](#_Toc74153878)

[ACCESS ONLY PART OF THE DATE 46](#_Toc74153879)

[CHANGE PART OF THE DATE 46](#_Toc74153880)

[HTML DOM (DOCUMENT OBJECT MODEL) 47](#_Toc74153881)

[MAIN CONCEPTS 47](#_Toc74153882)

[IN THE BROWSER 47](#_Toc74153883)

[JAVASCRIPT IN HTML 48](#_Toc74153884)

[ACCESS DOM NODES 48](#_Toc74153885)

[ACCSESS THE ATTRIBUTES OF A DOM NODE 49](#_Toc74153886)

[SORT HTML COLLECTIONS 49](#_Toc74153887)

[MODIFY DOM NODES 50](#_Toc74153888)

[CREATE/DELETE DOM ELEMENTS 50](#_Toc74153889)

[EVENTS 51](#_Toc74153890)

[GOOGLE APIS 52](#_Toc74153891)

[ERRORS IN JS 52](#_Toc74153892)

[THROWING ERRORS (EXCEPTIONS) 52](#_Toc74153893)

[TESTING THE CODE 53](#_Toc74153894)

[MODULES IN JS 53](#_Toc74153895)

[TYPES OF MODULES 53](#_Toc74153896)

[EXPORT AND IMPORT MODULES 53](#_Toc74153897)

[MODULE WRAPPER FUNCTION 54](#_Toc74153898)

[MOCHA AND CHAI 54](#_Toc74153899)

[INSTALLATION 54](#_Toc74153900)

[WRITE AND RUN TESTS 55](#_Toc74153901)

[HTTP AND REST SERVICES 55](#_Toc74153902)

[HTTP REQUESTS 55](#_Toc74153903)

[HTTP RESPONSES 56](#_Toc74153904)

[AJAX 56](#_Toc74153905)

[SET TIME OUT 56](#_Toc74153906)

[XHR OBJECTS (XML HTTP REQUEST) 56](#_Toc74153907)

[FETCH API 57](#_Toc74153908)

[ASYNC FUNCTIONS 57](#_Toc74153909)

[REMOTE DATA AND AUTHENTICATION 58](#_Toc74153910)

[SINGLE PAGE APPLICATION (SPA) 60](#_Toc74153911)

[JS BACK END 61](#_Toc74153912)

[NODE.JS 61](#_Toc74153913)

[NODE.JS CORE MODULES 61](#_Toc74153914)

[URL MODULE 61](#_Toc74153915)

[QUERYSTRING MODULE 62](#_Toc74153916)

[PATH MODULE 62](#_Toc74153917)

[OPERATING SYSTEM MODULE 62](#_Toc74153918)

[FILE SYSTEM MODULE 62](#_Toc74153919)

[EVENTS MODULE 64](#_Toc74153920)

[HTTP MODULE 65](#_Toc74153921)

[EXPRESS.JS 65](#_Toc74153922)

[INSTALLATION AND MAIN CHARACTERISTICS 65](#_Toc74153923)

[ROUTER PATHS 67](#_Toc74153924)

[ROUTER RESPONSES 67](#_Toc74153925)

[MIDDLEWARE (PLUGINS, EXTENSIONS) 67](#_Toc74153926)

[STATIC FILES 68](#_Toc74153927)

[VIEW ENGINES 69](#_Toc74153928)

[HANDLEBARS 69](#_Toc74153929)

[JOI 70](#_Toc74153930)

[EXTERNAL LIBRARIES 70](#_Toc74153931)

[STREAMS 71](#_Toc74153932)

[PUBLISH-SUBSCRIBE PATTERN 72](#_Toc74153933)

[DATABASES 73](#_Toc74153934)

[TYPES OF DATABASES 73](#_Toc74153935)

[MONGO DB 73](#_Toc74153936)

[MONGOOSE 74](#_Toc74153937)

[INSTALLATION AND CONNECTION TO DB 74](#_Toc74153938)

[CREATE A MODEL. VALIDATION 74](#_Toc74153939)

[READ DATA 76](#_Toc74153940)

[UPDATE DATA 76](#_Toc74153941)

[REMOVE DATA 77](#_Toc74153942)

[COUNT DOCUMENTS (ENTRIES) 77](#_Toc74153943)

[REFERENCES AND POPULATION 77](#_Toc74153944)

[APPLICATION SECURITY 78](#_Toc74153945)

[COOKIES 78](#_Toc74153946)

[SESSIONS 78](#_Toc74153947)

[AUTHENTICATION 79](#_Toc74153948)

[JSON WEB TOKEN 80](#_Toc74153949)

# WRITING A PROGRAM: BASIC SOURCE CODE

|  |  |  |
| --- | --- | --- |
| File Name | Only lowercase (when possible). Files containing a class or a DB model should have the name of the class/model with initial capital letter. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourceCode) C++ C# Java |
| File Extension | .js |
| Run in Command Prompt | Command Prompt > type node, press 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 backticks (``) instead of quotes (''/"") or using + for concatenation. |
| Comments (Not Executable Code) | // a single-line comment  /\* a multiline comment \*/ |
| Strict Mode (Turns Bad Syntax into Actual Errors) | 'strict mode'; // at the beginning of the code/function; automatically used in modules |
| Simple Executable Statements (End with Semicolon) | let x = 5; |  |
| Code Blocks (No Semicolon) | if (x < 7) { *code block* } |  |

# IDENTIFIERS (VARIABLE/FUNCTION/PROPERTY NAMES)

|  |  |  |
| --- | --- | --- |
| Lower Camel Case | let firstName = 'John'; | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Identifiers) C++ C# Java |
| Starting with a Number | no |
| Case-Sensivite | yes |
| Hyphens in Identifiers | no |
| jQuery Objects | const $email = $('#email'); |  |
| Private Properties | \_id |  |
| Parameters To Be Ignored: \_ | ['a', 'b'].map((\_, i) => i); // [0, 1] |  |

# 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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 // 'Mark'  name // '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 // '1,2,31'  numbers // [1, 2, 3] |

## FIND THE DATA TYPE OF A VARIABLE

|  |  |  |
| --- | --- | --- |
| Find the Data Type of a Variable | typeof 'John' // string  typeof 35 // number  isNaN(35) // false Number.isInteger(35) // true  Number.isSafeInteger(x) // true for x between -(253 - 1) and +(253 - 1)  isFinite(x) // false if x is Infinity or NaN typeof NaN // number typeof Infinity // number typeof true // boolean  typeof { name: 'John',  age: 35 } // object typeof ['John', '35'] // object  Array.isArray(['John', '35']) // true  ['John', '35'] instanceof Array // true  ['John', '35'].constructor.toString().includes('Array') // true typeof new Date() // object  typeof (() => { console.log('Hi!') }) // function  typeof undefined // undefined | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#FindTheDataTypeOfAVariable) C++ C# Java |
| Find the Constructor Function for a Variable | 'John'.constructor // f String() (3.14).constructor // f Number() false.constructor // f Boolean() [1, 2, 3].constructor // f Array() { name: 'John', age: 35 }.constructor // f Object() new Date().constructor // f Date() function () { }.constructor // f Function() |
| Find Out if an Object is an Array | Array.isArray([1, 2]) // true [1, 2].constructor == Array // true |
| Find Out if an Object is a Date | function isDate(myDate) {  return myDate.constructor.toString().indexOf('Date') > -1;  } // true or false  function isDate(myDate) {  return myDate.constructor == Date;  } // true or false |

## CONVERT DATA TYPES. TYPE COERCION

|  |  |  |
| --- | --- | --- |
| Convert Other Types to String | String(3) // '3' (3).toString() // '3'  JSON.stringify({ age: 34 }) // '{"age":34}'  { age: 34 }.toString() // '[object Object]'  [1, [2, 3]].toString() // '1,2,3'  JSON.stringify([1, [2, 3]]) // '[1,[2,3]]'  true.toString() // 'true'  new Date().toString() // 'Thu Jun 10 2021 12:02:40 GMT+0300 (Eastern European Summer Time)'  newDate().toUTCString() // 'Thu, 10 Jun 2021 09:03:55 GMT'  new Date.toISOString() // '2021-06-10T09:07:08.295Z'  newDate().toDateString() // 'Thu Jun 10 2021' | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#ConvertDataTypes) C++ C# Java |
| Convert Other Types to Number | Number('3') // 3 parseInt('3') // 3 parseFloat('3.14') // 3.14 Number('John') // NaN Number('') // 0 Number([20]) // 20 Number([10, 20]) // NaN Number([]) // 0 Number({}) // NaN Number(false) // 0 Number(true) // 1 Number(new Date('Mar 25 2015')) // 1427234400000 (milliseconds counted from Jan 01 1970 00:00:00 UTC) |
| Convert Other Types to Boolean (Truthy and Falsy Values) | Boolean('3') // true Boolean(0) // false Boolean(3) // true  Boolean(3 / 2) // true Boolean(3 / 'd') // false Boolean('0') // true Boolean('') // false Boolean([]) // true (BUT [] == true returns false!) Boolean({}) // true  Boolean(NaN) // false Boolean(null) // falseBoolean(undefined) // false |
| Type Coercion | '' + 3 // '3' `${3}` // '3' +'3' // 3 2 \* '10' // 20 4 < '14' // true if (3) { ... } // 3 coerced to true |  |

# DECLARE DATA. ASSIGN VALUES

## DECLARE A VARIABLE

|  |  |  |
| --- | --- | --- |
| Assignment Operators | = // assigns a value to a variable (the statement returns the value) += // 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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; |  |

## LIFE OF A VARIABLE. HOISTING

|  |  |  |
| --- | --- | --- |
| Global Scope (*var*, *let* and *const* outside a Function) | let firstName = 'John'; // code here can use firstName function myFunc() {  // code here can use firstName  } // code here can use firstName | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#LifeOfAVariable) C++ C# Java |
| Function Scope (*var*, *let* and *const* in a Function) | // code here CANNOT use firstName function myFunc() {  let firstName = 'John';  // code here can use firstName  } // code here CANNOT use firstName |
| Block Scope (*let* and *const* in a Code Block) | let i = 5;  for (let i = 1; i < 10; i++) { ... } // another variable with the same name  // i is 5 here |

## CONSTANTS IN JAVASCRIPT

|  |  |  |
| --- | --- | --- |
| Constants in JS | Recommended use with objects (reference values) when we do not intend to reassign or redeclare them in the same scope. Must be assigned a value when declared. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#LifeOfAVariable) C++ C# Java |
| Constants with Primitive Values | const pi = 3.14;  pi = 3.14159; // will throw an error, constants cannot be reassigned |
| Constants with Reference Values: Arrays | const numbers = [1, 2];  numbers.push(3); // numbers is now [1, 2, 3]  numbers = numbers.slice(0); // not allowed |
| Constants with Reference Values: Objects | const person = { name: 'Peter', age: 23 };  person.name = 'John'; // person is now { name: 'John', age: 23 }  person = { name: 'John', age: 23 }; // not allowed |  |

## DECLARE AN OBJECT (CONST RECOMMENDED)

|  |  |  |
| --- | --- | --- |
| Declare an Object {Property/Key: Value} | const person = { name: 'John', age: 35 }; | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare an Array Object [List of Values] | const people = ['John', 'Mary', 'George']; |
| Declare a Map Object [[Property/Key, Value]] | const person = new Map([['name', 'John'], ['age', 35]]); |
| Declare a Set Object  [List of Unique Values] | const 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 | function funcName() {  // content of the function  } |  |

# CONDITIONS

## TRUTHY/FALSY VALUES

|  |  |  |
| --- | --- | --- |
| Truthy Values | Coerce to true in a boolean context: 'a', 5, {}, [], etc. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#TruthyFalsyValues) C++ C# Java |
| Falsy Values | Coerce to true in a boolean context: 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Operators) C++ C# Java |
| Logical Operators | && // and; returns the leftmost falsy value or the last truthy if all are true  || // or; returns the leftmost truthy value or the last falsy if all are false  ! // not; returns false if its operand can be converted to true, otherwise true |
| Ternary Operator | age < 18 ? 'young' : 'old' // returns 'young' if the age < 18, 'old' if age > 18 |

## COMPARE DATA

|  |  |  |
| --- | --- | --- |
| Comparing Numbers | 1 < 2 // true | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Comparing a Number to a String | 2 < '12' // true (numeric strings are converted to numbers)  2 < 'John' // false 0 == '' // true 0 === '' // false |
| Comparing Strings | 'John' <= 'John' // true 'a' < 'b' // true (alphabetically ordered) '2' < '12' // false (alphabetically ordered) |
| Comparing Objects: Always Returns False | [1, 2, 3] == [1, 2, 3] // false  { name: 'John' } == { name: 'John' } // false |

## CONDITIONAL STATEMENTS

|  |  |  |
| --- | --- | --- |
| If – (Else If) – Else  if (*condition*) {...}  else if (*condition*) {...}  else {...} | if (x > y) {  // code to be executed if x > y } else if (x < y) {  // code to be executed if x < y } else {  // code to be executed if x = y } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| If with Only One Statement in the Code Block | if (x > y) return x; |
| Switch | switch (num) {  case 1: day = 'Monday'; break; // if num = 1 day = 'Monday'  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 7  default: day = 'unknown'; break; // optional } |

