JavaScript

[INTRODUCTION 10](#_Toc106140375)

[WHERE TO USE JAVASCRIPT 11](#_Toc106140376)

[FILE NAMES 11](#_Toc106140377)

[SOURCE CODE COMPONENTS 12](#_Toc106140378)

[IDENTIFIERS (VARIABLE NAMES) 12](#_Toc106140379)

[WRITING A PROGRAM: BASIC RULES 13](#_Toc106140380)

[DATA TYPES & VALUES 14](#_Toc106140381)

[ABOUT JAVASCRIPT DATA TYPES 14](#_Toc106140382)

[JAVASCRIPT DATA TYPES 14](#_Toc106140383)

[PRIMITIVE/REFERENCE DATA TYPES 15](#_Toc106140384)

[GET THE DATA TYPE OF A VALUE 15](#_Toc106140385)

[CHECK WHETHER A VALUE IS OF SPECIFIC DATA TYPE 16](#_Toc106140386)

[CONVERT DATA TYPES. TYPE COERCION 17](#_Toc106140387)

[CONVERT OTHER TYPES TO BOOLEAN 17](#_Toc106140388)

[CONVERT OTHER TYPES TO STRING 17](#_Toc106140389)

[CONVERT OTHER TYPES TO NUMBER 18](#_Toc106140390)

[CONVERT OTHER TYPES TO BIGINT 18](#_Toc106140391)

[CONVERT OTHER TYPES TO OBJECT 19](#_Toc106140392)

[TYPE COERCION 19](#_Toc106140393)

[LIFE OF A VARIABLE 21](#_Toc106140394)

[UNDECLARED VARIABLES 21](#_Toc106140395)

[DECLARE & INITIALIZE A VARIABLE 21](#_Toc106140396)

[ASSIGN VALUES. ASSIGNMENT OPERATORS 21](#_Toc106140397)

[DESTRUCTURING ASSIGNMENT 22](#_Toc106140398)

[REDECLARE VARIABLES 23](#_Toc106140399)

[SCOPE 23](#_Toc106140400)

[HOISTING 24](#_Toc106140401)

[COMPARISONS & CONDITIONS 25](#_Toc106140402)

[OPERATORS 25](#_Toc106140403)

[COMPARE DATA 25](#_Toc106140404)

[IF – ELSE 26](#_Toc106140405)

[SWITCH 27](#_Toc106140406)

[LOOPS 28](#_Toc106140407)

[FOR LOOP 28](#_Toc106140408)

[WHILE LOOP 28](#_Toc106140409)

[BREAK A LOOP 29](#_Toc106140410)

[STRINGS. TEXT PROCESSING 30](#_Toc106140411)

[JS SPECIFICS 30](#_Toc106140412)

[STRING CHARACTERS. STRING LENGTH 30](#_Toc106140413)

[SEARCH FOR CHARACTER(S) 30](#_Toc106140414)

[EXTRACT STRING PARTS 31](#_Toc106140415)

[COMBINE AND CHANGE STRINGS 31](#_Toc106140416)

[ESCAPE SEQUENCES 32](#_Toc106140417)

[CHARACTER ENCODING 33](#_Toc106140418)

[COMPARE STRINGS 33](#_Toc106140419)

[NUMBERS & BIGINTS 34](#_Toc106140420)

[JS SPECIFICS 34](#_Toc106140421)

[ARITHMETIC OPERATIONS 35](#_Toc106140422)

[ARITHMETIC OPERATIONS WITH BIGINTS 35](#_Toc106140423)

[ARITHMETIC OPERATIONS WITH STRINGS 36](#_Toc106140424)

[FORMAT NUMBERS 36](#_Toc106140425)

[CHECK WHETHER A NUMBER IS OF SPECIFIC TYPE 37](#_Toc106140426)

[NUMBER SYSTEMS 37](#_Toc106140427)

[RANDOM NUMBERS 38](#_Toc106140428)

[SYMBOLS IN JAVASCRIPT 39](#_Toc106140429)

[OBJECTS 40](#_Toc106140430)

[JS SPECIFICS & TYPES OF OBJECTS 40](#_Toc106140431)

[OBJECT OBJECTS 41](#_Toc106140432)

[CREATE AN OBJECT 41](#_Toc106140433)

[CREATE A COPY OF AN OBJECT 42](#_Toc106140434)

[CREATE AN OBJECT BASED ON ANOTHER OBJECT 42](#_Toc106140435)

[ADD, CHANGE, ACCESS & DELETE PROPERTIES 43](#_Toc106140436)

[CHECK WHETHER A PROPERTY EXISTS 43](#_Toc106140437)

[DESTRUCTURING ASSIGNMENT 43](#_Toc106140438)

[GET A LIST OF THE PROPERTIES 44](#_Toc106140439)

[VISUALIZE THE OBJECT CONTENT 44](#_Toc106140440)

[LOOP THROUGH AN OBJECT 45](#_Toc106140441)

[METHODS 45](#_Toc106140442)

[GETTER & SETTER METHODS 46](#_Toc106140443)

[NESTED OBJECTS 47](#_Toc106140444)

[INTERNAL PROPERTIES 48](#_Toc106140445)

[CONVERT AN OBJECT TO AN ARRAY/A JSON STRING 50](#_Toc106140446)

[USE AN OBJECT AS A FUNCTION LIBRARY 50](#_Toc106140447)

[USE AN OBJECT AS A SWITCH REPLACEMENT 51](#_Toc106140448)

[PROTOTYPE- AND CLASS-BASED INHERITANCE 51](#_Toc106140449)

[DECLARE A CLASS AND CREATE INSTANCES 52](#_Toc106140450)

[THE FOUR PILLARS OF OBJECT-ORIENTED PROGRAMMING 55](#_Toc106140451)

[ARRAY OBJECTS 56](#_Toc106140452)

[JAVASCRIPT ARRAY CHARACTERISTICS 56](#_Toc106140453)

[CREATE AN ARRAY AND ACCESS ITS ELEMENTS 56](#_Toc106140454)

[INSPECT AN ARRAY 57](#_Toc106140455)

[EXTRACT ARRAY PARTS. FILTER 58](#_Toc106140456)

[CHANGE/ADD/REMOVE/SWAP ARRAY ELEMENTS 59](#_Toc106140457)

[CONVERT AN ARRAY TO STRING. CONCATENATE ARRAYS AND STRINGS 60](#_Toc106140458)

[CONVERT A NESTED ARRAY TO AN OBJECT 60](#_Toc106140459)

[ARRANGE THE ELEMENTS OF AN ARRAY IN CERTAIN ORDER. SORT AN ARRAY 60](#_Toc106140460)

[CALL A FUNCTION FOR EACH ARRAY ELEMENT 61](#_Toc106140461)

[REGULAR EXPRESSIONS 62](#_Toc106140462)

[MATCHING RULES 62](#_Toc106140463)

[PREDEFINED CLASSES 62](#_Toc106140464)

[QUANTIFIERS 63](#_Toc106140465)

[GROUP CONSTRUCTS 63](#_Toc106140466)

[FLAGS/MODIFIERS 64](#_Toc106140467)

[ANCHORS 64](#_Toc106140468)

[REGEX IN JAVASCRIPT 64](#_Toc106140469)

[EXAMPLES 66](#_Toc106140470)

[MAP OBJECTS 66](#_Toc106140471)

[SET OBJECTS 67](#_Toc106140472)

[DATE OBJECTS 68](#_Toc106140473)

[CREATE A DATE OBJECT 68](#_Toc106140474)

[CONVERT A DATE TO A STRING/NUMBER. COMPARE DATES 68](#_Toc106140475)

[ACCESS ONLY PART OF THE DATE 69](#_Toc106140476)

[CHANGE PART OF THE DATE 70](#_Toc106140477)

[FUNCTIONS 70](#_Toc106140478)

[FUNCTION DECLARATION/EXPRESSION 70](#_Toc106140479)

[FUNCTION HOISTING 71](#_Toc106140480)

[CALL/INVOKE A FUNCTION. PARAMETERS AND ARGUMENTS 71](#_Toc106140481)

[FUNCTION CONTEXT. THIS OBJECT 72](#_Toc106140482)

[FIRST-CLASS AND HIGHER-ORDER FUNCTIONS. CALLBACKS 73](#_Toc106140483)

[OUTPUT 75](#_Toc106140484)

[RECEIVE DATA FROM A WEB SERVER 75](#_Toc106140485)

[JSON (JAVASCRIPT OBJECT NOTATION) 75](#_Toc106140486)

[XML 76](#_Toc106140487)

[HTML DOM (DOCUMENT OBJECT MODEL) 77](#_Toc106140488)

[MAIN CONCEPTS 77](#_Toc106140489)

[IN THE BROWSER 77](#_Toc106140490)

[JAVASCRIPT IN HTML 78](#_Toc106140491)

[THE DOCUMENT OBJECT 78](#_Toc106140492)

[ACCESS DOM NODES (NULL IF NOT FOUND) 79](#_Toc106140493)

[GET NUMBER OF ELEMENTS 80](#_Toc106140494)

[ACCSESS THE ATTRIBUTES OF A DOM NODE 80](#_Toc106140495)

[CHANGE THE VALUE OF A DOM NODE ATTRIBUTE 81](#_Toc106140496)

[SORT HTML COLLECTIONS 82](#_Toc106140497)

[MODIFY DOM NODES 83](#_Toc106140498)

[CREATE/DELETE DOM ELEMENTS 83](#_Toc106140499)

[EVENTS 84](#_Toc106140500)

[VALIDATE INPUT 86](#_Toc106140501)

[DOM ANIMATIONS 86](#_Toc106140502)

[GOOGLE APIS 86](#_Toc106140503)

[JQUERY 87](#_Toc106140504)

[JQUERY INTRODUCTION 87](#_Toc106140505)

[JQUERY SELECTORS 87](#_Toc106140506)

[JQUERY FILTERING 88](#_Toc106140507)

[JQUERY EVENTS 88](#_Toc106140508)

[EVENT TYPES 88](#_Toc106140509)

[EVENT EXAMPLES 88](#_Toc106140510)

[JQUERY EFFECTS 89](#_Toc106140511)

[JQUERY HTML 90](#_Toc106140512)

[JQUERY AJAX 90](#_Toc106140513)

[JQUERY MISC 91](#_Toc106140514)

[JS BROWSER OBJECT MODEL (BOM) 91](#_Toc106140515)

[MAIN CONCEPTS 91](#_Toc106140516)

[WEB API 95](#_Toc106140517)

[MAIN CONCEPTS 95](#_Toc106140518)

[ERRORS IN JS 99](#_Toc106140519)

[THROWING ERRORS (EXCEPTIONS) 99](#_Toc106140520)

[TESTING THE CODE 99](#_Toc106140521)

[MODULES IN JS 100](#_Toc106140522)

[TYPES OF MODULES 100](#_Toc106140523)

[EXPORT AND IMPORT MODULES 101](#_Toc106140524)

[MODULE WRAPPER FUNCTION 102](#_Toc106140525)

[MOCHA AND CHAI 102](#_Toc106140526)

[INSTALLATION 102](#_Toc106140527)

[WRITE AND RUN TESTS 102](#_Toc106140528)

[WEB APPLICATIONS: DESIGN MODELS 103](#_Toc106140529)

[WEB APPLICATIONS: FRONT END 103](#_Toc106140530)

[HTTP AND REST SERVICES 103](#_Toc106140531)

[HTTP REQUESTS / Methods 103](#_Toc106140532)

[HTTP RESPONSES 104](#_Toc106140533)

[RESTFUL SERVICE/API 104](#_Toc106140534)

[AJAX 105](#_Toc106140535)

[ASYNCHRONOUS PROGRAMMING 105](#_Toc106140536)

[PROMISES 105](#_Toc106140537)

[SET TIME OUT 106](#_Toc106140538)

[XHR OBJECTS (XML HTTP REQUEST) 107](#_Toc106140539)

[FETCH API 108](#_Toc106140540)

[ASYNC FUNCTIONS 109](#_Toc106140541)

[REMOTE DATA AND AUTHENTICATION 110](#_Toc106140542)

[SINGLE PAGE APPLICATION (SPA) 112](#_Toc106140543)

[WEB APPLICATIONS: BACK END 113](#_Toc106140544)

[BAAS 113](#_Toc106140545)

[NODE.JS 114](#_Toc106140546)

[NODE.JS CORE MODULES 114](#_Toc106140547)

[URL MODULE 114](#_Toc106140548)

[QUERYSTRING MODULE 115](#_Toc106140549)

[PATH MODULE 115](#_Toc106140550)

[OPERATING SYSTEM MODULE 115](#_Toc106140551)

[FILE SYSTEM MODULE 115](#_Toc106140552)

[EVENTS MODULE 117](#_Toc106140553)

[HTTP MODULE 118](#_Toc106140554)

[EXPRESS.JS 118](#_Toc106140555)

[INSTALLATION AND MAIN CHARACTERISTICS 118](#_Toc106140556)

[ROUTER PATHS 120](#_Toc106140557)

[ROUTER RESPONSES 120](#_Toc106140558)

[MIDDLEWARE (PLUGINS, EXTENSIONS) 121](#_Toc106140559)

[STATIC FILES 122](#_Toc106140560)

[VIEW ENGINES 122](#_Toc106140561)

[HANDLEBARS 122](#_Toc106140562)

[JOI 124](#_Toc106140563)

[EXTERNAL LIBRARIES 124](#_Toc106140564)

[STREAMS 125](#_Toc106140565)

[PUBLISH-SUBSCRIBE PATTERN 126](#_Toc106140566)

[DATABASES 126](#_Toc106140567)

[TYPES OF DATABASES 126](#_Toc106140568)

[MONGO DB 127](#_Toc106140569)

[MONGOOSE 128](#_Toc106140570)

[INSTALLATION AND CONNECTION TO DB 128](#_Toc106140571)

[CREATE A MODEL. VALIDATION 128](#_Toc106140572)

[READ DATA 129](#_Toc106140573)

[UPDATE DATA 130](#_Toc106140574)

[REMOVE DATA 131](#_Toc106140575)

[COUNT DOCUMENTS (ENTRIES) 131](#_Toc106140576)

[REFERENCES AND POPULATION 131](#_Toc106140577)

[APPLICATION SECURITY 132](#_Toc106140578)

[COOKIES 132](#_Toc106140579)

[SESSIONS 132](#_Toc106140580)

[AUTHENTICATION 133](#_Toc106140581)

[JSON WEB TOKEN 134](#_Toc106140582)

[VALIDATION AND ERROR HANDLING 135](#_Toc106140583)

[VALIDATOR.JS 135](#_Toc106140584)

[EXPRESS VALIDATOR 135](#_Toc106140585)

[MONGOOSE VALIDATOR 137](#_Toc106140586)

[EXPRESS REST API 137](#_Toc106140587)

[ANGULAR JS 138](#_Toc106140588)

[PURPOSE AND INSTALLATION 138](#_Toc106140589)

[TEMPLATES AND DATA BINDING 139](#_Toc106140590)

[MODULES 141](#_Toc106140591)

[COMPONENTS 142](#_Toc106140592)

[DIRECTIVES 143](#_Toc106140593)

[SERVICES 145](#_Toc106140594)

[REACTIVE PROGRAMMING AND RxJS 145](#_Toc106140595)

[HTTP CLIENT 147](#_Toc106140596)

[PIPES 148](#_Toc106140597)

[INTERCEPTORS 149](#_Toc106140598)

[LAZY LOADING 150](#_Toc106140599)

[ROUTING 150](#_Toc106140600)

[ABOUT ROUTING 150](#_Toc106140601)

[USE A ROUTING MODULE 151](#_Toc106140602)

[FETCH QUERY PARAMETERS 152](#_Toc106140603)

[ROUTE GUARDS 152](#_Toc106140604)

[ROUTER RESOLVER 153](#_Toc106140605)

[HANDLING FORMS 153](#_Toc106140606)

[TEMPLATE-DRIVEN FORMS 153](#_Toc106140607)

[REACTIVE FORMS 156](#_Toc106140608)

[STATE MANAGEMENT AND NGRX 158](#_Toc106140609)

[NGRX STORE 158](#_Toc106140610)

[NGRX STORE DEVTOOLS 161](#_Toc106140611)

[NGRX EFFECTS 161](#_Toc106140612)

[AN ANGULAR APP 161](#_Toc106140613)

[START WORKING 161](#_Toc106140614)

[CREATE MODULES FOR THE APP PARTS 162](#_Toc106140615)

[CREATE COMPONENTS FOR THE APP VIEWS 163](#_Toc106140616)

[ADD ROUTING 163](#_Toc106140617)

[CREATE INTERFACES 164](#_Toc106140618)

[CREATE SERVICES 164](#_Toc106140619)

[CREATE INTERCEPTORS AND PIPES 165](#_Toc106140620)

[AUTHENTICATION AND GUARDS 167](#_Toc106140621)

[RENDER CONTENT 168](#_Toc106140622)

[CREATE CUSTOM VALIDATION 170](#_Toc106140623)

[LOGIN/REGISTER FORMS (REACTIVE) 171](#_Toc106140624)

[ADD/EDIT ITEM FORMS (TEMPLATE-DRIVEN) 172](#_Toc106140625)

[DOCUMENTATION 173](#_Toc106140626)

[TYPESCRIPT 173](#_Toc106140627)

[PURPOSE AND INSTALLATION 173](#_Toc106140628)

[TYPE ANNOTATIONS 174](#_Toc106140629)

[TYPE INFERENCE 175](#_Toc106140630)

[TYPE ASSERTION 175](#_Toc106140631)

[FUNCTIONS 175](#_Toc106140632)

[INTERFACES 176](#_Toc106140633)

[CLASSES 177](#_Toc106140634)

[GENERICS 179](#_Toc106140635)

[MODULES 180](#_Toc106140636)

[NAMESPACES 180](#_Toc106140637)

[REACT JS 181](#_Toc106140638)

[PURPOSE, INSTALLATION, SETTINGS 181](#_Toc106140639)

[CREATE A REACT APP 182](#_Toc106140640)

[JSX 182](#_Toc106140641)

[ELEMENTS 183](#_Toc106140642)

[COMPONENTS 184](#_Toc106140643)

[GENERAL 184](#_Toc106140644)

[PASS DATA FROM PARENT TO CHILD 185](#_Toc106140645)

[PRESERVE STATE BETWEEN RERENDERS 186](#_Toc106140646)

[CHANGE THE PARENT'S STATE FROM A CHILD 186](#_Toc106140647)

[COMPONENT LIFECYCLE HOOKS/METHODS 187](#_Toc106140648)

[HANDLING EVENTS 188](#_Toc106140649)

[CONDITIONAL RENDERING 189](#_Toc106140650)

[LISTS AND KEYS 189](#_Toc106140651)

[CSS STYLES 189](#_Toc106140652)

[IMAGE FILES 190](#_Toc106140653)

[ROUTING 191](#_Toc106140654)

[PURPOSE AND INSTALLATION 191](#_Toc106140655)

[IMPLEMENT ROUTING 191](#_Toc106140656)

[ADD NAVIGATION AND LINKS 192](#_Toc106140657)

[ACCESS DATA FROM URL 193](#_Toc106140658)

[REDIRECT 193](#_Toc106140659)

[NESTED ROUTES 194](#_Toc106140660)

[LAZY LOADING 195](#_Toc106140661)

[FORMS 195](#_Toc106140662)

[UNCONTROLLED COMPONENTS 195](#_Toc106140663)

[CONTROLLED COMPONENTS 196](#_Toc106140664)

[HOOKS 197](#_Toc106140665)

[CONTEXT 198](#_Toc106140666)

[ADVANCED TECHNIQUES 199](#_Toc106140667)

[HIGHER-ORDER COMPONENTS 199](#_Toc106140668)

[USEREDUCER 199](#_Toc106140669)

[ERROR BOUNDARIES 200](#_Toc106140670)

[JEST 200](#_Toc106140671)

[REACT QUERY 200](#_Toc106140672)

[REACT REMEMBERS 200](#_Toc106140673)

[CREATING A REACT APP 201](#_Toc106140674)

[CREATE THE BASE 201](#_Toc106140675)

[DEPLOYING AN APPLICATION 201](#_Toc106140676)

[FIREBASE 201](#_Toc106140677)

[HEROKU 201](#_Toc106140678)

# INTRODUCTION

|  |  |  |
| --- | --- | --- |
| About JavaScript | JavaScript (JS) is a programming language, i.e., a set of rules that converts lines of text into instructions executable by a computer. These instructions are called **source code**. | |
| JavaScript in the Front End | JavaScript is known as the programming language of the Web. It is mostly used to program the behavior of web pages built with HTML and CSS, and particularly the part of them that a user will interact with – the client side, or **front end**.  No need to download JavaScript for the front end – it is already running in any browser on any device. | |
| Web Pages Without JavaScript | Without JavaScript, web pages are but on-screen magazines – they have some content that they cannot change according to the reader's actions.  Here, the user is clicking the button, yet nothing happens. |  |
| Web Pages With JavaScript | JavaScript can make the pages actually **do something** when interacted with.  Here, a simple JS function is executed when the button is clicked, making a pop-up window appear. Another function saves the input from the user and displays a welcome message with it. | request to the back end |
| JavaScript in the Back End | JavaScript can also be used in back-end development that focuses on the server side of a web page (invisible to the users). The **back end** communicates with the front end, sending and receiving information to be displayed as a web page.  **Node.js** (needs to be downloaded) is the environment that allows us to use JavaScript outside of the browser, just like the other programming languages. | a function in the back end checks the database for a user named "John"  the record exists and the back end sends the front end a response that is not an error |
| Full-Stack JavaScript | When we use JavaScript both in the front end and back end, we say we are using **full-stack** JavaScript and we are full-stack developers (programmers). | |
| Documentation | <https://developer.mozilla.org/en-US/docs/Web/JavaScript>  <https://nodejs.org/api/> | |

# WHERE TO USE JAVASCRIPT

|  |  |  |
| --- | --- | --- |
| HTML Document | 1. Create an HTML document in a text editor. 2. Add a <script> tag. 3. Write some code in JavaScript in it. 4. Open the document in a browser to see the result. |  |
| Console Window | 1. Download and install Node.js. 2. Press Win, type cmd and press Enter to open the console window. 3. Type node and press Enter. 4. Type anything in JavaScript. 5. Press Enter to execute each line of code. |  |
| Browser Console | 1. Press F12 to open the browser Developer Tools. 2. Open the "Console" tab. 3. Type anything in JavaScript. 4. Press Enter to execute each line of code. |  |
| Integrated Development Environments (IDE) | To use JS in **Visual Studio Code**:   1. Create a .js file in VSC. 2. Type anything in JavaScript in it. 3. Press F5 to run the program and see the output in the Debug Console |  |

# FILE NAME CONVENTIONS

|  |  |  |
| --- | --- | --- |
| Most JS Files | **Camel case**: each word after the first one starts with an uppercase letter. |  |
| Classes and Database Models | **Pascal case**: all words start with an uppercase letter. |  |

# SOURCE CODE COMPONENTS

|  |  |  |  |
| --- | --- | --- | --- |
| Variables | A variable is a **container that stores data**. Created with a keyword const/let and a name, it can be used instead of its value. | const x = 5; | const name = 'Jim'; |
| > x + 1  < 6 | > name < 'Jim' |
| Expressions | An expression is code that **evaluates to a single value**. It can contain variables, operators or just a simple value.  In the browser console, it returns its value on the next line. | > x = 5 < 5 | > x + 1 < 6 |
| > 3 < 3 | > 'text' < 'text' |
| Statements | A statement is a line of code that **performs an action** and is written with a semicolon at the end. Some expressions can be transformed into statements with the semicolon.  In the browser console, statements return undefined on the next line. | > const x = 5;  < undefined | |
| > console.log('hi');  hi  < undefined | |
| Code Blocks | Code blocks are **parts of the code that are grouped together** (often enclosed in {}).  They don't take a semicolon at the end. | if (x < 7) { *code block* } | |
| for (i = 0; i < 3; i++) { *code block* } | |
| Functions | A function is similar to a procedure – a **set of statements that performs a task**. In the browser console, function declarations return undefined on the next line. | > function sum(a, b) {  return a + b;  }  < undefined | |
| In order to perform the task they were created for, **functions need to be called** (executed), taking some input and returning an output. | > sum(3, 2);  < 5 | |

# IDENTIFIERS (VARIABLE NAMES)

|  |  |  |
| --- | --- | --- |
| General Convention | **Camel case**: each word after the first one starts with an uppercase letter. | const firstName = 'John'; |
| Numbers | We can use numbers in identifiers, but **not in the beginning**. | > const 1stName = 'John'; |
| Hyphens | We **cannot use hyphens** in identifiers. | > const first-name = 'John'; |
| Case-Sensivite | JS identifiers are **case-sensitive**: firstName and firstname are different variables. | > const firstName = 'John';  const firstname = 'Peter';   firstName  < 'John' |

# WRITING A PROGRAM: BASIC RULES

|  |  |  |
| --- | --- | --- |
| Optimal Line Length | No longer than **80 characters**.  Some developers break it after an operator, others – before. | const fullHeight = innerHeight +  border; |
| const fullHeight = innerHeight  + border; |
| Comments | Comments are **non-executable lines of code** written in order to make the source code easier for other developers to understand. | // a single-line comment |
| /\* a multiline comment \*/ |
| Strict Mode | Normally, JavaScript allows some mistakes that might cause problems later on, e.g., declaring variables without a keyword (const/let). | > myNumber = 5; // no keyword → OK < 5 |
| Strict mode **turns bad syntax into actual errors**, helping developers write cleaner and safer code. | > 'use strict'; // strict mode  myNumber = 5; // no keyword → error |

# DATA TYPES & VALUES

## ABOUT JAVASCRIPT DATA TYPES

|  |  |  |  |
| --- | --- | --- | --- |
| Data Types | Any information (piece of data) represented in JavaScript is of certain data type that tells the computer system how to interpret its value.  The main data types in JavaScript are **boolean**, **string**, **number**, **undefined**, **null**, and **object**. | boolean | true |
| string | 'John' |
| number | 5 |
| Dynamic Types | Unlike other programming languages, JavaScript types are **dynamic**, which means that the same variable can be used to hold different data types. | let x = 5; // x is a number  x = 'John'; // x is a string | |

