| **Topic** | **Basic** | | **Advanced** | **Expert** |
| --- | --- | --- | --- | --- |
| **JavaScript Core** | | | | |
| **Syntax** | 1. Know the **comments syntax:**    1. Single    2. Multiple    3. JSDoc 2. Know the **data literals** syntax:    1. Null    2. Boolean    3. Numeric       1. Decimal       2. Binary       3. Octal       4. Hexadecimal       5. BigInt       6. Numeric separators    4. Object    5. Array    6. String       1. Hexadecimal escape sequences       2. Unicode escape sequences    7. Template    8. Regular expression 3. Know **syntax of loops:**    1. for    2. for of    3. for in    4. for await .. of    5. while / do while    6. break / continue 4. Be aware of **block** statement; 5. Know the **conditions syntax**;    1. If .. else    2. Switch / break / default 6. Ability to explain the purpose of each loop type, describe the situations where each of them might be the right one | | 1. Understand what is **enumerability**; 2. Understand what is **iterable protocol**; 3. Be aware of **labels;** | \_empty\_ |
| **Arrays** | 1. Understand **what is an array**, its purpose; 2. Understand **what data can be stored in array;** 3. Understand **when to use arrays instead of objects;** 4. Know several **options of array creation**, use the **literal creation** on real projects; 5. Understand how to make a **multidimensional array;** 6. Know the **length property** and main use cases of it; 7. Know how to use and purpose of **basic array methods**:    1. push / pop    2. unshift / shift    3. indexOf / lastIndexOf    4. join    5. slice / splice    6. concat    7. sort    8. reverse    9. Array.isArray 8. Know how to use and purpose of **functional array methods**. Understand **immutability** concept:    1. forEach    2. filter    3. find    4. map    5. some / every    6. reduce / reduceRight 9. Know how to **retrieve an item** from array; 10. Know how to **insert an item** to the array; 11. Know how to **iterate through the array**:     1. loops     2. forEach 12. Know **array-like objects**. Know how to **transform them** into normal arrays:     1. string     2. argument     3. list of dom elements     4. Array.from method 13. Know what is a **sparse array** and how the “length”; property behaves; 14. Understand why an **Array constructor** usage **is** a **bad** practice; 15. Know how to apply the **chaining** of **functional array methods**; 16. Know the **ES2015+** array **methods**:     1. copyWithin     2. fill     3. findIndex     4. flat     5. flatMap     6. includes     7. entries     8. keys     9. values | | 1. Know what is an **iterator/iterable protocol**. Sync/Async iterables. Its purpose and use cases. 2. Know the **performance aspects of arrays**:    1. Costs to retrieve an item    2. Costs to insert/delete an item    3. Costs to concat arrays    4. indexOf vs. simple loop 3. Ability to **compare Hash** vs. **Array performance**; 4. Understand the purpose of **typed arrays** and how to work with them:    1. Int8Array    2. Uint8Array    3. BigInt64Array    4. Etc.. 5. Understand the purpose of **keyed collections** and how to work with them:    1. Map    2. Set    3. WeakMap    4. WeakSet | 1. Know the **aspects of memory allocations** for array by the **interpreter**; 2. Know the **arrays optimisation patterns**; 3. Understand the concept of **buffers** & **views** in typed arrays architecture; |
| **Functions** | 1. Understand **what is a function**, its purpose; 2. Know the various **types of functions declarations**:    1. Function declaration    2. Function expression 3. Understand what is a **context of a function**, its purpose and how to identify what is a **context value** in a certain situation; 4. Know the different ways of **setting a context** to the function:    1. apply / call    2. bind 5. Know the **arguments key-word**, when it might be useful; 6. Know how to deal with **optional arguments;** 7. Understand a **callback pattern** and **anonymous/lambda** function; 8. Know the **arrow function** type, it’s main differences with the normal function; 9. Know the rules/indicators signing that the code should be move to a function; 10. Understand what is a **prototype** of a function and when it might be useful? 11. Know how the function behaves if it’s used as a **constructor**? 