# LOOPS

## FOR LOOP

|  |  |  |
| --- | --- | --- |
| For Loop (*set variable*; *condition*; *change variable*) | for (let i = 0; i < 10; i++) {  // code block to be executed 10 times  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| For Loop without the First Statement | let i = 0; for (; i < 10; i++) {  // code block to be executed 10 times  } |
| For Loop without the Second Statement | for (let i = 0; ; i++) {  // code block to be executed 10 times  if (i >= 10) {  break;  }  } |
| For Loop without the Third Statement | for (let i = 0; i < 10;) {  // code block to be executed 10 times  i++;  } |
| For Loop with a Condition Statement Only | const cars = ['BMW', 'Audi']; let i = 0; for (; cars[i];) { // cars[i] will return false when i = 2  // code block to be executed 2 times  i++;  } |  |
| For Loop without Any Statement (Avoid an Infinite Loop!) | let i = 0; for (; ;) {  // code block to be executed 10 times  if (i >= 10) {  break;  }  i++;  } |  |

## WHILE LOOP

|  |  |  |
| --- | --- | --- |
| While Loop | while (*condition*) {  // code block to be executed while the condition returns true } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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

## FUNCTION DECLARATION/EXPRESSION

|  |  |  |
| --- | --- | --- |
| Function Declaration | function funcName() {  // content of the function  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Input) C++ C# Java |
| Function Expression (Define a Function Inside an Expression) | const hello = function () {  const name = 'John';  return 'Hello ' + name; } |
| Arrow (Anonymous) Functions | const hello = () => {  const name = 'John';  return 'Hello ' + name; } |
| Arrow Functions with Only One Statement | const hello = () => 'Hello!';  const hello = name => 'Hello ' + name; const hello = (name, greeting) => 'Hello ' + name + greeting; |
| not hoisted |  |

## CALL/INVOKE A FUNCTION. FUNCTION PARAMETERS

|  |  |  |
| --- | --- | --- |
| Parameters and Arguments | function by2(num) { // num is parameter, behaves as variable between {}  return num \* 2; } by2(3); // 3 is argument | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Input) C++ C# Java |
| arguments.length | an array of all arguments |
| String Parameters | function greetingByName(firstName, lastName) {  return 'Your name is ' + firstName + lastName; } greetingByName('John', 'Doe'); // 'Your name is John Doe' |
| 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 HOISTING

|  |  |  |
| --- | --- | --- |
|  |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Input) C++ C# Java |
|  |  |

## FUNCTION CONTEXT. THIS OBJECT

|  |  |  |
| --- | --- | --- |
| Global Invoke without a Function | console.log(this); // {} in both strict and sloppy mode | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Global Invoke in a Function | function myFunction() {  return this; }  myFunction(); // Object [global] in node.js (undefined in strict mode), [object Window] in the browser |
| Global Invoke in an Arrow Function | (() => console.log(this))(); // {} in both strict and sloppy mode |
| Object Method | let myObj = {  name: 'Peter',  func() {  return this;  } };  myObj.func() // returns { name: 'Peter', func: *f* } |
| DOM Event (NOT in an Arrow Function!) | button.addEventListener('click', onClick);  function onClick() {  console.log(this); // <button>Click me</button>  } |  |
| 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 |  |
|  | function outer() {  const inner = () => console.log(this);  inner();  }  const obj = {  name: 'Peter',  func: outer  };  obj.func(); // Object {name: "Peter"} |  |
| This in Constructor Invocation | Refers to the newly created instance. |  |
| 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 Math.max.apply(null, [1, 2, 3]); // 3  let boundFunc = func.bind(myObj); // creates a new function 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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!' |
| Callback | a function passed into another function as an argument |  |
| 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 even after the parent function has closed. |  |
| IIFE (Immediately Invoked Function Expression) | (function () { let name = 'Peter' })(); // name cannot be used let result = (function () {  return name = 'Peter'; })() // result = 'Peter' |  |
| Using Closure and IIFE to Create a Counter | const add = (() => {  let counter = 0;  return () => ++counter;  })();  add(); // 1  add(); // 2  add(); // 3 |  |
| 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 |  |
|  | in objects are called methods, used to make objects: constructors |  |

# OUTPUT

|  |  |  |
| --- | --- | --- |
| Print Some Data to the Console | console.log('Hi there, John!'); // prints 'Hi there, John!'  console.log(35); // prints 35 | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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 in the Browser | alert('Hi there!');  window.alert('Hi there!'); |  |
| Ask the User for Confirmation in an Alert Window in the Browser | confirm('Are you sure you want to delete this?'); |  |
| Print Page | window.print(); |  |

# NUMBERS IN JAVASCRIPT

## BASIC MATH

|  |  |  |
| --- | --- | --- |
| Number Storage in JS | JS numbers are always 64-bit floating point, where the number (the fraction) is stored in bits 0 to 51, the exponent in bits 52 to 62, and the sign in bit 63. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicMath) C++ C# Java |
| 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 |
| 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'  47.44.toPrecision(2) // '47'; BUT (9.5).toPrecision(1) returns '1e+1' |  |
| Round Down | parseInt(47.98) // returns 47  Math.trunc(47.98) // returns 47  Math.floor(47.98) // returns 47  47.98 | 0 // 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'  47.445.toPrecision(4) // '47.45'; BUT (99.5).toPrecision(2) = '1.0e+2' |  |
| Check the Sing of a Number (Positive/Negative) | Math.sign(4) // returns 1  Math.sign(0) // returns 0  Math.sign(-4) // returns -1 |  |
| 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  5 \*\* 2 // returns 25 |  |
| Square Root | Math.sqrt(36) // returns 6 Math.sqrt(-1) // returns NaN |  |
| Cube Root | Math.cbrt(8) // 2 |  |
| 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' |  |
| Exponential Notation | (9.656).toExponential() // returns '9.656e+0' (not rounded)  (9.656).toExponential(2) // returns '9.66e+0'  (9.656).toExponential(4) // returns '9.6560e+0' |  |
| 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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' |
| Replace All Occurences | 'blue house, blue car'.replace(/blue/g, '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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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 // [A-Za-z\_] | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Match Any Non-Alphanumeric Character | \W // [^A-Za-z\_] |
| Match Any White-Space Character | \s // [ \t\n\r\0] |  |
| Match Any Non-White-Space Character | \S // [^ \t\n\r\0] |  |
| Match Any Digit | \d // [0-9] |  |
| Match Any Non-Digit Character | \D // [^0-9] |  |
| Match the Unicode Character Specified by the Hexadecimal Number (\uxxxx) | \u0061 // String.fromCharCode((0x0061).toString(10)) |  |
|  | \p{...} |  |

### QUANTIFIERS

|  |  |  |
| --- | --- | --- |
| Zero or One of | /a?/ // matches zero or one time 'a' | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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>>/ |  |

# SYMBOLS IN JAVASCRIPT

|  |  |  |
| --- | --- | --- |
|  |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#CombineStrings) C++ C# Java |
|  |  |

# ARRAY OBJECTS

## JAVASCRIPT ARRAY CHARACTERISTICS

|  |  |  |
| --- | --- | --- |
| Zero-based | const numbers = [1, 2];  numbers[0] // returns 1 | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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) | const cars = ['Audi', 2004, { model: 'BMW', year: 2010 }]; |
| Dynamic Length | const numbers = [1, 2]; numbers.length = 4; numbers // returns [1, 2, undefined, undefined] |
| NOT Guaranteed to Be Dense | const 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) | const cars = [];  const cars = new Array(); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create an Array Object | const cars = ['Audi', 'BMW'];  const cars = new Array('Audi', 'BMW'); |
| Destructuring Assignment | const [car1, car2] = cars; // creates two variables: car1 = 'Audi', car2 = 'BMW' |
| Destructuring with the Rest Operator | const [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  } |  |
| Create a Nested Array (Matrix) and Access Its Elements | const matrix = [[1, 2], [3, 4]];  matrix[0][1] // returns 2 |  |

## INSPECT AN ARRAY

|  |  |  |
| --- | --- | --- |
| Find the Number of Array Elements | cars.length // returns 2 | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| 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((v, i, a) => v < 6); // returns true |  |
| Check if Some Array Values Pass a Test | [2, 5, 4].some((v, i, a) => v > 6); // returns false |  |
| Find the Index of the First Array Element that Passes a Test | [2, 5, 4].findIndex((v, i, a) => v < 6); // returns 0 |  |

## EXTRACT ARRAY PARTS. FILTER

|  |  |  |
| --- | --- | --- |
| Extract Array Parts (Creates a New Array, Does Not Change the Original Array) | children.slice(1, 3); // returns elements 1 to 2: ['John', 'Mary'] children.slice(1); // returns elements 1 to the end | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Filter an Array (Value, Index, Array) – Must Be Saved in Another Variable | const 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 Function | [2, 5, 4].find((v, i, a) => v < 6) // returns 2  [21, 50, 43].find(x => x < 6) // returns undefined |  |

## CHANGE/ADD/REMOVE/SWAP ARRAY ELEMENTS

|  |  |  |
| --- | --- | --- |
| Change an Array Element | const cars = ['Audi', 'BMW'];  cars[0] = 'Opel';  cars // returns ['Opel', 'BMW'] | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Concatenate Arrays (and Strings): Always Returns a New Array | let boys = ['Peter', 'John']; let girls = ['Mary', 'Jill']; let children = boys.concat(girls, 'Ann'); // returns ['Peter', 'John', 'Mary', 'Jill', 'Ann'] |

## CONVERT A NESTED ARRAY TO AN OBJECT

|  |  |  |
| --- | --- | --- |
|  | const arr = [['name', 'John'], ['age', 23]];  const obj = arr.reduce((a, c) => Object.assign(a, { [c[0]]: c[1] }), {}); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

## 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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) Using a Compare Function | [40, 2, 179].sort((a, b) => 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((a, b) => b – a) // returns [179, 40, 2] |  |
| Sort an Array in Random Order | numbers.sort((a, b) => 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 | const cars = [  { type: 'Volvo', year: 2016 },  { type: 'Saab', year: 2010 },  { type: 'BMW', year: 2010 } ];  cars.sort(function (a, b) => 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a New Array by Performing a Function on Each Element (Does Not Change the Original Array) | const nums = [2, 5]; const numbersBy2 = nums.map((v, i, a) => v \* 2); // [4, 10] const 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. Does NOT Reduce the Original Array | [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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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 | const person = { name: 'John', age: 35 }; const anotherPerson = Object.assign({ weight: 80 }, person); // { weight: 80, name: 'John', age: 35 } const thirdPerson = { ...person, weight: 90 }); // { name: 'John', age: 35, weight: 90 } |  |
| 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 35 (true in an if statement) 'age' in person // returns true |  |

## 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | Object.getOwnPropertyNames(obj) |
| 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' JSON.stringify(person) // returns |
|  | 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 }); |  |
| Non-Writable (if Contains an Object, the Reference to the Object Is Non-Writable, the Object Itself Can Be Modified) |  |  |
| 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) | Object.seal(person); |  |
|  | Object.defineProperty(this, 'fullName', {  set: function(value) {  // set value + validation  },  get: function() {  // calculate and return value  }  }  ); |  |
|  | preventExtensions() |  |
| Example: Counter | // Define object  const obj = {counter:0};  // Define setters  Object.defineProperty(obj, "reset", {  get : function () {this.counter = 0;}  });  Object.defineProperty(obj, "increment", {  get : function () {this.counter++;}  });  Object.defineProperty(obj, "decrement", {  get : function () {this.counter--;}  });  Object.defineProperty(obj, "add", {  set : function (value) {this.counter += value;}  });  Object.defineProperty(obj, "subtract", {  set : function (i) {this.counter -= i;}  });  // Play with the counter:  obj.reset;  obj.add = 5;  obj.subtract = 1;  obj.increment;  obj.decrement; |  |

## 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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]] |  |

## PROTOTYPE- AND CLASS-BASED INHERITANCE

|  |  |  |
| --- | --- | --- |
| Prototype-Based Inheritance | JavaScript is a prototype-based language. The class keyword is introduced in ES20215, but it is syntactical sugar – the "classes" we simulate are just a function object.  Objects inherit directly from other objects through a prototype property. Prototype-based programming allows the creation of an object without first defining its class.  When it comes to inheritance, JavaScript only has one construct: objects. Each object has a private property which holds a link to another object called its prototype. That prototype object has a prototype of its own, and so on until an object is reached with NULL as its prototype. By definition, NULL has no prototype, and acts as the final link in this prototype chain.  Nearly all objects in JS are instances of Object which sits on the top of a prototype chain.  JS objects are dynamic "bags" of properties (referred to as own properties). They have a link to a prototype object. When trying to access a property of an object, the property will not only be sought on the object but on the prototype of the object, the prototype of the prototype, and so on until either a property with a matching name is found or the end of the prototype chain is reached. hasOwnProperty is the only thing in JS which deals with properties and does NOT traverse the prototype chain. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Assign Prototype to an Object | class Person { };  const woman = {}; Object.setPrototypeOf(woman, Person.prototype); // or: const woman = Object.create(Person.prototype); // or: const woman = new Person(); |  |
| Get Prototype of an Object | Object.getPrototypeOf(woman) // returns Person {} woman.\_\_proto\_\_ // returns Person {}, the prototype of woman |  |
| Get Prototype of a Function | Person.prototype // returns Person {}, the prototype to be assigned to all instances of objects created by the function when used as a constructor |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Class-Based Inheritance | A class defines a type which can be instantiated at runtime. A child of an ES6 class is another type definiton which extends the parent with new properties and methods, which in turn can be instantiated at runtime.  A class constructor creates an instance of the class. When invoked with the NEW keyword, it assigns its prototype as the prototype of the returned object. JS classes are primarily syntactical sugar over JS's existing prototype-based inheritance. |  |

