deeecode

JS INTERVIEW QUESTIONS PART 1

JS

swipe left

deeecode

JS

JS INTERVIEW QUESTIONS - PART 1

In JavaScript interviews, there are so many questions that can be asked.

I may not be able to cover every single question but in this new series I'm starting, I will try to answer a couple of them of them and hopefully they can help you!

The questions I'll be sharing in these series are based on research I made from different platforms, and also questions I formed myself which may be asked.

If you enjoy this post, please like and share with others. I will appreciate it.

So let's start with Part 1.

swipe left

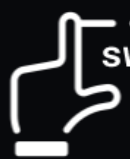
JS

JS INTERVIEW QUESTIONS - PART 1

1. What is the difference between let, const and var?

let, **const** and **var** are three keywords you can use to create variables in JavaScript. The difference is that:

- **var** can be reassigned, and it also hoists variables to the top of its local or global scope it is defined in
- **let** can be reassigned, but it does not hoist variables to the top of its local or global scope
- **const** does not hoist variables to top of its local or global scope, and more importantly, such variables cannot be reassigned to another value



swipe left

JS

JS INTERVIEW QUESTIONS - PART 1

2. What does hoisting mean in JavaScript?

Hoisting is a concept in JS where the declaration of a variable moves to the top of the scope (global or local) it is defined in, above the line it was declared in the code.

```
function print() {
  console.log("before: "+ variable)
  // some other code here
  var variable = "hello"
  console.log("after: "+ variable)
}
```

```
print()
// before: undefined
// after: hello
```

Here is an example of hoisting. The **variable** is declared and assigned a value on the 4th line, but it is accessed on the 2nd line. Because of hoisting, the variable would be accessed but since its only the declaration that is hoisted, and not the assigment, the variable is undefined.

swipe left

JS

JS INTERVIEW QUESTIONS - PART 1

3. What are arrow functions and how are they different from normal functions?

Arrow functions are a short & easier way of writing functions in JS. The syntax involves an equal and a greater than sign to indicate an arrow:

```
const fn = (args) => {
  // do something
}
```

The differences between these and normal functions are:

- the **arguments** variable is not defined in arrow functions while they are in normal
- normal functions create their own **this**, while arrow functions do not have theirs
- arrow functions cannot be used as **constructors**.

swipe left

JS

JS INTERVIEW QUESTIONS - PART 1

4. Explain different methods you can use to select HTML elements from the DOM

The document object in JS has different methods you can use to select HTML elements. Here are some of them:

- **getElementById**: used to get an element by the value of its id attribute
- **getElementsByClassName**: used to get elements by the values of their class attribute
- **querySelector**: used to get an element that matches a CSS selector
- **getElementsByTagName**: used to get elements by the tags used for them
- **getElementsByName**: used to get elements by the values of their name attribute.

swipe left

JS

JS INTERVIEW QUESTIONS - PART 1

5. What is the name for == and ===, and what are the differences between these operators?

== is known as a **loose equality operator** while === is known as a **strict equality operator**.

Both operators are used for comparing values.

== compares values by checking if the values are the same. The types of the values are not strictly checked. So **"20"** (string) and **20** (number) compared using this operator will be **true**, because the values are the same.

=== compares values by checking if the types of the values, as well as the values themselves are equal. For this operator **"20"** (string) and **20** (number) will not be equal because the types differ.

swipe left

deeecode

JS

JS INTERVIEW QUESTIONS - PART 1

If you enjoyed this part, please share with others so they can learn from and improve their JavaScript knowledge.

Also, you can find the video version of this series on my YouTube channel - **DEEECODE**

In the video version, I wrote code examples and did more explanations, so do check it out 😇