12. Know how the function behaves if it’s used as a **method**; 13. Know what does the **instanceof** operator do; 14. Understand **IIFE** and when it might be useful; 15. Understand the “**Callback Hell**” problem; 16. Ability to use **recursion**; 17. Know the **hoisting** of function declarations behaviour; 18. Understand what is a **call stack** and when this knowledge might be useful in development; 19. Know the **rest** parameters / **destructuring** rest parameters and when it might be useful; | | 1. Understand how the **prototypal inheritance** can be organized; 2. Know the **higher order functions** and when it might be useful; 3. Know the **currying** and when it might be useful; 4. Understand the **memoization** concept and its application; 5. Understand the **partial invocation** of a function concept and its application; 6. Know how to organize JS code in **modules**. Know what is **namespacing** and how to apply these techniques in the application; 7. Know what is a **pure function**, know it’s advantages; 8. Know how to write **testable code**; 9. Understand the **mix-ins pattern** and its application; | 1. Understand the difference between; **functional** & **prototypal inheritance**; 2. Understand the application of a **“Function” constructor;** 3. Understand **functional programming** paradigm; 4. Ability to characterize the **OOP vs. FP** paradigms; 5. Understand **functional inheritance** and its application; |
| **Scope & closure** | 1. Understand what is the **lexical environment & scope**, it’s main characteristics; 2. Know the features of **nested LE’s**. Understand the problems that could be associated with them. Know the techniques of minimizing them. 3. Understand the difference between the **global & local scopes**. Know how to access the global scope from local ones; 4. Understand the **hoisting** feature. Know the problems that can occur without knowing of hoisting; 5. Know the ways of **LE creation**:    1. Functions as scope    2. Blocks as scope 6. Understand how the various types of **variables declarations** behave along with the **scopes**; | | 1. Understand the mechanism of variable identification through the **chain of LE’s**; 2. Understand **LE’s inheritance** mechanism; 3. Why it’s generally a good idea to prevent **global scope** from **pollution** and don’t touch it. | \_empty\_ |
| [**Async behavior**](https://docs.google.com/document/d/1t79HO4n_sEm9aWC3ZN5skhcp_SO3z8WNuZrBERf3QuA/edit#bookmark=id.evfofci973yz) | 1. Know how **setTimeout** / **setInterval** work. Understand their application; 2. Know the purpose of using **setTimeout(0)**, which effects can be achieved by that; 3. Understand the **Promise** concept, its advantages comparing to callback pattern:    1. Know the promise chaining feature and why it can be handy.    2. Know the Promise.all application.    3. Be aware of Promise.race functionality. 4. Understand the **async/await** concept, its advantages comparing to Promise pattern; 5. Understand how the **exception handling** behaves with async model; | | 1. Understand what is a **promisification pattern** and its application; 2. Understand how **event loop** works and **concurrency model**. Know the ways of influencing on the JS instructions handling through the event loop; | 1. Understand the **performance issues** that could be provoked by the negligence in async model usage; 2. Understand how **generators** work. Their applications; 3. Understand how the error handling is treated in generators; |
| [**Data types**](https://docs.google.com/document/d/1t79HO4n_sEm9aWC3ZN5skhcp_SO3z8WNuZrBERf3QuA/edit#bookmark=id.euvm56obevy8) | 1. Understand what is **dynamic typing**, the difference with static typing & strict static typing; 2. Know what are the **primitive types** and **reference types**, key differences between them; 3. Ability to list all the **data types** and their **purpose**:    1. Number    2. String    3. Boolean    4. Object    5. Undefined    6. Null    7. BigInt    8. Symbol 4. Understand how to deal with **Date object;** 5. Know the **Number methods:**    1. parseInt    2. parseFloat    3. isNaN    4. toFixed 6. Know the **String methods:**    1. charAt    2. substring    3. slice    4. indexOf    5. split    6. replace    7. toUpperCase    8. search 7. Know all the variants of **variables declaration**. Ability to explain the features of each and pitfalls:    1. var    2. let    3. const 8. Understand how and when **implicit / automatic types coercion** works:    1. String <-> Number    2. Boolean <-> Number    3. Object <-> String    4. valueOf function 9. Understand how to do an **explicit coercion**; 10. Understand how the **comparison** of values of **different types** works; 11. Know the “**falsy values**” concept, its application and pitfalls; 12. Understand the **mutable** / **immutable** values possibility. When and why it can be used; 13. Know the basic features of **Math** standard object; | | 1. Know the features of **JSON** object; 2. Know what is a **dead-zone concept** for “let”; 3. Know the **String object** in depth:    1. Escape characters    2. Encodings, UTF-8, UTF-16 4. Know the **Number object** in depth:    1. Special values of Number       1. Infinity/-Infinity       2. POSITIVE\_INFINITY/NEGATIVE\_INFINITY       3. MAX\_VALUE       4. MIN\_VALUE       5. MAX\_SAFE\_INTEGER/MIN\_SAFE\_INTEGER       6. NaN    2. Numbers rounding issues    3. Format types of numbers (2,8,16)    4. Scientific notation 5. Understand how the **objects-wrappers** mechanism works; | 1. Understand the purpose of **Reflect** object; 2. Understand the purpose of **Proxy** object; 3. Understand the purpose of **Intl** object; |