## JAVASCRIPT DATA TYPES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Types That Can Contain Values | boolean | can have two values: **true** or **false** | true | false |
| string | a **sequence of characters** enclosed in single/double quotation marks or backticks | 'John' "John" | `John` |
| number | whole **numbers** or **decimals** | 4 | 48.5 |
| bigint | used for integers **beyond the safe integer limit**: ±(253 - 1) | 1n | BigInt(1) |
| symbol | a **unique and immutable** value; may be the key of an object property | Symbol() | Symbol('a') |
| object | stores **multiple values**; there are many types of objects in JS: **object**, **array**, **map**, **set**, **date**, etc. | { name: 'John', age: 25 } | |
| [1, 2, 3] // array new Date() // date | |
| function | a set of statements **that performs a task**, taking some input and returning an output | function sum(a, b) {  return a + b; } | |
| Data Types That Cannot Contain Values | undefined | has the value undefined, automatically set for variables which **haven't been assigned a value** yet | > let name;  name  < undefined | |
| null | has the value null (the **intentional absence of any value**) | null | |

## PRIMITIVE/REFERENCE DATA TYPES

|  |  |  |  |
| --- | --- | --- | --- |
|  | Primitive Data Types | | Reference Data Types |
| Data Types | **boolean**, **string**, **number**, **bigint**, **symbol**, **undefined** and **null** | | **objects** and **functions** |
| Values | **without properties** or methods | | **with properties** or methods |
| Size in Memory | fixed | | not fixed |
| Accessed | **by value**: the variable is a name associated with a value | | **by reference**: the variable name is associated with a reference to a location in the memory where the value is |
| Using the Assignment Operator (=) | Creates a **copy of the value**. When we change the copy, the original value does not change. | | Makes a **copy of the reference** to the same location in the memory. When we change the copy, we actually change the original through the copied reference. |
| Example | > let hometown = 'London';  let residence = hometown;  residence = 'NY';   hometown < 'London'  > residence < 'NY' | > let residence = { code: 'NY' };  let payingTaxesIn = residence;  payingTaxesIn.code = 'CA';   residence < ► *{code: 'CA'}*  > payingTaxesIn < ► *{code: 'CA'}* | |

## GET THE DATA TYPE OF A VALUE

|  |  |  |  |
| --- | --- | --- | --- |
| Data Type | The typeof Operator | | The constructor Method |
| Primitive Data Types That Can Hold a Value | > typeof true < 'boolean' | | > true.constructor < *f Boolean() { [native code] }* |
| > typeof 'John' < 'string' | | > 'John'.constructor < *f String() { [native code] }* |
| > typeof 48 < 'number' | > typeof NaN < 'number' | > (48).constructor < *f Number() { [native code] }* |
| > typeof 1n < 'bigint' | | > 1n.constructor < *f BigInt() { [native code] }* |
| > typeof Symbol('a') < 'symbol' | | > Symbol('a').constructor < *f Symbol() { [native code] }* |
| Primitive Data Types That Cannot Hold a Value | > typeof undefined < 'undefined' | | > undefined.constructor |
| > typeof null < 'object' // legacy reasons; null is NOT an object | | > null.constructor |
| Reference Data Types | > typeof { name: 'Ben', age: 13 } < 'object' | | > ({ name: 'Ben', age: 13 }).constructor < *f Object() { [native code] }* |
| > typeof [1, 2, 3] < 'object' | | > [1, 2, 3].constructor < *f Array() { [native code] }* |
| > typeof function myFunc() {} < 'function' | | > (function myFunc() {}).constructor < *f Function() { [native code] }* |

## CHECK WHETHER A VALUE IS OF SPECIFIC DATA TYPE

|  |  |  |
| --- | --- | --- |
| Data Type | The typeof Operator | The constructor Method |
| Primitive Data Types That Can Hold a Value | > typeof true == 'boolean' < true | > true.constructor == Boolean < true |
| > typeof 'John' == 'string' < true | > 'John'.constructor == String < true |
| > typeof 23 == 'number' < true | > (23).constructor == Number < true |
| Primitive Data Types That Cannot Hold a Value | > typeof x == 'undefined'  < true | // not working |
| > typeof null == 'null'  < false // typeof null → 'object' > null === null  < true | // not working |
| Reference Data Types | // all types of object return 'object' | > ({}).constructor == Object < true |
| > [].constructor == Array < true |
| > new Date().constructor == Date < true |
| > typeof (() => {}) == 'function' < true | > (() => {}).constructor == Function < true |

## CONVERT DATA TYPES. TYPE COERCION

### EXPLANATION AND EXAMPLES

|  |  |  |
| --- | --- | --- |
| About | In programming, we often need to change (or convert) one data type into another. This conversion can be **implicit**, executed behind the scenes by the programming language itself, or **explicit**, executed by the programmer. | |
| Implicit Type Conversion | The implicit type conversion is also called **coercion**. It happens when you apply **operators** to values of different types or it can be triggered by the **surrounding context**. | |
|  | In this case, the arithmetic opertator \* expects numbers as operands and **converts the string** '5' **into the number** 5 so that it can be multiplied by 2. | > 2 \* '5' < 10 |
|  | Here, the comparison operator ==, One operator that does not trigger implicit type coercion is ===, which is called the strict equality operator. |  |
|  | or it can be triggered by the surrounding context, like with if (value) { ... }, where value is coerced to boolean. |  |

### CONVERT OTHER TYPES TO BOOLEAN

|  |  |  |
| --- | --- | --- |
|  |  |  |
| String to Boolean | > Boolean('0') < true | > Boolean('') < false |
| Number to Boolean | > Boolean(3) < true | > Boolean(0) < false |
| > Boolean(Infinity) < true | > Boolean(NaN) < false |
| BigInt to Boolean | > Boolean(3n) < true | > Boolean(0n) < false |
| Symbol to Boolean | > Boolean(Symbol()) < true | > Boolean(Symbol(0)) < true |
| Undefined and Null to Boolean | > Boolean(null) < false | > Boolean(undefined) < false |
| Object to Boolean | > Boolean({}) < true | > Boolean([]) < true |
| Type Coercion | All values are coerced to boolean in a boolean context – the condition in an if – else statement or loop or one of the operands of a logical operator.  Values that are coerced to true in a boolean context are called truthy values, those coerced to false – falsy. The falsy values in JS are false, 0, -0, 0n, '', null, undefined and NaN. | if (3) { ... }  // 3 is coerced to true |
| > 0n || 'а' // 0n is coerced to false < 'а' |
| > !null // null is coerced to false < true |
| > !!5 // 5 is coerced to true < true |

### CONVERT OTHER TYPES TO STRING

|  |  |  |
| --- | --- | --- |
| Boolean to String | > String(true) < 'true' | > true.toString() < 'true' |
| Number to String | > String(3) < '3' | > (3).toString() < '3' |
| BigInt to String | > String(1n) < '1' | > 1n.toString() < '1' |
| Symbol to String | > String(Symbol('a')) < 'Symbol(a)' | > Symbol('a').toString() < 'Symbol(a)' |
| Object to String | > String({ name: 'Ben', age: 3 }) < '[object Object]' | > JSON.stringify({ name: 'Ben', age: 3 }) < '{"name":"Ben","age":3}' |
| Array Object to String | > [1, [2, 3]].toString() < '1,2,3' | > JSON.stringify([1, [2, 3]]) < '[1,[2,3]]' |
| Date Object to String | > new Date().toString() < 'Tue May 03 2022 17:22:54 GMT+0300 (Eastern European Summer Time)' | > new Date().toISOString() < '2022-05-03T14:26:16.888Z' |
| > new Date().toUTCString() < 'Tue, 03 May 2022 14:25:04 GMT' | > new Date().toDateString() < 'Tue May 03 2022' |
| Map and Set Object to String | > new Map([]).toString() < '[object Map]' | > JSON.stringify(new Map([[1, 1]])) < '{}' |
| RegExp Object to String | > String(/[a-z.\*]/) < '/[a-z.\*]/' | > JSON.stringify(new RegExp('/[a-z.\*]/')) < '{}' |
| Function to String | > String((a) => { return a + 2; }) < '(a) => { return a + 2; }' | > ((a) => { return a + 2; }).toString() < '(a) => { return a + 2; }' |

### CONVERT OTHER TYPES TO NUMBER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Boolean to Number | > Number(false) < 0 | | > Number(true) < 1 | |
| String to Number | > Number('3.14') < 3.14 | | > Number('') < 0 | |
| > parseInt('3.1') < 3 | > parseInt('3a') < 3 | > parseFloat('3.14') < 3.14 | > parseFloat('3.1a') < 3.1 |
| > Number('John') < NaN // Not a Number | | | |
| BigInt to Number | > Number(12n) < 12 | | | |
| Object to Number | > Number({}) < NaN | | | |
| Array Object to Number | > Number([20]) < 20 | > Number([]) < 0 | > Number([10, 20]) < NaN | |
| Date Object to Number | > Number(new Date('Mar 25 2015')) < 1427234400000 // milliseconds from Jan 01 1970 00:00:00 UTC | | | |

### CONVERT OTHER TYPES TO BIGINT

|  |  |  |
| --- | --- | --- |
| Number to BigInt | > BigInt(1) < 1n | > BigInt(0.5) |
| String to BigInt | > BigInt('') < 0n | > BigInt('5') < 5n |
| Boolean to BigInt | > BigInt(true) < 1n | > BigInt(false) < 0n |

### CONVERT OTHER TYPES TO OBJECT

|  |  |  |
| --- | --- | --- |
| About JavaScript Objects | In JS, almost everything is an object. Booleans, numbers and strings can be objects (when defined with the new keyword), objects and functions are always objects. | |
| Boolean to Object | The keyword new creates a Boolean object instead of a boolean primitive value. Not recommended. | > new Boolean(true) < *▼* *Boolean* *{true}*  ► [[Prototype]]: Boolean  [[PrimitiveValue]]: true |
| String to Object | The keyword new creates a String object instead of a string primitive value. Not recommended. | > new String('a') < *▼* *String* *{'a'}*  **0**: "a"  **length**: 1  ► [[Prototype]]: String  [[PrimitiveValue]]: 'a' |
| JSON String to Object | We can convert JSON strings into objects using JSON.parse(*string*). | > JSON.parse('{"name":"John","age":25}') < ► *{name: 'John', age: 25}* |
| Number to Object | The keyword new creates a Number object instead of a number primitive value. Not recommended. | > new Number(5) < *▼* *Number* *{5}*  ► [[Prototype]]: Number  [[PrimitiveValue]]: 5 |
| Array Object/Map to Object | The Object.fromEntries() method transforms a list of key-value pairs into an object.  The first element of every pair will be used as property key, the second is the value associated with that key. | > Object.fromEntries([  ['name', 'John'],  ['age', 25]  ]) < ► *{name: 'John', age: 25}* |
| > Object.fromEntries(new Map([  ['name', 'John'],  ['age', 25]  ])) < ► *{name: 'John', age: 25}* |

### CONVERT OTHER TYPES TO OBJECT ARRAY

|  |  |  |
| --- | --- | --- |
|  |  | |
|  |  |  |
|  |  |  |

### TYPE COERCION

|  |  |  |  |
| --- | --- | --- | --- |
| Coerce to String | > '' + 3 < '3' | | > `${3}` < '3' |
| Coerce to Number | > +'3' < 3 | > 2 \* '10' < 20 | > '4' < 14 // '4' is coerced to 4 < true |
| Coerce to Number as per ASCII Code (asciitable.com) | > 'John' <= 'a' < true // 74 < 97 | | > '2' >= '12' < true // 50 > 49 |
| Coerce to Boolean (Truthy/Falsy Values) | All values are coerced to boolean in a boolean context – the condition in an if – else statement or loop or one of the operands of a logical operator.  Values that are coerced to true in a boolean context are called truthy values, those coerced to false – falsy. The falsy values in JS are false, 0, -0, 0n, '', null, undefined and NaN. | |  |
|  |
|  |
|  |

# LIFE OF A VARIABLE

## UNDECLARED VARIABLES

|  |  |  |
| --- | --- | --- |
| Value and Type | An undeclared variable is assigned the value undefined at execution and is also of type *undefined*. | > typeof text < 'undefined' |
| Accessing the Value | A ReferenceError is thrown when we try to access a previously undeclared variable. | > text |

## DECLARE & INITIALIZE A VARIABLE

|  |  |  |
| --- | --- | --- |
| Keyword const | For all objects and most primitive data type variables (unless their value is about to change). | > const person = { name: 'Ben', age: 5 }; > const text = 'a'; |
| Keyword let | When the value is likely to change. | > let text = 'a'; |
| Keyword var | Used until ECMAScript 2015. No longer recommended. | > var text = 'a'; |
| No Keyword | Not possible in strict mode and not recommended. | > text = 'a'; |
| Multiple Variables | We can declare (and initialize) multiple variables at once. | > const text = 'a', number = 1; |
| Declare a Variable without a Value | We can create a variable without initializing it (not possible with const). The variable is automatically assigned the value undefined. | > let text; |

## ASSIGN VALUES. ASSIGNMENT OPERATORS

|  |  |  |
| --- | --- | --- |
| Assignment | > myVariable = 35; < 35 | |
| Addition Assignment | > myVariable += 5; < 40 // addition with numbers | > myVariable += 'a'; < '35a' // concatenation with strings |
| Increment Operator (+= 1) | > myVariable++; < 40 > myVariable < 41 | > ++myVariable; < 41 > myVariable < 41 |
| Subtraction Assignment | > myVariable -= 10; // subtracts from previous value < 31 | |
| Decrement Operator (-= 1) | > myVariable--; < 31 > myVariable < 30 | > --myVariable; < 30 > myVariable < 30 |
| Multiplication Assignment | > myVariable \*= 2; // multiplies previous value < 60 | |
| Exponentiation Assignment | > myVariable \*\*= 3; // raises previous value to the value of the right operand < 216000 | |
| Division Assignment | > myVariable /= 1000; // divides previous value < 216 | |
| Remainder Assignment | > myVariable %= 10; // divides previous value and assigns the remainder < 6 | |

## DESTRUCTURING ASSIGNMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| About Destructuring Assignment | The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables. | | | |
| Array Destructuring | Instead of declaring the variables one by one (const a = numbers[0];  const b = numbers[1];), we can use the destructuring syntax for arrays. | const numbers = [1, 2, 3]; const [a, b] = numbers; | | |
| > a < 1 | | > b < 2 |
| Object Destructuring Keeping the Name of the Property | Instead of declaring the variables one by one (const age = person.age;  const name = person.name;), we can use the destructuring syntax for arrays. | const person = { name: 'Ben', age: 5 }; const { age, name } = person; | | |
| > age < 5 | | > name < 'Ben' |
| Object Destructuring Changing the Name of the Property | If we want to give the variable a new name, we can add it after a colon (:). | const { name: firstName } = person; | | |
| > firstName < 'Ben' | | |
| Nested Destructuring | const person = {  name: 'John',  emails: ['jd@dot.de', 'john@doe.uk']  }; | const { emails:  [email]  } = person; | > email < 'jd@dot.de' | |

## REDECLARE VARIABLES

|  |  |  |  |
| --- | --- | --- | --- |
| Keyword const | As a rule, we cannot redeclare variables with const. |  | |
| We can now redeclare variables with const in the Chrome browser console. | > const text = 'a';  text < 'a' > const text = 'b';  text < 'b' | |
| Keyword let | We simply assign a new value.  When no new value is specified, the new value is undefined. | > let text = 'a';  text < 'a' > let text = 'b';  text < 'b' | > let text = 'a';  text < 'a' > let text;  text < undefined |
| Keyword var | We assign a new value.  When no new value is specified, the redeclaration is ignored. | > var text = 'a';  text < 'a' > var text = 'b';  text < 'b' | > var text = 'a';  text < 'a' > var text;  text < 'a' |

## SCOPE

|  |  |  |
| --- | --- | --- |
| Global Scope (NOT Recommended) | A variable declared outside a function can be used everywhere in the code. | > const text = 'a';  text < 'a' > (function myFunc() {  return text;  })(); < 'a' |
| A variable declared without a keyword can be used everywhere in the code. | > (function myFunc() {  text = 'a';  })();  text < 'a' |
| Function Scope | A variable declared in a function can only be used in this function. | > (function myFunc() {  const text = 'a';  })(); > text |
| Block Scope | A variable declared in a code block can only be used in this code block. | > for (let i = 1; i <= 2; i++) {  console.log(i);  }  1  2 > i |
| Variables declared with the keyword var don't have block scope. | > for (var i = 1; i <= 2; i++) {  console.log(i);  }  1  2 > i < 3 |
| Redeclare a Variable in Another Scope | When we redeclare a variable in a scope different than its original one, we are actually creating another variable with the same name. | > let i = 5;  for (let i = 1; i <= 2; i++) {  console.log(i);  }  1 // second variable named "i"  2 // second variable named "i" > i < 5 // first variable named "i" |

## HOISTING

|  |  |  |
| --- | --- | --- |
| About Hoisting | All variable declarations are hoisted to the top of their scope and processed before any code is executed. | |
| Keywords let and const | The hoisted variable remains uninitialized. | > console.log(hoist);  const hoist = 'hoisted'; |
| Keyword var | The hoisted variable is initialized with the value undefined. | > console.log(hoist);  var hoist = 'hoisted';  undefined |

# COMPARISONS & CONDITIONS

## OPERATORS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Comparison Operators | *equal to* | == | 5 == 5 // true | 5 == '5' // true |
| *equal in value and type* | === | 5 === 5 // true | 5 === '5' // false |
| *not equal* | != | 5 != 15 // true | |
| *not equal value or type* | !== | 5 !== '5' // true | |
| *greater than* | > | 5 > 4 // true | |
| *less than* | < | 5 < 4 // false | |
| *greater than or equal to* | >= | 5 >= 4 // true | |
| *less than or equal to* | <= | 5 <= 5 // true | |
| Logical Operators | *and*: the first falsy value or the last truthy if all are true | && | 1 < 2 && 2 <= 3 // true (2 <= 3) 1 > 2 && 2 <= 3 // false (1 > 2) | |
| *or*: the first truthy value or the last falsy if all are false | || | 1 < 2 || 2 <= 3 // true (1 < 2) 1 > 2 || 2 <= 3 // true (2 <= 3) | |
| *nullish coalescing operator*: the right-hand operand when the left-hand is null/undefined | ?? | 1 > 2 ?? 2 <= 3 // false (1 > 2) null ?? 2 <= 3 // true (2 >= 3) | |
| *not*: false if the operand is a truthy value, otherwise true | ! | !(1 < 2) // false !0 // true | !'D' // false  !![] // true |
| Ternary Operator | alternative to if – else | ?: | 1 < 2 ? 'less' : 'more' // 'less'  3 < 2 ? 'less' : 'more' // 'more' | |

## COMPARE DATA

|  |  |  |
| --- | --- | --- |
| Compare Numbers to Numbers/BigInts | > 1 < 2 < true | > 1 < 2n < true |
| > 1 == 1n < true | > 1 === 1n < false |
| Compare Numbers to Strings | > 0 < 'John' < false | > 0 < '2' < true // numeric strings → numbers |
| > 0 == '' < true // '' is coerced to 0 | > 0 === '' < false |
| Compare Strings | > '0' < 'John' < true // as per [ASCII table](https://www.asciitable.com/) | > 'John' <= 'a' < true // as per [ASCII table](https://www.asciitable.com/) |
| > '2' <= '12' < false // as per [ASCII table](https://www.asciitable.com/) | > '0' > '' < true // as per [ASCII table](https://www.asciitable.com/) |
| Compare Objects | > { age: 18 } == { age: 18 } < false | > [1, 2] == [1, 2] < false |
| > JSON.stringify({ age: 18 }) === JSON.stringify({ age: 18 }) < true | |

## IF – ELSE

|  |  |  |
| --- | --- | --- |
| If Statement | Part of the code is executed only if a certain condition is met. | const x = 1, y = 0; if (x > y) {  console.log('x is greater than y'); }  x is greater than y |
| A code block of one statement can be written on the same line without curly braces (NOT recommended). | > if (x > y) 'x is greater than y'; < x is greater than y |
| If – Else Statement | Another block of code is executed only if the condition is NOT met. | const x = 1, y = 2; if (x > y) {  console.log('x is greater than y'); } else {  console.log('y is greater than x');  }  y is greater than x |
| If – Else If – ... – Else Statement | Multiple conditions are evaluated one by one from top to bottom. | const x = 1, y = 1; if (x > y) {  console.log('x is greater than y'); } else if (y > x) {  console.log('y is greater than x');  } else {  console.log('x is equal to y');  }  x is equal to y |

## SWITCH

|  |  |  |
| --- | --- | --- |
| Switch Statement | An expression is evaluated against multiple possible cases and one or more code blocks are executed based on matching cases. | const number = 2; switch (number) {  case 1: console.log('Spring'); break;  case 2: console.log('Summer'); break;  case 3: console.log('Autumn'); break;  case 4: console.log('Winter'); break; }  Summer |
| The break Keyword | The break keyword ends the switch block after a match.  When omitted, all code blocks after the match are executed (until the end of the switch statement or until break;). | const number = 3; switch (number) {  case 1: console.log('Spring');  case 2: console.log('Summer');  case 3: console.log('Autumn');  case 4: console.log('Winter'); }  Autumn Winter |
| Switch Statement with a Default Case | When there is no match, the default code block is executed. | const number = 6; switch (number) {  case 1: console.log('Spring'); break;  case 2: console.log('Summer'); break;  case 3: console.log('Autumn'); break;  case 4: console.log('Winter'); break;  default: console.log('unknown'); break;  }  unknown |

# LOOPS

## FOR LOOP

|  |  |  |
| --- | --- | --- |
| Purpose and Basic Syntax | A block of code that is repeated until a specified condition evaluates to false. | for (*start*; *end*; *step*) {  // code to be repeated } |
| Start (Optional) | We initialize a variable setting the value for the first iteration of the loop. | let i = 1 |
| End (Optional) | When the condition for ending the loop evaluates to false, the loop will break. | i <= 2 |
| Step (Optional) | Describes how the value of the variable will change. Will be executed after each iteration of the loop. | i++ |
| Full For Loop | The variable i will have the value 1 for the first iteration of the loop. After each iteration, it will increase by 1 (i++). When it reaches 3, i <= 2 evaluates to false and the loop breaks. | for (let i = 1; i <= 2; i++) {  console.log(i); } 1 2 |
| Infinite Loop | If we set a condition that always evaluates to true or omit the condition altogether, the loop goes on forever. | for (let i = 1; i > 0; i++) {  // code here will be repeated forever }  for (let i = 1; ; i++) { ... } |

## WHILE LOOP

|  |  |  |
| --- | --- | --- |
| Purpose and Basic Syntax | A block of code that is repeated until a specified condition evaluates to false. | while (*condition*) {  // code to be repeated } |
| Condition | When the condition evaluates to false, the loop will break. | x < 0 |
| Full While Loop | The condition x > 0 is checked before every iteration and the code block is repeated until x reaches 0. Then x > 0 returns false and the loop is broken. | let x = 2;  while (x > 0) {  console.log(x);  x--; } 2 1 |
| While Loop with No Iterations | The condition x > 2 evaluates to false before the first iteration, so the code block is not executed at all. | let x = 2;  while (x > 2) {  console.log(x);  x--; } |
| Do – While Loop | The code block is executed once, then the condition is checked and the loop is broken. | let x = 2;  do {  console.log(x);  x--; } while (x > 2); 2 |

## BREAK A LOOP

|  |  |  |
| --- | --- | --- |
| Keyword break | Breaks the loop when a specified condition evaluates to true. | for (let i = 1; i <= 3; i++) {  if (i == 2) break;  console.log(i); } 1 |
| Keyword continue | Breaks only the current iteration of the loop when a specified condition evaluates to true. | for (let i = 1; i <= 3; i++) {  if (i == 2) continue;  console.log(i); }  1  3 |

# STRINGS. TEXT PROCESSING

## JS SPECIFICS

|  |  |  |
| --- | --- | --- |
| About Strings in JS | Strings are sequences of characters enclosed in single or double quotation marks ('' or ""). | 'John' "John" |
| Interpolation | We can use backticks (``) instead of quotation marks to create strings with placeholders. | const name = 'John';  > `Hello, ${name}!` < 'Hello, John!' |

## STRING CHARACTERS. STRING LENGTH

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Character Indexing | Strings are ordered sequences, the first character corresponding to index 0. | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | 'a | | b | c | d' | | 0 | 1 | 2 | 3 | | |
| Character Access | > 'abcd'[0] < 'a' | > 'abcd'[2] < 'c' | > 'abcd'.charAt() < 'a' | > 'abcd'.charAt(3) < 'd' |
| Get String Length | > 'abcd'.length < 4 | | > ''.length < 0 | |
| Loop Through String Characters | for (let i = 0; i < 'abc'.length; i++) {  console.log('abc'[i]);  }  'a'  'b'  'c' | | for (let char of 'abc') {  console.log(char);  }  'a'  'b'  'c' | |

## SEARCH FOR CHARACTER(S)

|  |  |  |
| --- | --- | --- |
| Search from the Beginning | > 'abcd'.indexOf('c') < 2 // 'c' is at index 2 | > 'abcd'.search('b') < 1 // 'b' is at index 1 |
| > 'Hello! Hello!'.indexOf('Hello') < 0 // 'Hello' is at index 0 | > 'Hello! Hello!'.search('Hello') < 0 // 'Hello' is at index 0 |
|  | > 'abcd'.indexOf('g') < -1 // not found | > 'abcd'.search('g') < -1 // not found |
| Search from Specified Start Index | > 'Hello! Hello!'.indexOf('Hello', 1) < 7 // searching from index 1 to the end | |
| Search for Last Occurrence | > 'Hello! Hello!'.lastIndexOf('o') < 11 // searching from the end | > 'Hello! Hello!'.lastIndexOf('o', 10) < 4 // searching from index 10 to 0 |
| Check if a String Contains Specified Character(s) | > 'Hello!'.includes('o') < true | > 'Hello!'.includes('E') < false // case-sensitive |
| Check if a String Begins/Ends with Specified Character(s) | > 'Hello!'.startsWith('Hell') < true | > 'Hello!'.endsWith('!') < true |