## DECLARE A CLASS AND CREATE INSTANCES

|  |  |  |
| --- | --- | --- |
| Declare a Class (a Template for Creating Objects), NOT Hoisted | class Rectangle {  constructor(height, width) {  this.height = height;  this.width = width;  }  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare a Class Using a Constructor Function (Legacy) | function Rectangle(height, width) {  this.height = height;  this.width = width;  } |  |
| Add a Method to a Declared Class | Rectangle.prototype.calcArea = function() {  return this.height \* this.width;  }; |  |
| Create an Instance of a Class | const square = new Rectangle(3, 3); // automatically executes the Rectangle constructor and returns Rectangle { height: 3, width: 3 } |  |
| Create an Instance Step By Step | function Person(firstName, lastName) {  this.firstName = firstName;  this.lastName = lastName;  }  function newOperator(constructor, ...params) {  const result = {};  Object.setPrototypeOf(result, Person.prototype);  constructor.apply(result, params);  return result;  }  const square = newOperator(Person, 'Jane', 'Smith'); |  |
| Create a Static Method for a Class (Part of the Class, NOT of the Instances) | class Person {  constructor(name) {  this.name = name;  }  static sayHi(obj) {  console.log(`Hi, ${obj.name}`);  } }  const myPerson = new Person('Tim');  myPerson.sayHi({ name: 'John' }); // no?  Person.sayHi(myPerson); // prints 'Hi, Tim' |  |
| Check if an Object is an Instance of a Specified Class | square instanceof Rectangle // returns true square instanceof Object // returns true |  |
| Accessor Properties (Getter and Setter) | class Circle {  constructor(r) {  this.r = r;  }    get diameter() {  return this.r \* 2;  }  set diameter(value) {  if (value <= 0) {  throw new Error('Diameter must be positive');  }  this.r = value / 2;  } }  const myCircle = new Circle(3); myCircle.diameter // returns 6 myCircle.diameter = 10; myCircle.r // returns 5 |  |
| Add Properties to a Class Using Its Prototype | Circle.prototype.color = 'green'; myCircle.hasOwnProperty('color') // returns false myCircle.color // returns 'green' |  |
| Class Inheritance (Does NOT Create Copies, Inserts a Reference Instead) | class Person {  constructor(name) {  this.name = name;  }  sayHi() {  console.log(`${this.name} says hi!`);  }  }  class Employee extends Person {  constructor(name, salary) {  super(name);  this.salary = salary;  }    collectSalary() {  console.log(`${this.name} collected ${this.salary}`);  } }  const myEmployee = new Employee('Peter', 60000);  myEmployee // returns Person { name: 'Peter', salary: 60000 } myEmployee.collectSalary() // prints 'Peter collected 60000' myEmployee.sayHi() // prints 'Peter says hi!' |  |
| Overriding Methods and Properties |  |  |
| Private Properties (\_) |  |  |
| Class Inheritance (Legacy) | function Person(name) {  this.name = name; }  Person.prototype.sayHi = function() {  console.log(`${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!' |  |

## THE FOUR PILLARS OF OBJECT-ORIENTED PROGRAMMING

|  |  |  |
| --- | --- | --- |
| Abstraction |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Encapsulation |  |  |
| Inheritance |  |  |
| Polymorphism |  |  |

# 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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 |  |
| JS ISO Dates (Preferred) | newDate('2015-03-25') // returns 'Wed Mar 25 2015 00:00:00 GMT+0200 (Eastern European Standard Time)'  newDate('2015-03-25T12:00:00Z') // returns 'Wed Mar 25 2015 14:00:00 GMT+0200 (Eastern European Standard Time)': Z for UTC time  newDate('2015-03') // returns between February 28th and March 1st according to the time zone  newDate('2015') // returns between December 31st 2014 and January 1st 2015 according to the time zone |  |
| 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  newDate('March 25 2015')  newDate('MARCH, 25, 2015')  newDate('25 Mar 2015') |  |
| 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 | new Date.toString() // 'Thu Jun 10 2021 12:02:40 GMT+0300 (Eastern European Summer Time)'  newDate().toUTCString() // 'Thu, 10 Jun 2021 09:03:55 GMT'  new Date.toISOString() // '2021-06-10T09:07:08.295Z' | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Convert a Date to a String (without Time) | newDate().toDateString() // 'Thu Jun 10 2021' |
| Convert a Date to a Number (Milliseconds from January 1st, 1970) | Date.now() // 1427234400000 (milliseconds from Jan 01 1970 00:00 UTC) Date.parse('Mar 25 2015') // 1427234400000  Number(new Date('Mar 25 2015')) // 1427234400000  new Date('Mar 25 2015').getTime() // 1427234400000 |  |
| Compare Dates | new Date('Mar 25 2015') < new Date('Mar 30 2015') // true |  |

## ACCESS ONLY PART OF THE DATE

|  |  |  |
| --- | --- | --- |
| Get the Year from a Date Object | new Date('Mar 25 2015').getFullYear() // 2015 | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Get the Month from a Date Object (0 to 11) | new Date('Mar 25 2015').getMonth() // 2 (0 – 11) |
| 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()] // 'March' |  |
| Get the Day from a Date Object (1 to 31) | new Date('Mar 25 2015').getDate() // 25 (1 – 31) |  |
| Get the Hours from a Date Object (0 to 23) | new Date('Mar 25 2015').getHours() // 0 (0 – 23) |  |
| Get the Minutes from a Date Object (0 to 59) | new Date('Mar 25 2015').getMinutes() // 0 (0 – 59) |  |
| Get the Seconds from a Date Object (0 to 59) | new Date('Mar 25 2015').getSeconds() // 0 (0 – 59) |  |
| Get the Milliseconds from a Date Object (0 to 999) | new Date('Mar 25 2015').getMilliseconds() // 0 (0 – 999) |  |
| Get the Weekday from a Date Object (0 to 6) | new Date('Mar 25 2015').getDay() // 3 for Wednesday (0 – 6) |  |

## 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) // 'Wed Mar 25 2020 00:00:00 GMT+0200 (Eastern European Standard Time)'  new Date('Mar 25 2015').setFullYear(2020, 11, 3) // 'Thu Dec 03 2020 00:00:00 GMT+0200 (Eastern European Standard Time)' | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Change the Month in a Date Object (0 to 11) | new Date('Mar 25 2015').setMonth(11) // '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) // '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) // '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) // '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) // '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) // 'Wed Mar 25 2015 00:00:22 GMT+0200 (Eastern European Standard Time)' |  |

# RECEIVE DATA FROM A WEB SERVER

## JSON (JAVASCRIPT OBJECT NOTATION)

|  |  |  |
| --- | --- | --- |
|  | A language independent text format for storing and transporting data.  Data is in name/value pairs separated by commas, curly braces hold objects, square brackets hold arrays. Strings must be written in double quotes!  Used for sending/receiving data from a web server or saving/retrieving data from local storage.  Values can be: string, number, object, array, boolean, null. (whereas in JS object values can also be: function, date, undefined).  File type for JSON: .json  MIME type for JSON text: "application/json" | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | JSON.parse() // convert data from JSON to JS object; second parameter is called reviver:  const text = '{"name":"John", "birth":"1986-12-14", "city":"New York"}';  const obj = JSON.parse(text, function (key, value) {  if (key == "birth") {  return new Date(value);  } else {  return value;  }  });  // parsing functions (avoid it!):  const text = '{"name":"John", "age":"function () {return 30;}", "city":"New York"}';  const obj = JSON.parse(text);  obj.age = eval("(" + obj.age + ")"); |
|  | JSON.stringify() // convert data (object/array) to JSON  Converts dates into strings.  Removes any functions (unless previously converted to strings) |  |
|  | {"employees":[  { "firstName":"John", "lastName":"Doe" },  { "firstName":"Anna", "lastName":"Smith" },  { "firstName":"Peter", "lastName":"Jones" }  ]} |  |

## XML

|  |  |  |
| --- | --- | --- |
|  | A markup language, cannot use arrays.  <employees>  <employee>  <firstName>John</firstName> <lastName>Doe</lastName>  </employee>  <employee>  <firstName>Anna</firstName> <lastName>Smith</lastName>  </employee>  <employee>  <firstName>Peter</firstName> <lastName>Jones</lastName>  </employee>  </employees> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |

# HTML DOM (DOCUMENT OBJECT MODEL)

## MAIN CONCEPTS

|  |  |  |
| --- | --- | --- |
| Document Object Model | An interface that treats an HTML/XML document as a tree structure. A standard for how to get, chande, add and delete HTML elements. In the DOM, all HTML elements are defined as objects. The HTML DOM can be accessed with JavaScript, as well as with other programming languages. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| DOM Node | Every element of a DOM tree. According to the node type, the Node.nodeType property can have one of the following values: Node.ELEMENT\_NODE, Node.ATTRIBUTE\_NODE, Node.TEXT\_NODE, Node.COMMENT\_NODE, Node.DOCUMENT\_NODE, etc.  In a node tree, the top node is called the root (node); ex. <html>. Every node has exactly one parent, except the root, which has no parent. A node can have a number of children. Siblings are nodes with the same parent. |
| DOM Element | Every node of the type Node.ELEMENT\_NODE.  Variables holding HTML elements are live: when their content is modified, the DOM is updated, when inserted somewhere in the DOM, the original is moved. |  |
| Node List | A collection of DOM nodes of any type. Accessed using using childNodes (then it is a live, automatically updated collection) or querySelectorAll() (then it is static). Can be indexed and iterated, has length. Items can only be accessed by their index. |  |
| HTML Collection | An array-like list of only element nodes and is live. Has an extra namedItem method. Accessed using children, getElementsByClassName(), getElementsByTagName(). Can be indexed and iterated, has length. Items can be accessed by their name, id or index. |  |

## IN THE BROWSER

|  |  |  |
| --- | --- | --- |
| From the Folder | Open the 'index.html' file in the browser. Refresh manually (F5). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| From the Terminal (Using a Local Server) - plays the role of a static web server: only returns the files from our web app | lite-server // automatically starts and updates 'index.html' from the same folder, runs on http://localhost:3000; resolves the CORS problem |
|  | Start web service (our REST service): server.js, so that the app can make HTTP requests; dynamic server? |  |

## JAVASCRIPT IN HTML

|  |  |  |
| --- | --- | --- |
| Inline Script in <head> or <body> | <script>  (() => document.getElementById('demo').innerHTML = 'Surprise!')();  </script> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| External File in <head> or <body> | <script *src*="/js/myScript.js"></script> |
| Links to JS Documents (Scripts Declarations) | <script src="main.js"></script> |  |
| Link to JS Documents to Be Executed after the Page Loading | <script src="main.js"></script> <!-- at the end of <body> -->  <script src="main.js" defer="true"></script> <!-- at any place --> |  |

## THE DOCUMENT OBJECT

|  |  |  |
| --- | --- | --- |
|  | document.cookie // display the cookies associated with this document | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | document.domain // the domain of the server that loaded the document.lastModified // date and time the document was last modified  document.title // title of the document  document.URL // full URL of the document |
| Replace the Content of a Document | document.open('text/html', 'replace');  document.write('<h2>Learning about HTML DOM is fun!</h2>');  document.close(); |  |
| Open a New Window and Add Some Content | var w = window.open();  w.document.open();  w.document.write("<h2>Hello World!</h2>");  w.document.close(); |  |

## ACCESS DOM NODES (NULL IF NOT FOUND)

|  |  |  |
| --- | --- | --- |
|  | always start with accessing the document object, it represents the web page | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| By ID | document.getElementById('title') // element with ID 'title' or null document.querySelector('#title') // returns the element with ID 'title' |
| By Class Name | document.querySelectorAll('.small) // returns a static node list of all elements with the class 'small' document.getElementsByClassName('small') // returns a live HTML collection of all elements with the class 'small'  document.querySelector('.small) // returns the first element with the class 'small' |
| By Tag Name | document.querySelectorAll('p') // returns a node list of all <p> elements document.getElementsByTagName('p') // returns a live HTML collection of all <p> elements  document.querySelector('p') // returns the first <p> element |  |
| By Tag and ID | document.forms['first'] // the form with ID 'first' - HTMLFormElement (HTMLCollection?); form.elements[0] - the first input element |  |
| By Tag and Class | document.querySelectorAll('article.list') // returns a static node list of all <article> elements with the class 'list' |  |
| By Name | document.getElementsByName('login') // returns a live node list of all elements with the name 'login'  document.querySelectorAll('input[name="login"]') // returns a static node list of all <input> elements with name 'login' |  |
| By Parent Node | document.querySelectorAll('div p') // returns a static node list of all <p> elements inside <div> elements document.querySelectorAll('#content div') // returns a static node list of all <div> elements inside the element with ID 'content'  element.children // returns a live HTML collection of all the child elements of the element element.childNodes // returns a static node list of all children nodes of the element  firstChild (childNodes[0]), lastChild  document.forms['form1']['input1'] // the input element with the name 'input1' in the form with ID 'form1' |  |
| By Parent Node in a Numbered Order | const thirdLi = document.querySelector('ul').querySelector('li')[2]; // third <li> from the first <ul> const thirdLi = document.querySelector('ul li:nth-child(3)'); // third <li> from the first <ul> |  |
| By Child Node | element.parentElement element.parentNode |  |
|  | nextSibling, previousSibling |  |
| All <a> Elements with a "name" Attribute | document.anchors  document.links? |  |
| All <form> Elements | document.forms |  |
| All <img> Elements | document.images |  |
| All <script> Elements | document.scripts |  |
| <body> | document.body |  |
| <head> | document.head |  |
| <title> | document.title |  |
| <html> | document.documentElement |  |
|  | .dataset // obtain DOMStringMap of custom data attributes (case-insensitive) |  |
|  |  |  |