| **Expressions & operators** | 1. Know what is an **operand** and how many operands the operators can operate with; 2. Know the **arithmetic** operators    1. Addition (+)    2. Subtraction (-)    3. Division (/)    4. Multiplication (\*)    5. Remainder (%)    6. Exponentiation (\*\*)    7. Increment (++)    8. Decrement (--)    9. Unary negation (-)    10. Unary plus (+) 3. Know the **shorthand assignments;** 4. Understand what is **eval**, how it works and what are the pitfalls and risks it can cause;    1. Know the difference between **direct** and **indirect** eval calls;    2. Know the **safer** and **faster** alternative: **new Function**; 5. Know the **comparison** operators:    1. Equality (==)    2. Inequality (!=)    3. Identity / strict equality (===)    4. Non-identity / strict equality (!==) 6. Know the **relational** operators:    1. Greater than(>)    2. Greater than or equal (>=)    3. Less than (<)    4. Less than or equal (<=) 7. Know the **logical** operators:    1. Logical AND (&&)    2. Logical OR (||)    3. Logical NOT (!)    4. Short-circuit evaluation    5. Operator precedence    6. Double NOT (!!) 8. Know the conditional (**ternary**) operator; 9. Know **conditional chains** by the ternary operator technique; 10. Know the **grouping** operator; 11. Know the **destructuring** assignment     1. Array destructuring possibilities     2. Object destructuring 12. Know the **operators precedence;** 13. Understand what is **operators associativity**; 14. Know the “**in**” operator; 15. Know the **void** operator; | | 1. Know the **bitwise** operators:    1. AND (&)    2. OR (|)    3. XOR (^)    4. NOT (~)    5. Left Shift (<<)    6. Sign-propagating right shift (>>)    7. Zero-fill right shift (>>>)    8. Flags & bitwise masks | 1. Understand the **Nullish coalescing** operator / optional chaining operator; |
| **Objects** | 1. Know the ways of **object initialization:**    1. new.    2. literal.    3. Object.create.    4. ES2015 notation.    5. Be aware of **computed property names** feature;.    6. Via function-constructor. 2. Know the **accessing properties**:    1. Dot notation    2. Bracket notation 3. Know the **property definitions** variants:    1. defineProperty       1. Know the API of **object property descriptor**;    2. simple assignment 4. Know the **method definitions** variants; 5. Know how **delete operator** works, it’s influence on memory management; 6. Know the **spread properties** technique; 7. Know the **build-in Object methods**:    1. hasOwnProperty    2. isPrototypeOf    3. assign    4. create    5. defineProperty/Properties    6. entries    7. freeze    8. fromEntries 8. Know the structure of **class** expression / declaration and the application of each member:    1. getters/setters    2. super    3. this    4. instanceof    5. class field    6. constructor    7. extends    8. static    9. private members    10. class extension prohibition | | 1. Understand what is a **prototype mutation**, **monkey patching**; 2. Know the **OOP emulation patterns** in ES5; | 1. Be aware of “**new.target**” pseudo-property; |
| **Error handling** | 1. Understand the **try / catch / finally** statement and the mechanism behind it; 2. Know **throw** keyword and its purpose; 3. Know the **error object properties**, **methods** and how it can be useful; 4. Be aware of the common **standard errors**; 5. Know the possibility to define **custom errors**, understand the reason behind it; 6. Understand how **errors** are **propagated** across the call stack; | | 1. Know the common **error handling patterns** & **practices**; 2. Understand the **performance issues** caused by **error handling**; | \_empty\_ |
| [**Misc**](https://docs.google.com/document/d/1t79HO4n_sEm9aWC3ZN5skhcp_SO3z8WNuZrBERf3QuA/edit#bookmark=id.abea43hw7arp) | 1. Know the **basics of regular expressions** and which types of tasks can be covered by them; 2. Know the **JS modules** concept, various techniques to organize modularization; 3. Know what does **strict mode** do, which limitations and improvements it brings to developers; | | 1. Understand how the JS script is executed & **control flow** is structured; 2. Understand how **garbage collection** works and how developer can influence on it; | 1. Understand the **memory management** mechanisms when running the JS script; 2. Understand the **mark-and-sweep algorithm** (in memory management); 3. Know the **advanced techniques of regular expressions**; |