## EXTRACT STRING PARTS

|  |  |  |
| --- | --- | --- |
| Extracted Part | The substring() Method | The slice() Method |
| From Specified Start Index | > 'Hi, John!'.substring(4) < 'John!' | > 'Hi, John!'.slice(4) < 'John!' |
| Start to End Index: [*start* – *end*) | > 'Hi, John!'.substring(4, 8) < 'John' | > 'Hi, John!'.slice(4, 8) < 'John' |
| > 'Hi, John!'.substring(0, 0 + 2) < 'Hi' | > 'Hi, John!'.slice(0, 0 + 2) < 'Hi' |
| End Index to Start Index: (*end* – *start*] | > 'Hi, John!'.substring(8, 4) < 'John' | // does NOT work |
| Start Index from the End | // does NOT work | > 'Hi, John!'.slice(-5) < 'John!' |
| Start to End Index from the End: [*start* – *end*) | // does NOT work | > 'Hi, John!'.slice(-5, -1) < 'John' |

## COMBINE AND CHANGE STRINGS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| About String Change | Strings in JavaScript are immutable – they cannot change, we can only make new strings. | | > text = 'abc';  text[1] = 'B';  text < 'abc' | | > text = 'abc';  text = 'aBc'; // new string  text < 'aBc' |
| If we want to use the result of a change more than once, we must assign it to a variable. | | > text = 'abc';  text + 'd' < 'abcd'  > text < 'abc' | | > text = 'abc';  text += 'd'; // assignment < 'abcd'  > text < 'abcd' |
| Concatenation | The process of merging multiple strings. | | > 'a' + ' ' + 'b' < 'a b' | | > 'a'.concat(' b') < 'a b' |
| We can use it to break a long string into several lines. | | 'This is a very very long line' + ' that I should break.' | | |
| Remove Whitespace | > ' abc '.trim() < 'abc' | > ' abc '.trimStart() < 'abc ' | | | > ' abc '.trimEnd() < ' abc' |
| Repeat Character(s) | > 'abc'.repeat(2) < 'abcabc' | | | | |
| Add Characters to Reach Specified Length | > 'abc'.padStart(5, '\*') < '\*\*abc' | | | > 'abc'.padEnd(5, '\*') < 'abc\*\*' | |
| > ('\*'.repeat(5) + 'abc').slice(-5) < '\*\*abc' | | | > ('abc' + '\*'.repeat(5)).slice(0, 5) < 'abc\*\*' | |
| Replace Character(s) | > 'abcabc'.replace('a', 'A') < 'Abcabc' // only the first occurrence | | | | |
| Convert to Lower Case | > 'iPad'.toLowerCase() < 'ipad' | | | > 'iPad'.toLocaleLowerCase() < 'ipad' // uses current locale | |
| Convert to Upper Case | > 'iPad'.toUpperCase() < 'IPAD' | | | > 'iPad'.toLocaleUpperCase() < 'İPAD' // if browser language is TR | |

## ESCAPE SEQUENCES

|  |  |  |
| --- | --- | --- |
| Single Quote (Apostrophe): \' | > 'It's nice' | > 'It\'s nice' < "It's nice" |
| Double Quote: \" | > "He said "Hi!"" | > "He said \"Hi!\"" < 'He said "Hi!"' |
| Backslash: \\ | > '\\_/'  < '\_/' | > '\\\_/'  < '\\_/' |
| Newline: \n  Tab: \t | console.log('line1\nline2');  line1  line2 | console.log('word1\tword2');  word1 word2 |
| Unicode Code  (<https://unicode-table.com>) | \x + *2 hexadecimal digits* | > '\x61'  < 'a' // U+0061 is the code for 'a' |
| \u + *4 hexadecimal digits* | > '\u0100'  < 'Ā' // U+0100 is the code for 'Ā' |
| \u{*6 hexadecimal digits*} | > '\u{01f60d}'  < '😍' // U+1f60d is the code for '😍' |

## CHARACTER ENCODING

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Convert Characters to Corresponding Code ([UTF-16](https://www.fileformat.info/info/charset/UTF-16/list.htm)) | > 'abcd'.charCodeAt() < 97 // 'a' → 97 | > 'abcd'.charCodeAt(0) < 97 // 'a' → 97 | | | |
| > 'abcd'.charCodeAt(2) < 99 // 'c' → 99 | > 'abcd'.charCodeAt(9) < NaN // no character at index 9 | | | |
| Convert Special Characters to Corresponding Code ([UTF-16](https://www.fileformat.info/info/charset/UTF-16/list.htm) vs. [Unicode](https://unicode-table.com/en/)) | > '😍'.charCodeAt() < 55357 // UTF-16: NOT full code | > '😍'.codePointAt() < 128525 // Unicode: full code | | | |
| Convert Code to Character | > String.fromCharCode(97) < 'a' | > String.fromCodePoint(97) < 'a' | | | |
| Convert Code to Special Character ([UTF-16](https://www.fileformat.info/info/charset/UTF-16/list.htm) vs. [Unicode](https://unicode-table.com/en/)) | > String.fromCharCode(128525) < '' // UTF-16: incorrect character | > String.fromCodePoint(128525) < '😍' // Unicode: correct character | | | |
| Normalize Strings to Work with Special Characters | Some characters can be represented in more than one way, e.g., we can either use the code for 'ñ' or combine the codes for 'n' and '~'. | > const a = '\u00f1'; // 'ñ'  const b = '\u006e\u0303'; // 'n' + '~' | | | |
| > a  < 'ñ' | | > b  < 'ñ' | |
| The variables a and b look the same but have different values and even different lenghts. | > a == b  < false | > a.length  < 1 | | > b.length  < 2 |
| To solve this problem, we use the normalized form of the characters. | > a.normalize() == b.normalize()  < true | | | |

## COMPARE STRINGS

|  |  |  |
| --- | --- | --- |
| Compare Strings Directly ([Unicode](https://unicode-table.com/en/)) | Strings are coerced to the corresponding Unicode code. | > 'a' > 'B' < true // 'a' → 97, 'B' → 66 |
| > 'cafe' > 'café' < false // 'e' → 101, 'é' → 233 |
| The localeCompare() Method | Returns 0 for strings that remain in the same order, a positive number (often 1) when the reference string comes after the compare string and a negative number (often -1) when it comes before the compare string. | > 'b'.localeCompare('a') < 1 // 'b' comes after 'a' |
| > 'a'.localeCompare('a') < 0 // 'a' is the same as 'a' |
| > 'a'.localeCompare('A') < -1 // 'a' comes before 'A' |

# NUMBERS & BIGINTS

## JS SPECIFICS

|  |  |  |  |
| --- | --- | --- | --- |
| About Numbers in JS | JavaScript has two types of numbers: *number* (integers or floating-point) and *bigint* (big integers). | | |
| Number Storage | JavaScript numbers are always stored as double precision floating-point numbers in 64 bits, 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. | | |
| Number Value Range | Number.MAX\_VALUE is the largest positive number possible to represent in JS. Any higher number is considered Infinity. | > Number.MAX\_VALUE < 1.7976931348623157e+308  // 1.7976931348623157 \* 10308 | |
| Number.MAX\_SAFE\_INTEGER is the largest integer which can be used safely in calculations. | > Number.MAX\_SAFE\_INTEGER < 9007199254740991 // 253 - 1  > Number.MAX\_SAFE\_INTEGER + 1 < 9007199254740992  > Number.MAX\_SAFE\_INTEGER + 2 < 9007199254740992 | |
| Number.MIN\_VALUE is the smallest positive number possible to represent in JS. | > Number.MIN\_VALUE < 5e-324 // 5 \* 10-324 | |
| Number.MIN\_SAFE\_INTEGER is the smallest integer which can be used safely in calculations. | > Number.MIN\_SAFE\_INTEGER < -9007199254740991 // -(253 - 1)  > Number.MIN\_SAFE\_INTEGER - 1 < -9007199254740992  > Number.MIN\_SAFE\_INTEGER - 2 < -9007199254740992 | |
| BigInt | Used for integers outside the number value range. Cannot be used with methods of the built-in Math object. | > 9007199254740991n + 2n < 9007199254740993n | |
| Integer Precision | JS integers are accurate up to 15 digits. | > 999999999999999 < 999999999999999 | > 9999999999999999 < 10000000000000000 |
| BigInt Precision | JS bigints are accurate even with more than 15 digits. | > 999999999999999999999n < 999999999999999999999n | |
| Floating Precision | Floating-point arithmetic is not always accurate. | > 0.1 + 0.2 < 0.30000000000000004 > (0.1 \* 10 + 0.2 \* 10) / 10 < 0.3 | |

## ARITHMETIC OPERATIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Addition | > 1 + 2 < 3 | > 1 + 2.345 < 3.345 | |
| Subtraction | > 1 - 2 < -1 | > 1 - 2.345 < -1.3450000000000002 | |
| Multiplication | > 1 \* 2 < 2 | > 2 \* 2.345 < 4.69 | |
| Division | > 1 / 2 < 0.5 | > 1 / 0 < Infinity | > -1 / 0 < -Infinity |
| Remainder | > 1 % 2 < 1 // remainder of 1 / 2 | > 10 % 2.6 < 2.1999999999999997 | |
| Exponentiation | > 3 \*\* 2 < 9 // raises 3 to the power of 2 | > Math.pow(3, 2) < 9 | |
| Square Root | > Math.sqrt(9) < 3 | > Math.sqrt(-1) < NaN | |
| Cube Root | > Math.cbrt(8) < 2 | > Math.cbrt(-1) < -1 | |
| Absolute Value | > Math.abs(2) < 2 | > Math.abs(-2) < 2 | |
| Parentheses | > 1 + 2 \* 3 < 7 | > (1 + 2) \* 3 < 9 | |
| Find Largest/Smallest Number | > Math.max(2, 5, -32) < 5 | > Math.min(2, 5, -32) < -32 | |

## ARITHMETIC OPERATIONS WITH BIGINTS

|  |  |  |
| --- | --- | --- |
| Number and BigInt | > 1n + 2 | |
| Convert Number to BigInt | > 1n + 2n // append "n" < 3n | > 1n + BigInt(2) < 3n |
| Convert BigInt to Number | > Number(1n) + 2 < 3 | > Number(5n) / 2 < 2.5 |
| BigInt Division | > 4n / 2n < 2n | > 5n / 2n < 2n // truncated |

## ARITHMETIC OPERATIONS WITH STRINGS

|  |  |  |  |
| --- | --- | --- | --- |
| Addition/Concatenation | > 1 + 1 + '2' < '22' | > '2' + 1n + 1  < '211' | |
| Subtraction | > '5' - 2 < 3 // '5' was coerced to 5 | > '5' - '2' < 3 | |
| Multiplication | > '5' \* 2 < 10 | > '5' \* '2' < 10 | |
| Division | > '5' / 2 < 2.5 | > '5' / '2' < 2.5 | |
| Remainder | > '5' % 2 < 1 | > '5' % '2' < 1 | |
| Exponentiation | > '5' \*\* 2 < 25 | > Math.pow('5', '2') < 25 | |
| Non-Numeric Strings | > 1 + 'a' < '1a' | > 1 - 'a' < NaN | > 1 \* 'a' < NaN |

## FORMAT NUMBERS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Round to Nearest Integer | > Math.round(2.4) < 2 | > Math.round(2.5) < 3 | > Math.round(-2.5) < -2 | | > Math.round(-2.6) < -3 |
| Round Down | > Math.trunc(2.9) < 2 | > Math.floor(2.9) < 2 | > parseInt('2.9') < 2 | | > 2.9 | 0 < 2 |
| Round Up | > Math.ceil(2.1) < 3 | | > Math.ceil(-2.1) < -2 | | |
| Round to Specific Decimal Place | > 2.346.toFixed(2) < '2.35' | > -2.5.toFixed() < -3 | > 2.35.toExponential(1) < '2.4e+0' // 2.4 \* (10 \*\* 0) | | |
| > 2.346.toLocaleString(undefined, { minimumFractionDigits: 2,  maximumFractionDigits: 2 }) < '2.35' | | | | |
| Round to Specific Digit | > 2.35.toPrecision(2) < '2.4' | | > 9.5.toPrecision(1) < '1e+1' // 1 \* (10 \*\* 1) | | |
| Remove Trailing Zeroes | > Number(2.000) < 2 | | > parseFloat('2.000') < 2 | | |
| Set Number of Significant Digits | > (2346).toLocaleString(undefined, { maximumSignificantDigits: 2 }) < '2,300' | | | | |
| Exponential Notation | > 123e5 < 12300000 // 123 \* (10 \*\* 5) | | | > 123e-5 < 0.00123 // 123 \* (10 \*\* -5) | |
| > (123000).toExponential() < '1.23e+5' // 1.23 \* (10 \*\* 5) | | | > 1.23.toExponential(1) < '1.3e+0' // 1.3 \* (10 \*\* 0) | |
| Language-Sensitive Number Formatting | > 1234.56.toLocaleString() < '1,234.56' | | | > 1234.56.toLocaleString('bg') < '1234,56' | |
| Currency Format | > 1234.56.toLocaleString('de', { style: 'currency', currency: 'EUR' }) < '1.234,56 €' | | | | |
| > 1234.56.toLocaleString('bg', { style: 'currency', currency: 'BGN' }) < '1234,56 лв.' | | | | |

## CHECK WHETHER A NUMBER IS OF SPECIFIC TYPE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Integer | > Number.isInteger(1) < true | | > Number.isInteger(1.5) < false | |
| Safe Integer | > Number.isSafeInteger(2 \*\* 53) < false | | > Number.isSafeInteger(2 \*\* 53 - 1) < true | |
| Finite Number | > isFinite(1253) < true | | > isFinite(Infinity) < false | |
| Positive Number | > Math.sign(1) == 1 < true | | > Math.sign(-5) == 1 < false | |
| Negative Number | > Math.sign(-5) == -1 < true | | > Math.sign(1) == -1 < false | |
| Not a Number | > isNaN(1) < false | > isNaN('1') < false | > isNaN(Infinity) < false | > isNaN('a') < true |

## NUMBER SYSTEMS

|  |  |  |  |
| --- | --- | --- | --- |
| Main Number Systems | Decimal (base 10): any number represented using the digits 0 to 9. |  | > 13 < 13 // 3\*100 + 1\*101 |
| Binary (base 2): any number represented using the digits 0 and 1. | 0b | > 0b1101 < 13 // 1\*20 + 0\*21 + 1\*22 + 1\*23 |
| Octal (base 8): any number represented using the digits 0 to 7. | 0o | > 0o15 < 13 // 5\*80 + 1\*81 |
| Hexadecimal (base 16): any number represented with 0 to 9 + a (10), b (11), c (12), d (13), e (14), f (15). | 0x | > 0xd < 13 // 13\*160 |
| Convert Decimals to Another Base | > (13).toString(2) < '1101' | > (13).toString(16) < 'd' | |
| Convert Other Numbers to Decimals | > parseInt('1101', 2) // binary < 13 | > parseInt('d', 16) // hexadecimal < 13 | |

## RANDOM NUMBERS

|  |  |  |
| --- | --- | --- |
| Range [0 – 1) | > Math.random() < 0.5803786374865132 | > Math.random() < 0.15326638974306683 |
| Range [0 – 9] | > Math.floor(Math.random() \* 10) < 5 | > Math.floor(Math.random() \* 10) < 3 |
| Range [1 – 10] | > Math.floor(Math.random() \* 10) + 1 < 2 | > Math.floor(Math.random() \* 10) + 1 < 9 |
| Range [7 – 21] | > Math.floor(Math.random() \* 15) + 7 < 14 | > Math.floor(Math.random() \* 15) + 7 < 20 |
| Range [x – y] | > Math.floor(Math.random() \* (y - x) + 1) + x | |

# SYMBOLS IN JAVASCRIPT

|  |  |  |
| --- | --- | --- |
|  |  |  |

# OBJECTS

## JS SPECIFICS & TYPES OF OBJECTS

|  |  |  |
| --- | --- | --- |
| About JS Objects | JS objects are containers that can store multiple values at once in order to represent real-life collections of things and/or properties. | |
| Object Objects | Objects are like a collection of comma-separated related variables (called *properties*) under a single name. Each object property is a key-value pair, the key being either a string or a symbol. | const person = {  name: 'John',  age: 25  }; |
| In the browser console, an object's name returns an expandable alphabetically ordered list of the properties. |  |
| Array Objects | An array is a zero-based indexed collection (list) of values under a single name. | const people = [  'John',  'Mary'  ]; |
| In the browser console, an array's name returns an expandable indexed list of the array elements. | > people  < ▼ *(2)* *['John', 'Mary']*  **0**: "John"  **1**: "Mary"  **length**: 2  ► [[Prototype]]: Array(0) |
| Map Objects | Maps are similar to objects in that they are collections of key-value pairs, but the keys in a map can be of any data type and the properties are always listed in order of insertion. | const person = new Map([  ['name', 'John'],  ['age', 25]  ]); |
| In the browser console, a map's name returns an expandable indexed list of key-value pairs. | > person  < ▼ *Map(2)* *{'name' => 'John', 'age' => 25}*  ▼ [[Entries]]:  ▼ **0**: {"name" => "John"}  **key**: "name"  **value**: "John"  ▼ **1**: {"age" => 25}  **key**: "age"  **value**: 25  size: 2  ► [[Prototype]]: Map |
| Set Objects | Sets are similar to arrays in that they are indexed collections of values, but the elements of a set are guaranteed to be unique. | const people = new Set([  'John',  'John',  'Mary'  ]); |
| In the browser console, a set's name returns an expandable indexed list of values. | > people  < ▼ *Set(2)* *{'John', 'Mary'}*  ▼ [[Entries]]:  ▼ **0**: "John"  **value**: "John"  ▼ **1**: "Mary"  **value**: "Mary"  size: 2  ► [[Prototype]]: Set |
| Date Objects | Date objects represent a moment in time (in the browser's time zone). | const date = new Date(); |
| In the browser console, a date object's name returns information about the date and time. | > date  < Sat Jun 04 2022 11:09:38 GMT+0300 (Eastern European Summer Time) |
| RegExp Objects | A regular expression is an object that describes a pattern of characters. | const pattern = /[a-z]/;  const pattern = new RegExp('[a-z]'); |
| In the browser console, a regular expression's name returns the pattern. | > pattern  < /[a-z]/ |

## OBJECT OBJECTS

### CREATE AN OBJECT

|  |  |  |
| --- | --- | --- |
| Create an Empty Object | We can declare an object using curly braces ({}). | const person = {}; |
| > person  < ► *{}* |
| Create an Object with Properties | We can list all the properties we need enclosed in curly braces ({}). | const person = { name: 'John', age: 25 }; |
| In the browser console, the object's name returns an expandable alphabetically ordered list of the properties. | > person  < *▼* *{name: 'John', age: 25}*  **age**: 25  **name**: "John"  ► [[Prototype]]: Object |
| Create an Object with Properties That Already Exist As Variables | We can turn variables into object properties by enclosing them in curly braces. | const name = 'John';  const age = 25;  const person = { name, age }; |
| > person  < ► *{name: 'John', age: 25}* |

### CREATE A COPY OF AN OBJECT

|  |  |  |
| --- | --- | --- |
| Create a Copy of the Reference | The assignment operator creates a copy of the reference pointing to the same object.  When we change the object through any of the references, all the other references show the changes as well. | const student = person; |
| > student.name = 'Mary';  student  < ► *{name: 'Mary', age: 25}*  > person  < ► *{name: 'Mary', age: 25}* |
| Create a New Reference to a Copy of the Object | We can create an actual copy of the object using Object.assign().  This way, a new reference is created and changing the copy does not affect the original. | const student = Object.assign({}, person); |
| > student.name = 'Mary';  student  < ► *{name: 'Mary', age: 25}*  > person  < ► *{name: 'Jonh', age: 25}* |
| Create a New Reference Adding New Properties | The new properties must come before the original object, otherwise we will create a copy of the reference and change the original as well. | const student = Object.assign(  { id: 1234 }, person  ); |
| > student  < ► *{id: 1234, name: 'John', age: 25}* |
| Create a New Object with Copies of the Object's Properties | Another way to create a copy of an object is using the spread operator (...).  This way, a new object is created with the original one's properties. Changing the copy does not affect the original. | const student = { ...person }; |
| > student.name = 'Mary';  student  < ► *{name: 'Mary', age: 25}*  > person  < ► *{name: 'Jonh', age: 25}* |
| Create a New Object Adding New Properties | The order of the properties does not matter. | const student = { id: 1234, ...person }; |
| > student  < ► *{id: 1234, name: 'John', age: 25}* |

### ADD, CHANGE, ACCESS & DELETE PROPERTIES

|  |  |  |  |
| --- | --- | --- | --- |
|  | The Dot Notation | | The Bracket Notation |
| Handling Object Properties | We can add, change, access and delete an object property using the dot notation (. + property name).  It only works for property names without special characters (like a hyphen or whitespace). | | We can add, change, access and delete any object property using the bracket notation ([*property name*]).  It works with special characters, as well as with variables holding property names. |
| Add/Change a Property | person.job = 'actor'; | | person['marital-status'] = 'single'; |
| person.marital-status = 'single'; | | const property = 'marital-status';  person[property] = 'single'; |
| > person  < ► *{name: 'John', age: 25, job: 'actor', marital-status: 'single'}* | | |
| Access Property Value | > person.name  < 'John' | > person.weight  < undefined | > person['marital-status']  < 'single' |
| Delete Properties | delete person.name | | delete person['marital-status'] |

### CHECK WHETHER A PROPERTY EXISTS

|  |  |  |
| --- | --- | --- |
| The hasOwnProperty() Method | > person.hasOwnProperty('name')  < true | > person.hasOwnProperty('weight')  < false |
| The in Operator | > 'name' in person  < true | > 'weight' in person  < false |

### MAKE AN OBJECT NON-EXTENDABLE

|  |  |  |
| --- | --- | --- |
| The preventExtensions() Method | The preventExtensions() method prevents new properties from ever being added to an object. | Object.preventExtensions(person) |
| > person.weight = 80;  person  < ► *{name: 'John', age: 25}* |

### DESTRUCTURING ASSIGNMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Destructuring Assignment Keeping the Name of the Property | Instead of declaring the variables one by one (const age = person.age;  const name = person.name;), we can use the destructuring syntax for arrays. | const person = { name: 'Ben', age: 5 }; const { age, name } = person; | | |
| > age < 5 | > name < 'Ben' | |
| Destructuring Assignment Changing the Name of the Property | If we want to give the variable a new name, we can add it after a colon (:). | const { name: firstName } = person; | | |
| > firstName < 'Ben' | | |
| Nested Destructuring | const person = {  name: 'John',  address: {  street: 'Main Str.',  number: 31  }  }; | const { address:  { street }  } = person; | | > street < 'Main Str.' |
| const { address:  { street: str }  } = person; | | > str < 'Main Str.' |
| Nested Destructuring: Objects and Array Objects | const person = {  name: 'John',  emails: ['jd@dot.de', 'john@doe.uk']  }; | const { emails:  [email]  } = person; | | > email < 'jd@dot.de' |
| const person = {  name: 'John',  phones: [  { code: '+33', phone: '123' },  { code: '+43', phone: '567' }  ]  }; | const { phones:  [{ phone: number }]  } = person; | | > number < '123' |

### GET A LIST OF THE PROPERTIES

|  |  |  |
| --- | --- | --- |
| Get a List of the Property Keys | We can get an array of all the property keys in an object using Object.keys(*object name*). | > Object.keys(person)  < ► *(2)* *['name', 'age']* |
| Get a List of the Property Values | We can get an array of all the property values in an object using Object.values(*object name*). | > Object.values(person)  < ► *(2)* *['John', 25]* |
| Get a List of the Property Keys | We can get an array of arrays (a two-dimensional array or a matrix) containing all the property key-value pairs in an object using Object.entries(*object name*). | > Object.entries(person)  > people  < ▼ *(2)* *[Array(2), Array(2)]*  ► **0**: (2) ['name', 'John']  ► **1**: (2) ['age', 25]  **length**: 2  ► [[Prototype]]: Array(0) |

### VISUALIZE THE OBJECT CONTENT

|  |  |  |
| --- | --- | --- |
| Display Object in Browser Console | In the browser console, we can get an expandable alphabetically ordered list of the object properties by typing its name. | > person  < *▼* *{name: 'John', age: 25}*  **age**: 25  **name**: "John"  ► [[Prototype]]: Object |
| Display Object as a JSON String | We can get a string with the object's content using JSON.stringify(). | > JSON.stringify(person)  < '{"name":"John","age":25}' |
| In order to see the object in a more readable way, we can format it while printing it on the console. | console.log(JSON.stringify(person, null, 2))  {  "name":"John",  "age":25  } |
| Display Object as a Table | We can see the object's content in a table using console.table(). Under the table, there is an expandable list of the object properties. | console.table(person) |
| Display Several Objects in a Table | We can compare several objects by listing them enclosed in {} in the console.table() method. | console.table({ first, second, third }) |

### LOOP THROUGH AN OBJECT

|  |  |  |
| --- | --- | --- |
| Loop Trough Object Property Keys | We can use the for – in loop to get access to all the object property keys one by one. | for (let key in person) {  console.log(key);  }  name  age  job |
| Loop Through Object Property Values | We can also loop through the object property values using the bracket notation. | for (let key in person) {  console.log(person[key]);  }  'John'  25  'actor' |
| Loop Through Object Property Keys & Values | We can get both the key and value for each object property. | for (let [k, v] of Object.entries(person)) {  console.log(k, v);  }  name 'John'  age 25  job 'actor' |

### METHODS

|  |  |  |
| --- | --- | --- |
| About Object Methods | Methods are functions as object properties.  Inside an object method, the keyword this allows us to refer to the object itself. | const person = {  name: 'John',  introduce: function () {  return `I am ${this.name}`;  }  }; |
| Call Object Methods | In order to access the function's output, we need to call the method with (). | > person.introduce() < 'I am John' |
| Use an Already Existing Function as a Method | We can declare the function outside the object and use it just as we would any other variable as property. | function introduce() {  return `I am ${this.name}`;  }  const person = { name: 'John', introduce }; |
| > person.introduce() < 'I am John' |
| Built-In Methods | Each object has some automatically added built-in methods. We can see them in the browser console under [[Prototype]]: Object. | > {};  < *▼* *{}*  *▼* [[Prototype]]: Object  ► **constructor**: *f Object()*  ► **hasOwnProperty**: *f hasOwnProperty()*  ► **isPrototypeOf**: *f isPrototypeOf()*  ► **propertyIsEnumerable**:   *f propertyIsEnumerable()*  ► **toLocaleString**: *f toLocaleString()*  ► **toString**: *f toString()*  ► **valueOf**: *f valueOf()* |
| We can call the built-in methods just like the ones we created – with (). | > person.toString() < '[object Object]' |
| Override Built-In Methods | We can override built-in methods by just adding a method with the same name. | const person = {  name: 'John',  toString() {  return `I am ${this.name}`;  }  }; |
| > person.toString() < 'I am John' |