## GET NUMBER OF ELEMENTS

|  |  |  |
| --- | --- | --- |
|  | document.getElementsByClassName('small').length | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | document.anchors.length |

## ACCSESS THE ATTRIBUTES OF A DOM NODE

|  |  |  |
| --- | --- | --- |
| Get Text Content | const element = document.getElementById('demo'); element.textContent // if the element has children, returns all text concatenated | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Get Text Content + Children | const element = document.getElementById('demo'); element.innerHTML // if the element has children, returns the HTML code as string  element.firstChild.nodeValue, element.childNodes[0].nodeValue |
| Get Input Content | element.value // returns the element value as string |
| Get all Class Values of the Node | element.classList // returns a read-only collection |  |
| Get the Value of a Specified Attribute | element.getAttribute('type'); |  |
| Check if the Node Has a Specified Attribute | element.hasAttribute('type'); // returns true/false |  |
| Get the ID of the Form That Contains a Button | listEl.form.id |  |
| Get the ID of the Form a Button Belongs to | button.form.id |  |
| Get the Index of the Selected Option | listEl.selectedIndex |  |
| Get the Value of the Selecter Option from a Dropdown List | listEl.options[listEl.selectedIndex].text; |  |
|  |  |  |
| Get the Number of Elements in a Form | formEl.length |  |
|  | nodeName: read-only, uppercase, the same as the tag name, of a text node: #text  nodeType |  |

## CHANGE THE VALUE OF A DOM NODE ATTRIBUTE

|  |  |  |
| --- | --- | --- |
| Change Text Content | element.textContent = 'Some new content'; | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | innerHTML? |
| Change the Style of an HTML Element | element.style.display = 'none'; |  |
| Change the Source of an Image | imageEl.src = 'myPicture.jpg'; |  |
| Change the Color of a Text | element.style.color = 'blue'; |  |
| Hide/Show an Element | element.style.display = 'none'/'inline-block'  element.style.visibility = 'hidden'/visible; |  |
| Disable a Button (Dropdown List) | button.disabled = true; |  |
| Turn a Dropdown List into a Multiline List | listEl.size = 4; |  |
| Select Multiple Options in a Dropdown List | listEl.multiple = true; // use Shift or Ctrl |  |
| Change the Text of the Selected Option | listEl.options[listEl.selectedIndex].text = 'Melon'; |  |
| Remove Options from a Dropdown Menu | listEl.remove(listEl.selectedIndex); |  |
| Reset a Form | formEl.reset(); |  |
| Submit a Form | formEl.submit(); |  |

## 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

## MODIFY DOM NODES

|  |  |  |
| --- | --- | --- |
| Add Text | element.textContent = 'some text'; // text will be escaped element.innerHTML = 'some text'; // text will be parsed and turned into HTML elements (XSS attacks) | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Hide a Node | element.style.display = 'none'; |
| Reveal a Hidden Node | element.style.display = ''; // or 'block', 'inline-block' |
| Change Background Color | element.style.background = 'teal'; |  |
| Change the Font Size | element.style.fontSize = '35px'; |  |
| Disable a Button | button.setAttribute('disabled', 'true'); |  |
| Add Class Value | liElement.classList.add('myClass'); |  |
| Remove Class Value | liElement.classList.remove('myClass'); |  |
| Set Value to a Specified Attribute | element.setAttribute('type', 'text'); |  |
| Remove Value of a Specified Attribute | element.removeAttribute('spellCheck'); |  |
| Custom Data Attributes | dataset DOMStringMap |  |

## CREATE/DELETE DOM ELEMENTS

|  |  |  |
| --- | --- | --- |
| Create DOM Elements | const paragraph = document.createElement('p'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create Text Node | const newNode = document.createTextNode('some text'); |
| Create Document Fragment Node | const fragment = document.createDocumentFragment(); |
| Copy/Clone DOM Elements | const newParagraph = paragraph.cloneNode(true); // 'true' for a deep copy |
| Add a New Child (at the End) | document.body.appendChild(paragraph);  parent.insertBefore(newEl, referenceEl); |  |
| Add a New Child (at the Beginning) | document.body.prepend(paragraph); |  |
| Delete DOM Elements | liElement.remove(); |  |
| Delete DOM Elements from Parent Element | let ulElement = document.querySelector('ul'); let liElement = document.querySelector('li'); ulElement.removeChild(liElement); |  |
|  |  |  |
| Replace one Element with Another | replaceChild  liElement.replaceWith(newElement); |  |
| Insert Element before Selected Node | liElement.before(newElement); |  |
| Insert Element after Selected Node | liElement.after(newElement); |  |

## EVENTS

|  |  |  |
| --- | --- | --- |
| Mouse Events | click, mouseover, mouseout, mouseup, mousedown | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Touch Events | touchstart, touchend, touchmove, touchcancel |
| DOM/UI Events | load, unload, resize, dragstart/drop |
| Keyboard Events | keydown, keypress, keyup |
| Focus Events | focus, blur |
| Form Events | input, change, submit, reset |
|  | element.onclick = function () {...} |
| Attach Event Listener to an Element | element.addEventListener('click', (event) => console.log(event)); // addEventListener(*type of event*, *event handler function, a boolean value specifying whether to use event bubbling or event capturing*) |
|  | document.getElementById("myBtn").onclick = displayDate; |
| Attach Event Listener to an Element in the HTML | <button *onclick*="console.log('click')">Click me!</button> |
| Attach Event Listener to an Element in the HTML with a <script> Tag | <button *onclick*="sayHi()">Click me!</button>  <script>  function sayHi() {  console.log('Hi!');  }  </script> |
| Attach Event Listener to an Element in an External File | <button *id*="btnGreeting">Click me!</button>  <script *src*="/myScript.js"></script>  // in myScript.js:  const btn = document.getElementById('btnGreeting');  element.addEventListener('click', () => console.log('Hi!')); |
| Access Event Target | event.target // returns element that triggered event event.currentTarget // returns element that has the event listener |
|  | event.offsetX event.target.clientWidth |
| Stop Event Propagation | event.stopPropagation(); // used to stop an event handler for the same type of event attached to a parent node |
| Stop Browser's Default Behavior | event.preventDefault(); // in <a> to stay on the same page, in <form> to stop submitting an HTTP request and refreshing |
| Remove Event Listener | element.addEventListener('click', evHandler); |
|  | removeEventListener() // when event listener was attached using addEventListener |
| Event Propagation | A way of defining the element order when an event occurs. If you have a <p> element inside a <div> element, and the user clicks on the <p> element, which element's "click" event should be handled first?  In bubbling the inner most element's event is handled first and then the outer: the <p> element's click event is handled first, then the <div> element's click event.  In capturing the outer most element's event is handled first and then the inner: the <div> element's click event will be handled first, then the <p> element's click event.  With the addEventListener() method you can specify the propagation type by using the "useCapture" parameter:  addEventListener(event, function, useCapture);  The default value is false, which will use the bubbling propagation, when the value is set to true, the event uses the capturing propagation. |  |

## VALIDATE INPUT

|  |  |  |
| --- | --- | --- |
|  | input.checkValidity() // returns false if input value is invalid according to the HTML input attributes (missing value in a required field, number where type is specified as text, 100 where max is set to 50, etc.)  if (!input.checkValidity()) {  console.log(input.validationMessage);  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | if (input.validity.rangeOverflow) {  console.log('Value too big');  } |
| validity properties (true or false) | customError, patternMismatch, rangeOverflow, rangeUnderflow, stepMismatch, tooLong, typeMismatch, valueMissing, valid |

## DOM ANIMATIONS

|  |  |  |
| --- | --- | --- |
|  | function myMove() {  let id = null;  const elem = document.getElementById("animate");  let pos = 0;  clearInterval(id);  id = setInterval(frame, 5);  function frame() {  if (pos == 350) {  clearInterval(id);  } else {  pos++;  elem.style.top = pos + 'px';  elem.style.left = pos + 'px';  }  }  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

## GOOGLE APIS

|  |  |  |
| --- | --- | --- |
| Google Maps |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Google Fonts |  |
| Google Charts |  |  |

# JQUERY VS JAVASCRIPT

|  |  |  |
| --- | --- | --- |
| jQuery | A JavaScript library used in the past to simplify HTML DOM manipulation, event handling, animations and AJAX. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | const element = $('#id01'); // document.getElementById('id01')  const elements = $('p'); // document.querySelectorAll('p')  element.text('Hello Sweden!'); // element.textContent = 'Hello Sweden!';  element.html('<p>Hi</>'); // element.innerHTML = '<p>Hi</p>'  element.hide(); // element.style.display = 'none'; |

# JS BROWSER OBJECT MODEL (BOM)

## MAIN CONCEPTS

|  |  |  |
| --- | --- | --- |
|  | Allows JS to "talk to" the browser. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| The Window Object | Represents the browser window. Global variables are properties of the window object, global functions are methods of the window object. |
| Window Size | window.innerHeight, window.innerLength // in pixels; NOT including toolbars and scrollbars |  |
| Window Methods | window.open() // opens a new window  window.close() // closes the current window  window.moveTo() // move the current window  window.resizeTo() // resize the current window |  |
| (Window) Screen | screen.width // the width of the visitor's screen in pixels  screen.height // the height of the visitor's screen in pixels  screen.availWidth // the width of the visitor's screen in pixels minus interface features like Windows Taskbars  screen.availHeight // the height of the visitor's screen in pixels minus interface features like Windows Taskbars  screen.colorDepth // the number of bits used to display one color  screen.pixelDepth // the pixel depth of the screen, number; for modern computers, color depth and pixel depth are equal |  |
| (Window) Location: can be used to get the current page address (URL) and to redirect the browser to a new page | window.location.href // the href (URL) of the current page: https://www.w3schools.com/js/js\_window\_location.asp  window.location.hostname // the domain name of the web host: www.w3schools.com  window.location.pathname // the path and filename of the current page: /js/js\_window\_location.asp  window.location.protocol // the web protocol used (http: or https:): https:  window.location.port // the number of the internet host port of the current page: If the port number is default (80 for http and 443 for https), most browsers will display 0 or nothing  window.location.assign("https://www.w3schools.com") // loads a new document |  |
| (Window) History | window.history // contains the browser history  history.back() // same as clicking back in the browser  history.forward() // same as clicking forward in the browser |  |
| Web Worker | function startWorker() {  if(typeof(w) == "undefined") {  w = new Worker("demo\_workers.js"); // the worker is in an external js file  }  w.onmessage = function(event) {  document.getElementById("result").innerHTML = event.data;  };  }  function stopWorker() {  w.terminate();  w = undefined;  }  worker:  let i = 0;  function timedCount() {  i ++;  postMessage(i);  setTimeout("timedCount()",500);  }  timedCount(); |  |
| (Window) Navigator: contains information about the visitor's browser. this info can often be misleading and should not be used to detect browser versions | navigator.appName // the application name of the browser ("Netscape" is the application name for IE11, Chrome, Firefox, and Safari)  navigator.appCodeName // the application code name of the browser ("Mozilla" is the application code name for both Chrome, Firefox, IE, Safari, and Opera)  navigator.platform // the browser platform (operating system): Win32  navigator.cookiesEnabled // returns true if cookies are enabled, otherwise false  navigator.product // the product name of the browser engine (Most browsers returns "Gecko" as product name!!!)  navigator.appVersion // version information about the browser (5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36)  navigator.userAgent // the user-agent header sent by the browser to the server (Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.114 Safari/537.36)  navigator.language // the browser's language (bg)  navigator.onLine // true of the browser is online  navigator.javaEnabled() // true if Java is enabled |  |
| JS Popup Boxes  To display line breaks inside a popup box, use '\n' | (window.)alert() // often used to make sure information comes through to the user. When an alert box pops up, the user will have to click "OK" to proceed.  (window.)confirm() // often used to make the user verify or accept something. When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed. If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.  (window.)prompt() // used to make the user input a value before entering a page. When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value. If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null. |  |
| Timing Events: specified time intervals at which some code is executed | myVar = setTimeout(function, milliseconds); // executes a function after waiting a specified number of milliseconds  clearTimeout(myVar) // uses the variable returned from setTimeout and stops the execution of the function specified in it  setInterval(function, milliseconds) // repeats the execution of the function every [specified number] milliseconds  clearInterval(myVar) // uses the variable returned from setInterval and stops the execution of the function specified in it |  |
| JS Cookies: data stored in small text files on the computer | When a web server has sent a web page to a browser, the connection is shut down, and the server forgets everything about the user.  Cookies were invented to solve the problem "how to remember information about the user": When a user visits a web page, his/her name can be stored in a cookie. Next time the user visits the page, the cookie "remembers" his/her name.  Cookies are saved in name-value pairs. When a browser requests a web page from a server, cookies belonging to the page are added to the request. This way the server gets the necessary data to "remember" information about users.  document.cookie = 'username=JohnDoe'; // create a cookie  document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC"; // add expiry date; by default, the cookie is deleted when the browser is closed  document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/"; // tell the browser what path the cookie belongs to; by default: current page  read a cookie:  document.cookie will return all cookies in one string much like: cookie1=value; cookie2=value; cookie3=value;  change a cookie:  document.cookie = "username=John Smith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/"; (the old cookie is overwritten)  delete a cookie:  document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;"; // just set the expiry date to a past date; no cookie value needed, but path! |  |
|  | function setCookie(cname, cvalue, exdays) {  const d = new Date();  d.setTime(d.getTime() + (exdays \* 24 \* 60 \* 60 \* 1000));  let expires = "expires="+d.toUTCString();  document.cookie = cname + "=" + cvalue + ";" + expires + ";path=/";  }  function getCookie(cname) {  let name = cname + "=";  let ca = document.cookie.split(';');  for(let i = 0; i < ca.length; i++) {  let c = ca[i];  while (c.charAt(0) == ' ') {  c = c.substring(1);  }  if (c.indexOf(name) == 0) {  return c.substring(name.length, c.length);  }  }  return "";  }  function checkCookie() {  let user = getCookie("username");  if (user != "") {  alert("Welcome again " + user);  } else {  user = prompt("Please enter your name:", "");  if (user != "" && user != null) {  setCookie("username", user, 365);  }  }  } |  |
|  |  |  |

