Advanced JavaScript

Concetti avanzati e “cose losche”

Di cosa parleremo

- I float in javascript sono un pò loschi..
- Il punto e virgola è necessario in javascript?
- regular vs arrow function
- Le closure, le usiamo tutti i giorni…
- La ricorsione
- Curring function
- Iterators/Generators
- Proxy
- Gestione delle eccezioni

Di cosa parleremo

- ECMAScript 2024 features
- Concurrency vs Parallelism
- ES FEATURES

ES 6

Default parameters let & const Template strings Object declarations
Classes set & get Arrow functions for of **Promises** Deconstruction
Maps & Sets Spread Operator Symbols Modules Iterators
Proxies Generators

ES 2016

`Array.includes` Exponentation operator

ES 2017

`Object.entries()` `Object.values()` **Async functions**

ES 2018

Asynchronous iteration Rest properties Regular expression enhancements
Spreading into object literals **`Promise.finally()`**

ES 2019

Optional catch binding **`Array.flat()`** **`Array.flatMap()`**
`Object.fromEntries()` `String.trimStart()` `String.trimEnd()`

ES 2020

Optional chaining Nullish coalescing operator Bigints
Dynamic imports `globalThis` **`Promise.allSettled()`**
`String.matchAll()`

ES 2021

Local assignment operators Numeric separators **`Promise.any()`**
`String.replaceAll()`

ES 2022

Private slots **Top-level await** **`Array./String.at()`** `error.cause`
Class field declarations Regular match indices Static initialization blocks

ES 2023

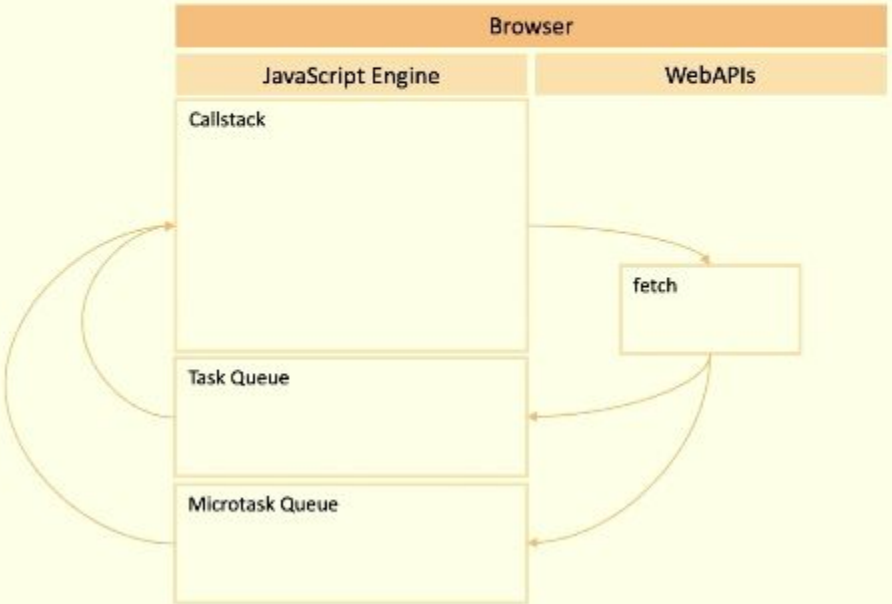
`Array.findLast()` **`Array.findLastIndex()`** **`Array.toReversed()`**
`Array.toSorted()` **`Array.toSpliced()`** **`Array.with()`** WeakMaps

ES 2024

`Promise.withResolvers()` **`Object./Map.groupBy()`**
`Atomics.waitAsync()` `String.isWellFormed()`

The need for Promises

The JavaScript Event Loop



JavaScript Interview Cheatsheet
(Part 2)

12. Modules
Modules allow you to organize and encapsulate code into reusable pieces, exporting and importing functionalities across different files.

They help maintain clean and modular code.
JavaScript modules are essential for building scalable applications, especially in modern frameworks and libraries.

13. Debouncing & Throttling
Debouncing and throttling are techniques used to limit the rate at which a function is executed. Debouncing delays execution, while throttling ensures the function runs at fixed intervals.

www.developerupdates.com

These techniques improve performance by reducing the frequency of function calls in response to events like scrolling or resizing.

14. Destructuring
Destructuring is a syntax that allows you to unpack values from arrays or properties from objects into distinct variables. This makes your code cleaner and easier to read.
It's a convenient way to work with complex data structures, reducing the need for repetitive code.



Chetan Mahajan
www.developerupdates.com

Are you looking for Front-end Developer Job?
If Yes, Check the link in bio to get Interview Kit.