### GETTER & SETTER METHODS

|  |  |  |  |
| --- | --- | --- | --- |
| About Getters & Setters | Getters and setters are methods that behave like properties: getters get the property value, setters set the property value. They are also called *object accessors* or *computed properties*. | | |
| Getter | The get syntax binds the property name to a function that will be called when that property is looked up. | | |
| A getter creates read-only properties with dinamically computed values. We cannot have a regular property and a getter with the same name. | const circle = {  r: 5,  get diameter() { return this.r \* 2; }  }; | |
| We can read the getter but cannot assign a new value to it (throws error in strict mode). | > circle.diameter < 10 | > circle.diameter = 7;  circle.diameter < 10 |
| Setter | The set syntax is used to execute a function whenever we try to change a property. | | |
| Setters are often used when we need some validation before assigning a value to a property (in this case, the value must be positive). | const circle = {  set diameter(value) {  if (value > 0) this.r = value / 2;  }  }; | |
| > circle.diameter < undefined | > circle.diameter = 5;  circle < ► *{r: 2.5}* |
|  | In the browser console, we see the getter as a property and as a method, and the setter – only as a method. | > circle < *▼* *{r: 5}*  **diameter**: (...)  **r**: 5  ► get diameter: *f diameter()*  ► set diameter: *f diameter(value)*  ► [[Prototype]]: Object | |

### NESTED OBJECTS

|  |  |  |
| --- | --- | --- |
| Objects Within Objects | As an object property's value can be of any data type, sometimes we have objects within objects. | const person = {  names: {  firstName: 'John',  lastName: 'Doe'  }  }; |
| Add, Change and Access Nested Properties | We can add, change and access nested properties just the way we do with first-level properties: using the dot notation and the bracket notation. | person.names.fullName = 'John Doe'; |
| person.names.firstName = 'Mary'; |
| > person.names.fullName < 'John Doe' |
| Optional Chaining Operator | When we try to access a property of a nested object that does not exist, we get an error. | > person.name.fullName // omitted "s" |
| When we use the optional chaining operator (?.), we get undefined instead of error. | > person.name?.fullName  < undefined |

### PROPERTY DESCRIPTORS

|  |  |  |
| --- | --- | --- |
| About Property Descriptors | When we create a JS object, each property gets its own object called *a property descriptor*. The property descriptor contains the value/accessor and some special metadata attributes (or *flags*). We can use the defineProperty() method to change them later (unless configurable is set to false). | |
| Access Property Descriptors | The method getOwnPropertyDescriptor() returns the full information about a property. | > Object.getOwnPropertyDescriptor(  person, 'age'  );  < ► *{value: 25, writable: true,*  *enumerable: true, configurable: true}* |
| Create a New Data Property | Instead of creating a property the simple way (person.age = 25;), we can set not only the value, but the metadata for the property as well.  All the flags (writable, enumerable and configurable) are true by default. | Object.defineProperty(  person,  'age',  {  value: 25,  writable: true,  enumerable: true,  configurable: true  }  ); |
| Create a New Accessor Property | We can only add an accessor property to an existing object using the defineProperty() method. This way, we set a getter, a setter, and the metadata flags all at once.  The accessor property flags (enumerable and configurable) are false by default. | Object.defineProperty(  circle,  'diameter',  {  get() { return this.r \* 2; },  set(v) { if (v > 0) this.r = v / 2; },  enumerable: false,  configurable: false  }  ); |

#### NON-WRITABLE PROPERTIES

|  |  |  |
| --- | --- | --- |
| Non-Writable Primitive Value Property | When we set the writable flag of a property to false, we are no longer able to change the property value. | Object.defineProperty(  person, 'age', { writable: false }  ); |
| > person.age = 20;  person  < ► *{name: 'John', age: 25}* |
| Non-Writable Object Property | If the value is an object, only the reference to it is non-writable. | person.names = { first: 'Jim', last: 'Doe' }; Object.defineProperty(  person, 'names', { writable: false }  ); |
| We can access and change the object properties, but if we try to reassign the object, the expression is simply ignored. | > person.names.first = 'John';  person  < *▼* *{names: {...}}*  **names**: {first: 'John', last: 'Doe'}  ► [[Prototype]]: Object |
| > person.names = 'John Doe';  person  < *▼* *{names: {...}}*  **names**: {first: 'John', last: 'Doe'}  ► [[Prototype]]: Object |

#### NON-ENUMERABLE PROPERTIES

|  |  |  |  |
| --- | --- | --- | --- |
| Lists of Object Keys/Values | Non-enumerable properties are not listed in Object.keys(), Object.values(), or Object.entries(). | Object.defineProperty(  person, 'age', { enumerable: false }  ); | |
| > Object.keys(person)  < ► *['name']* | > Object.values(person)  < ► *['John']* |
| JSON Strings | Non-enumerable properties will be missing from a JSON string. | > JSON.stringify(person)  < '{"name":"John"}' | |
| Loops | Also, they won't be considered in a for – in loop. | for (let key in person) {  console.log(key);  }  name | |
| Full List of Properties | We can still see a full list of the properties using the method getOwnPropertyNames(). | > Object.getOwnPropertyNames(person)  < ► *(2)* *['name', 'age']* | |

#### NON-CONFIGURABLE PROPERTIES

|  |  |  |
| --- | --- | --- |
| Change Non-Configurable Property Enumerable Flag | If configurable is set to false, we can no longer change the property enumerable flag. | Object.defineProperty(  person, 'age', { configurable: false }  ); |
| > Object.defineProperty(  person, 'age', { enumerable: false }  ); |
| Change Non-Configurable Property Writable Flag | We can, however, change the writable flag, but only to false. | > Object.defineProperty(  person, 'age', { enumerable: false }  ); < ► *{name: 'John', age: 25}* |
| Delete Non-Configurable Property | We cannot delete non-configurable properties. | > delete person.age  < false // error in strict mode  > person.age  < 25 |

#### LOCK ALL OBJECT PROPERTIES

|  |  |  |
| --- | --- | --- |
| Object.freeze() | The Object.freeze() method sets all properties to non-writable and non-configurable. | > Object.freeze(person);  Object.getOwnPropertyDescriptor(person, 'age'); < ► *{value: 25, writable: false,*  *enumerable: true, configurable: false}* |
| Object.seal() | The Object.seal() method sets all properties to non-configurable. | > Object.seal(person);  Object.getOwnPropertyDescriptor(person, 'age'); < ► *{value: 25, writable: true,*  *enumerable: true, configurable: false}* |

## PROTOTYPE INHERITANCE & CLASSES

|  |  |  |
| --- | --- | --- |
| The Prototype Chain | JavaScript is a prototype-based language. 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 like the final link in the prototype chain. | |
| Inheritance with the Prototype Chain | When we try to access a property, the property is first sought among the object’s own properties. If not found – among the properties of the object’s prototype, then the prototype of the prototype, until the end of the prototype chain.  The only exception is the method hasOwnProperties(), which only checks the object itself and does not go further the prototype chain. | |
| JS Classes | A class is a template for objects: a function with a constructor method that creates objects. The new objects are called *instances* of the class and share its prototype.  Nearly all objects in JS are instances of the Object class and inherit properties from Object.prototype. | |
| Create an Object with a Specified Prototype | The new object seems epmty, it does not have properties of its own yet. However, it has access to its prototype’s properties. | const student = Object.create(person); |
| const student = Object.setPrototypeOf({}, person); |
| > student  < *▼* *{}*  *▼* [[Prototype]]: Object  **age**: 25  **name**: "John"  ► [[Prototype]]: Object |
| Create an Object with a Specified Prototype and Own Properties | The new object has a property of its own right away (id). It has also access to its prototype’s properties. | const student = Object.create(person, {  id: { value: 1, writable: false }  }); |
| > student  < *▼* *{id: 1}*  **id**: 1  *▼* [[Prototype]]: Object  **age**: 25  **name**: "John"  ► [[Prototype]]: Object |
| Override a Prototype’s Property | We can create an object’s own property with the same name as one of the properties of its prototype. | student.name = 'Mary'; |
| > student  < *▼* *{name: 'Mary'}*  **name**: "John"  *▼* [[Prototype]]: Object  **age**: 25  **name**: "John"  ► [[Prototype]]: Object |
| Set Prototype of an Existing Object | We can explicitly specify an existing object’s prototype using the Object.setPrototypeOf() method. | const student = { id: 1 };  Object.setPrototypeOf(student, person); |
| > student  < *▼* *{id: 1}*  **id**: 1234  *▼* [[Prototype]]: Object  **age**: 25  **name**: "John"  ► [[Prototype]]: Object |

### ACCESS PROPERTIES

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Looking for Property | Looking in Object | Result |
| > student.name  < 'Mary' | name | student | YES → get value |
| > student.age  < 25 | age | student | NO → go a step further ↓ |
| age | person | YES → get value |
| > student.constructor < *f Object() { [native code] }* | constructor | student | NO → go a step further ↓ |
| constructor | person | NO → go a step further ↓ |
| constructor | Object | YES → get value |
| > student.grades  < undefined | grades | student | NO → go a step further ↓ |
| grades | person | NO → go a step further ↓ |
| grades | Object | NO → go a step further ↓ |
| grades | null | NO → get undefined |
| > student.hasOwnProperty('grades')  < false | grades | student | NO → get false |

### ACCESS PROTOTYPE

|  |  |  |
| --- | --- | --- |
| Access Object Prototype | We can access an object’s prototype, i.e., the value of the internal [[Prototype]] property with the following method: Object.getPrototypeOf(). | > Object.getPrototypeOf(student)  < ► *{name: 'John', age: 25}* |
| Another way to access the prototype is the \_\_proto\_\_ property. NOT recommended. | > student.\_\_proto\_\_  < ► *{name: 'John', age: 25}* |
| Access Function Prototype | A function’s prototype is accessible via the prototype property. | > function myFunction() {}  muFunction.prototype  < ► *{constructor: f}* |

### CLASSES AND CONSTRUCTOR FUNCTIONS

|  |  |  |  |
| --- | --- | --- | --- |
| JS Classes and Constructor Functions | There are three ways to easily create multiple objects with some common properties or behavior: classes, constructor functions and factory functions.  Constructor functions and classes are basically the same, the class keyword being a newer syntax.  A class is a template for objects: a function with a constructor method that creates objects. The new objects are called *instances* of the class and share its prototype.  Nearly all objects in JS are instances of the Object class. | | |
| Declare a Class | We use the class keyword with the name of the class (capitalized). Within the class constructor, this is a substitute for the object to be created. | class Person {  constructor(name, age) {  this.name = name;  this.age = age;  }  } | |
| Create an Instance of a Class | We can create a new instance of a class using the keyword new. | const woman = new Person('Ann', 15); | |
| We can also use the Object.create() method based on the prototype of the class. | const woman = Object.create(Person.prototype, {  name: { value: 'Ann'},  age: { value: 15 }  }); | |
| We can also create a regular object and then set its prototype to the prototype of the class. | const woman = { name: 'Ann', age: 15 };  Object.setPrototypeOf(woman, Person.prototype); | |
| Access a Class Instance | In the browser console, we can see the properties of the new object, as well as a reference to the class. | > woman  < *▼* *Person* *{name: 'Ann', age: 15}*  **age**: 15  **name**: "Ann"  *▼* [[Prototype]]: Object  **constructor**: *class Person*  ► [[Prototype]]: Object | |
| Add Properties and Methods to a Class | We cannot add properties and methods directly to an already declared class, we need to add them to its prototype instead. | Person.prototype.introduce = function () {  return `I am ${this.name}`;  } | |
| > woman.introduce()  < 'I am Ann' | |
| Static Properties and Methods | The keyword static allows us to create properties and methods as part of the class but NOT the instances. | class Person {  constructor() { ... }  static description = 'This a class';  }  const woman = new Person(); | |
| > woman.description  < undefined | > Person.description  < 'This a class' |
| Check Whether an Object is Instance of a Class | The instanceof operator allows us to check whether an object is instance of a specific class. | > woman instanceof Person  < true | |
| > woman instanceof Object  < true | |

### CONSTRUCTOR FUNCTIONS

|  |  |  |
| --- | --- | --- |
| Create a Constructor | Constructor functions are almost identical to classes and were the only way of creating object templates before the class syntax was implemented. | function Person (name, age) {  this.name = name;  this.age = age;  } |
| Create an Instance of a Constructor | We create instances of a constructor in the same ways we do for classes. | const woman = new Person('Ann', 15); |
| Access a Constructor Instance | In the browser console, we see the properties of the new object, as well as a reference to the constructor function. | > woman  < *▼* *Person* *{name: 'Ann', age: 15}*  **age**: 15  **name**: "Ann"  *▼* [[Prototype]]: Object  **constructor**: *f Person(name, age)*  ► [[Prototype]]: Object |
| Check Whether an Object is Instance of a  Constructor | The instanceof operator works the same way as with classes. | > woman instanceof Person  < true |
| > woman instanceof Object  < true |

## DECLARE A CLASS AND CREATE INSTANCES

|  |  |  |
| --- | --- | --- |
| 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, ${obj.name}`);  } }  const myPerson = new Person('Tim');  myPerson.sayHi({ name: 'John' }); // no?  Person.sayHi(myPerson); // prints 'Hi, Tim' |  |
| Class Inheritance (Does NOT Create Copies, Inserts a Reference Instead) | class Person {  constructor(name) {  this.name = name;  }  sayHi() {  console.log(`${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!' |  |
| Private Properties (\_) |  |  |

## 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 |  |  |

## USE AN OBJECT AS A FUNCTION LIBRARY

|  |  |  |
| --- | --- | --- |
|  | Often used to expose public API in a module. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | const compareNumbers = {  ascending: (a, b) => a - b,  descending: (a, b) => b - a  }; |

## USE AN OBJECT AS A SWITCH REPLACEMENT

|  |  |  |
| --- | --- | --- |
|  | switch (command) {  case 'increment': count++; break;  case 'decrement': count--; break;  case 'reset': count = 0; break;  }  const parser = {  increment: () => count++,  decrement: () => count--,  reset: () => count = 0  };  parser[command](); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |

|  |  |
| --- | --- |
| Factory Functions (Create and Return 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.' |

# ARRAY OBJECTS

## JAVASCRIPT ARRAY CHARACTERISTICS

|  |  |  |
| --- | --- | --- |
| Zero-based | Array.isArray(['John', '35']) // true  ['John', '35'] instanceof Array // true  const numbers = [1, 2];  numbers.push(3); // [1, 2, 3]  numbers = numbers.slice(0); // not allowed  const person = { name: 'Peter', age: 23 };  person.name = 'John'; // person is now { name: 'John', age: 23 }  person = { name: 'John', age: 23 }; // not allowed  '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']  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 | const 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 | const cars = ['Audi', 'Fiat', 'BMW'];  cars[cars.indexOf('Audi')] = cars.splice(cars.indexOf('BMW'), 1, 'Audi').join(''); // returns ['BMW', 'Fiat', 'Audi'] |  |
| Shallow Copy Part of an Array to Another Location in the Same Array without Changing the Array Length | ['a', 'b', 'c'].copyWithin(1) // ['a', 'a', 'b']: section from index 0 till the end is copied to target index 1  ['a', 'b', 'c'].copyWithin(2) // ['a', 'b', 'a']: section from index 0 till the end is copied to target index 2  ['a', 'b', 'c'].copyWithin(1, 2) // ['a', 'c', 'c']: section from index 2 till the end is copied to target index 1  ['a', 'b', 'c'].copyWithin(0, 1) // ['b', 'c', 'c']: section from index 1 till the end is copied to target index 0  ['a', 'b', 'c'].copyWithin(0, 1, 2) // ['b', 'b', 'c']: section from index 1 to 2 is copied to target index 0 |  |
| Transpose a Matrix | const matrix = [[1, 2], [3, 4]]; const transpose = matrix[0].map((\_, i) => matrix.map(x => x[i])); // [[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 |
| Convert an Array to a Locale Specific String | ['a', new Date()].toLocaleString() // a,8/7/2021, 4:13:46 PM |
| 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) |  |

# 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>>/ |  |

# 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 Locally Formatted String | new Date().toLocalString() // '8/6/2021, 11:18:25 AM'  new Date().toLocalString('de-DE') // '6.8.2021, 11:18:25'  new Date().toLocalString('bg') // '6.08.2021 г., 11:18:25 ч.' |
| 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)' |  |

# FUNCTIONS

## FUNCTION DECLARATION/EXPRESSION

|  |  |  |
| --- | --- | --- |
| Function Declaration | function sayHi() {  // content of the function  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Input) C++ C# Java |
| Function Expression (Anonymous Function) | const sayHi = function () {  const name = 'John';  return 'Hi, ' + name; } |
| Arrow Function | const sayHi = () => {  const name = 'John';  return 'Hi, ' + name; } |
| Arrow Function with Only One Statement | const sayHi = () => 'Hi!';  const sayHi = name => 'Hi, ' + name; const sayHi = (name, greeting) => 'Hi, ' + name + greeting; |

## FUNCTION HOISTING

|  |  |  |
| --- | --- | --- |
| Function Declaration: Hoisted | sayHi(); // 'Hi there!'  function sayHi() { return 'Hi there!'; } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#Input) C++ C# Java |
| Function Expression, Arrow Functions: NOT Hoisted | sayHi(); // not allowed  const sayHi = () => 'Hi there!'; |

## CALL/INVOKE A FUNCTION. PARAMETERS AND ARGUMENTS

|  |  |  |
| --- | --- | --- |
| Parameters and Arguments | function by2(num) { // parameter *num* behaves as a variable between {}  return num \* 2; } by2(3); // argument: 3 | [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 fullName(firstName, lastName) {  return 'Your name is ' + firstName + ' ' + lastName; } fullName('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); // 'You will be able to vote in 3 years' |
| Multiple Inputs (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); // '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]); // 'My name is John and I am 35 years old.' |  |
| Default Parameters | function printStars(count = 5) {  console.log('\*'.repeat(count)); }  printStars(); // '\*\*\*\*\*' printStars(2); // '\*\*' printStars(3, 5, 8); // '\*\*\*' |  |

## 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; // 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 | const myObj = {  name: 'Peter',  func() { return this; } };  myObj.func() // { 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 | const myObj = {  name: 'Peter',  func() {  console.log(this); // { name: 'Peter', func: *f* }  function innerFunc() {  console.log(this); // global/window  }  innerFunc();  } }; |  |
| Arrow Function Context | const myObj = {  name: 'Peter',  func() {  console.log(this); // { name: 'Peter', func: *f* }  const innerFunc = () => {  console.log(this); // { name: 'Peter', func: *f* }  };  innerFunc();  } }; |  |
| This in Constructor Invocation | class Person {  constructor(name) { this.name = name; } }  const person = new Person('Mark'); // this refers to the new class instance  person.name // 'Mark' |  |
| Explicit Binding (Changing the Context) | const myObj = { name: 'Peter' }; function func(a, b) { console.log(this, a, b); }  func(2, 3) // Object [global] { ... } 2 3 func.call(myObj, 2, 3); // { name: 'Peter' } 2 3 |  |
| Explicit Binding with an Array of Arguments | func.apply(myObj, [2, 3]); // { name: 'Peter' } 2 3 |  |
| Explicit Binding in Another Function | const bound = func.bind(myObj); bound(2, 3); // prints { name: 'Peter' } and then 2 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'); |

|  |  |
| --- | --- |
| 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!' |

## FIRST-CLASS AND HIGHER-ORDER FUNCTIONS. CALLBACKS

|  |  |  |
| --- | --- | --- |
| First-Class Functions | Functions that are treated like any other variable: they do something specific and are meant to be passed as arguments or be returned by other functions. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| First-Class Function Example | function sayHello() { return 'Hello!'; } const greeting = () => 'Hello!'; console.log(sayHello(), greeting()); // 'Hello! Hello!' |
| Higher-Order Functions | Functions that take other functions as arguments (callbacks) or return a function. |
| Higher-Order Function Example | function sayHello() {  return function () { console.log('Hello!'); } } sayHello(); // 'Hello!' |
| Built-In Higher-Order Functions | array.map(x => x \* 2) array.filter(x => x > 5) array.reduce((acc, curr) => acc + curr, 0) |  |
| Callback | A function passed into another function as argument. |  |
|  |  |  |
| Predicates (Return a Boolean: Nil/Non-Nil) | [2, 4, 12, 37].find(e => e > 10); // 12  [2, 4, 12, 37].find(e => e > 50); // undefined |  |
| 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; } const 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 () { const name = 'Peter' })(); // name cannot be used const result = (function () { return name = 'Peter'; })() // result = 'Peter' |  |
| Create a Counter Using Closure and IIFE | 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) // 9 |  |
| Function Decoration | function pow(pow, num) { return num \*\* pow; } const sqr = pow.bind(null, 2); sqr(3) // 9 |  |
| Currying (Function Decomposition) | function sum3(a) {  return (b) => {  return (c) => {  return a + b + c;  }  }  }  sum3(5)(6)(8) // 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 a Value Stored in a Variable | let name = 'John';  console.log(name); // prints 'John' |
| Print Styled Text to the Console | console.log('%c My Friends', 'color: orange; font-weight: bold'); |  |
| Print to the Console Several Objects without Losing Their Names | const foo = { name: 'Ben', age: 23 };  const bar = { name: 'Tim', age: 35 };  console.log({ foo, bar }); // or console.log([foo, bar]) |  |
| Print to the Console Several Objects in the Form of a Table | const foo = { name: 'Ben', age: 23 };  const bar = { name: 'Tim', age: 35 };  console.table([foo, bar]); |  |
| Print to the Console an Object as It Is | console.dir(divElement); // displays all properties of the element |  |
| 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(); |  |

# 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

## JQUERY INTRODUCTION

|  |  |  |
| --- | --- | --- |
| About jQuery | jQuery is a lightweight JavaScript library that simplifies HTML DOM and CSS manipulation, event handling, animations and AJAX. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Using jQuery | Either download it from jQuery.com or include it from a CDN (like Google). |
| jQuery in HTML | <head>  <script *src*="https://ajax.googleapis.com/.../jquery.min.js" ></script>  </head> |
|  | element.text('Hello Sweden!'); // element.textContent = 'Hello Sweden!';  element.html('<p>Hi</>'); // element.innerHTML = '<p>Hi</p>'  element.hide(); // element.style.display = 'none'; |

## JQUERY SELECTORS

|  |  |  |
| --- | --- | --- |
| By Element Name | $('p') // document.querySelectorAll ('p') | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| By Class Name | $('.important') // document.getElementsByClassName('important') |
| By ID | $('#demo') // return a collection!, != document.getElementById('demo') |
| Current Element | $(this) // e.target |  |
| By Element & Class | $('p.intro') // document.querySelectorAll ('p.intro') |  |
| By Attribute | $('[href]') // |  |
| By Attribute Value | $('a[target="\_blank"]') //  $('a[target!="\_blank"]') // |  |
| By Order | $('p:first') //  $('p:last') //  $('ul li:first') //  $('ul li:first-child') //  $('tr:even') //  $('tr:odd') // |  |
| Buttons | $(':button') // all <button> and inputs of type button |  |
| By Child |  |  |
| By Parent |  |  |
| By Ancestor |  |  |
| By Sibling |  |  |
| Select All Elements | $('\*') // |  |

## JQUERY FILTERING

|  |  |  |
| --- | --- | --- |
|  | first, last, eq, filter, not | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |
|  |  |

## JQUERY EVENTS

### EVENT TYPES

|  |  |  |
| --- | --- | --- |
| Mouse Events | click, dblclick, mouseenter, mouseleave | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Keyboard Events | keypress, keydown, keyup |
| Form Events | submit, change, focus, blur |
| Document Events | ready, load, resize, scroll, unload |  |

### EVENT EXAMPLES

|  |  |  |
| --- | --- | --- |
| Wait for the Document to Load | $(document).ready(function() { ... }); // should wrap all jQuery code | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Hide Element When Clicked | $('p').click(function() {  $(this).hide();  }); |
| Alert a Message by Double Click | $('p').dblclick(function() {  alert('You just double clicked this');  }); |
|  | $('#myParagraph').mouseenter(function() {  alert('You entered myParagraph');  }); |  |
|  | $('#myParagraph').mouseleave(function() {  alert('You left myParagraph');  }); |  |
|  | $('#myParagraph').mousedown(function() {  alert('Mouse down on myParagraph');  }); |  |
|  | $('#myParagraph').mouseup(function() {  alert('Mouse up over myParagraph');  }); |  |
|  | $('#myParagraph').hover(function() {  alert('You entered myParagraph');  }, function() {  alert('You left myParagraph');  }); |  |
|  | $('input').focus(function() {  $(this).css('background-color', 'pink');  }); |  |
|  | $('input').blur(function() {  $(this).css('background-color', 'white');  }); |  |
| The on() Method | $('p').on('click', function() {  $(this).hide();  });  $('p').on({  mouseenter: function() {  $(this).css('background-color', 'gray');  },  mouseleave: function() {  $(this).css('background-color', 'white');  }  }); |  |

## JQUERY EFFECTS

|  |  |  |
| --- | --- | --- |
| Hide and Show | $("button").click(function(){ // selects ALL the buttons on the page  $("p").hide(1000);  }); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Toggle | $("button").click(function(){  $("p").toggle(1000, *callback*);  }); |
| Fade In and Out |  |
| FadeTo, fadeToggle() |  |  |
| Slide | slideDown(), slideUp(), slideToggle() |  |
| Animate |  |  |
| Stop |  |  |
| Callback | $("button").click(function(){  $("p").hide("slow", function(){  alert("The paragraph is now hidden");  });  }); |  |
| Chaining |  |  |