# WEB API

## MAIN CONCEPTS

|  |  |  |
| --- | --- | --- |
| About | A Web API is an application programming interface for the Web. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Browser API | Browsers have built-in Web APIs: Geolocation API  Third-party APIs: to use them you need to download the code from the Web (YouTupe API, Facebook API) |
| Web Forms API | Constraint Validation DOM Methods:  checkValidity() // true if an element contains valid data  setCustomValidity() // sets the validationMessage property of an input element  if (!inpObj.checkValidity()) {  console.log(inpObj.validationMessage);  }  Constraint Validation DOM Properties:  validity // contains boolean properties related to the validity of an input element: customError (true if a custom validity message is set), patternMismatch (true if an element value does not match its pattern attribute), rangeOverflow (true if value > its max attribute), rangeUnderflow (true if value < its min attribute), stepMismatch (true if an element's value is invalid per its step attribute), tooLong (true if an element's value exceeds its maxLength attribute), typeMismatch (true if an element's value is invalid per its type attribute), valueMismatch (true if an element (with a required attribute) has no value), valid (true if an element's value is valid)  validationMessage // the message a browser will display when the validity is false  willValidate // indicates if an input element will be validated  if (document.getElementById("id1").validity.rangeOverflow) {  text = "Value too large";  } else {  text = "Input OK";  } |  |
| Web History API: easy methods to access the window.history object (the URLs visited by the user) | history.back() // loads the previous URL in the window.history list  history.forward() // loads the next URL in the history list  history.go(-2) // loads a specific URL from the history list (back 2 pages)  history.length // returns the number of URLs in the history |  |
| Web Storage API: a simple syntax for storing and retrieving data in the browser | The localStorage object provides access to a local storage for a particular web site. It allows you to store, read, add, modify, and delete data items for that domain. Stores data with no expiration date.  localStorage.setItem('name', 'John'); // stores a data item in a storage (name, value)  localStorage.getItem('name'); // retrieves a data item from the storage (name)  The sessionStorage object stores data for one session. The data is deleted when the browser is closed.  sessionStorage.setItem('name', 'John');  sessionStorage.getItem('name')  Storage Object Properties:  key(n) // the name of the nth key in the storage  length // the number of data items in the storage object  getItem(keyname) // the value of a specified key name  setItem(keyname, value) // adds key to the storage or update if existing  removeItem(keyname) // removes key from storage  clear() // empty all key out of the storage |  |
| Web Geolocation API | The HTML Geolocation API is used to get the geographical position of a user.  Since this can compromise privacy, the position is not available unless the user approves it.  <script>  const x = document.getElementById("demo");  function getLocation() {  if (navigator.geolocation) {  navigator.geolocation.getCurrentPosition(showPosition);  } else {  x.innerHTML = "Geolocation is not supported by this browser.";  }  }  function showPosition(position) {  x.innerHTML = "Latitude: " + position.coords.latitude +  "<br>Longitude: " + position.coords.longitude;  }  </script>  // If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)  // The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location  function showError(error) {  switch(error.code) {  case error.PERMISSION\_DENIED:  x.innerHTML = "User denied the request for Geolocation."  break;  case error.POSITION\_UNAVAILABLE:  x.innerHTML = "Location information is unavailable."  break;  case error.TIMEOUT:  x.innerHTML = "The request to get user location timed out."  break;  case error.UNKNOWN\_ERROR:  x.innerHTML = "An unknown error occurred."  break;  }  }  // To display the result in a map, you need access to a map service, like Google Maps.  function showPosition(position) {  let latlon = position.coords.latitude + "," + position.coords.longitude;  let img\_url = "https://maps.googleapis.com/maps/api/staticmap?center=  "+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR\_KEY";  document.getElementById("mapholder").innerHTML = "<img src='"+img\_url+"'>";  } |  |
| Geolocation Object: properties and methods | coords: latitude, longitude, accuracy, altitude, altitudeAccuracy, heading, speed  timestamp  watchPosition() // continuously returns updated position of the user  clearWatch() // stops watchPosition()  <script>  const x = document.getElementById("demo");  function getLocation() {  if (navigator.geolocation) {  navigator.geolocation.watchPosition(showPosition);  } else {  x.innerHTML = "Geolocation is not supported by this browser.";  }  }  function showPosition(position) {  x.innerHTML = "Latitude: " + position.coords.latitude +  "<br>Longitude: " + position.coords.longitude;  }  </script> |  |

# ERRORS IN JS

## THROWING ERRORS (EXCEPTIONS)

|  |  |  |
| --- | --- | --- |
| General Error | throw new Error('Invalid state'); // { name: '...', message: '...' } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Range Error | throw new RangeError('Invalid index'); // number not in the allowed range |
| Type Error | throw new TypeError('String expected'); // unexpected value type |
| Reference Error | throw new ReferenceError('Missing age'); // non-declared variable |  |
| URI Error | throw new URIError('Can't decode %'); // illegal characters in decodeURI() |  |

## TESTING THE CODE

|  |  |  |
| --- | --- | --- |
| Test a Block of Code for Errors | try {  new Array(-1); } catch (err) {  console.log(err); // 'RangeError: Invalid array length'  console.log(err.message); // 'Invalid array length'  console.log(err.name); // 'RangeError' } | try |
| Validate Input and Clear Input Field | try {  if (input == '') throw 'is empty';  if (isNaN(input)') throw 'is not a number';  if (Number(input) > 10) throw 'is too high'; } catch (err) {  alert(`Input ${err}`); // 'Input is empty' } finally {  document.getElementById('input').value = ''; // regardless of try/catch  } |
| Write an Error Message to the Console | console.log(err.message); // displays the message onto the console  console.error(err.message); // displays a red message onto the console |
| Test Asynchronous Code | .then().catch()  async function myF() {  try {  } catch (err) {    }  } |
| Unit Testing |  |  |
| The Debugger Keyword | debugger; |  |

# MODULES IN JS

## TYPES OF MODULES

|  |  |  |
| --- | --- | --- |
| Local Modules (Created Locally) | const muModule = require('./myModule'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Core Modules (Node.js Built-in Modules) | const fs = require('fs'); |
| Third-Party Modules (Need Installation from NPM) | npm install express --save-exact // installed from Node Package Manager (NPM)  const express = require('express'); |

## EXPORT AND IMPORT MODULES

|  |  |  |
| --- | --- | --- |
| Export Modules | function sum(a, b) {  return a + b; }  module.exports = sum; | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Export Several Objects (Functions) | function sum(a, b) {  return a + b; }  function mult(a, b) {  return a \* b; }  module.exports = {  sum,  mult }; |
| Import Modules | const sum = require('./myModule'); // executes all code from myModule.js const { mult } = require('./myModule');  sum(4, 6); // returns 10 mult(4, 6); // returns 24 |
| Export Modules Using ES6 Syntax (HTML <script type="module">, local server) | export function sum(a, b) {  return a + b; }  export {  sum,  mult }; |  |
| Import Modules Using ES6 Syntax | import { sum } from './myModule.js'; // does not execute all code from myModule.js, only loads the function sum import { sum, mult } from './myModule.js'; import \* as calculations from './myModule.js'; |  |

## MODULE WRAPPER FUNCTION

|  |  |  |
| --- | --- | --- |
| Parameters | exports, require, module, \_\_filename, \_\_dirname // act as variables global to the module | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

# 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Check if It Worked | mocha --version |  |
| Set Node Path for Current Session | set NODE\_PATH=%AppData%\npm\node\_modules  + restart IDE |  |
| Set Node Path for Any Future Sessions | setx NODE\_PATH %AppData%\npm\node\_modules  + restart IDE |  |

## WRITE AND RUN TESTS

|  |  |  |
| --- | --- | --- |
| Test File Names | myModule.test.js // tests for file myModule.js | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Load Chai Library | const { expect } = require('chai');  const { assert } = require('chai'); |
| Load File to Be Tested | const { mult } = require('./myModule'); |  |
| Test Code | describe('Mult function', () => {  it('works', () => {  expect(mult(2, 3)).to.equal(6);  });  ); |  |
|  | beforeEach() |  |
| Compare Arrays and Objects | deepEqual |  |
| Run Test (in the Terminal) | mocha myModule.test.js // returns Mult function, works, 1 passing |  |

# WEB APPLICATIONS: DESIGN MODELS

|  |  |  |
| --- | --- | --- |
| MVC (Model-View-Controller) |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| DDD (Domain Driven Design) |  |  |

# WEB APPLICATIONS: FRONT END

|  |  |  |
| --- | --- | --- |
|  |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |  |

# HTTP AND REST SERVICES

## HTTP REQUESTS / Methods

|  |  |  |
| --- | --- | --- |
| HTTP | Used to make CRUD requests to restful APIs |  |
| GET Request to Retrieve a Resource | GET /rest/v2/name/Bulgaria HTTP/1.1 // HTTP request line  Host: restcountries.eu  Accept: \*/\*  ... // other HTTP headers  <CRLF> // the request body is empty | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| POST Request to Create/Store a Resource | POST /jsonstore/phonebook/phonebook/ HTTP/1.1  Host: localhost:3030  Content-Type: application/json  <CRLF> // an empty line  {  "name": "John",  "number": "123456789"  }  <CRLF> |  |
| PUT | update a resource |  |
| DELETE | delete a resource |  |
| PATCH | update a resource partially |  |
| HEAD | retrieve the resource's headers |  |

## HTTP RESPONSES

|  |  |  |
| --- | --- | --- |
| GET Request to Retrieve a Resource | HTTP/1.1 200 OK // HTTP response status line  Date: Friday, 11 November 2016 16:09:18 GMT+2  Server: Apache/2.2.14 (Linux)  Content-Type: html/text  <CRLF>  <html>  <head><title>Test</title></head>  <body>Test HTML file.</body>  </html> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | http.cat |  |
| POST Request to Create/Store a Resource |  |  |

## RESTFUL SERVICE/API

|  |  |  |
| --- | --- | --- |
| REST (REpresentational State Transfer) | A client-server architecture for building web applications. Client-server communication over HTTP.  Stateless (in practice - not)  We execute REST requests using HTTP.  Send HTTP requests to endpoints to access the different resources (collections?) | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| RESTful API | An Application Programming Interface (on the server?) that follows the REST principles.  API: access to an application that developers can include into (use in) their code. Something that developers consume/use. Some data that developers can access and modify.  = backend? |  |
| Accessing a Restful Ressource (Service?) | GET https://some-service.org/api/posts  GET https://some-service.org/api/posts/17  POST https://some-service.org/api/posts |  |

# AJAX

## ASYNCHRONOUS PROGRAMMING

|  |  |  |
| --- | --- | --- |
|  | Sends the tasks which are not to be executed immediately (ex. callbacks in setTimeout(), addEventListener()) to the (event) queue. When done with synchronous code in the stack, goes to the event queue. That's why the setTimeout callback is executed after everything else even when we set it to be executed after 0 milliseconds. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | The main thread includes the stack and the queue. The background (worker) threads is where the requests (to the server) are executed (after being removed from the stack in order not to block the browser), when ready they (the callbacks in them!) are sent to the queue.  Our JS code remains single-threaded, we just use the fact that the browser (Node.js) doesn't have such restrictions. |
| AJAX | misleading name, no more xml, json instead |

## PROMISES

|  |  |  |
| --- | --- | --- |
| Producing Code | Code that can take some time. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Consuming Code | Code that must wait for producing code. |
| Promise | Contains the producing code and calls to the consuming code. The background thread gives us a promise that it will do something. |
| Promise Object Properties (Not Accessible) | state: pending, fulfilled, rejected  result: undefined (while pending), value (when fulfilled), error object (when rejected) |  |
| Example | const promise = new Promise((resolve, reject) => {  // producing code  resolve(); // when successful  reject(); // when error  });  // consuming code  promise.then(  (value) => { /\* code if successful \*/},  (error) => { /\* code if some error \*/},  ); |  |

## SET TIME OUT

|  |  |  |
| --- | --- | --- |
|  | console.log('Hello'); // executed first  setTimeout(function () {  console.log('Hi!'); }, 2000); // executed third; if 0, once again after everything else  console.log('Hello again'); // executed second | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | setTimeout(() => console.log('Hi!'), 3000); // executed three minutes after the page loads |  |
|  | requestAnimationFrame() - better than setTimeout |  |
| Using a Promise | const promise = new Promise((resolve, reject) => {  setTimeout(() => resolve('I love you!'), 3000);  });  promise.then((value) => console.log(value)); |  |
| Create Ticking Timer in an HTML Element | setInterval(() => {  const d = new Date();  element.innerHTML = `${d.getHours()}:${d.getMinutes()}:${d.getSeconds()}`;  }, 1000); |  |

