Mohammed Ramadan

# JAVASCRIPT INTERVIEW .JS

Mohammed Ramadan

# IS JAVASCRIPT A CASE-SENSITIVE LANGUAGE?

Yes, JavaScript is **a case-sensitive** language.

The language keywords, variables, function names, and any other identifiers **must** always be typed with a consistent capitalization of letters.

So the identifiers Time and TIME will **convey** different meanings in JavaScript.

Mohammed Ramadan

# DEFINE ANONYMOUS FUNCTION

It is a function that has **no name**.

These functions are **declared dynamically** at **runtime** **using the function operator** instead of the function declaration.

The function operator is **more flexible** than a function declaration.

It can be easily used in the place of an expression

Mohammed Ramadan

For example:

```
var display=function()
{
 alert("Anonymous Function is
invoked");
}
display();
```

Mohammed Ramadan

# WHAT IS THE MAIN DIFFERENCE BETWEEN VAR, CONST, AND LET?

- Variables declared with let and const are **block-scoped**; Variables declared with var are **globally-scoped** or function-scoped.
- var variables can be updated and re-declared **within its scope**; let variables **can be updated but not re-declared**; const variables can neither be **updated nor re-declared**.

- var can be **hoisted** to the top of their scope. Where var variables are initialised as undefined, let and const variables are **not initialised**(Temporary Dead Zone, TDZ).

- While var and let can be declared **without being initialised**, const must be **initialised during declaration.**

Mohammed Ramadan

# WHAT ARE PROMISES AND ASYNC-AWAIT?

Promises is a way to **enable asynchronous** programming in JavaScript. In more general terms,

Promise means that a program **calls** a function **with the expectation** that it will return the result that the calling program can use in further computation.

Mohammed Ramadan

Async-await also **helps in asynchronous programming**.

It is **syntactic sugar** for promises.

Async-await has a simple syntax and is **easy to maintain** lots of asynchronous calls **in** a single function.

Also, async-wait **prevents callback hell**.

Mohammed Ramadan

# WHAT IS A CALLBACK AND HOW DOES IT WORK BEHIND THE SCENE?

In short, Callback is a function that is **passed to another function**.

- The callback is possible only because JavaScript supports the first-class function.
- A function that has passed as an argument to another function or it can be executed in that other function is called a **callback**.

- In Node.js, it consists of 4 default threads that are responsible for maintaining the main stack and other queues, most asynchronous functions call other asynchronous functions and then call the callback. You can think of it as a chain of functions and callbacks. All asynchronous and IO operations are directly **not handled by the main thread**, All callbacks and asynchronous calls have been handled by the other queue which presents in the JS engine.

Mohammed Ramadan

# WHAT IS "CALLBACK HELL 😈"?

Callback hell is a situation where we **nest callbacks in callbacks**.

```
getBeef("John", (rawBeef) => {
    sliceBeef(rawBeef, (slicedBeef) => {
        cookBeef(slicedBeef, (cookedBeef) => {
            serveBeef(cookedBeef, (servedBeef) => {
                console.log(servedBeef)
            });
        });
    });
});
```

The code above is very ugly and definitely not something that we want to read or debug.

There is actually a solution to this: to use **Promises** in JavaScript.

Mohammed Ramadan

# HOW MANY WAYS DO WE HAVE FOR DECLARING A FUNCTION AND HOW ARE THEY DIFFERENT FROM EACH OTHER?

- A function declaration has made of a **function keyword**, followed by an obligatory function name, a list of parameters in a pair of parenthesis.

- **Shorthand method** definition can be possible to use in a method declaration on object literals and ES2015 classes.

Mohammed Ramadan

- **An Arrow function** is defined using a pair of parenthesis that contains the list of parameters. Followed by a fat arrow => and a pair of curly braces that delimits the body statements.
- In a **function expression**, you assign a function to a variable.

- A function can be dynamically created using the **Function constructor**, but it suffers from security and performance issues and is not advisable to use.

Mohammed Ramadan

# WHAT IS THE USE OF THE NAN() FUNCTION ?

In JavaScript NaN is short for "**Not-a-Number**".

The isNaN() method returns **true** if a value is **NaN**.

The isNaN() method **converts** the value to a number **before testing it**.

The NaN() function has only one purpose — to check whether a value is an **illegal number**.

Mohammed Ramadan

# DIFFERENCE BETWEEN ISNAN() AND NUMBER.ISNAN() ?

isNaN() method returns **true** if a **value** is **Not-a-Number**.

Number.isNaN() returns **true** if a **number** is **Not-a-Number**.

**SO**

isNaN() **converts** the value to a number **before testing** it.

Mohammed Ramadan

# WHAT DOES THE ‘THIS’ KEYWORD DO?

This keyword is a **self-reference** for JavaScript objects.

**Which object?**

it depends on how this is being **invoked:**

1. In an object method, this refers to **the object**.
2. Alone, this refers to the **global object**.
3. In a function, this refers to the **global object**.

4. In a function, in strict mode, this is **undefined**.

5. In an event, this refers to the **element that received the event**.

6. Methods like call(), apply(), and bind() can refer this to **any object**.

## Note

this is **not a variable**. It is a **keyword**. You **cannot change the value** of this.

working on next part

Thanks for reading

# I hope you enjoyed it and found it useful

## Mohammed Ramadan

## FOLLOW ON LINKEDIN

https://www.linkedin.com/in/mohammed-ramadan-1374771b7/