## JQUERY HTML

|  |  |  |
| --- | --- | --- |
| Get Content and Attributes | text(), html(), val()  attr() | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Set Content and Attributes | text(), html(), val()  attr() |
| Add Elements | append, prepend // inside the element  after, before  clone() |
| Remove Elements | remove()  hide(), show(), toggle()  fadeOut(), fadeIn(), fadeToggle(), slideUp(), slideDowd(), slideToggle() |  |
| CSS Classes | css(), addClass(), removeClass(), toggleClass()  hasClass() |  |
| CSS |  |  |
| Dimensions | width, height, innerWidth, innerHeight, outerWidth, outerHeight |  |
| Animations | animate() |  |

## JQUERY AJAX

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

## JQUERY MISC

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

# 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> |  |
| Server-Sent Events | // when a webpage automatically gets updates from the server  var source = new EventSource("demo\_sse.php");  source.onmessage = function(event) {  document.getElementById("result").innerHTML += event.data + "<br>";  };  if(typeof(EventSource) !== "undefined") {  // Yes! Server-sent events support!  // Some code.....  } else {  // Sorry! No server-sent events support..  }  // demo\_sse.php:  <?php  header('Content-Type: text/event-stream');  header('Cache-Control: no-cache');  $time = date('r');  echo "data: The server time is: {$time}\n\n";  flush();  ?> |  |

# 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; // creates a breakpoint in the code |  |

# MODULES IN JS

## TYPES OF MODULES

|  |  |  |
| --- | --- | --- |
| Local Modules (Created Locally) | const myModule = 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

|  |  |  |
| --- | --- | --- |
| See Your Web App on Your Phone | On your phone, go to Settings > About Phone, scroll to the bottom and tap 7 times on the Build number. Then go to Settings > System > Advanced > Developer Options and enable USB Debugging. Then go to Settings > Network 7 Internet > Hotspot & tethering and enable USB tethering  Type ipconfig in the command prompt in your computer and find the ipv4 for "ethernet adapter local area connection x" (mostly the first one) Now go to your mobile browser, type that ipv4 with the port number of your web application. eg: 192.168.42.83:3000 | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |  |

# HTTP AND REST SERVICES

## HTTP REQUESTS / Methods

|  |  |  |
| --- | --- | --- |
| HTTP | A text-based request-response model used to make CRUD requests to restful APIs. Relies on a unique resource URLs, provides resource metadata (ex. encoding) and is stateless (cookies can overcome that). |  |
| 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.  Essencially: endpoints that are accessible via the HTTP protocol.  Just like we can send a ship to a port to get some load, the client app can send an HTTP request (ship) to an HTTP endpoint (port) to get some data.  = backend?  Backend: one or more DBs and APIs that make the application available to the client. This is where the business logic is. The frontend, on the other hand, is all about the presentation: HTML templates + presentation logic of our application (displaying data and responding to user actions). |  |
| 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);  // Object.fromEntries(formData) returns an object of all form fields  // [...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');  // const { email, password, rePass } = Object.fromEntries(formData);  // const [email, password, rePass] = [...formData.entries()].map(x => x[1]);  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)

|  |  |  |
| --- | --- | --- |
|  | Uses routing in order to achieve navigation without reloading the page.  A router loads the appropriate content when the location changes (ex. the user manually enters an address). Conversely, a change in the content is reflected in the address bar (ex. when the user clicks on a link).  Benefits: load all the scripts only once, maintain state across multiple pages, browser history can be used, build user interfaces that react quickly. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | <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'; |
|  |  |  |

# 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 |
|  | Backend: one or more DBs and APIs that make the application available to the client. This is where the business logic is. The frontend, on the other hand, is all about the presentation: HTML templates + presentation logic of our application (displaying data and responding to user actions). |

## 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...')); |  |
|  | res.status(200).sendFile(\_\_dirname + '/index.html'); |  |
| 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

|  |  |  |
| --- | --- | --- |
|  | MongoDB sign up: username (DBeneva), pass (1qaz@WSX)  Database > Connect > Connect Using MongoDB (copy link)  VSC: install mongoDB for VSC extension, then click on Add Connection and paste the copied link.  MongoDB Compass: Download, open, New Connection > paste copied link, replacing <password> with the pass for 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 |  |  |
| Add a Value to an Array (unless Already Present) | await Student.findByIdAndUpdate('dk123hb', {  $addToSet: { subscribers: userId }  }); |  |

## 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 using 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) |
| CORS Problem | 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();  }); |

# ANGULAR JS

## PURPOSE AND INSTALLATION

|  |  |  |
| --- | --- | --- |
| About AngularJS | A JavaScript framework for building complex frontend applications (routing, forms, tests, etc.). It has a built-in compiler (compiles all the components and the HTML templates into JavaScript) and works internally with webpack. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Install Angular CLI (Command Line Interface) | npm install -g @angular/cli // installs the 'ng' command |  |
| Check AngularJS Version | ng --version |  |
| Documentation | angular.io |  |
| Create a New Project and Install Dependencies | ng new first-app  cd first-app  npm i |  |
| Build, Deploy, Open and Automatically Refresh an Angular App | ng s -o // ng serve --open (localhost:4200)  ng s --port 4201 // specify another port |  |
| Stop the Automatic Update | Ctrl + C |  |
| Build and Deploy an Angular App | ng build first-app // creates a folder 'dist/' in production configuration |  |

## TEMPLATES AND DATA BINDING

|  |  |  |
| --- | --- | --- |
| About Angular Templates | Each template is a section of HTML to include as a part of a page that the browser displays. It renders a view (user interface) in the browser with a lot more functionality than regular HTML. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Template Expressions (Not JS Expressions!) | <p>The sum of 1 + 1 is {{1 + 1}}</p> <!-- The sum of 1 + 1 is 2 --> |
| Display Values with Interpolation | app.component.ts: title = 'My Title';  app.component.html: <h1>{{title}}</h1> <!-- My Title --> |  |
| Bind Values to HTML Properties | app.component.ts: name = 'Maria';  app.component.html:  <input *[value]*="name"> <!-- Maria -->  <input *bind-value*="name"> <!-- Maria -->  <input *value*="name"> <!-- name --> |  |
| Bind Values to HTML Attributes | app.component.ts: imgUrl = 'some image URL';  app.component.html: <img *[attr.src]*="imgUrl" /> |  |
| Bind CSS Classes or Specific Class Names | app.component.ts: classes = ['red', 'fancy']; isSpecial = true;  app.component.html:  <p *[class]*="classes">Lorem ipsum</p> <!-- class="red fancy" -->  <p *[class.special]*="isSpecial">Lorem ipsum</p> <!-- class="special" --> |  |
| Bind Styles or Styles with Units | app.component.ts: isSpecial = true;  app.component.html:  <p *[style.color]*="isSpecial ? 'red' : 'green'">Lorem</p> <!-- red -->  <p *[style.font-size.%]*="!isSpecial ? 50 : 100">Lorem</p> <!-- 50 --> |  |
| Attach Events | app.component.ts:  showEvent = (event: MouseEvent) => alert(`${event.type}`);  app.component.html:  <button *(click)*="showEvent($event)">Click me</button> <!-- click -->  <button *on-click*="showEvent($event)">Click me</button> <!-- click --> |  |
| Show/Hide Text | app.component.ts:  visible = true; toggleText = () => this.visible = !this.visible;  app.component.html:  <p *\*ngIf*="visible">Lorem ipsum</p>  <button *(click)*="toggleText()">{{visible ? 'HIDE' : 'SHOW'}}</button> |  |
| Reference Other Elements | app.component.ts:  sayHi = (name: string) => alert(`Hi, ${name}`);  app.component.html:  <input *#inputElement*>  <button *(click)*="sayHi(inputElement.value)">Hi</button> |  |
| Null-Safe Operator | <p>The user is {{user?.age}} years old.</p>  <p>The user is {{user && user.age}} years old.</p> |  |
| Two-Way Data Binding (Banana Box Syntax) | app.module.ts: @NgModule({ imports: [FormsModule] })  app.component.ts: name = 'Ben';  app.component.html:  <input *[(NgModel)]*="name"> or <input *bindon-ngModel*="name">  <p>{{name}}</p> <!-- mirrors input value --> |  |
| Conditional Statements | app.component.ts: users = [];  app.component.html: <p *\*ngIf*="users.length == 0">No current users</p> |  |
| Render Arrays Using the \*ngFor Directive | app.component.ts:  users = [{ name: 'Ben', age: 3 }, {name: 'Jim', age: 4 }];  app.component.html:  <ul>  <li *\*ngFor*="let user of users">{{user.name}}</li> <!-- • Ben, • Jim -->  </ul> |  |

## MODULES

|  |  |  |
| --- | --- | --- |
| About NgModules | AngularJS modules are containers (classes marked by the @NgModule decorator) for different parts of an application (related components, services, directives, pipes, etc.). Unlike JavaScript modules, NgModules have a metadata object that describes how they fit together with the other parts of the app. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Root Module | Every AngularJS application contains at least one module (usually app.module.ts), used to bootstrap/launch the application. Only the root module should contain BrowserModule. |
| Root Module Content (app.module.ts) | import { NgModule } from '@angular/core';  import { BrowserModule } from '@angular/platform-browser';  import { AppComponent } from './app.component';  @NgModule({  declarations: [AppComponent], // components, directives and pipes  imports: [BrowserModule], // @NgModule classes  providers: [], // register service providers and inject them into components  bootstrap: [AppComponent] // root component  })  export class AppModule { } |
| Create Modules Manually | core.module.ts:  @NgModule({  imports: [CommonModule], // in all custom-made modules!  declarations: [ItemListComponent], // components; private by default  exports: [ItemListComponent], // in all custom-made modules!  providers: [ItemsService]  })  export class CoreModule { }  app.module.ts: @NgModule({ imports: [CoreModule] }) |
| Create Modules Using the Angular CLI | ng g m core // creates core/core.module.ts  app.module.ts: @NgModule({ imports: [CoreModule] }) |  |
| Suggested Common Modules | SharedModule: for all components/directives/pipes used in many places  CoreModule: for single services and components used only once  AuthenticationModule: for register, login, logout  FeatureModule: for feature specific components |  |

## COMPONENTS

|  |  |  |
| --- | --- | --- |
| About Components | Components are the building blocks of every AngularJS application. A component is a class that controls part of the screen (view). Each component has its own HTML/CSS template as well as the logic for rendering the view. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create Components Manually (app.component.ts) | import { Component } from '@angular/core';  @Component({// the Component decorator providing metadata  selector: 'app-root', // html tag in index.html  templateUrl: './app.component.html', // template: `<h1>{{title}}</h1>`  styleUrls: ['./app.component.css'], // styles: `[h1 { color: red; }]`  providers: [] // list of providers that can be injected using DI  })  export class AppComponent { title = 'App Title'; } |
| Import Components into the App Module (app.module.ts) | import { AppComponent } from './app.component';  @NgModule({  declarations: [AppComponent] // components, directives and pipes  }) |  |
| Insert Components into Parent Component | <app-home></app-home> |  |
| Create Components Using the Angular CLI | ng g c home // ng generate component home; creates src/app/home/ with component files, imports and inserts the component into the app |  |
| Lifecycle Hooks Used in Component Class | ngOnInit() { // called shortly after creation, after all inputs }  ngOnDestroy() { // used for cleanup }  ngOnChanges(simpleChanges) { // called when data is changed }  ngDoCheck() { // detect your own changes }  ngAfterContentInit() { // called when external content is received }  ngAfterContentChecked() { // called when external content is checked }  ngAfterViewInit() { // called when the views and child views are created }  ngAfterViewChecked() { // called when the views are checked } |  |
| Component Interaction: from Parent to Child | app.component.ts: users = ['Ben', 'Jim'];  app.component.html: <app-child *[userArr]*="users"></app-child>  child.component.ts: @Input() userArr: string[] = []; // comes from app  child.component.html:  <app-child2 *\*ngFor*="let user of userArr" *[user]*="user"></app-child2>  child2.component.ts: @Input() user!: string; // from child component  child2.component.html: <p>{{user!}}</p>  child2.component.css: :host { display: block; } |  |
| Component Interaction: from Child to Parent | child.component.ts:  @Output() addUser = new EventEmitter<string>();  addNewUser(userNameInput: HTMLInputElement) {  const { value: name } = userNameInput;  this.addUser.emit(name);  }  app.component.ts:  addNewUserHandler(name: string): void {  this.users = this.users.concat(name);  }  app.component.html:  <app-child *[userArr]*="users" *(addUser)*="addNewUserHandler($event)">  </app-child> |  |

## DIRECTIVES

|  |  |  |
| --- | --- | --- |
| About Directives | Directives are classes that add additional behaviour to elements in an AngularJS application. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Types of Directives | Components (directives with templates: <app-root>), attribute directives (change the appearance/behaviour of an element: *ngStyle*, *ngClass*, *ngModel*) and structural directives (change the DOM layout by adding/removing elements: *\*ngIf*, *\*ngFor*). |
| Create an Attribute Directive Manually (highlight.directive.ts) | @Directive({ selector: '[appHighlight]' })  export class HighlightDirective {  @Input() set appHighlight(isActive: boolean) {  if (isActive) {  this.renderer.setStyle(this.el.nativeElement, 'background-color', 'red');  } else {  this.renderer.setStyle(this.el.nativeElement, 'background-color', 'gray');  }  }  constructor(private renderer: Renderer2) { }  }  app.module.ts: declarations: [HighlightDirective] |  |
| Create a Directive with Angular CLI | ng g d highlight |  |
| Create a Structural Directive | export class MyIfDirective {  @Input() set appMyIf(isVisible: boolean) {  if (isVisible) {  this.viewContainerRef.createEmbeddedView(this.templateRef, data);  } else {  this.viewContainerRef.clear();  }  constructor(private viewContainerRef: ViewContainerRef,  private templateRef: TemplateRef<any>) { }  } |  |
| Use a Directive in the Template | <p *[appHighlight]*="isActive">Some text</p> // attribute  <ng-container *\*appMyIf*="isVisible">...</ng-container> // structural with \*  <ng-template *[appMyIf]*="isVisible"> // structural without \*  <ng-container>...</ng-container>  </ng-template> |  |
| Pass Data through a Directive | <ng-container *\*appMyIf*="isVisible; let data=data">  {{data}}</ng-container> |  |
| Bind to DOM Properties without Renderer | export class HighlightDirective {  @HostBinding('style.backgroundColor') backgroundColor = 'gray';  @Input() set isActive(isActive: boolean) {  if (isActive) {  this.backgroundColor = 'red';  } else {  this.backgroundColor = 'gray';  }  }  } |  |
| Respond to Events | @HostListener('mouseenter') { onMouseEnter(e) {  this.highlight('yellow'); }  }  @HostListener('mouseleave') { onMouseLeave(e) {  this.highlight('blue'); }  } |  |

## SERVICES

|  |  |  |
| --- | --- | --- |
| About Secrices | In AngularJS, a service is a class with an @Injectable() decorator that encapsulates non-UI logic and code that can be reused across an application. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Service Manually (user.service.ts) | import { Injectable } from '@angular/core';  @Injectable({ providedIn: 'root'}) // or app.module.ts: @NgModule({ providers: [UserService] })  export class UserService {  addUser(user: string): void { this.users = this.users.concat(user); }  } |
| Create a Service with Angular CLI | ng g s user // ng generate service user; creates src/app/user.service.ts |
| Inject a Service into a Module/Component (Dependency Injection) | app.module.ts: @NgModule({ providers: [UserService] })  app.component.ts:  @Component({ providers: [UserService] })  constructor(public userService: UserService) { } |  |
| Use a Service in the Template | app.component.html: <app-user-list *[userArray]*="userService.users"  *(addUser)*="userService.addNewUserHandler($event)"></app-user-list> |  |
| Lifecycle Hook in Services | ngOnDestroy(): void { } // if provided on component level |  |

## REACTIVE PROGRAMMING AND RxJS

|  |  |  |
| --- | --- | --- |
| Reactive Programming | It is an asynchronous programming paradigm concerned with data streams and the propagation of change. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Observables | In AngularJS, streams are handled as arrays using observables. Observables are containers that hold values, just like promises. As opposed to promises, observables can be canceled if we don't need them and can work with more than one value. Naming convention: $ after the name (time$). |
| Interactive Diagrams of Rx Observables | https://rxmarbles.com/ |
| Cold vs. Hot Observables | Observables are either hot (a movie at the cinema, shared) or cold (a movie on a plane). Cold observables create the data producer themselves (of, from, range, interval and timer). Hot observables have their data producer outside the observable itself (fromEvent). |
| Subjects | A special type of observable which allows values to be multicasted to many observers (like event emitters or hot observables). Every subject is an observable (has subscribe()) and an observer (has next(), error() and complete()). |
| Behavior Subjects | Store the latest value emitted to their consummers. Whenever a new observer subscribes, it receives the current value from the behavior subject. While an event stream of birthdays would be a subject, a stream of a person's age would be a behavior subject. |
| Replay Subject | Like the behavior subjects, they can send old values to new subscribers. They can also record multiple parts of the observable execution and replay them to new subscribers. |
| Async Subjects | Only the last value of the observable is sent to its observers and only when the execution completes. |
| RxJS | RxJS (Reactive Extensions for JavaScript) is a library for reactive programming using observables that makes it easier to compose asynchronous code. |
| Install RxJS | npm install rxjs |  |
| Use RxJS in Common JS Code | const { range } = require('rxjs');  const { map, filter } = require('rxjs/operators'); |  |
| Use RxJS with Import/Export | import { of } from 'rxjs';  import { map } from 'rxjs/operators';  const obs$ = of(1, 2); // cold observable  obs$.pipe(map(x => x + 10)  .subscribe((x) => console.log(x))); // subscription |  |
| Subscribe to an Observable | obs$.pipe(map(x => x + 10)  .subscribe({  next: (x) => console.log(x),  error: (err) => console.error(err),  complete: () => console.log('Load users stream completed')  }); |  |
| The Map Operator | const obs$ = range(1, 10).pipe(map(x => x \*\* 2)); |  |
| The Tap Operator (Similar to ForEach) | const obs$ = range(1, 10).pipe(tap(x => console.log(`Hello, ${x}`))); |  |
| The Filter Operator | const obs$ = range(1, 10).pipe(filter(x => x % 2 == 0)); |  |
| The Reduce Operator | const obs$ = range(1, 10).pipe(reduce((acc, c) => acc + c, 0))); |  |
| The StartWith Operator | const obs$ = interval(1000).pipe(startWith(), map(() => new Date()));  // starts at app loading, then reloads every second |  |
| Create a Timer Using Observables | time.component.ts:  export class TimeComponent implements OnDestroy {  timeSubsc!: Subscription;  time$ = interval(1000).pipe(startWith(''), map(() => new Date()));  time!: Date;  constructor() {  this.timeSubsc = this.time$.subscribe(time => this.time = time);  }  ngOnDestroy(): void { this.timeSubsc.unsubscribe(); }  }  time.component.html: {{time}} |  |

## HTTP CLIENT

|  |  |  |
| --- | --- | --- |
| About HTTPClient | HTTPClient is a service that can be used to fetch data from a server. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Import the HTTPClient Module | @NgModule({ imports: [HttpClientModule] }) |
| Use the HTTP Client in Services | @Injectable()  export class UserService {  constructor(private http: HttpClient) { }  loadUsers(search?: string) {  const query = search ? `?email\_like=${search}` : '';  return this.http.get<IUser[]>  (`https://jsonplaceholder.typicode.com/users${query}`);  }  } |  |
| Inject a Service that Loads Data and Subscribe to the Observable | export class AppComponent implements OnInit {  users: IUser[] | undefined;  constructor(public userService: UserService) { }  ngOnInit(): void {  this.userService.loadUsers(search).subscribe(  users => this.users = users,  err => console.log(JSON.stringify(err))  );  }  } |  |

## PIPES

|  |  |  |
| --- | --- | --- |
| About Pipes | Pipes are simple functions to use in template expressions that accept an input value and return a transformed value. They can be used throughout the entire application. |  |
| Built-In Angular Pipes | DatePipe: formats a date value according to local rules  UpperCasePipe: transforms text to all upper case  LowerCasePipe: transforms text to all lower case  CurrencyPipe: numbers to currency strings according to local rules  DecimalPipe: numbers to strings with a decimal point according to local rules  PercentPipe: numbers to percentage strings according to local rules | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Use Pipes in a Template | <p>His birthday is {{ day | date }}</p> <!-- |: the pipe operator -->  <p>Amount: {{ amount | currency:'EUR' }}</p> <!-- with a parameter -->  <p>Amount: {{ amount | currency:'EUR':'Euros' }}</p> <!-- 2 params -->  <p>{{ birthday | date | uppercase }}</p> <!-- chaining pipes --> |
| Create Custom Pipe to Shorten Data | ng g p shared/shorten  shorten.pipe.ts:  transform(value: string, limit: number) {  if (value.length > limit) { return `${value.substr(0, limit)}...`; }  return value;  }  shared.module.ts: exports: [ShortenPipe] |  |
| Create Custom Pipe to Retrieve Property Value | ng g p shared/get-prop  get-prop.pipe.ts:  transform(value: Record<string | number, any> | any[], path: (string | number)[]): any {  if (value === null || typeof value != 'object') { return null; }  let result = value;  for (let currentPart of path) {  result = result[currentPart];  if (!result) { return result; }  }  return result;  }  shared.module.ts: exports: [GetPropPipe] |  |
| Use Custom Pipes in a Template | app.component.ts:  text: 'This is a very long text';  data = [{ test: 111 }, { test: 222 }, { test: 333 }];  app.component.html:  {{ text | shorten: 10 }} <!-- This is a ... -->  {{ data | getProp: [ 0, 'test'] }} <!-- 111 --> |  |
| Async Pipe | app.component.ts:  text = new Promise((resolve, reject) => {  setTimeout(() => { resolve('show some text'); }, 3000); });  app.component.html:  {{ text | async }} |  |
| Async Pipe: Observables | app.component.ts:  export class PostComponent implements OnInit {  posts$: Observable<Post[]>;  ngOnInit() { this.posts$ = this.contentService.getAllPosts(); }  }  app.component.html:  <div *\*ngFor*="let post of posts$ | async">...</div> |  |

## INTERCEPTORS

|  |  |  |
| --- | --- | --- |
| About Interceptors | Interceptors automatically attach authentication information (often tokens) to requests or modify the requests in some way. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create an Interceptor to Modify the HTTP Requests | core/app-interceptor.ts:  const { apiURL } = environment;  @Injectable()  export class AppInterceptor implements HttpInterceptor {  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  constructor(private router: Router) { }  let reqStream$ = next.handle(req);  if (req.url.startsWith('/api')) {  reqStream$ = next.handle(req.clone({  url: req.url.replace('/api', apiURL),  withCredentials: true  }));  }  return reqStream$.pipe(catchError(err => {  this.router.navigate(['/'],  { queryParams: { error: 'Oops! Something went wrong!' } });  return throwError(err);  }));  }  }  export const appInterceptorProvider: Provider = {  provide: HTTP\_INTERCEPTORS,  useClass: AppInterceptor,  multi: true  };  core.module.ts: providers: [appInterceptorProvider]  user.service.ts, content.service.ts:  replace API\_URL with '/api' and remove { withCredentials: true } |

## LAZY LOADING

|  |  |  |
| --- | --- | --- |
| About Lazy Loading | It helps load the page in chunks (faster). Not working with routerLink. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Feature Module to Use with Lazy Loading | furniture.module.ts  funrniture-routing.module.ts: RouterModule.forChild(routes);  app.component.ts: do NOT import FurnitureModule! |
| Set the main Routing and Protect the Module | app-routing.module.ts:  { path: 'furniture',  loadChildren: () => import('./furniture/furniture.module')  .then(m => m.FurnitureModule)  }  export const appRoutingModule = RouterModule.forRoot(routes,  { preloadingStrategy: PreloadAllModules }); |  |

## ROUTING

### ABOUT ROUTING

|  |  |  |
| --- | --- | --- |
| About Routing | Allows navigation without reloading the page; pivotal element of writing single page applications (SPA). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Single Page Applications | A router loads the appropriate content when the location changes (ex. when the user manually enters an address). Conversely, a change in content is reflected in the address bar (ex. when the user clicks on a link). |
| Benefits of SPAs | They load all the scripts only once, maintain state across multiple pages, allow the use of browsing history and build user interfaces (UI) that react quickly. |  |

### USE A ROUTING MODULE

|  |  |  |
| --- | --- | --- |
| Create a Routing Module Manually | app.routing.module.ts:  import { Routes, RouterModule} from '@angular/router'  const routes: Routes = [  { path: '', redirectTo: 'home', pathMatch: 'full' },  { path: '/home', component: HomeComponent },  { path: '/user/:id', component: UserDetailsComponent },  { path: '\*\*', component: NotFoundComponent } // wildcard  ];  export const AppRoutingModule = RouterModule.forRoot(routes); // forChild() for child/nested rputes  app.module.ts: @NgModule({ imports: [AppRoutingModule] }) // must be the last of the imports!  app.component.html: <router-outlet></router-outlet> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Routing Module Using the Angular CLI | ng g m app --routing --defaults // generates a an application routing module called AppRoutingModule |
| Setting Up Child (Nested) Routes | { path: 'users', component: UsersComponent, children: [  { path: ':id', component: UserComponent },  { path: ':id/details', component: UserDetailsComponent }  ] }  users.component.html:  <router-outlet></router-outlet> |  |
| Define the Template | index.html: <base *href*="/"> // automatically added by the CLI  app.component.html:  <nav>  <a *routerLink*="/home">Home</a>  <a *routerLink*="/about">About</a>  </nav>  <router-outlet></router-outlet> |  |
| The RouterLink Directive | <a *routerLink*="/user/profile">Profile Page</a>  <a *[routerLink]*="['/user', 1, '/profile']">Profile Page</a> |  |
| Make Active Link Stand Out | header.component.ts:  <a *routerLink*="/" *routerLinkActive*="active"  *[routerLinkActiveOptions]*="{ exact: true }">Home</a>  <a *routerLink*="/about" *routerLinkActive*="active">About</a>  header.component.css:  a.active { color: red; } |  |
| Navigate Programmatically | inject router in components:  constructor(private router: Router) { }  use it to navigate from one component to another:  loadData() {  // service call  this.router.navigate(['/home']); // not recommended  } |  |