## XHR OBJECTS (XML HTTP REQUEST)

|  |  |  |
| --- | --- | --- |
| retro, not done this way | const httpRequest = new XMLHttpRequest(); // an object which allows us to execute asynchronous code and not block the browser  httpRequest.addEventListener('readystatechange', () => {  if (httpRequest.readyState == 4 && httpRequest.status == 200) {  console.log(httpRequest.responseText);  }  });  httpRequest.open('GET');  httpRequest.send(); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| retro, not done this way | const httpRequest = new XMLHttpRequest(); // an object which allows us to execute asynchronous code and not block the browser  httpRequest.addEventListener('readystatechange', () => {  switch (httpRequest.readyState) {  case 0:  console.log('...');  break;  case 4:  if (httpRequest.status == 200) {  console.log(httpRequest.responseText);  }  break;  });  httpRequest.open('GET');  httpRequest.send(); |  |
| Waiting for a File (Callback) | function getFile(myCallback) => {  const req = new XMLHttpRequest();  req.open('GET', 'mycar.html');  req.onload = () => {  if (req.status == 200) myCallback(req.responseText);  else myCallback('Error:', req.status);  };  req.send();  }  getFile(myDisplayer); |  |
| Waiting for a File (Promise) | const promise = new Promise((resolve, reject) => {  const req = new XMLHttpRequest();  req.open('GET', 'mycar.html');  req.onload = () => {  if (req.status == 200) resolve(req.response);  else reject('File not found');  };  req.send();  });  promise.then((value) => myDisplayer(value), (err) => myDisplayer(err)); |  |

## FETCH API

|  |  |  |
| --- | --- | --- |
| Fetch API (Uses Promises) | fetch('./api/some.json')  .then(function(response) ...) // if successful  .catch(function(error) ...); // The response of a fetch request is a stream object. The reading of the stream happens asynchronously. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| GET Request | fetch('https://.../api/some.json') // returns a Promise (Stream), we receive only the headers of the response  .then(response => response.json()) // returns another Promise  .then((data) => console.log(data))  .catch((error) => console.log(error)); |  |
|  | response.json()  response.text()  response.blob() // for images |  |
| POST Request | fetch(url, {  method: 'post',  headers: { 'Content-Type': 'application/json' },  body: JSON.stringify(data)  })  .then((response) => response.json())  .then((data) => console.log(data))  .catch((error) => console.log(error)); |  |
| Promises: Objects Holding Asynchronous Operations; States – Pending, Fulfilled, Failing | console.log('Hello'); // executed first  new Promise((resolve, reject) {  setTimeout(() => resolve('done'), 500); // resolved after 500 ms  })  .then((result) => console.log('Then returned: ' + result)) // 'Then returned: done'  .catch((error) => console.log(error));  console.log('Hello again'); // executed second |  |
|  | const [response1, response2] = Promise.All([ fetch(url1), fetch(url2) ]); |  |
|  | async |  |

## ASYNC FUNCTIONS

|  |  |  |
| --- | --- | --- |
| Async Functions | Asynk makes a function return a promise.  Await makes a function wait for a promise.  Returns a promise that can await other promises in a way that looks synchronous. Contains an await expression that pauses the execution of the function and waits for the promise's resolution. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | async function sayHi() {  return 'Hello';  }  // same as:  async function sayHi() {  return Promise.resolve('Hello');  } |  |
| use the promise | sayHi().then((value) => ..., (err) => ...); |  |
|  | async function surprise() {  const promise = new Promise((resolve, reject) => {  setTimeout(() => resolve('I love you!'), 3000);  });  console.log(await promise);  }  surprise(); |  |
| waiting for a file | async function getFile() {  const promise = new Promise((resolve, reject) => {  const req = new XMLHttpRequest();  req.open('GET', 'mycar.html');  req.onload = () => {  if (req.status == 200) resolve(req.response);  else reject('File not found');  };  req.send();  });  console.log(await promise);  }  getFile(); |  |
| Promise.then vs. Async/Await | function logFetch(url) {  return fetch(url)  .then(response => response.text)  .then(text => console.log(text))  .catch(err => console.error(err));  }  async function logFetch(url) {  try {  const response = await fetch(url);  console.log(response.text());  } catch (err) {  console.log(err);  } |  |

# REMOTE DATA AND AUTHENTICATION

|  |  |  |
| --- | --- | --- |
| 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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\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)

|  |  |  |
| --- | --- | --- |
|  | <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](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |  |

# WEB APPLICATIONS: BACK END

## BAAS

|  |  |  |
| --- | --- | --- |
| About Backend as a Service (BaaS) | A cloud service model in which developers outsource all the behind-the-scenes aspects of a web or mobile application so that they only have to write and maintain the frontend. BaaS vendors provide pre-written software for activities that take place on servers, such as user authentication, database management, remote updating, etc. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

## NODE.JS

|  |  |  |
| --- | --- | --- |
| Initialize a Node Project | npm init // asks questions and creates a package.json file with all dependencies (libraries needed for code execution)  npm init -y // creates a package.json with automatic answers | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Download an External Library and Add It to Dependencies | npm install lodash --save // adds lodash to dependencies  npm install -E lodash // short for --save |
| Download Several External Modules | npm install -E express express-handlebars // downloads both |
| Download All External Libraries Needed for the Project | npm install // downloads all external libraries necessary for the project  npm i // short for 'npm install' |
| Run a Node.js Application | node index.js // when any changes are made, Ctrl + C to stop the server and again node to restart it |  |
| Run and Automatically Update a Node.js Application | npm install -g nodemon  nodemon index.js |  |
| Run and Ignore Updates in a Specific File | nodemon --ignore models/data.json |  |
|  | in package.json:  "scripts": {  "start": "nodemon app.js"  }  npm start // type it to run the server app |  |
|  | index.js - automatically imports it from the folder |  |
| Stop the Server | Ctrl + C |  |

## NODE.JS CORE MODULES

### URL MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const url = require('url'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Parse a URL Object | const urlObj = url.parse(req.url); // returns an object with info about the url  const host = urlObj.host; // 'localhost:8080'  const path = urlObj.pathname; // '/home'  const query = urlObj.query // '?year=2017&month=february'  const search = urlObj.search // '?year=2017&month=february' |

### QUERYSTRING MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const queryString = require('querystring'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Parse a Querystring Object (Stream Data) | const qs = queryString.parse('year=2017&month=february');  const year = qs.year; // 2017  const month = qs.month; // 'february' |

### PATH MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const path = require('path'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Parse a Path Object | const pathObj = path.parse(\_\_filename); // \_\_dirname  pathObj.root // 'C:\\'  pathObj.dir // 'C:\\Users\\Project'  pathObj.base // 'myFile.js'  pathObj.ext // '.js'  pathObj.name // 'myFile' |
| Path Methods | path.join(\_\_dirname, '\\Demo', '\\asdf', '..') // 'C:\\Users\\Project\\Demo'  path.normalize('C:\\Demo\\\\temp\\..\\') // 'C:\\Demo' |  |

### OPERATING SYSTEM MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const os = require('os'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Get Total/Free OS Memory | const totalMemory = os.totalmem();  const freeMemory = os.freemem(); |

### FILE SYSTEM MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const fs = require('fs'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Import the FS Promises Property | const fs = require('fs').promises; |
| List Files in a Directory | const files = fs.readdir('./', 'utf-8', (err, files) => {  if (err) {  console.log('Error', err);  return;  } else {  console.log('Result', files); // an array of file and folder names  }  }); |
| List Files with FS Promises Property | (async () => {  const files = await fs.readdir('.');  console.log(files); // an array of file and folder names  })(); |  |
| Read File Synchronously | const data = fs.readFileSync('./package.json');  data // returns buffer data  data.toString() // returns json data |  |
| Read File Asynchronously | fs.readFile(path, 'utf-8', (err, data) => { // second parameter is optional  if (err) console.log('Error', err);  else console.log('Result', data.toString());  }); |  |
| Read File with FS Promises Property | const promise = fs.readFile('./package.json');  promise.then(data => console.log(data.toString())); |  |
| Read File with FS Promises Property in an Async Function | async function handleFiles() {  const data = await fs.readFile('./package.json');  console.log(data.toString());  } |  |
| Read File in Chunks | fs.createReadStream(filePath); |  |
| Create a Directory | fs.mkdir('./myDir', err => {  if (err) {  console.log(err);  return;  }  }); |  |
| Create a Directory with FS Promises Property | (async () => {  await fs.mkdir('./myDir');  })(); |  |
| Delete a Directory | fs.rmdir('./muDir', err => {  if (err) {  console.log(err);  return;  }  }); |  |
| Delete a Directory with FS Promises Property | (async () => {  await fs.rmdir('./myDir');  })(); |  |
| Rename a File/Directory | fs.rename('./oldName', './newName', err => {  if (err) {  console.log(err);  return;  }  }); |  |
| Rename a File/Directory with FS Promises Property | (async () => {  await fs.rename('./myDir', 'myNewDir');  })(); |  |
| Write a File Synchronously | fs.writeFileSync('./package\_copy.json', data); // creates new file with data |  |
| Write a File Asynchronously | fs.writeFile('./data.txt', 'Some text', err => {  if (err) {  console.log(err);  return;  }  }); |  |
| Write a File with FS Promises Property | (async () => {  await fs.writeFile('myFile', 'Some text');  })(); |  |
| Delete a File | fs.unlink('./target.txt', err => {  if (err) {  console.log(err);  return;  }  }); |  |
| Delete a File with FS Promises Property | (async () => {  await fs.unlink('myFile');  })(); |  |

### EVENTS MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const os = require('events'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Register an Event Listener | const publisher = new events.EventEmitter();  publisher.on('ping', (a, b) => console.log(a, b)); // 'Hello world' |
| Raise an Event | publisher.emit('ping', 'Hello', 'world'); // synchronous! |  |

### HTTP MODULE

|  |  |  |
| --- | --- | --- |
| Import It | const http = require('http'); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Web Server (to Handle Web Requests) | http.createServer((req, res) => { // request/response wrappers  res.write('Hello, world!');  res.end();  }).listen(3000, () => console.log('Listening on port 3000...')); // server is an instance of EventEmitter |
| Request Wrapper Properties | req.httpVersion // '1.1' or '1.0'  req.headers // object for request headers  req.method // 'GET', 'POST', etc.  req.url // the URL of the request |  |
| Response Wrapper Methods | res.writeHead(200, { 'Content-Type': 'text/plain' }); // creates response header  res.write('Hello from Node.js'); // sends content to the client (UTF-8 encoding)  res.end(); // ends the response |  |
|  | server.on('connection', (socket) => {  console.log('New connection...');  }); |  |
|  | server.on('request', (req, res) => {  const src = fs.createReadStream('./bigfile.txt');  src.pipe(res);  }); |  |

# EXPRESS.JS

## INSTALLATION AND MAIN CHARACTERISTICS

|  |  |  |
| --- | --- | --- |
| Installation | npm install express --save-exact | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Loading | const express = require('express'); const app = express(); // new instance of the application |
| Automatically Added Response Header | { 'X-Powered-By': 'Express' } |  |
| Hide a URL from Users, Set a ENV Port, Production | npm install dotenv  create a '.env' file, inside: PORT=5000 // DB\_CONNECTION=mongodb://testboy:rhino94@ds155396.mlab.com:55396/rest or in terminal: $set PORT=5000  require('dotenv/config');  const port = process.env.PORT || 3000;  app.listen(port);  mongoose.connect(  process.env.DB\_CONNECTION,  { useNewUrlParser: true },  () => console.log('connected to DB!')  ); |  |
| Router (Server?) app.*method*(*path*, *handler*) | app.get('/', (req, res) => {  res.status(200).send('Welcome to Express.js!'); // can only be used once (automatically ends the response) ).listen(3000, () => console.log('Listening on port 3000...')); |  |
| All Methods Route (Used with '\*' for 404 Page) | app.all('/about', (req, res, next) => {  console.log('Middleware execution...');  next();  }, (req, res) => {  res.send('Show about page');  }).listen(3000); |  |
| Create Chainable Route Handlers | app.route('/home')  .get((req, res) => res.send('GET home page'));  .post((req, res) => res.send('POST home page'));  .all((req, res) => res.send(404, '404 Not Found')); // 'all' always at the end, otherwise it cancel the previous methods |  |
| Create Modular Routers (Mounted on a Route/Endpoint) | const router = express.Router();  router.get('/catalog', (req, res) => res.send('Catalog Page')); |  |
| Use Modular Routers | const catalogRouter = require('./catalog')  app.use(catalogRouter); |  |
| Body Parser (Gets the Fields from a <form>: req.body = { name: ..., id: ...} | app.use(express.urlencoded({ extended: true })); // only if the <form> does NOT have enctype="multipart/form-data" |  |
| Access Query String | .../?search=alabala&difficulty=3  req.query // returns { search: 'alabala', difficulty: 3 } |  |
| Store Local Variables Scoped to the Specific Request/Response Cycle | res.locals = userData; |  |

## ROUTER PATHS

|  |  |  |
| --- | --- | --- |
| Match Everything (Including an Empty String) | app.get('/catalog/\*', (req, res) => {  res.send('Product Page');  }); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Use String Patterns | app.get('/ab\*cd', (req, res) => {  res.send('abcd, abANYTHINGcd');  }); |
| Use Regular Expressions | app.get(/.\*fly$/, (req, res) => {  res.send('butterfly, dragonfly');  }); |
| Use Parameters | app.get('/users/:userId', (req, res) => {  const paramsObj = req.params;  res.send(paramsObj); // { userID: '123' }  }); |  |
| Validate Parameters Using RegExp (Not Recommended) | app.get('/users/:userId(\\d+)', (req, res) => {  const paramsObj = req.params;  res.send(paramsObj); // Cannot GET /users/123a  }); |  |