JavaScript Interview Cheatsheet
(Part 2)

15. Object-Oriented Programming (OOP)
OOP in JavaScript involves using objects to represent real-world entities, encapsulating data and behavior within them. It promotes code reusability and modularity.

Understanding OOP concepts like inheritance, polymorphism, and encapsulation is vital for building scalable, maintainable applications.

16. Error Handling
Error handling involves using try...catch blocks to manage exceptions in your code. Proper error handling ensures your application can handle unexpected situations gracefully.

www.developerupdates.com

Effective error handling is crucial for creating robust, user-friendly applications that fail gracefully.

17. Type Coercion
Type coercion refers to the automatic or implicit conversion of values from one data type to another in JavaScript. It happens when operations involve mixed types.

Being aware of type coercion helps avoid unexpected results and bugs in your code, especially with comparisons.



Chetan Mahajan
www.developerupdates.com

Are you looking for Front-end Developer Job?
If Yes, Check the link in bio to get Interview Kit.

JS E6 Cheatsheet

- **Async/Await**
  - **Async functions**: async function() {}.
  - **Await keyword**: await promise.
- **Modules**
  - Importing: **import { module } from 'source'.**
  - Exporting: **export const module.**
- **Map and Set**
  - **Map**: Key-value pairs, **new Map()**.
  - **Set**: Unique values, **new Set()**.

www.developerupdates.com

- **Symbols**
  - Unique and immutable identifiers: **const sym = Symbol('description')**.
- **Iterators and Generators**
  - Iterators: Custom iteration logic.
  - Generators: **function* generator() { yield value; }**.
- **Enhanced Object Literals**
  - Shorthand properties: **const obj = { a, b }**.
  - Method definitions: **method() {}**.
- **For...of Loop**
  - Iterate over iterable objects: **for (const item of iterable) {}**.
- **Block-Scoped Variables**
  - **let** and **const** are block-scoped, unlike **var**.



Chetan Mahajan
www.developerupdates.com

JS E6 Cheatsheet

- **Arrow Functions and 'this'**
  - Arrow functions do not have their own this.
- **Modules (Import/Export)**
  - Import: **import { myModule } from 'myModule.js'.**
  - Export: **export const myModule = ...**
- **Enhanced Function Parameters**
  - Default, rest, and spread parameters in functions.
- **Object.assign()**
  - Merge objects: **Object.assign(target, source1, source2).**
- **Object.entries() and Object.values()**
  - Entries: **Object.entries(obj)**.
  - Values: **Object.values(obj)**.

www.developerupdates.com

- **Array.from() and Array.of()**
  - Array from iterable: **Array.from(iterable)**.
  - Array from arguments: **Array.of(arg1, arg2)**.
- **String Methods**
  - Includes: **str.includes('substring')**.
  - Starts with: **str.startsWith('prefix')**.
  - Ends with: **str.endsWith('suffix')**.
- **Number Methods**
  - Is finite: **Number.isFinite(value)**.
  - Is NaN: **Number.isNaN(value)**.



Chetan Mahajan
www.developerupdates.com

JS E6 Cheatsheet

- **Let and Const**
  - **let**: Block-scoped variable declaration.
  - **const**: Block-scoped, immutable variable declaration.
- **Arrow Functions**
  - Shorter syntax: **(params) => expression.**
  - No **this** binding.
- **Template Literals**
  - String interpolation: **`Hello, ${name}!`**.
  - Multi-line strings.
- **Destructuring**
  - Object destructuring: **const {a, b} = obj.**
  - Array destructuring: **const [x, y] = arr.**
- **Spread and Rest Operators**
  - Spread: **'...'** to expand arrays/objects.
  - Rest: **'...'** to collect function arguments.
- **Default Parameters**
  - Set default values for function parameters: **function(x = 10) {}**.
- **Classes**
  - Class syntax: **class MyClass {}**.
  - Constructors and methods.
- **Promises**
  - Asynchronous operations: **new Promise((resolve, reject) => {})**.
  - Methods: **.then(), .catch(), .finally()**.

www.developerupdates.com



Chetan Mahajan
www.developerupdates.com

JS E6 Cheatsheet

- **Promises and Async/Await**
  - Promises: **new Promise((resolve, reject) => {})**.
  - Async/Await: **async function() { await promise; }**.
- **Set**
  - Unique values collection: **new Set([iterable])**.
- **WeakMap and WeakSet**
  - WeakMap: **new WeakMap()**.
  - WeakSet: **new WeakSet()**.
- **Modules (Dynamic Import)**
  - Dynamic import: **import('module').then(...)**.

Are you looking for a Front-end Developer Job?

If yes, Check the link in bio to get Interview Kit or Visit www.developerupdates.com



Chetan Mahajan
www.developerupdates.com