### FETCH QUERY PARAMETERS

|  |  |  |
| --- | --- | --- |
| Fetch Parameters | inject router in components: // only works in the rendering component  constructor(private route: ActivatedRoute) { }  retrieve parameters directly from the snapshot:  ngOnInit() { const id = this.route.snapshot.params['id']; } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Fetch Parameters Reactively | ngOnInit() { this.route.params.subscribe((params: Params) => {  const id = params['id'];  }) } |
| Query Parameters and Fragments | header.component.ts:  <a *[routerLink]*="['/users', user.id, user.name]"  *[queryParams]*="{ search: 'Peter' }" *fragment*="loading">...</a>  retrieve parameters from the snapshot:  this.route.snapshot.queryParams  this.route.snapshot.fragment |  |

### ROUTE GUARDS

|  |  |  |
| --- | --- | --- |
| Create a Route Guard Manually (auth.guard.ts) | export class AuthGuard implements CanActivate {  canActivate(route: ActivatedRouteSnapshot,  state: RouterStateSnapshot): boolean {  return this.checkIfLogged(state.url);  }  checkIfLogged(url: string): boolean { ... }  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Route Guard Using the Angular CLI | ng g guard auth // generates auth.guard.ts that implements chosen interfaces (ex. CanActivate) |

### ROUTER RESOLVER

|  |  |  |
| --- | --- | --- |
| The Resolve Property of Angular Router | { path: 'users', component: UsersComponent, children: [  { path: ':id',  component: UserComponent,  resolve: { user: UsersResolver } }  ] } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create the Resolver Guard | export class UsersResolver implements Resolve<User> {  resolve(route: ActivatedRouteSnapshot,  state: RouterStateSnapshot) {  return this.usersService.getUserById(route.params['id']);  }  } |
| Use the Resolver inside a Component | constructor (private route: ActivatedRoute) { }  ngOnInit() { // fetch data before navigating to the route  this.user = this.route.snapshot.data['user'];  } |

## HANDLING FORMS

### TEMPLATE-DRIVEN FORMS

|  |  |  |
| --- | --- | --- |
| About Template-Driven Forms | They are built by writing templates using the Angular temlate syntax. Most of the logic is in the template. |  |
| Import FormsModule | @NgModule({ imports: [FormsModule] }) // ngModel, ngSubmit, ngForm | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare a Template Variable to the Form | <form *#form*="ngForm" *(ngSubmit)*="loginHandler(form)">...</form>  // monitors and validates properties |
| Display, Listen and Extract Data | <input *name*="email" *ngModel* *#email*="ngModel">  // doesn't work without name |
| Access Form in Code | @ViewChild('form', { read: NgForm }) form: NgForm // w/o a function  loginHandler(form: ngForm): void { // as function parameter  const content = form.value;  } |  |
| Use Angular CSS Classes to Mark (In)valid Data | input.ng-valid { /\* no required attribute/required attribute + valid value \*/  border-left: 5px solid green;  }  input.ng-invalid.ng-touched { /\* input field has been visited, invalid value \*/  border-left: 5px solid red;  } |  |
| Display Error Messages | <input *type*="text" *#emailInput*="ngModel" *ngModel* *required email*>  <ul *\*ngIf*="emailInput.touched">  <li *\*ngIf*="emailInput.errors?.required">Email is required</li>  <li *\*ngIf*="emailInput.errors?.email">Email is invalid</li>  </ul> |  |
| Disable Submit Button | <button *type*="submit" *[disabled]*="form.invalid">Submit</button> |  |
| Two-Way Data Binding | app.component.ts: myName = 'John';  <input *name*="name" *[ngModel]*="myName"> <!-- initially "John" -->  <p>{{form.value.name}}</p> <!-- "John", then changes with input -->  <input *[ngModel]*="myName" *#input*="ngModel"> <!-- initially "John" -->  <p>{{input.value}}</p> <!-- initially "John", then changes with input -->  <input *type*="text" *[(ngModel)]*="myName"> <!-- initially "John" -->  <p>{{myName}}</p> <!-- initially "John", then changes with input --> |  |
| Display Data in Input Field | <input *[ngModel]*="user.username" *#input*="ngModel">  <input *ngModel*="myName" *#input*="ngModel"> |  |
| Group Similar Input Fields | <div *ngModelGroup*="passData" *#passData*="ngModelGroup">...</div>  <div *\*ngIf*="passData.invalid && passData.touched">Pass invalid!</div> |  |
| Use Custom Validator | shared.validators.ts:  export function emailValidator(control) {  if (!control.value) { return; }  return /^.{6,}@gmail\.(com|bg)$/.test(control.value) ? null :  { invalidEmail: true };  }  ng g d shared/custom-validator  custom-validator.directive.ts:  selector: '[appCustomValidator]',  providers: [  { provide: NG\_VALIDATORS,  useExisting: CustomValidatorDirective,  multi: true }  ]  export class CustomValidatorDirective implements Validator {  @Input('appCustomValidator') validatorFn: ValidatorFn;  constructor() { }  validate(control: AbstractControl): ValidationErrors {  if (!this.validatorFn) { return null; }  return this.validatorFn(control);  }  }  edit-profile.component.html:  <input *ngModel* *#emailInput*="ngModel"  *[appCustomValidator]*="emailValidator">  shared.module.ts: exports: [CustomValidatorDirective]  user.module.ts: imports: [SharedModule] |  |
| Validate Passwords | ng g d same-value  selector: '[ngModel][appSameValue]',  providers: [  { provide: NG\_VALIDATORS,  useExisting: SameValueDirective,  multi: true }  ]  export class SameValueDirective implements OnDestroy, Validator {  @Input() appSameValue = '';  @Input() name: string;  otherControl: AbstractControl;  subscription: Subscription;  constructor(private form: NgForm) { }  validate(control: AbstractControl) {  const otherControl = this.form.controls[this.appSameValue];  if (this.subscription) { this.subscription.unsubscribe(); }  this.subscription = otherControl.valueChanges.subscribe(() => {  control.updateValueAndValidity({ onlySelf: true });  });  return control.value != otherControl.value ? {  sameValue: { [this.appSameValue]: otherControl.value,  [this.name]: control.value }  } : null;  }  ngOnDestroy() { this.subscription.unsubscribe(); }  }  register.component.html:  <input *name*="pass" *#passInput*="ngModel" *ngModel* *minlength*="6">  <input *#repassInput*="ngModel" *ngModel* *appSameValue*="pass">  <ul *\*ngIf*="passInput.invalid && passInput.touched">  <li *\*ngIf*="passInput.errors?.minlength">At least 6 characters!</li>  </ul>  <ul *\*ngIf*="repassInput.invalid && repassInput.touched">  <li *\*ngIf*="repassInput.errors?.sameValue">Don't match!</li>  </ul> |  |
| Select Lists with Default Values | <select *#car*="ngModel" *[ngModel]*="BMW" *name*="car" *id*="car">  <option *\*ngFor*="let car of cars">{{car}}</option>  </select>  or:  <select *name*="car" *id*="car"> <!-- no ngModel! -->  <option *\*ngFor*="let car of cars" *[selected]*="car == 'BMW'">{{car}}<...>  </select> |  |
| Setting and Patching Form Value | changeInput() { this.laptopForm.form.patchValue(  { ram: '16 GB', proc: 'Intel Core i7' }  ); } |  |
| Reset a Form | onSubmit(): void { this.form.reset(); } |  |

### REACTIVE FORMS

|  |  |  |
| --- | --- | --- |
| About Reactive Forms | They are used when we need form arrays or when we need the form to be created dynamically. Most of the logic is inside the component class. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Import ReactiveFormsModule | @NgModule({ imports: [ReactiveFormsModule] }) // formGroup, formControl, formArrayName |
| Reactive Form in the Template | <form *[formGroup]*="myForm" *(ngSubmit)*="onSubmit()">  <input *name*="username" *formControlName*="username" *required*>  </form> |
| Accsess Properties of the Form Model | myForm.controls.username.valid  myForm.get('username').valid |  |
| Display Error Messages | <input *type*="text" *formControlName*="email" *required email*>  <ul *\*ngIf*="form.get('email').touched">  <li *\*ngIf*="form.get('email').errors?.required">Email is required</li>  <li *\*ngIf*="form.get('email').errors?.email">Email is invalid</li>  </ul> |  |
| Use FormBuilder Service | form: FormGroup;  constructor(private fb: FormBuilder) {  this.form = this.fb.group({  username: ['', [Validators.required, Validators.minLength(3)]]  });  } |  |
| Group Similar Input Fields | register.component.html:  <div *formGroupName*="passwords">  <input *formControlName*="pass"><input *formControlName*="rePass">  </div>  <p *\*ngIf*="form.get('passwords.pass').invalid">Pass invalid!</p>  register.component.ts:  this.form = this.fb.group({  passwords: this.fb.group({  password: ['', [Validators.required, Validators.minLength(3)]],  rePass: ['', [sameValueValidator(...)]] // custom validator  })  }); |  |
| Select Lists with Default Values | <select *formControlName*="car" *name*="car" *id*="car">  <option *\*ngFor*="let car of cars" *[selected]*="car == 'BMW'">  {{car}}  </option>  </select> |  |

## STATE MANAGEMENT AND NGRX

|  |  |  |
| --- | --- | --- |
| About State Management | It is the data management of multiple components in a complex application. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| State Management Tools | Libraries that allow developers to create a model of their app state, update the state of components, monitor and observe changes to the state, and read state values. Redux is the most popular one for JavaScript applications. |

### NGRX STORE

|  |  |  |
| --- | --- | --- |
| About NgRx Store | An RxJS powered state management tool (framework) for building reactive applications in Angular, inspired by Redux. NgRx provides libraries for managing global state (store), isolation of side effects (effects), entity collection management (entity), intergation with the Angular router (router store) and developer tooling (store devtools, schematics). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Use NgRx Store in an Angular Application | All the actions, reducers and selectors for a module are usually created in a folder "+store" (that stays on top of the other files and folders in the module). |
| Installation with Angular CLI | ng add @ngrx/store |
|  |  |  |
| Export Interface and Reducers for Root Module (+store/index.ts) | export interface IState { readonly global: IGlobalState; };  export const reducers: ActionReducerMap<IState> =  { global: globalReducer }; |  |
| Export Interface and Reducers for Feature Module (+store/index.ts) | export interface IUserState  { readonly list: IUserListState; readonly details: IUserDetailsState; };  export const reducers: ActionReducerMap<IUserState> =  { list: userListReducer, details: userDetailsReducer }; |  |

#### ACTIONS

|  |  |  |
| --- | --- | --- |
| About NgRx Actions | Actions in NgRx Store express unique events that happen throughout the application. An action is made of a simple interface with a single property describing the action that will be dispatched: { type: '[Source] Event' }. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create Actions (+store/actions.ts) | export const increment = createAction('[Counter] Increment');  export const login = createAction('[Login Page] Login',  props<{ name: string; pass: string; }>()); // will receive name & pass |
| Dispatch Actions from a Component | constructor(private store: Store<any>) { }  login(name: string, pass: string) { this.store.dispatch(login); } |  |

#### REDUCERS

|  |  |  |
| --- | --- | --- |
| About Reducers | A reducer is responsible for handling transitions from one state to another. It is a pure function and handles each state transition synchronously. It takes the latest action dispatched and the current state and determines whether to return the original state or a modified one based on the action's type. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create Reducers for Root Module | +store/reducer.ts:  export interface IGlobalState {  readonly counter: number;  readonly value: any;  readonly users: any[] | null;  }  const initialState: IGlobalState = { count: 0, value: null, users: null };  export const globalReducer = createReducer(  initialState,  on(incrementCount, state => ({ ...state, count: state.count + 1 })),  on(setValue, (state, { value }) => ({ ...state, value })),  on(loadUsersSuccess, (state, { users }) => ({ ...state, users })),  on(clearGlobalState, () => initialState)  );  app.module.ts: imports: [StoreModule.forRoot(reducers)] |
| Create Reducers for Feature Module | user/+store/reducer.ts:  export interface IUserListState { readonly users: any[] | null; }  const initialUserListState: IUserListState = { users: null };  export const userListReducer = createReducer(initialUserListState, ...);  export interface IUserDetailsState { readonly user: any | null; }  const initialUserDetailsState: IUserDetailsState = { user: null };  export const userDetailsReducer = createReducer(  initialUserDetailsState, ...);  user.module.ts: imports: [StoreModule.forFeature('user', reducers)] |  |
| NgRx Meta-Reducers | Meta-reducers allow developers to pre-process actions before normal reducers are invoked. |  |
| Use Meta-Reducer to Clear App State | +store/meta-reducers.ts:  function clearAppStateMetaReducer(reducer: ActionReducer<any>):  ActionReducer<any> {  return function (state: IState, action: Action) {  if (action.type == clearAppState.type) {  return reducer(undefined, action);  }  return reducer(state, action);  };  }  export const metaReducers: MetaReducer<any>[] =  [clearAppStateMetaReducer];  app.module.ts:  imports: [StoreModule.forRoot(reducers, { metaReducers })] |  |

#### SELECTORS

|  |  |  |
| --- | --- | --- |
| About NgRx Selectors | Selectors are pure functions used to obtain slices of store state. NgRx Store keeps track of the latest arguments in which the selector function was invoked, returning the result without reinvoking the function if the arguments match. Those performance benefits are part of the memoization technique. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create Selectors for Root Module (+store/selectors.ts) | const selectGlobal = (state: IState) => state.global;  export const selectGlobalCounter = createSelector(  selectGlobal, state => state.counter); |
| Create Selectors for Feature Module (user/ +store/selectors.ts) | const userModuleSelector = createFeatureSelector<IUserState>('user');  const selectUserList = createSelector(  userModuleSelector, state => state.list);  const selectUserDetails = createSelector(  userModuleSelector, state => state.details);  export const selectUserListUsers = createSelector(  selectUserList, s => s.users);  export const selectUserDetailsUser = createSelector(  selectUserDetails, s => s.user); |  |
| Use Selectors in a Component | constructor(private store: Store<any>) { }  users$ = this.store.select(selectGlobalUsers); |  |

### NGRX STORE DEVTOOLS

|  |  |  |
| --- | --- | --- |
| About NgRx Store Devtools | Tools that allow developers to track app state changes easily: F12 > Redux or extension for Chrome. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Installation with Angular CLI | ng add @ngrx/store-devtools |

### NGRX EFFECTS

|  |  |  |
| --- | --- | --- |
| About NgRx Effects | The NgRx Store effects decrease the responsibility of the component, isolating it from the services. The component needs to only declare an intent to fetch data, the effects do the rest. They are long-running services that listen to an observable of every action dispatched from the Store, perform tasks and return a new action. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Installation with Angular CLI | ng add @ngrx/effects |
| Create Effects | +store/effects.ts  @Injectable()  export class GlobalEffects {  constructor(  private actions$: Actions, private userService: UserService) { }  increment = createEffect(() => this.actions$.pipe(  ofType(setValue), map(action => incrementCounter())  ));  loadUsers = createEffect(() => this.actions$.pipe(  ofType(loadUsers), switchMap(() => this.userService.loadUsers.pipe(  takeUntil(this.actions$.pipe(ofType(loadUsersCancel))),  map(users => loadUsersSuccess({ users })),  catchError(error => [loadUsersFailure({ error })])  ))  ));  }  app.module.ts: imports: [EffectsModule.forRoot([GlobalEffects])] |  |

## AN ANGULAR APP

### START WORKING

|  |  |  |
| --- | --- | --- |
| Create a New App | ng new softuni-forum // create a project named "Softuni Forum"  cd softuni-forum // go to project directory  npm i // istall dependencies | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Test the App in the Browser | ng s -o // build, deploy, open and automatically refresh (localhost:4200)  ng s // start server and automatically refresh app (localhost:4200) |
| Use Back End & Database | npm start or nodemon // in the server directory  open MongoDB Compass + mongod.exe // C:/Program Files/MongoDB/server/4.4/bin |  |
| "npm ERR! jasmine-core@">=3.8"..." | package.json:  "devDependencies": { "jasmine-core": "3.8" } |  |
| "Module tslib cannot be found" | tsconfig.json:  "compiler options": {  "baseUrl": "./",  "paths": { "tslib": ["./node-modules/tslib/tslib.d.ts"] }  } |  |
| Set Environment Variable for API URL | environments/environment.ts:  export const environment = {  production: false,  apiURL: 'http://localhost:3000/api'  };  environments/environment.prod.ts:  export const environment = {  production: true,  apiURL: 'http://mywebsite.com/api'  }; |  |

### CREATE MODULES FOR THE APP PARTS

|  |  |  |
| --- | --- | --- |
| App Module | automatically generated | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Core Module | ng g m core // for components used only once (ex. header, footer)  app.module.ts: imports: [CoreModule] |
| Shared Module | ng g m shared // for components/interfaces used more than once  app.module.ts: imports: [SharedModule] |
| User Module | ng g m user // for register, login, profile and user service  app.module.ts: imports: [UserModule] |
| Theme Module | ng g m theme // for catalog, details, add theme, aside  app.module.ts: imports: [ThemeModule] |

### CREATE COMPONENTS FOR THE APP VIEWS

|  |  |  |
| --- | --- | --- |
| Header & Footer | ng g c core/header // creates a header component in the core module  ng g c core/footer // creates a footer component in the core module  core.module.ts: exports: [HeaderComponent, FooterComponent]  app.component.html: <app-header></...><app-footer></...> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Home | ng g c home // creates a home component in the app module |
| Welcome Message | ng g c shared/welcome-message  shared.module.ts: exports: [WelcomeMessageComponent] |
| Themes, Theme, New Theme & Aside | ng g c theme/themes  ng g c theme/theme  ng g c theme/new-theme  ng g c theme/aside |
| Register, Login, Profile | ng g c user/register  ng g c user/login  ng g c user/profile |  |
| Not Found (404) | ng g c not-found |  |

### ADD ROUTING

|  |  |  |
| --- | --- | --- |
| App Routing Module | ng g m app-routing  app-routing.module.ts:  import { Routes, RouterModule } from '@angular/router';  const routes: Routes = [  { path: '', pathMatch: 'full', redirectTo: 'home'},  { path: 'home', component: HomeComponent },  { path: '\*\*', component: NotFoundComponent }  ];  export const AppRoutingModule = RouterModule.forRoot(routes);  app.module.ts: imports: [AppRoutingModule] // must be the last import!  app.component.html: <router-outlet></router-outlet> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Theme Routing Module | ng g m theme/theme-routing  theme-routing.module.ts:  import { Routes, RouterModule } from '@angular/router';  const routes: Routes = [  { path: 'themes', children: [  { path: '', pathMatch: 'full', component: ThemesComponent },  { path: ':themeId', component: ThemeComponent }  ] },  { path: 'add-theme', component: NewThemeComponent }  ];  export const ThemeRoutingModule = RouterModule.forChild(routes);  theme.module.ts: imports: [ThemeRoutingModule] |  |
| User Routing Module | ng g m user/user-routing  user-routing.module.ts:  import { Routes, RouterModule } from '@angular/router';  const routes: Routes = [  { path: 'register', component: RegisterComponent },  { path: 'login', component: LoginComponent },  { path: 'profile', component: ProfileComponent }  ];  export const ThemeRoutingModule = RouterModule.forChild(routes);  user.module.ts: imports: [UserRoutingModule] |  |

### CREATE INTERFACES

|  |  |  |
| --- | --- | --- |
| Theme Interface | shared/interfaces/theme.ts:  export interface ITheme {  subscribers: string[];  userId: { themes: string[], \_id: string, username: string, ... };  ...  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Post Interface | shared/interfaces/post.ts:  export interface IPost { likes: string[]; \_id: string; ... } |  |
| User Interface | shared/interfaces/user.ts:  export interface IUser { firstName: string; lastName: string; ... } |  |
| Set an index.ts File to Export Interfaces | shared/interfaces/index.ts:  export \* from './theme'; export \* from './post'; export \* from './user'; |  |

### CREATE SERVICES

|  |  |  |
| --- | --- | --- |
| Content Service | ng g s theme/content  content.service.ts:  const API\_URL = environment.apiURL;  export class ContentService {  constructor(private http: HttpClient) { }  loadTheme(id: string) {  return this.http.get<ITheme>(`/api/themes/${id}`);  }  loadThemes() { ... } // sorting should be done in the back end!  saveTheme(theme: any) {  return this.http.post<ITheme>(`/api/themes`, theme);  }  loadPosts(limit?: number) {  const query = limit ? `?limit=${limit}` : '';  return this.http.get<IPost[]>(`/api/posts${query}`);  }  }  theme.module.ts: imports: [HttpClientModule] | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| User Service | ng g s user/user  user.service.ts:  export class UserService {  constructor(private http: HttpClient) { }  user: IUser | null | undefined = undefined;  get isLogged(): boolean { return !!this.user; }  getProfileInfo() { // to initially check if there is a logged in user  return this.http.get<IUser>('/api/users/profile')  .pipe(tap((user) => this.user = user));  }  register(user: { username: string; email: string; phone: string; password: string }) {  return this.http.post<IUser>('/api/register', user)  .pipe(tap((user) => this.user = user));  }  login(user: { email: string; password: string }) { ... }  logout() {  return this.http.post<IUser>('/api/logout', {})  .pipe(tap((user) => this.user = null));  }  updateProfile(user: { username: string; email: string; phone: string; password: string }) { ... }  } |  |

### CREATE INTERCEPTORS AND PIPES

|  |  |  |
| --- | --- | --- |
| Interceptor | core/app-interceptor.ts:  const { apiURL } = environment;  @Injectable()  export class AppInterceptor implements HttpInterceptor {  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  constructor(private router: Router) { }  let reqStream$ = next.handle(req);  if (req.url.startsWith('/api')) {  reqStream$ = next.handle(req.clone({  url: req.url.replace('/api', apiURL),  withCredentials: true  }));  }  return reqStream$.pipe(catchError(err => {  this.router.navigate(['/'],  { queryParams: { error: 'Oops! Something went wrong!' } });  return throwError(err);  }));  }  }  export const appInterceptorProvider: Provider = {  provide: HTTP\_INTERCEPTORS,  useClass: AppInterceptor,  multi: true  };  core.module.ts: providers: [appInterceptorProvider] | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Shorten Pipe | ng g p shared/pipes/shorten  shorten.pipe.ts:  transform(value: string, limit: number): string {  return value.length > limit ? `${value.substr(0, limit)}...` : value;  }  shared.module.ts: exports: [ShortenPipe] |  |
| Time Difference Pipe | ng g p shared/pipes/time-diff  time-diff.pipe.ts:  transform(value: string): string {  const timePassed = Number(new Date()) - Number(new Date(value));  if (timePassed < 60000) {  return `${Math.floor(timePassed / 1000)} seconds`;  }  if (timePassed < 60000 \* 60) {  return `${Math.floor(timePassed / 60000)} minutes`;  }  if (timePassed < 60000 \* 60 \* 24) {  return `${Math.floor(timePassed / (60000 \* 60))} hours`;  }  if (timePassed < 60000 \* 60 \* 24 \* 30) {  return `${Math.floor(timePassed / (60000 \* 60 \* 24))} days`;  }  if (timePassed < 60000 \* 60 \* 24 \* 30 \* 12) {  return `${Math.floor(timePassed / (60000 \* 60 \* 24 \* 30))} months`;  }  return `${Math.floor(timePassed / (60000 \* 60 \* 24 \* 30 \* 12))} years`;  }  shared.module.ts: exports: [TimeDiffPipe] |  |

### AUTHENTICATION AND GUARDS

|  |  |  |
| --- | --- | --- |
| Check if There Is a Logged User by Initial Page Loading | app.component.ts:  get isAuthenticating(): boolean {  return this.userService.user === undefined;  }  constructor(private userService: UserService) {  this.userService.getProfileInfo().subscribe({  error: (error) => {  this.userSrevice.user = null;  }  });  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Implement UserService for Login/Register | login.component.ts:  constructor(  private userService: UserService,  private activatedRoute: ActivatedRoute,  private router: Router) { }  login(email: string, password: string): void {  this.userService.login(email, password);  const redirectUrl = this.activatedRoute.snapshot.queryParams.redirectUrl || '/';  this.router.navigate([redirectUrl]);  } |
| Create Guards | auth.activate.ts:  @Injectable()  export class AuthActivate implements CanActivate {  constructor(private router: Router, private userService: UserService){}  canActivate(route: ActivatedRouteSnapshot): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {  const {authRequired, authFailureRedirectUrl } = route.data;  if (typeof authRequired == 'boolean' && authRequired == this.userService.isLogged) { return true; }  let authRedirectUrl = authFailureRedirectUrl;  if (authRequired) {  const loginRedirectUrl = route.url.reduce((acc, s) => `${acc}/${s.path}`, '');  authRedirectUrl += `?redirectUrl=${loginRedirectUrl}`;  }  return this.router.parseUrl(authRedirectUrl || '/');  }  }  core.module.ts: providers: [AuthActivate] |
| Use Guards in Routing | theme-routing.module.ts:  { path: 'add-theme',  component: NewThemeComponent,  canActivate: [AuthActivate],  data: {authRequired: true, authFailureRedirectUrl: '/login' }  } |
| Fake Authentication |  |  |