## ROUTER RESPONSES

|  |  |  |
| --- | --- | --- |
| Download a File | app.get('/pdf', (req, res) => {  res.download('./demo.pdf');  }); // adds a header to the response: 'Content-Disposition': attachment; filename="demo.pdf" | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Send a File as an Octet Stream (Open It in the Browser) | app.get('/tos', (req, res) => {  res.sendFile(\_\_dirname + 'demo.pdf');  }); |
| End the Response | res.end(); |
| Send a JSON Response | res.json(); |
| Redirect Request to Another Page | app.get('/contact', (req, res) => {  res.redirect('/about'); // adds a header 'Location': '/about'  }); |  |
| Render a View Template | res.render(); |  |

## MIDDLEWARE (PLUGINS, EXTENSIONS)

|  |  |  |
| --- | --- | --- |
| Definition | A function that has access to the request and respons object and the next middleware in the application's request-response cycle.  between the router and the action; always has to call next() | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Types of Middleware | application, route, error |
| Create Middleware | function isAdmin(req, res, next) {  if (req.headers.hasOwnProperty('x-admin')) next();  else res.status(401).send('Admins only. Please sign in.');  } |
| Use Middleware for a Specific Path | const isAdmin = require('./guard');  app.get('/admin', isAdmin, (req, res) => {  res.send('Admin Page');  }); |  |
| Use Middleware on Application Level (for All Paths) | const logger = require('./logger');  app.use(logger); |  |
| Create Middleware for Error Handling | function fallback(err, req, res, next) {  console.error(err.message);  res.status(500).send('500 Server error');  } |  |
| Use Middleware for Error Handling (Does Not Catch Asynchronous Errors) | const fallback = require('./fallback');  app.get('/', (req, res, next) {  next(new Error('Test error'));  });  app.use(fallback); // at the end; a global error handler |  |
| Third-Party Middleware | app.use(cookieParser());  app.use(session({ secret: 'magic unicorns' }));  app.use(passport.initialize());  app.use(passport.session());  app.use(express.static(config.rootPath + '/public')); |  |

## STATIC FILES

|  |  |  |
| --- | --- | --- |
| Serving Static Files (All Files from the Directory Will Be Public) | app.use(express.static('public')); // all files from the folder 'public' will be considered static | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | app.use('/static', express.static('public')); // find all files the user tries to access with the url starting with '/static' in folder 'public' instead |
|  | app.use('/static', express.static(\_\_dirname + '/public')); |

## VIEW ENGINES

|  |  |  |
| --- | --- | --- |
| Set File Extension and Default Files/Folders | app.engine('.hbs', hbs({  extname: '.hbs',  layoutsDir: 'myLayouts', // default: 'layouts'  defaultLayout: 'site' // default: 'main'  }));  app.set('view engine', '.hbs'); // default extension if none in render() | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

# HANDLEBARS

|  |  |  |
| --- | --- | --- |
| Installation | npm install handlebars  npm install express-handlebars //integration in Express | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Loading | const hbs = require('express-handlebars'); |
| Set File Extension for Express and HBS | app.engine('.hbs', hbs({ extname: '.hbs' }));  app.set('view engine', '.hbs'); // default extension if none in render() |  |
| custom select helper | app.engine('.hbs', hbs.create({  helpers: {  select: function (value, options) {  return options.fn()  .split('\n')  .map(v => {  const t = `value="${value}"`;  return new RegExp(t).test(v) ? v.replace(t, t + ' selected') : v;  })  .join('\n');  }  },  defaultLayout: 'main',  layoutsDir: 'layouts',  extname: '.hbs'  }).engine); |  |
| Create Default Layout (Folder Views > Folder Layouts > main.hbs) | default HTML (! + Enter), in <body>:  {{{body}}} |  |
| Create View (Folder Views > home.hbs) | <h1>Home Page</h1> |  |
| Change Default Folder for View Engine | app.set('views', 'templates'); |  |
| Render Created View | app.get('/', (req, res) => res.render('home')); |  |
| Render View with Dynamic Data (Context) | app.get('/', (req, res) => res.render('home', { title, body })); |  |
| When Layouts is Empty | app.get('/', (req, res) => res.render('home', { layout: false })); |  |
| Initialize Expressions in HBS | <h1>{{title}}</h1> |  |
| Comments in HBS | {{!-- This is a comment --}} |  |
| Loop through an Array (Each Helper) | <ul>  {{#each numbers}} {{!-- numbers: [1, 2, 3]) --}}  <li>{{this}}</li>  {{else}} {{!-- if the array is empty --}}  <li>No numbers</li>  {{/each}}  </ul> |  |
| Loop through an Array of Objects | <ul>  {{#each items}} {{!-- items: [{ type, qty }, { type, qty }]) --}}  <li>{{type}}: {{qty}}</li>  {{/each}}  </ul> |  |
| Conditional Statements | {{#if user}}  <span>Hello, {{user.username}}!</span>  {{else}}  <span><a *href*="/login">Login</span>  {{/if}} |  |
| Partials (Folder Views > Partials) | <ul>  {{#each contacts}}  {{> contact}} {{!-- inserts the template contact.hbs --}}  {{else}}  <i>(empty)</i>  {{/each}}  </ul> |  |
| HTML Escaping | {{{title}}} {{!-- 'About <p> Tags' => 'About &lt;p&gt; Tags' --}} |  |
| Change the Context | {{{#with numbers}}}  ...  {{{/with}}} |  |

# JOI

|  |  |  |
| --- | --- | --- |
| Installation | npm install joi | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Validate Input with Joi |  |

# EXTERNAL LIBRARIES

|  |  |  |
| --- | --- | --- |
| Mocha |  | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |  |
|  |  |  |
| Formidable | npm install -E formidable  const formidable = require('formidable');  const form = new formidable.IncomingForm();  form.parse(req, (err, fields, files) => {  database.addItem(fields); // { name: 'John', age: 23 }  res.writeHead(301, { 'Location': '/' });  res.end();  }); |  |
| Joi | npm install -E joi |  |

# STREAMS

|  |  |  |
| --- | --- | --- |
| Stream Definition | Collection of data that is not available at once (may come continuously in chunks). Obviously asynchronous. We use events to manage them. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Readable Streams | can only be read (process.stdin)  Functions:  read() // get chunks from the stream after pause()  pause() // switch to paused mode  resume() // switch to flowing mode  pipe() // allows a readable stream to output directly to a writable stream  Events: (used when the stream is flowing)  data // chunk is available for reading  end // no more data  error // an exception has occurred  HTTP request is a readable stream:  const http = require('http');  http.createServer((req, res) => {  if (req.method == 'POST') {  let body = '';  req.on('data', data => body += data);  req.on('end', () => console.log(body));  }  }).listen(5000);  ==================================  const fs = require('fs');  const server = require('http').createServer();  server.on('request', (req, res) => {  const src = fs.createReadStream('./bigfile.txt');  src.pipe(res);  });  server.listen(5000); |
| Writable Streams | can only be written to (process.stdout)  Functions:  write() // send chunks to the stream  end() // close the stream  Events:  drain // stream can receive more data  finish // all data has been flushed (buffer is empty)  error // an exception has occurred  HTTP Response is a writable stream  const fs = require('fs');  const server = require('http').createServer();  server.on('request', (req, res) => {  const src = fs.createReadStream('./bigfile.txt');  src.on('data', data => res.write(data));  src.on('end', () => res.end());  });  server.listen(5000); | u |
| Duplex Streams | Implements both readable and writable interfaces (ex. TCP sockets). |  |
| Transform Streams | A special kind of duplex stream where the output is a transformed version of the input (ex. zlib, crypto). |  |

# PUBLISH-SUBSCRIBE PATTERN

|  |  |  |
| --- | --- | --- |
| Publishers | The senders of messages (data) that do not program the messages to be sent directly to specific receivers, but instead categorize published messages into classes without knowledge of which subscribers, if any, there may be. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Subscribers | Express interest in one or more events and only receive messages that are of interest. |  |
| Event Bus (Message Broker) | An intermediary that retrieves published messages and forwards them to the subscribers who are registered to receive them. |  |
| Example | button.addEventListener('click', handler) // the button is publisher, the event handler is subscriber |  |

# DATABASES

## TYPES OF DATABASES

|  |  |  |
| --- | --- | --- |
| Relational (MariaDB, Oracle) | Data organized in tables of columns and rows with unique (for the table) keys identifying each row. SQL is used to extract data (SELECT \* FROM Students), relations between tables are done using Foreign Keys.  Strict rules! rules are on engine level | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Non-Relational (NoSQL: MongoDB, Cassandra) | All non-relational types of databases. Data stored in key-value pairs. SQL is not used, therefore also called NoSQL. More scalable, provide superior performance.  No strict rules! or: rules are on code level (our code) |

## MONGO DB

|  |  |  |
| --- | --- | --- |
| Setup | Create the nested folders "data" > "db" in C:/.  In Program Files > Mongo DB > Server > 42 > bin:  mongod // the primary daemon process for the MongoDB system  mongo // run the MongoDB shell | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| In the MongoDB Shell | show dbs // print a list of all databases on the server  use myBD // create a new database called "myDB"  db.courses.insertOne({ name: "mongoDB course" }) // create a collection called "courses" in "myDB"  show collections // courses  db.courses.find() // { "\_id" : ObjectId("5e56c9f2dc..."), "name" : "mongoDB course" } |
| Configuration | <path to mongod.exe>mongod --dbpath <path to store data> |
| Installation in Node.js | npm install mongodb -g |  |
| Work with MongoDB in NodeJS | const mongodb = require('mongodb');  const MongoClient = mongodb.MongoClient;  const connectionStr = 'mongodb://localhost:27017';  const client = new MongoClient(connectionStr, { useUnifiedTopology: true });  client.connect((err) => {  const db = client.db('testdb'); // use testdb in terminal  const people = db.collection('people');  people.insertOne({ name: 'Ivan' }, (err, data) => {  people.find({ name: 'Ivan' }).toArray((err, data) => {  console.log(data);  });  });  }); |  |
| MongoDB Hosting | Go to 'mongo atlas' or mlab.com and register in order to store up to 500 MB of content. |  |
| MongoBD Graphic User Interface | MongoDB Compass |  |

# MONGOOSE

## INSTALLATION AND CONNECTION TO DB

|  |  |  |
| --- | --- | --- |
| About the Library | An object-document model (ODM) module in Node.js for MongoDB (works without mongoDB, though). Provides a straight-forward, schema-based solution to model the application data. Includes built-in type casting and validation. Extends the native queries (much easier to use). (like ORM in relational DB) | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Installation | npm install mongoose --g // for each project |
| Loading | const mongoose = require('mongoose'); |
| Connect to DB and Create an Instance of a Model | (async () => {  await mongoose.connect('mongodb://localhost:27017/unidb', {  useUnifiedTopology: true,  useNewUrlParser: true,  useFindAndModify: false,  autoIndex: false  });  await new Student({ name: 'Peter', age: 23 }).save();  })(); |  |

## CREATE A MODEL. VALIDATION

|  |  |  |
| --- | --- | --- |
| Create a Model (Usually a Separate File in "Models" Folder) | const { Schema, model } = require('mongoose');  const studentSchema = new Schema({  name: { type: String, required: true, minlength: 3 },  age: { type: Number, default: 25}  });  module.exports = model('Student', studentSchema); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Set Value Type | type: String |  |
|  | name: [String, 'Name must be string'] |  |
| Set Property as Required | required: true  required: function () { return this.age >= 18 }  required: [true, 'Name is required'] |  |
| Set a Unique Value | unique: true // DB indexes need to be set |  |
| Set a Minimum Value for a Number | min: 0  min: [0, 'Age cannot be negative'] |  |
| Set a Minimum Length for a String | minLength: 10 // or minlength |  |
| Set a Default Value | default: 'This is the content of the publication'  default: Date.now |  |
| Preset Values | enum: {  values: ['male', 'female'],  message: 'Sex must be either "male" or "female", got {VALUE} instead'  } |  |
| Set a Required First Capital Letter | validate: {  validator: function (v) {  const letter = v.slice(0, 1);  return letter == letter.toLocaleUpperCase();  },  message: props => `${props.value} doesn't start with a capital letter`  } |  |
| Validate Records after Creating the Model | studentSchema.path('firstName')  .validate(function () {  return this.firstName.length >= 2 && this.firstName.length <= 10;  }, 'First name must be between 2 and 10 symbols long!'); |  |
| Add Methods to a Model | studentSchema.methods.getInfo = function () { // avoid arrow functions  console.log(`I am ${this.name} and I am ${this.age} years old`);  }; |  |
| Add Virtual Properties (Not Persisted to the Database) | studentSchema.virtual('fullName').get(function () {  return this.firstName + ' ' + this.lastName;  }); |  |

## READ DATA

|  |  |  |
| --- | --- | --- |
| General Read Syntax (Returns a Promise) | Student.find({}).then(arr => console.log(arr)).catch((err) => ...);  await Student.find({}); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Get a JS Object instead of a Mongoose Document Class | await Student.find({}).lean(); // for Handlebars  ?? // await Student.find({}).toObject(); // same as lean, but can be used later (in controllers) |
| Get All Instances (No Filter) | await Student.find({}); // returns an array |
| Filter Instances by a Specific Value | await Student.find({ name: 'John' }); // students whose name is John  await Student.find({}).where('facultyNumber').equals('12399');  await Student.find({}).where('name').equals('John').or(...); |  |
| Filter Indexed Instances by a Search Value | await Student.find({ $text: { $search: 'John' } }); // schema.index('$\*\*': 'text'): all string fields are indexed |  |
| Filter Instances with a RegEx | await Student.find({ name: /o/i }); // students whose names contain o/O  await Student.find({ name: { $regex: 'o', $options: 'i' }});  await Student.find({ name: { match: /o/i }}); |  |
| Filter Instances by a Number Range | await Student.find({ age: { $gt: 19 } }); // students older than 19  await Student.find({ age: { $lte: 19 } }); // all students yonger than or 19  await Student.find({ age: { min: 18, max: 26 } });  await Student.find({}).where('age').gt(7).lt(14); |  |
| Get Instances That Are (Not) Part of a Predefined Array | await Student.find({ name: { $in: ['John', 'Ben'] } }); // John or Ben  await Student.find({ name: { $nin: { ['Mary', 'Jill'] } }); // NOT Mary or Jill |  |
| Get Sorted Instances | await Student.find({}).sort({ age: -1 }); // sort descending  await Student.find({}).sort({ age: -1 }).skip(10).limit(10); // pagination |  |
| Get the First Filtered Instance | await Student.findOne({ name: 'John' }); // an object |  |
| Get an Instance by Its ID | await Student.findById('e453ne6r'); |  |
| Sort Instances (Never in Business Logic) | await Student.find({}).sort({ age: -1 }); // sort by age, descending  await Student.find({}).sort({ subjects: -1 }); // sort by subjects.length, descending  await Student.find({}).sort({ subjects: 'desc' }); // sort by subjects.length, descending |  |
| Get Only First Three Instances | await Student.find({}).limit(3); |  |

## UPDATE DATA

|  |  |  |
| --- | --- | --- |
| Find, Update and Save | const student = await Student.findById('dk123hb');  student.age++;  await student.save(); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | findOneAndUpdate({ runValidators: true }) // will not skip validation |
| Find and Update | await Student.findByIdAndUpdate('dk123hb', {  $set: { name: 'Ben' }  }); |
| Update the First Match | await Student.updateOne(  { name: 'Peter' },  { $set: { name: 'Ben' } }  }); |  |
| Update Many |  |  |

## REMOVE DATA

|  |  |  |
| --- | --- | --- |
| Find and Remove | await Student.findByIdAndRemove('dk123hb'); // useFindAndModify: false | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Remove | await Student.deleteOne({ name: 'John' }); |
| Remove Many |  |

## COUNT DOCUMENTS (ENTRIES)

|  |  |  |
| --- | --- | --- |
| Get a Number of All Documents in a Collection | await Student.countDocuments({}); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Get a Number of Filtered Documents | await Student.countDocuments({ age: { $gt: 19} }); |

## REFERENCES AND POPULATION

|  |  |  |
| --- | --- | --- |
| Create Models That Reference Each Other | const studentSchema = new Schema({  name: String,  teacher: { type: Schema.Types.ObjectId, ref: 'Teacher' },  subjects: [{ type: Schema.Types.ObjectId, ref: 'Subject' }]  });  const subjectSchema = new Schema({  title: String,  students: [{ type: Schema.Types.ObjectId, ref: 'Student' }]  }); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Population | await Student.findOne({}).populate('subjects').populate('teacher'); |  |
| Nested Population | await Subject.find({}).populate({  path: 'students',  populate: 'teachers'  }); |  |
|  |  |  |

# APPLICATION SECURITY

## COOKIES

|  |  |  |
| --- | --- | --- |
| About | Stored on the client, preferred when in need of long-term information/values storage. Not very safe: expiration can be set and they can last for years. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Set Cookie in an HTTP Response Header | res.setHeader('Set-Cookie': 'sessionId=902fkeu64hshfhf'); |
| Access Cookie | req.headers.cookie |
|  | httpOnly // cookie not accessible through the client's JS |
| Cookie Parser Installation | npm install cookie-parser --save(-exact) // exact version |
| Loading | const cookieParser = require('cookie-parser'); // in an Express app |  |
| Setting as Middleware | app.use(cookieParser()); |  |
| Usage (First Delete All Cookies From Browser: F12 > Application > Storage > Cookies) | app.get('/setCookie', (req, res) => {  res.cookie('message', 'hello');  res.end('Cookie set');  });  app.get('/readCookie', (req, res) => {  res.json(req.cookies);  }); |  |
| Set Cookie | res.cookie('message', 'hello'); |  |
| Access Cookie | req.cookies |  |

## SESSIONS

|  |  |  |
| --- | --- | --- |
| About | Stores information about a client on the server, used to persist state across requests. Matched to a client by their cookie. Preferred when in need of short-term information/values storage. Safer (stored on the server). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Express Session Installation | npm install express-session --save-exact |
| Loading | const session = require('express-session'); |  |
| Setting as Middleware | app.use(session({  secret: 'my secret',  resave: false,  saveUnitialized: true,  cookie: { secure: false } // true for https  })); |  |
| Usage: We Store Data Only in the Session on the Server, in the Cookie is Stored Only the Session ID | app.get('/setSession', (req, res) => {  req.session.message('hello');  res.end('Session set');  });  app.get('/readSession', (req, res) => {  res.json(req.session);  }); |  |

## AUTHENTICATION

|  |  |  |
| --- | --- | --- |
| About | An important part of application security, serves to verify whether the client is in fact who or what it declares itself to be. It's built on several layers of abstraction: cookies > sessions > security. A cross-cutting concern, best handled away from business logic. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| internal library: crypto |  |  |
| About Bcrypt (External Library) | A password hashing function. Incorporates a salt to protect agains rainbow table attacks and is an adaptive function. Over time, the iteration count can be increased to make it slower, so it remains resistant to brute-force search attacks even with increasing computation power. hashing algorithm |  |
|  | hashing cypher |  |
| Installation | npm install bcrypt |  |
| Loading | const bcrypt = require('bcrypt'); |  |
| Usage: Async Recommended!  Hash Password | const saltRounds = 9;  const myPlainTextPassword = 'password123';  bcrypt.genSalt(saltRounds, (err, salt) => {  bcrypt.hash(myPlainTextPassword, salt, (err, hash) => {  console.log(hash); // $2b$09$pdhUAoT4qE0tmku.ZkXWROeLcJCy.LDR q.1I4IVImjrUTGuUbYQMi  });  }); |  |
|  | const hashedPassword = await bcrypt.hash(password, 10); |  |
| Check Password | const myPlainTextPassword = 'password123';  const hash = '$2b$09$pdhUAoT4qE0tmku.ZkXWROeLcJCy.LDRq.1I4IVI mjrUTGuUbYQMi';  bcrypt.compare(myPlainTextPassword, hash, (err, res) => {  console.log(res); // true  }); |  |

## JSON WEB TOKEN

|  |  |  |
| --- | --- | --- |
| About | An open standard that defines a compact and self-contained way for securely transmitting information between parties as JSON object.  The information can be verified and trusted because it is digitally signed.  JWT can be signed using a secret or a public/private key pair usin RSA or ECDSA.  When we use only one server, the session is enough, no need for JWT.  in the headers or in the cookie: Viktor says better in the headers | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Authorization | Once the user is logged in, each subsequent request will include JWT, allowing the user to access routes, services and resources that are permitted with that token. |  |
| Information Exchange | JWT are a good way of securely transmitting information between parties. Because they are signed digitally. |  |
| JWT Structure | Compact form: header, payload and signature, separated by dots. |  |
| Installation | npm install jsonwebtoken |  |
| Loading | const jwt = require('jsonwebtoken'); |  |
| Encode Token | const payloads = { \_id, username };  const options = { expiresIn: '2d' };  const secret = 'MySuperPrivateSecret';  const token = jwt.sign(payload, secret, options);  console.log(token); //eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJwYXkiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.xzK8LJQz0lDkJqsng04BYxcUQzxWngyEBP |  |
| Decode Token | const token = req.cookies['token'] || sessionStorage.getItem('token'); // depends on where you store the token  const decodedToken = jwt.verify(token, secretKey); // or jwt.decode(token)  console.log(decodedToken); // { \_id: ..., username: ... } |  |

# VALIDATION AND ERROR HANDLING

|  |  |  |
| --- | --- | --- |
| Client-Side Validation | In HTML of JS before any request is sent. Optional; NOT a protection that secures us against incorrect data being sent to the server and stored into the DB, as the user can see, change or disable the code in the browser. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Server-Side Validation | This is the place to add validation and filter out invalid data. |
| Database Validation | Not required, there should be no scenario in which the DB works with invalid data. |  |
| Sanitization | Makes sure the data is in the right format, removes any illegal character from the data. Mutates the request.  normalizeEmail: canonicalizes an email address  trim: trim characters from both sides of the input  blacklist: remove characters that appear in the blacklist  escape: removes all HTML control symbols |  |

## VALIDATOR.JS

|  |  |  |
| --- | --- | --- |
| About | A library of string validators and sanitizers. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Installation | npm install validator |
| Server-Side Usage | const validator = require('validator');  const body = req.body;  validator.isEmail(body.email); // true or false |
|  | import isEmail() from ... |  |
| Client-Side Usage | <script *type=*"text/javascript" *src=*"validator.min.js"></script>  <script *type=*"text/javascript">  validator.isEmail($('#email').val()); // true or false  </script> |  |

## EXPRESS VALIDATOR

|  |  |  |
| --- | --- | --- |
| About | A set of express.js middlewares that wraps validator.js validator and sanitizer functions. Appropriate for user validation (data that we don't want to get to the DB) | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Installation | npm install express-validator |
| Server-Side Usage | const { check, validationResult } = require('express-validator');  check('email').isEmail(); // searches for 'email' in body, params, query...  check('password').isLength({ min: 5 });  const errors = validationResult(req);  if (!errors.isEmpty()) // return status 422 and export errors |
| Validation + Sanitizing Input | const { body } = require('express-validator');  body('email').isEmail().normalizeEmail();  body('password').isLength({ min: 5 }).isAlphanumeric().trim().escape(); |  |
|  | .isAlphanumeric()  .matches(/[a-zA-Z0-9]/).withMessage('Username must only consist of latin letters or numbers')  .notEmpty(), .withMessage() |  |
| Custom Validator | const { body } = require('express-validator');  app.post('/user', body.('email').custom(value, { req } => {  return User.findUserByEmail(value)  .then(user => {  if (user) {  return Promise.reject('E-Mail already in use');  }  });  }); |  |
| Custom Sanitizer | const { sanitizeParam } = require('express-validator');  app.post('/object/:id', sanitizeParam('id').customSanitizer(value => {  return ObjectId(value);  }), (req, res) => {  ...  }); |  |
|  | body('repass').custom((value, { req }) => {  if (value != req.body.password) {  throw new Error('Passwords don\'t match');  }  return true;  }); |  |
|  | .bail() // does not continue validations if previous failed |  |
|  | .withMessage() |  |

## MONGOOSE VALIDATOR

|  |  |  |
| --- | --- | --- |
| Mongoose Validation | It's a middleware defined in the SchemaType. Asynchronously recursive, customizable. The save() function triggers validate() hook. All pre('validate') and post('validate') hooks get called before any pre('save') hook.  \*/  schema.pre('validate', function() {  console.log('this gets printed first');  });  schema.post('validate', function() {  console.log('this gets printed second');  });  schema.pre('save', function() {  console.log('this gets printed third');  });  schema.post('save', function() {  console.log('this gets printed fourth');  });  /\*  All ShemaTypes have built-in required validator. Numbers have min and max validators, Strings have enum, match, minlength and maxlength. Custom validators: validate: { validator: function() ... }, message: props => ...  Errors returned after failed validation contain an error object whose values are ValidatorError object: has kind, path, value and message properties | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |
|  |  |

# EXPRESS REST API

|  |  |  |
| --- | --- | --- |
| two folders: server and client | Two separate applications.  In the terminal:  - cd client  - npm i (npm install)  In a second terminal:  - cd server  - npm init -y  - npm i express mongoose bcrypt jsonwebtoken  In server folder: index.js, controllers, middlewares, services  In client > src > api > data.js: const host = 'http://localhost:5000'  F12: make sure there is no local storage/session storage | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Domain Driven Design vs. Model View Controller | 1. folders: furniture, user (better for bigger apps)  2. folders: controllers, services, models, views (not good for front-end apps) |
|  | npm install cors // then use it as middleware  or:  app.use((req, res, next) => {  res.setHeader('Access-Control-Allow-Origin', '\*');  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, PATCH, DELETE');  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');  next();  }); |

SoftUni

Fundamentals: Workshop

Advanced Lections (Presentations): Objects Advanced, Functions Advanced

Advanced Exercises:

* + DOM Introduction Exercises with \*
  + Advanced Functions Lab ex. \* 10
  + Unit Testing Lab 7
  + Classes \* 11, 12, 14
  + Prototypes & Inheritance ex. \* 7
  + Workshop

JS Applications February 2020

My own JS Applications: the whole thing

JS Applications: Asynchronous Programming - Ex. (watch & do)

JS Applications? Webpack

JS Web: REST API Workshop (understand it and maybe redo it following DDD)

W3Schools

Math (sin, cos, tan...)

Bitwise Operators

AJAX

JSON PHP, JSONP

JS Examples: JS HTML Objects, JS HTML Events

v.kostadinov@softuni.bg