### RENDER CONTENT

|  |  |  |
| --- | --- | --- |
| Set Header Navigation | header.component.ts:  constructor(  private userService: UserService,  private router: Router  activatedRoute: ActivatedRoute  ) { this.errorMessage = activatedRoute.snapshot.queryParams.error; }  get isLogged(): boolean { return !!this.userService.user; }  get username(): string { return this.userService.user.username; }  errorMessage: string;  logout(): void { this.userService.logout(); this.router.navigate(['/']); }  header.component.html:  <ul *\*ngIf*="!isLogged"><li><a *routerLink*="login">Login</a></li>...</ul>  <li *(click)*="logout()">Logout</li>  <p *class*="notification error-message">{{errorMessage}}</p>  core.module.ts: imports: [RouterModule] // allows *routerLink* in templates | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Render Home Page | home.component.ts:  constructor(private userService: UserService) { }  get isLogged(): boolean { return !!this.userService.user; }  home.component.html:  <app-welcome-message *[isLogged]*="isLogged"><app-welcome-message>  welcome-message.component.ts: @Input() isLogged!: boolean;  welcome-message.component.html: <ul *\*ngIf*="!isLogged">...</ul> |
| Render Themes | themes.component.ts:  themes: ITheme[] | undefined;  constructor(private contentService: ContentService) {  this.fetchThemes();  }  fetchThemes(): void {  this.themes = undefined;  this.contentService.loadThemes()  .subscribe(themes => this.themes = themes);  }  themes.component.html:  <div *\*ngFor*="let theme of themes">...</div>  <a *[routerLink]*="['/themes', theme.\_id]">{{theme.themeName}}</a> |
| Render Posts in Aside | themes.component.ts:  recentPosts: IPost[] | undefined;  constructor(private contentService: ContentService) {  this.fetchRecentPosts();  }  fetchRecentPosts(): void {  this.recentPosts = undefined;  this.contentService.loadPosts(5)  .subscribe(posts => this.recentPosts = posts);  }  themes.component.html:  <app-aside *[recentPosts]*="recentPosts"></app-aside>  aside.component.ts: @Input() recentPosts: IPost[] | undefined;  aside.component.html: <div *\*ngFor*="let post of recentPosts">...</div> |  |
| Render Single Theme | theme.component.ts:  theme: ITheme | undefined;  constructor(  private contentService: ContentService,  private activatedRoute: ActivatedRoute) {  this.fetchTheme();  }  fetchTheme(): void {  this.theme = undefined;  const id = this.activatedRoute.snapshot.params.themeId;  this.contentService.loadTheme(id)  .subscribe(theme => this.theme = theme);  } |  |
| Render Profile Page | profile.component.ts:  user = this.userService.user;  isLoading = true;  constructor(private userService: UserService) {  this.userService.getProfileInfo()  .subscribe(() => this.isLoading = false);  }  profile.component.html:  <div *\*ngIf*="isLoading">Loading...</div>  <div *\*ngIf*="!isLoading">...</div> |  |

### CREATE CUSTOM VALIDATION

|  |  |  |
| --- | --- | --- |
| Create Custom Email Validator (shared/validators.ts) | export function emailValidator(control) {  if (!control.value) { return; }  return /^.{6,}@gmail\.(com|bg)$/.test(control.value) ? null :  { invalidEmail: true };  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create Custom Same Value Validator (shared/validators.ts) | export function sameValueAsFactory(getTargetControl: () =>  AbstractControl | null) {  let subscription: Subscription | null = null;  return function (control: AbstractControl) {  if (subscription) { subscription.unsubscribe(); subscription = null; }  const targetControl = getTargetControl();  if (!targetControl) { return null; }  subscription = control.statusChanges.pipe(  filter(() => false), startWith(null),  switchMap(() => targetControl.valueChanges)  ).subscribe({  next: () => { control.updateValueAndValidity(); },  complete: () => { subscription = null; }  });  return targetControl.value == control.value ? null : { noMatch: true };  }  } |
| Create a Custom Validator Directive | ng g d shared/custom-validator  custom-validator.directive.ts:  selector: '[appCustomValidator]',  providers: [  { provide: NG\_VALIDATORS,  useExisting: CustomValidatorDirective,  multi: true }  ]  export class CustomValidatorDirective implements Validator {  @Input('appCustomValidator') validatorFn: ValidatorFn;  constructor() { }  validate(control: AbstractControl): ValidationErrors {  if (!this.validatorFn) { return null; }  return this.validatorFn(control);  }  }  shared.module.ts: exports: [CustomValidatorDirective]  user.module.ts: imports: [SharedModule] |  |
| Use Custom Validator Directive in Template | <input *name*="email" *[appCustomValidator]*="emailValidator"> |  |
| Style Input Field Border | input { border-left: 0.3em solid green; }  input.ng-invalid.ng-touched { border-left: 0.3em solid red; } |  |
| Style Input Field Border for Grouped Fields | input { border-left: 0.3em solid green; }  .passwords.ng-invalid.ng-touched input { border-left: 0.3em solid red; } |  |
| Disable Submit Button | login.component.html: <button *[disabled]*="form.invalid">...</button>  login.component.css: button[disabled] { background-color: #4EBDED86 } |  |

### LOGIN/REGISTER FORMS (REACTIVE)

|  |  |  |
| --- | --- | --- |
| Import Forms Module | user.module.ts: imports: [ReactiveFormsModule] | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare Template Variables to the Form and Input Fields | login.component.html:  <form *[formGroup]*="form" *(ngSubmit)*="login(form)">  <input *name*="email" *formControlName*="email">  </form> |
| Display Form Data and Errors for Easy Debugging | {{form.value | json}}  <form>  <input ...>{{form.get('email').errors | json}}  </form> |  |
| Build the Form Using the FormBuilder | login.component.ts:  form: FormGroup;  emailValidator = emailValidator;  constructor(private fb: FormBuilder) {  this.form = this.fb.group({  username: ['', [Validators.required, Validators.minLength(3)]],  email: ['', [Validators.required, emailValidator]] // custom validator  });  } |  |
| Display Error Messages | <ng-container *\*ngIf*="form.get('email').touched">  <p *\*ngIf*="form.get('email').errors?.required">Email is required!</p>  </ng-container> |  |
| Implement User Service | login.component.ts:  constructor(private userService: UserService, private router: Router, private activatedRoute: ActivatedRoute) { }  login(form): void {  if (form.invalid) return;  const { email, password } = this.form.value;  this.userService.login({ email, password }).subscribe({  next: () => {  const redirectUrl = this.activatedRoute.snapshot.queryParams.redirectUrl || '/';  this.router.navigate([redirectUrl]); },  error: (err) => { console.error(err); }  });  } |  |

### ADD/EDIT ITEM FORMS (TEMPLATE-DRIVEN)

|  |  |  |
| --- | --- | --- |
| Import Forms Module | theme.module.ts: imports: [FormsModule] | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Declare Template Variables to the Form and Input Fields | login.component.html:  <form *#form*="ngForm">  <input *name*="email" *ngModel* *#emailInput*="ngModel">  </form> |
| Display Form Data and Errors for Easy Debugging | {{form.value | json}}  <form>{{form.value.username | json}}  <input ...>{{usernameInput.errors | json}}  </form> |  |
| Display Error Messages | <ng-container *\*ngIf*="emailInput.touched">  <p *\*ngIf*="emailInput.errors?.required">Email is required!</p>  </ng-container> |  |

### DOCUMENTATION

|  |  |  |
| --- | --- | --- |
| Embed Image into the .md File | ![](./image.png)  <img *src*="./image.png" *width*="100"> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  |  |

# TYPESCRIPT

## PURPOSE AND INSTALLATION

|  |  |  |
| --- | --- | --- |
| About TypeScript | TypeScript is a superset of JavaScript that provides powerful additional features (type annotations, classes, interfaces, modules, etc.) and compiles to plain JavaScript. It is a primary language for Angular application development. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| TypeScript in the Browser | Browsers can't execute TypeScript directly, it is transferred into JavaScript code via a compiler. The TypeScript compiler configuration in an Angular project can be found in tsconfig.json. |
| File Extension | .ts |
| Test TS Code Online | https://www.typescriptlang.org/play |
| Installation | npm install typescript // specific TS version for each project |
| Checking Version | tsc -v // tsc -version |  |
| Generate a tsconfig.json File | tsc --init |  |
| TypeScript Compiler (Creates a .js File) | npx tsc test.ts // if TS installed globally: tsc ./test.ts  npm run build // package.json > "scripts": { "build": "tsc test.ts" }  tsc // converts all .ts files in the tsconfig.json directory |  |
| Use Webpack | npm install ts-loader -save-dev |  |

## TYPE ANNOTATIONS

|  |  |  |
| --- | --- | --- |
| String | let color: string;  let color: string = 'blue';  color = 2; // throws a message 'Type "number" is not assignable to type "boolean"' | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Number | let decimal: number = 6;  let hex: number = 0xf00d;  let binary: number = 0b1010;  let octal: number = 0o744; |
| Boolean | let isOpen: boolean = true; |
| Array | const list: number[] = [1, 2, 3]; // const list: Array<number> = [1, 2, 3];  const list: (string | number)[] = ['Apples', 2, 'Bananas', 3];  const list: any[] = ['ABC', 12, false]; |
| Object | const point: { x: number, y: number } = { x: 1, y: 2 }; // inline annotation to define the shape of an object |  |
| Tuple | const employee: [number, string, boolean] = [1, 'Steve', true]; |  |
| Enum (Numeric, String, Heterogeneous) | enum Letters { A, B } // { '0': 'A', '1': 'B', A: 0, B: 1 }  Letters.A, Letters[0] // 0 A (reverse mapping for numeric members)  enum Letters { A = 1, B } // A = 1, B = 2 (first member initiated)  enum Letters { A, B = 10, C, D = B + 5, E, F = getV('f'), G } // A = 0, C = 11, D = 15, E = 16, G must have initializer  enum Letters { A = 'A', B = 'B' }  enum Status { Active = 'ACTIVE', Deactivate = 1, Pending } |  |
| Union | let code: string | number; // or (string | number)  code = 'ABC'; // OK  code = 32; // OK  code = true; // Compiler Error |  |
| Any | let code: any;  code = 'ABC'; // OK  code = true; // OK |  |
| Void | function sayHi(): void { console.log('Hi!'); } // no data returned  console.log(sayHi()); // undefined |  |
| Never | function throwError(msg: string): never { throw new Error(msg); }  function keepProcessing(): never { while (true) console.log('...'); } |  |
| Read-Only Type | interface IEmployee { name: string; }  const employee: Readonly<IEmployee> = { name: 'John' };  employee.name // 'John'  employee.name = 'Ben'; // Compiler Error |  |

## TYPE INFERENCE

|  |  |  |
| --- | --- | --- |
| Primitive Values | let a = 'some text'; // a: string | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Complex Objects | const arr = [10, null, 30]; // arr: number[]  const arr = [0, 1, 'test']; // arr: (number | string) |
| Return Type of a Function | function sum(a: number, b: number) { return a + b; } // return number  let total: string = sum(10, 20); // Compiler Error |  |

## TYPE ASSERTION

|  |  |  |
| --- | --- | --- |
| Primitive Values | const code: any = 123;  const employeeCode = <number> code; // typeof employeeCode = number  const employeeCode = code as number; // typeof employeeCode = number | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Object | interface IEmployee = { name: string, code: number };  const employee = <IEmployee> {}; // allows properties 'name' and 'code'  const employee = {} as IEmployee; // allows properties 'name' and 'code' |
| Non-Null Assertion | @Input() name!: string; // asserts that 'name' won't be undefined/null  @Input() name: string | undefined; // if no value, 'name' will be undefined |  |

## FUNCTIONS

|  |  |  |
| --- | --- | --- |
| Parameter Types And Number | function sum(x: number, y: number) { return x + y; }  sum(2, 3) // 5  sum(2) // Compiler Error: Expected 2 arguments, but got 1.  sum(2, 3, 4) // Compiler Error: Expected 2 arguments, but got 3. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Return Type | function sum(x, y): number { return x + y; } |
| Optional Parameters | function greet(greet: string, name?: string) { return greet + ', ' + name }  greet('hi', 'Steve') // 'hi, Steve'  greet('hi') // 'hi, undefined' |  |
| Default Parameters | function sum(x: number, y: number = 10) { return x + y; }  sum(2) // 12  function sum(x: number = 10, y: number) { return x + y; }  sum(undefined, 2) // 12 |  |
| Function Overloading | function add(a: string, b: string): string; // declaration (2 parameters)  function add(a: number, b: number): number; // declaration (2 parameters)  function add(a: any, b: any): any { return a + b; } // implementation  add('hello', 'Steve'); // 'hello Steve'  add(10, 20); // 30 |  |
| Rest Parameters | const hi = (greet: string, ...men: string[]) => greet + ' ' + men.join(', ');  hi() // 'hi '  hi('John') // 'hi John'  hi('John', 'Bob', 'Thom') // 'hi John, Bob, Thom' |  |

## INTERFACES

|  |  |  |
| --- | --- | --- |
| Purpose | An interface defines the structure of a class/object. It is NOT converted to JavaScript, only used for type checking (duck typing, structural subtyping). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Interface as Type | interface IKeyPair { key: number; value: string; }  const kv1 = { key: 1, value: 'Steve' };  const kv2 = { key: 1, val: 'Steve' }; // Compiler Error |
| Interface as Function Type | interface IGreet { (name: string): string; }  function hi(name: string): string { return `Hi, ${name}!`; }  function bye(name: string): string { return `Bye, ${name}!`; }  let greet: IGreet = hi;  greet('Bill') // 'Hi, Bill!'  greet = bye;  greet('Steve') // 'Bye, Steve!' |  |
| Interface for Array Type | interface INumList { [index: number]: number; }  const numArr: INumList = [1, 2, 3];  numArr[1] // 2  interface IStringList { [index: string]: string }  const stringArr: IStringList = { JS: 'JavaScript' };  stringArr['TS'] = 'TypeScript'; |  |
| Optional Properties/Methods | interface IEmployee { name: string; dept?: string; }  const first: IEmployee = { name: 'Steve' };  const second: IEmployee = { name: 'Steve', dept: 'IT' }; |  |
| Read-Only Properties/Methods | interface ICitizen { name: string; readonly SSN: number }  const person: ICitizen = { SSN: 110555444, name: 'James Bond' };  person.name = 'Steve Smith';  person.SSN = 123555444; // Compiler Error |  |
| Extend an Interface | interface IPerson { name: string; gender: string; }  interface IEmployee extends IPerson { empCode: number; }  const employee: IEmployee = { name: 'Ben', gender: 'M', empCode: 12 }; |  |
| Extend a Class | class Person { name: string; }  interface IEmployee extends Person { code: number; }  const employee: IEmployee = { code: 1, name: 'James' }; |  |
| Implement an Interface with a Class | interface IEmployee { name: string; code: number; }  class Employee implements IEmployee {  constructor(public code: number, public name: string) { }  getSalary(): number { return 20000; }  }  const employee = new Employee(123, 'Ben'); |  |

## CLASSES

|  |  |  |
| --- | --- | --- |
| Declare a Class in TypeScript | class Employee {  constructor(public code: number, public name: string) { }  getSalary(): number { return 20000; }  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Converted Class in .js File | var Employee = /\*\* *@class* \*/ (function () {  function Employee(code, name) { this.code = code; this.name = name; }  Employee.prototype.getSalary = function () { return 20000; };  return Employee;  }()); |
| Implement One or Several Interfaces | interface IPerson { name: string; display(): string; }  interface IEmployee { code: number; }  class Employee implements IPerson, IEmployee = {  constructor(public code: number, public name: string) { }  display(): string { return `${this.name} ${this.code}` }  }  const person: IPerson = new Employee(123, 'Ben');  person.display() // 'Ben 123'  const employee: IEmployee = new Employee(123, 'Ben');  employee.display() // Compiler Error |  |
| Method Overriding | class Car {  constructor(public name: string) { }  run(speed: number = 0) { return `${this.name} is fast: ${speed}` }  }  class Honda extends Car {  constructor(public name: string) { super(name); }  run(speed = 150) { return `A Honda started. ${super.run(speed)}` }  }  const honda = new Honda('Honda City');  honda.run() // 'A Honda started. Honda City is fast: 150' |  |
| Abstract Classes | abstract class Person { // cannot be instantiated  constructor(public name: string) { this.name = name; }  abstract getName(): string; // must be implemented in the derived class  }  class Employee extends Person {  constructor(name: string) { super(name); } // must call super()  getName(): string { return this.name; }  } |  |
| Access Modifiers | class Person {  public code!: number;  name!: string;  private phone!: number;  protected address!: string;  }  const person = new Person();  person.code = 123; // OK  person.name = 'Ben'; // OK  person.phone = 123456; // Compiler Error: Property 'phone' is only accesible within class 'Person'.  person.address = '12 A Str.'; // Compiler Error: Property 'address' is only accessible within class 'Person' and its subclasses.  person // { code: 123, name: 'Ben', phone: 123456, address: '12 A Str.' } (despite the compiler errors!) |  |
| Read-Only Properties/Methods | class Employee { constructor(readonly code: number) {} }  const employee = new Employee(10);  employee.code // 10  employee.code = 20; // Compiler Error |  |
| Static Properties/Methods | class Employee { static code: number = 1; name: string = 'A'; }  const employee = new Employee();  employee.name // 'A'  employee.code // Compiler Error  Employee.code // 1 |  |

## GENERICS

|  |  |  |
| --- | --- | --- |
| Use Generic Type | function getArr<T>(arr: T[]): T[] { return new Array<T>().concat(arr) }  const numArr = getArr<number>([1, 2, 3]);  const strArr = getArr<string>(['a', 'b', 'c']);  numArr.push(4); // OK  strArr.push('d'); // OK  numArr.push('d'); // Compliler Error  strArr.push(4); // Compliler Error | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Multiple Type Variables | function type<T, U>(id: T, str: U): string { return typeof id + typeof str; }  type<number, string>(2, 'hello'); // 'numberstring' |
| Use Generic Type Combined with Non-Generic Types | function type<T>(id: T, str: string): string { return typeof id + typeof str; }  type<number>(2, 'hello'); // 'numberstring' |
| Generic Type Methods and Properties | function type<T, U>(id: T, str: U): string {  str.toString(); // OK; available for all types  id.toFixed(); // Compiler Error  return typeof id + typeof str;  } |  |
| Generic Interfaces | interface IKeyPair<T, U> { key: T; value: U; }  const keyPair: IKeyPair<number, string> = { key: 1, value: 'a' }; |  |
| Generic Classes | class KeyValue<T, U> { constructor(public k: T, public v: U) {} }  const kvp = new KeyValue<number, string>(1, 'Ben');  kvp.val // 'Ben' |  |
| Generic Constraints | class Person { constructor(public name: string) {} }  function display<T extends Person>(per: T): string { return per.name; }  const person = new Person('Ben');  display(person); // 'Ben'  display({ age: 43, name: 'Ben' }); // 'Ben'  display('Ben'); // Compiler Error |  |

## MODULES

|  |  |  |
| --- | --- | --- |
| Modules in TypeScript | In TypeScript, the code we write is in the global scope by default. Modules are a way to create a local scope in the file. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Module and Export Data from It | export let greeting: string = 'Hello TS'; // greeting.ts is a module, but only exported data can be used in other files |
| Import Data from a Module and Use It | import { greeting } from './greeting';  console.log(greeting); // 'Hello TS'  greeting = 'Hello there!'; // CANNOT assign to greeting as it is an import |  |
| Import and Rename an Export | import { greeting as greetingTS } from './greeting'; |  |
| Import the Entire Module into a Variable | import \* as greeting from './greeting'; |  |
| Default Export and Import | export default class MyClass { ... }  import MyClass from './MyClass' // no {}!  import MyRenamedClass from './MyClass' // no {}! |  |
| Compile a TypeScript Module | tsc -m amd .\greeting.ts // or tsc --module amd .\greeting.ts |  |

## NAMESPACES

|  |  |  |
| --- | --- | --- |
| Namespaces in TypeScript | The namespace is used for logical grouping of functionalities and can include interfaces, classes, functions and variables. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Namespace | namespace StringUtility {  function toCapital(s: string): string { return s.toUpperCase(); }  function toLower(s: string): string { return s.toLowerCase(); }  } |
| Export Namespace Components | namespace StringUtility {  export function toCapital(s: string): string { return s.toUpperCase(); }  export function toLower(s: string): string { return s.toLowerCase(); }  } |  |
| Include a Namespace and Use It in Another File | /// <reference path='StringUtility.ts' />  const name = StringUtility.toCapital(employee.name); |  |
| Compile a Namespace | tsc --outfile File.js File.ts // then include File.js in <script> |  |

# REACT JS

## PURPOSE, INSTALLATION, SETTINGS

|  |  |  |
| --- | --- | --- |
| About React | React is a JS library for building user interfaces (UI), focused on creating reusable components in a declarative way (you only describe the desired result). It is easy to learn, can be used on both client and server, and has great documentation and error reporting. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Documentation | https://reactjs.org/docs/getting-started.html |
| VS Code Settings | Ctrl + , > Workspace > Extensions > Emmet > Edit in settings.json:  {  "emmet.includeLanguages": {  "javascript": "javascriptreact"  },  "emmet.syntaxProfiles": {  "javascript": "jsx"  }  }  (Select Language Mode from the VSC bottom bar: JavaScript React) |  |
| React Developer Tools for Chrome | 1. install React Developer Tools extension for Chrome 2. Chrome extensions settings > Allow access to file URLs 3. run npm install -g react-devtools 4. run react-devtools 5. F12 > Components and Profiler |  |
| Debug React in Chrome | VS Code > Run & Debug > Add Configuration > launch.json:  "configurations": {  "url": "http://localhost:3000",  "webRoot": "${workspaceFolder}/src"  }  Then use F5 with breakpoints. |  |

## CREATE A REACT APP

|  |  |  |
| --- | --- | --- |
| Run the React App Creator | npx create-react-app my-app // an app "my-app" in current directory | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| If Prompted to Uninstall Globally Installed CRA | npm uninstall -g create-react-app  npx clear-npx-cache npx create-react-app my-app |
| Start the App | cd my-app  npm start // npm run build, serve -s build for production |
| React App Structure | 1. package.json: project configuration 2. public/index.html: main HTML file (favicon: <link rel="icon" href="%PUBLIC\_URL%/favicon.ico?" />) 3. src/index.js: main JS file 4. src/App.js, App.css, App.test.js: React component "App" 5. src/components/Header.js, Footer.js: all custom-made components |  |
| Use a Free Website Template | 1. download a free HTML & CSS template 2. read LICENSE.txt 3. move folder "assets" to project folder "public" 4. copy "index.html" into public/index.html (<head>) and src/App.js (<body>) 5. clean up public/index.html and src/App.js (remove "../assets" from paths, *class* > *className*, *for* > *htmlFor*, *style*="background-image: url(...);" > *style*={{backgroundImage: 'url(...)'}}, delete comments, maybe remove strict mode from src/index.js)   create src/components/: Header.js, Footer.js, ... |  |
| Real-Time Communication Browser – Server | socket.io |  |

## JSX

|  |  |  |
| --- | --- | --- |
| About JSX | JSX is a JavaScript superset language which looks like a template language but comes with the full power of JavaScript. After compilation, JSX expressions become regular JavaScript function calls. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| React and JSX | React doesn't require JSX, but most people find it helpful as a visual aid when working with UI inside the JavaScript code. It also allows React to show more useful error and warning messages. |
| JSX Formatting | It is recommended to split JSX over multiple lines and wrap it in parentheses to avoid the pitfalls of automatic semicolon insertion:  const element = (  <header>  <h1>Hello, React!</h1>  </header>  ); |
| Embedding Expressions in JSX | const name = 'Josh Perez';  const element = <h1>Hello, {name}</h1>; |
| Specifying Attributes with JSX (""/{}) | const element = <div *tabIndex*="0"></div>; // camelCase, unlike HTML!  const element = <img *src*={user.avatarUrl} />; |
| Specifying Children with JSX | const element = <img *src*={user.avatarUrl} />; // self-closed empty tag  const element = <div><p>Hi!</p></div>; // tag with children |  |
| Custom Components | <MyComponent>, <Greeting>, <ScoreBoard> // always PascalCase! |  |

## ELEMENTS

|  |  |  |
| --- | --- | --- |
| About React Elements | Unlike browser DOM elements, React elements are plain objects representing the UI, and are cheap to create. They are immutable, so we need to create a new element if we want an update. The library ReactDOM takes care of updating the virtual DOM (VDOM) to match the React elements by only updating what's necessary. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Root Element | Usually a <div> element with id "root", showing that everything inside it will be managed by React DOM. |
| Create an Element | const element = <h1 *className*="title">Hi</h1>;  // Babel compiles the upper to:  const element = React.createElement('h1', {className: 'title'}, 'Hi');  // which creates an object like this (called React element):  const element = { type: 'h1', props: { className: 'title', children: 'Hi' } }; |  |
| Render an Element into a Root DOM Node | const element = <h1>Hello, world</h1>;  ReactDOM.render(element, document.getElementById('root'));  // the ONLY place in a React app to use 'document' (reference the real DOM) |  |
| Update the Rendered Element (= Create and Render a New Element) | function tick() {  const element = (  <>  <h1>Hi</h1>  <p>It is {new Date().toLocalTimeString()}.</p>  </>  );  ReactDOM.render(element, document.getElementById('root'));  }  setInterval(tick, 1000); // creates a ticking clock by rendering a new element each second |  |

## COMPONENTS

### GENERAL

|  |  |  |
| --- | --- | --- |
| About React Components | React components are like JavaScript functions. They accept arbitrary input ("props") and return React elements describing what should appear on the screen. Their names must start with a capital letter. All React components must act like pure functions with respect to their props (NOT change them). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Reasons Why a Component Renders | 1. initial render 2. the parent rerenders 3. the props changed 4. a hook changed |
| Types of Components | We distinguish between class components and functional components (more and more used): classes that create React components and functions that return React components. |
| Sample Functional Component | export default function Header() {  return (  <header> // there must be one root element (or fragment: <></>)  <h1>Hello React</h1><p>A short subtitle</p>  </header>  );  } |
| Sample Class Component | import React from 'react';  export default class Header extends React.Component {  render() {  return <header><h1>Hello React</h1><p>Subtitle</p></header>  }  } |  |
| Use a Child Component in the Root Component | src/App.js: <Header /> |  |
| Prevent a Component from Rendering | function WarningBanner(props) {  if (!props.warn) return null;  return <div>Warning!</div>;  } |  |

### PASS DATA FROM PARENT TO CHILD

|  |  |  |
| --- | --- | --- |
| About Props and State | Props are like immutable arguments passed to the function/class directly from its parent. All components have props. We can add them into the HTML like custom-made attributes or directly use the content of the HTML tag.  State starts with a default value when a component mounts and suffers from mutations in time. It is managed within the component (like local variables) and allows us to create dynamic and interactive components. Every time we modify the state, React will re-render the component. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Use Props to Pass Data | src/App.js: <ListItem *text*="Clean your room" />  src/components/ListItem.js:  export default const ListItem = (props) => {  return <li>{props.text}</li>;  } |
| Use Props Children to Pass Data | src/App.js:  <ListItem>Clean your room</ListItem> // plain text or nested HTML  src/components/ListItem.js:  export default const ListItem = (props) => {  return <li>{props.children}</li>;  } |
| Use Props in Class Components | src/App.js:  <ListItem>Clean your room</ListItem> // plain text or nested HTML  src/components/ListItem.js:  import React from 'react';  class ListItem extends React.Component {  render() { return <li>{this.props.children}</li>; }  } |  |
| Use State | import { useState } from 'react';  export default function Counter() {  let [count, setCounter] = useState(0);  if (*some condition*) setCounter(count + 1);  return <p>{count}</p>;  } |  |
| Use State in Class Components | import React from 'react';  class ListItem extends React.Component {  constructor(props) {  super(props);  this.state = { hobbies: [], selectedHobby: null };  }  render() { return <li>{this.props.children}</li>; }  } |  |

### PRESERVE STATE BETWEEN RERENDERS

|  |  |  |
| --- | --- | --- |
| Functional Components | import { useState, useEffect } from 'react';  const App = () => {  const [info, setInfo] = useState({ name: 'Ben', age: 30 });  const onClick = () => {  setInfo(oldState => ({ ...oldState, age: oldState.age + 1 }));  };  return (  <>  <p>{`${name}: ${age}-years old`}</p>  <button *onClick*={onClick}>+</button>  </>  );  }; | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Class Components | import { Component } from 'react';  class App extends Component {  constructor() {  super();  this.state = { name: 'Ben', age: 30 };  this.onClick = () => {  this.setState({ age: this.state.age + 1 }); // merges old & new state!  }  }  render() {  return (  <>  <p>{`${this.name}: ${this.age}-years old`}</p>  <button *onClick*={this.onClick}>+</button>  </>  );  }  } |

### CHANGE THE PARENT'S STATE FROM A CHILD

|  |  |  |
| --- | --- | --- |
| To Do List | export default function ToDoList() {  const [todos, setTodos] = useState([{id: 1, text: 'bla bla'}]);  const deleteItem = (id) => setTodos(state => state.filter(x.id !== id));  return (  <ul>  {todos.map(x => <ListItem *key*={x.id} *id*={x.id}  *text*={x.text} *onDelete*={deleteItem} />)}  </ul>  );  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| To Do List Item | export default function ListItem({id, text, onDelete}) {  return (  <li>{text} <button *onClick*={() => onDelete(id)}>x</button></li>  );  } |
|  | import React from 'react';  class ListItem extends React.Component {  constructor(props) {  super(props);  this.state = { hobbies: [], selectedHobby: null };  }  onClick(e) { this.setState({ selectedHobby: e.target.textContent }); }  render() {  return (  <li *onClick*={this.onClick.bind(this)}>{this.props.children}</li>  );  }  } |  |

### COMPONENT LIFECYCLE HOOKS/METHODS

|  |  |  |
| --- | --- | --- |
| Mouting Hook in Functional Components | Where the component and all its children are mounted (created and inserted into the DOM):  import { useEffect } from 'react';  function Component() {  useEffect(() => console.log('Mounted'), []);  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Mounting Method in Class Components | componentDidMount() {  fetch('http://localhost:3030/jsonstore/hobbies')  .then(res => res.json())  .then(res => this.setState({ hobbies: Object.values(res) }));  } |
| Updating Hook in Functional Components | Where the component is re-rendered because changes are made to its props or state. |
| Updating Method in Class Components | componentDidUpdate() {} |  |
| Unmounting Hook in Functional Components | When a component instance is unmounted (removed from the DOM):  import { useEffect } from 'react';  function Component() {  useEffect(() => return () => console.log('Unmounted'), []);  } |  |
| Unmounting Method in Class Components | componentWillUnmount() {} |  |

## HANDLING EVENTS

|  |  |  |
| --- | --- | --- |
| DOM Element Events vs. React Element Envents | DOM: <button *onclick*="clickHandler()">Click</button>  React: <button *onClick*={clickHandler}>Click</button> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Prevent Default Behavior | function Form() {  function handleSubmit(e) { e.preventDefault(); }  return (  <form *onSubmit*={handleSubmit}>  <button *type*="submit">Submit</button>  </form>  );  } |
| Handle Click Event | import { useState } from 'react';  export default function Counter() {  let [count, setCounter] = useState(0);  return (  <p>{count}</p>  <button *onClick*={() => setCounter(count + 1)}> // camelCase!  +  </button>  // same as:  <button *onClick*={setCounter.bind(null, count + 1)}>+</button>  );  } |
| Display Input Value Dynamically | import { useState } from 'react';  export default function ShowText() {  let [text, setText] = useState('');  const inputChangeHandler = (e) => { setText(e.target.value); };  return (  <p>{text}</p>  <input *onChange*={inputChangeHandler} />  );  } |

## CONDITIONAL RENDERING

|  |  |  |
| --- | --- | --- |
| Conditional Rendering with If Operator | function Greeting(props) {  if (props.user) return <h1>Hello, {formatName(props.user)}!</h1>;  return <h1>Hello, Stranger.</h1>;  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Conditional Rendering with Ternary Operator | function Greeting(props) {  return props.user  ? <h1>Hello, {formatName(props.user)}!</h1>  : <h1>Hello, Stranger.</h1>;  } |
| Conditional Rendering with Logical Operators | { name && <h2>Hello, {name}</h2> } // <h2> only if name is truthy  { name || <h2>No name</h2> } // <h2> only if name is falsy  { name ?? <h2>No name</h2> } // <h2> only if name is null/undefined |

## LISTS AND KEYS

|  |  |  |
| --- | --- | --- |
| Render Multiple Components | const numbers = [1, 2, 3, 4, 5];  function Numbers() {  return <ul>{numbers.map(n => <li>{n}</li>)}</ul>;  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| About Keys | Keys help React identify which items have changed, are added, or are removed. Keys should be given to the elements inside the array to give the elements a stable identity (the best keys are the IDs from the DB). They don't need to be globally unique, only among siblings. |
| Assign Keys to Elements | <li *key*={n.toString()}>{n}</li> |  |

## CSS STYLES

|  |  |  |
| --- | --- | --- |
| CSS Modules | CSS files in which all class names and animation names are scoped locally by default. All URLs and imports are relative. Importing a CSS module from a JS module exports an object with all mapping from local names to global names.  React supports CSS modules along with regular stylesheet using the [name].module.css file naming convention. CSS modules let you use the same class name in different files without worrying about naming clashes. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| CSS Module Example | ToDoItem.module.css: .todo-item { color: red; }  ToDoItem.js:  import styles from './ToDoItem.module.css';  function ToDoItem() {  return <li *className*={styles['todo-item']}>{todo.text}</li>;  } |
| Local CSS File Example | ToDoItem.css: .todo-item { color: red; } // the scope is global!!!  ToDoItem.js:  import './ToDoItem.css';  function ToDoItem() {  return <li *className*="todo-item">{todo.text}</li>;  } |  |
| Use React Bootstrap | npm install react-bootstrap bootstrap@5.1.3  App.js:  import 'bootstrap/dist/css/bootstrap.min.css'; // BEFORE App.css import!  src/components/Header.js:  import { Nav } from 'react-bootstrap'; // all custom components  const Header = () => {  return (  // chosen HTML from react-bootstrap.github.io/components  <Link *as*={Nav.Link} *to*="/login">Login</Link> // <Nav.Link> in HTML  );  };  export default Header; |  |

## IMAGE FILES

|  |  |  |
| --- | --- | --- |
| Standard Image Files | <img *src*="images/image.jpg" *alt*="Photo" /> // public/images/image.jpg | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| SVG Files | import logo from '../../logo.svg';  <img *src*={logo} *title*="title" *alt*="some text" />  // or with a React component:  import { ReactComponent as Logo } from '../../logo.svg';  <Logo /> |

## ROUTING

### PURPOSE AND INSTALLATION

|  |  |  |
| --- | --- | --- |
| About Routing | Allows navigation without reloading the page; pivotal element of writing single page applications (SPA). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Single Page Applications | A router loads the appropriate content when the location changes (ex. when the user manually enters an address). Conversely, a change in content is reflected in the address bar (ex. when the user clicks on a link). |
| Benefits of SPAs | They load all the scripts only once, maintain state across multiple pages, allow the use of browsing history and build user interfaces (UI) that react quickly. |  |
| Install React Router | npm install react-router-dom@5 // version 5  npm install react-router-dom // version 6 |  |

### IMPLEMENT ROUTING

|  |  |  |
| --- | --- | --- |
| BrowserRouter to Wrap the App | import { BrowserRouter } from 'react-router-dom';  ReactDOM.render(  <BrowserRouter> // injects match, location and history as props  <App />  </BrowserRouter>,  document.getElementById('root')  ); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Add Routes in V5 | import { Route } from 'react-router-dom';  function App() {  return (  <main> // without exact: both routes match and will be rendered  <Route *path*="/" *exact component*={Home} />  <Route *path*="/games" *component*={Games} />  </main>  );  } |
| Exclusive Rendering in V5 | import { Route, Switch } from 'react-router-dom';  function App() {  return (  <main> // the first will be rendered (without exact: loosely matched)  <Switch>  <Route *path*="/" *exact component*={Home} />  <Route *path*="/games" *component*={Games} />  </Switch>  </main>  );  } |  |
| Add Routes in V6 | import { Routes, Route } from 'react-router-dom';  function App() {  return (  <main> // the best match will be rendered  <Routes>  <Route *path*="/catalog/\*" *element*={<Home /> />  <Route *path*="/games" *element*={<Games />} />  </Route>  </main>  );  } |  |

### ADD NAVIGATION AND LINKS

|  |  |  |
| --- | --- | --- |
| Add Navigation in V5 | import { Link } from 'react-router-dom';  export default function Header() => (  <Link *to*="/" *component*={Home}>Home</Link>  <Link *to*="/login" *component*={Login}>Login</Link>  <Link *to*="/register" *component*={Register}>Register</Link>  ); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Style Active Links in V5 | import { NavLink } from 'react-router-dom';  export default function Header() => (  <NavLink *to*="/" *activeStyle*={{ color: 'red' }}>Home</NavLink>  <NavLink *to*="/login" *activeStyle*={{ color: 'red' }}>Login</NavLink>  <NavLink *to*="/all" *activeStyle*={{ color: 'red' }}>All</NavLink>  );  // or:  export default function Header() => (  <NavLink *to*="/" *activeClassName*={active}>Home</NavLink>  <NavLink *to*="/login" *activeClassName*={active}>Login</NavLink>  <NavLink *to*="/all" *activeClassName*={active}>All</NavLink>  ); // Header.css: .active { text-decoration: underline; } |
| Style Active Links in V6 | import { NavLink } from 'react-router-dom';  export default function Header() => (  <NavLink *to*="/"  *className*={({isActive} => "link" + (isActive ? " activated" : ""))}>  Home  </NavLink>  ); |  |

### ACCESS DATA FROM URL

|  |  |  |
| --- | --- | --- |
| In V5 | function GameDetails({ match }) {  const [game, setGame] = useState({});  useEffect(() => {  gameService.getById(match.params.gameId).then(g => setGame(g));  }, []);  ...  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| In V6 |  |

### REDIRECT

|  |  |  |
| --- | --- | --- |
| Redirect to a Direct Route Child | function Login({history}) {  const onFormSubmit = (e) => {  e.preventDefault();  ...  history.push('/games');  return <form *onSubmit*={onFormSubmit}>...</form>;  }  } | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Redirect | import { useHistory } from 'react-router-dom';  function Login() {  const history = useHistory();  const onFormSubmit = (e) => {  e.preventDefault();  ...  history.push('/games');  return <form *onSubmit*={onFormSubmit}>...</form>;  }  } |
| Redirect Directly in JSX | import { Redirect } from 'react-router-dom';  <Route *path*="/logout" *render*={(props) => {  console.log('Logged out!');  return <Redirect *to*="/" />;  }} /> |  |
| Redirect in V6 | import { useNavigate } from 'react-router-dom';  const Login = ({ onLogin }) => {  const navigate = useNavigate();  const onLoginHandler = (e) => ...; navigate('/');  }; |  |

### NESTED ROUTES

|  |  |  |
| --- | --- | --- |
| Nested Routes in V5 | const About = ({ match }) => (  <>  <h1>About Page</h1>  <Route *path*={match.url + '/contact'} *component*={Contact} />  </>  ); | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Nested Routes in V6 | const Dashboard = () => (  <>  <h1>Dashboard</h1>  <Link *to*="types">Types</Link> // '/dashboard/types'  </>  ); |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| <Outlet> | import { Outlet } from 'react-router-dom';  <section>  <Routes>...</Routes>  <Outlet />  </section>  App.js: <Routes><Route><Route /></Route></Routes> | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Match | The match object contains information about how the <Route path> matched the URL:  params: key/value pairs parsed from the URL  isExact: true if the entire URL was matched  path: the path pattern used to match  url: the matched portion of the URL |  |
| Location | The location object is never mutated and represents  where the app is now  where it was  where you want it to go. |  |
| History | The history object allows you to manage and handle the browser history inside your views or components:  length, action  location, push  replace, go  goBackgoForward, block |  |
|  |  |  |

## LAZY LOADING

|  |  |  |
| --- | --- | --- |
| Bundling | Most React apps have their files "bundeled" by tools like Webpack or Browserify. This is the process of following imported files and merging them into a single file (bundle). The bundle can be included on a webpage to load the entire app at once. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Code-Splitting | The opposite process to bundling. The best way to introduce code splitting into your app is through the dynamic import() syntax:  import { add } from './math'; // at compile time, before the app loads  import('./math').then(math.add(16, 24)); // at runtime |
| React Lazy | import { lazy, Suspense } from 'react';  const OtherComponent = lazy(() => import('./OtherComponent'));  function MyComponent() {  return (  <div>  <Suspense *fallback*={<h1>Loading...</h1>}>  <OtherComponent />  </Suspense>  </div>  );  } |  |

## FORMS

### UNCONTROLLED COMPONENTS

|  |  |  |
| --- | --- | --- |
| When to Use Uncontrolled Components | When you don't need to dynamically change input values, you should use uncontrolled components and let the DOM hold and control the input values. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Get Data from Forms using FormData | const submitHandler = (e) => {  e.preventDefault();  const formData = new FormData(e.currentTarget);  const { name, password } = Object.fromEntries(formData);  };  <form *method*="POST" *onSubmit*={submitHandler} /> |
| Set Initial Value for an Input Field | <input *type*="text" *name*="name" *defaultValue*="Ivan" /> |  |
| Implement an Initially Checked Checkbox | <input *type*="checkbox" *name*="isAdmin" *defaultChecked* /> |  |
| Implement Validation | const [isValid, setIsValid] = useState(true);  const onChange = (e) => {  const currentInput = e.target.value;  if (currentInput.length < 4) setIsValid(false);  else setIsValid(true);  };  <input *type*="text" *name*="name" *onChange*={onChange} />  {!isValid && <div *style*={{ color: 'red' }}>Invalid input!</div>} |  |
|  | implement debouncer for validation! |  |
| Dynamically Set Select Options | const [services, setServices] = useState([]);  useEffect(() => {  fetch('http://localhost:3030/jsonstore/services')  .then(res => res.json())  .then(res => setServices(Object.values(res)));  }, []);  <select>  {services.map(x =>  <option *key*={x.\_id} *value*={x.\_id}>{x.name}</option>)}  </select> |  |
| Use Reference to a Specific DOM Element | import { useRef } from 'react';  const passwordRef = useRef();  const onServiceChange = () => passwordRef.current.value = '';  <input *type*="password" *name*="password" *ref*={passwordRef} /> |  |

### CONTROLLED COMPONENTS

|  |  |  |
| --- | --- | --- |
| When to Use Controlled Components | When you need to dynamically change some input value (after an HTML request, another input value, etc.) and, therefore, need the value of the input to be controlled by React. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Controlled Component | const [username, setUsername] = useState('Pesho');  const onChange = (e) => { // input is recorded with event handlers  setUsername(e.target.value); // input value is kept in the state  console.log(username);  };  <input *type*="text" *value*={username} *onChange*={onChange} /> |
| Textarea in a Controlled Component | <textarea *value*={text} *onChange*={onChange} /> |  |
| Select in a Controlled Component | <select *multiple*={true} *value*={['A', 'B', 'C']} *onChange*={onChange}>  </select> |  |

## HOOKS

|  |  |  |
| --- | --- | --- |
| About React Hooks | JavaScript functions which can only be used inside functional components or other hooks. They let you use state and other React features (use hooks to share functionalities, NOT data) without writing a class. Their names always start with lowercase "use" followed by function name like "State": useState, useEffect, useContext. When a hook changes, the component will be automatically re-rendered. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| State Hook | The hook useState lets you add React state (and preserve it between re-renders!) to functional components. It returns an array of the current state value (the initial state argument is only used during the first render) and a function that lets you update it from anywhere (but doesn't merge the old and new state together!):  import { useState } from 'react';  const [name, setName] = useState(''); // can take a function as well!  const [age, setAge] = useState(0);  const [info, setInfo] = useState({ name: 'Ivo', age: 30, cars: [1, 2] });  setInfo(oldState => ({ ...oldState, age: oldState.age + 1 })); // merges new state to the old one  setInfo(oldState => ({ ...oldState, cars: [...oldState.cars, 3] })); |
| Effect Hook | The hook useEffect adds the ability to use all functionalities related to the component lifecycle (mounting, updating, unmounting) from a functional component. It is most used for data fetching, subscriptions or to manually change the DOM (operations like this are called side effects). It accepts a function that contains possibly effectful code that will run after the component is rendered. The function passed to useEffect may return a clean-up function to be executed before the component us unmounted.  import { useEffect } from 'react';  const [name, setName] = useState('');  useEffect(() => { console.log('mounted') }, []);  useEffect(() => { console.log('name was changed') }, [name]);  useEffect(() => { return () => console.log('unmounted') }, []); |  |
| Custom Hooks | A custom hook is a simple JS function whose name starts with "use" and that can call other hooks. It allows us to reuse some stateful logic between components. We can decide what it takes as arguments and what it should return. |  |
| Rules of Hooks | * Only call hooks at the top level * Only call hooks from functional components or custom hooks |  |

## CONTEXT

|  |  |  |
| --- | --- | --- |
| About Context | Context API provides a way to pass data through the component tree without passing the props manually. It is designed to share data like current authenticated user, theme (light/dark), preferred language, etc., and is primarily used when some data needs to be accessible by many components at different nesting levels. Apply it sparingly because it makes component reuse more difficult. When you use Context, only the top-most page component knows about your data. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a Context Object | src/contexts/AuthContext.js:  import { createContex } from 'react';  export const AuthContext = createContext(); |
| Create a Context Provider in the Root Component | const App = () => {  return (  <AuthContext.Provider *value*={{ count, user: info, addHobby }}>  <h1>Title</h1>  <div>Some content</div>  </AuthContext.Provider>  );  } |  |
| Use the Context Object | import { useContext } from 'react';  const { count } = useContext(AuthContext); |  |
| Set Context outside of the App Component | src/contexts/AuthContext.js:  import { createContex, useState } from 'react';  export const AuthContext = createContext();  const initialState = { email: '', accessToken: '' };  export const AuthProvider = ({ children }) => {  const [state, setState] = useState(initialState);  return (  <AuthContext.Provider>{children}</ AuthContext.Provider>  );  };  App.js:  <AuthProvider>  // the whole app  </ AuthProvider> |  |

## ADVANCED TECHNIQUES

### HIGHER-ORDER COMPONENTS

|  |  |  |
| --- | --- | --- |
| About Higher-Order Components | It is an advanced technique in React for reusing component logic. HOCs are not part of the React API, they are rather functions that take a component and return a new component (decorate a component). As components transform props into UI, a HOC transforms a component into another component. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create a HOC | src/hoc/isAuth.js:  export const isAuth = (Component) => {  return (props) => {  const { isAuthenticated } = useAuth();  return isAuthenticated  ? <Component { ...props }/>  : <Navigate *to*="/login" />;  };  }; |
| Use a HOC | src/components/MyList.js:  export default isAuth(MyList); |  |

### USEREDUCER

|  |  |  |
| --- | --- | --- |
| About UseReducer | It is an alternative to useState, which acceps a reducer of type (state, action) => new state and returns the current state paired with a dispatch method. Preferable when you have complex state logic. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Use UseReducer | const reducer = (state, action) => {  switch (action.type) {  case 'LOGIN': return { ...state, email: action.payload };  case 'LOGOUT': return initialState;  default: return state;  }  return { email: action };  };  const [user, dispatch] = useReducer(reducer, initialState);  const login = (email) => {  dispatch({  type: 'LOGIN',  payload: email  })  };  const logout = () => { dispatch({ type: 'LOGOUT' }) }; |

### ERROR BOUNDARIES

|  |  |  |
| --- | --- | --- |
| About Error Boundaries | They are React class components which catch, log and display JS errors anywhere in their child component tree. They catch errors during rendering and do NOT catch errors for event handlers, asynchronous code or server-side rendering. They work like a JS catch {} block for component. Declare an error boundary component once and use it throughout your application. You may wrap top-level route components to display some error message or wrap individual widgets inan error boundary to protect them from crashing the rest of the app. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Make a Component an Error Boundary | constructor(props) { super(props); this.state = { error: null }; }  // renders a fallback UI after an error has been thrown:  static getDerivedStateFromError(error) {  return { error };  }  componentDidCatch(error) { console.log(error) } |

### JEST

|  |  |  |
| --- | --- | --- |
| About JEST | It is a JavaScript unit testing framework used by Facebook to test services and React applications. It acts as a test runner, assertion library and mocking library, and provides Snapshot testing (create a rendered "snapshot" and compare it to a previous). | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| JEST Documentation | https://jestjs.io |
| Using JEST | create a JEST test |  |

### REACT QUERY

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

### REACT REMEMBERS

|  |  |  |
| --- | --- | --- |
|  | useMemo(): remembers the result from a function and gives it back every time the component rerenders | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
|  | useCallback(): remembers the reference to a function (the function itself) and gives it back every time the component rerenders |

## CREATING A REACT APP

### SETUP A REACT APP

|  |  |  |
| --- | --- | --- |
| Run the React App Creator | npx create-react-app my-app // an app "my-app" in current directory | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| If Prompted to Uninstall Globally Installed CRA | npm uninstall -g create-react-app  npx clear-npx-cache npx create-react-app my-app |
| Start the App | cd my-app  npm start // npm run build, serve -s build for production |  |
| Clean Up Code | 1. public/: remove logo files 2. public/index.html: remove comments, change title 3. src/index.js: remove comments, rename 'index.css' to './index.**scss**'; 4. src/App.js: remove "import logo from './logo.svg';" and content of <div className="App"> 5. src/index.css: remove everything, rename to index.scss |  |
| Use Pictures | Insert them in public/, then access them with "/image.jpg". |  |

### CREATE MAIN COMPONENTS

|  |  |  |
| --- | --- | --- |
| Folder | Create a folder src/components. | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Styling | src/index.scss: main styling and variables |
| Home Component | src/components/Home/Home.js:  import './Home.scss';  function Home() {  return (  <div className="Home">  ...  </div>  );  }    export default Home;  src/components/Home/Home.scss:  @import './../../index.scss'; |
| Header/Footer | src/components/common/Header/Header.js  src/components/common/Footer/Footer.js |  |
| Implement Header/Footer and Home in App Component | src/App.js:  function App() {  return (  <div className="App">  <Header></Header>  <Home />  </div>  );  }  export default App; |  |

### ADD ROUTING & NAVIGATION

|  |  |  |
| --- | --- | --- |
| Installation | npm i react-router-dom | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| BrowserRouter to Wrap the App | import { BrowserRouter } from 'react-router-dom';  ReactDOM.render(  <BrowserRouter> // injects match, location and history as props  <App />  </BrowserRouter>,  document.getElementById('root')  ); |
| Add Routes in V6 | import { Routes, Route } from 'react-router-dom';  function App() {  return (  <main> // the best match will be rendered  <Routes>  <Route *path*="/" *element*={<Home />} />  <Route *path*="/games" *element*={<Games />} />  </Route>  </main>  );  } |
| Style Active Links in V6 | import { NavLink } from 'react-router-dom';  export default function Header() => (  <NavLink *to*="/"  *className*={({isActive} => "link" + (isActive ? " activated" : ""))}>  Home  </NavLink>  ); |  |
| Nested Routing | <Routes>  <Route path="/" element={<Dashboard />}>  <Route  path="messages"  element={<DashboardMessages />}  />  <Route path="tasks" element={<DashboardTasks />} />  </Route>  <Route path="about" element={<AboutPage />} />  </Routes> |  |
| Redirect in V6 | import { useNavigate } from 'react-router-dom';  const Login = ({ onLogin }) => {  const navigate = useNavigate();  const onLoginHandler = (e) => ...; navigate('/');  }; |  |

# DEPLOYING AN APPLICATION

## FIREBASE

|  |  |  |
| --- | --- | --- |
| Create a Project | firebase.com | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Install Firebase | npm install firebase |
| Initialize Firebase | firebase login  firebase init > hosting (select the first one with Space), use existing project, public directory: build, SPA: yes, GitHub: no |
| Exclude Files from GitHub Repo | .gitignore: /.firebase |  |
| Deploy Application | firebase deploy |  |

## HEROKU

|  |  |  |
| --- | --- | --- |
| Initialize Server | npm init -y | [Python](file:///C:\Users\User\Desktop\Darina\Programming\JavaScript\Python.docx#BasicSourseCodePython) C++ C# Java |
| Create App | heroku.com: SoftUni Custom Server |
| Install Heroku CLI | npm install -g heroku |
| Initialize Heroku | heroku login  git init  git add .  git commit -am "Initial Commit"  heroku git: remote -a softuni-custom-server |
| Set Environment Variable | const port = process.env.PORT || 3030;  engine... |  |
| Deploy App | git push heroku master |  |
| Read More | Deploying Node.js app on Heroku |  |

SoftUni

Fundamentals: Workshop (already watched the video, just update the project itself as per Back-End pattern (MongoDB, validation, etc.))

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: Asynchronous Programming - Ex. (watch & do) have watched until 00:14:56

JS Applications? Webpack

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

AngularJS: проект на колегата Тодор: https://github.com/TodorBelchev/SoftUni

Academind: JavaScript Algorithms, Data Structures

https://www.atlassian.com/agile/scrum

W3Schools

Math (sin, cos, tan...)

Bitwise Operators

AJAX

JSON PHP, JSONP

JS Examples: JS HTML Objects, JS HTML Events

Books: Code Complete, The Pragmatic Programmer

v.kostadinov@softuni.bg