1. **What is the javascript interpreter? When does it run?**
   A good example of Javascript interpreter is Google's V8 engine, The V8 engine is written in C++ and used inside Chrome and Node.js. It runs on your browser every time you load or make a request into a web page, translates the javascript to machine code.

   Ps: There are different JavaScript engines including Rhino, JavaScriptCore, and SpiderMonkey. These engines follow the ECMAScript Standards. ECMAScript defines the standard for the scripting language. JavaScript is based on ECMAScript standards. These standards define how the language should work and what features it should have. You can learn more about ECMAScript here.

2. **What is a function?**
   Function is a contained block of code, that can be reuse. A function should be able to take different parameters and provide the same result, based on the parameters. Functions can be defined as the glue of your program, it's what connects, processes and gives you results, It gives us a way to structure larger programs, to reduce repetition, to associate names with subprograms, and to isolate these subprograms from each other.

   >>Resource

3. **What is the difference between function declaration and function invocation?**
   When declaring a function, you are basically writing a set of steps that you want the function to do with the parameter that are passed to it.
   Also, functions can have a set of parameters or no parameters, and can also generate a return value, or just side effects. (What's the name of this kind of functions?)
   Invoking a function is when you actually want the function to "work" for you and process the value that you are as passing as arguments or simple generate your side-effect. Always using () after the function name.

4. **What is hoisting? How does it work? Do variables get hoisted as well? Explain the process.**
   Hoisting is a JavaScript mechanism where **variables** and **function declarations** are moved to the top of their scope before code execution.
   >>Resource

5. **What is the difference between function scope and block scope?**
   Variable declare inside a function, can't be accessed outside of the function. If you have a block of if statements, if you declare this variable inside them, you can't access outside of the block as well.

6. **Explain the difference between *const*, *var*, and *let*.**
   const is used when the identifier/variable won't be reassigned.

   let, is used when **the variable may be reassigned**, such as a counter in a loop, or a value swap in an algorithm. It also signals that the variable will be used **only in the block it's defined in**, which is not always the entire containing function.

   var, is now the weakest signal available when you define a variable in JavaScript. The variable may or may not be reassigned, and the variable may or may not be used for an entire function, or just for the purpose of a block or function.

   >>Resource

7. **In what order will these print? Why?**

```
console.log('a');

setTimeout(function() {
  console.log('b');
}, 0);

console.log('c');

setTimeout(function() {
  console.log('d');
}, 0);

console.log('e');
```

All the console logs will print first, because they are not variables or call back functions declarations. And then the order would be: **A C E and then B D**

8. If you have the following constructor function, what does this refer to? To the best of your knowledge, is this in javascript always the same?

```
function Sandwich(bread, meat, topping, condiment) {
    this.bread = bread;
    this.meat = meat;
    this.topping = topping;
    this.condiment = condiment;
}

let mySandwich = new Sandwich("whole wheat", "bologna", "none", "ketchup");
```

In this example, "This" will always refer to a instance of the object Sandwich, basically saying that all the keys coming after "this." will be refer to the Sandwich object. Meaning that when creating a new object using this constructor, the keys will be always "Bread", "Meat", "Topping", "Condiment".

9. What will the following print? Why?

```
let warning = "Don't eat my food."

function trick() {
  console.log(warning);
  if (true) {
    let warning = "If you eat my sandwich, you better make me a new one.";
    console.log(warning);
  }
  console.log(warning);
}

warning = "If you steal my things, there will be trouble.";

trick();
```

Output >>>
If you steal my things, there will be trouble.
If you eat my sandwich, you better make me a new one.
If you steal my things, there will be trouble.

Because when we call the function trick(), we had already change the value of the variable warning. So when running the function, the initial value of the variable is already "If you steal my things, there will be trouble." And then inside the if statement we are actually creating a new variable when we declare "let warning = … ", that it's the variable that we are logging right after, and because we declare as let, that variable is only going to be accessible inside it's scope (if conditional).

10. What is a loop and how does it work? What are the different loops in Javascript?

Loops are created to perform the same set of actions on multiple items, and many times. Loops are useful when we need to process data in arrays and/or objects as well.

For Loops = Allow us to perform a task a multiple x amount of times. We use the counter variable to track iterations.
While Loops = While loops continue to iterate until a condition is find true.

For in / Do While/ forEach… Filter...

11. What is an object? What is an array? What is the difference?

An array is a method of storing values, just like an object, but arrays are "ordered list" because it is numerically ordered. We know it is an array when its values are surrounded by [ ]. E.g. myArray[2].

An object is another way of storing values and **keys**. Because is not exactly an ordered list, you can create a key for the value, so you can access using dot notation. E.g. myObject.name.

12. What are the Primitive data types in Javascript?

string, number, boolean, null, undefined.

13. What does this refer to in Javascript?

The meaning of this has changed with time in Javascript. And it will also depend on where and how you are using this.

If you are using “this” inside a function, it will always refer to the function itself within the scope. Or as we have seen in question 8.

You can check it out other uses of “this” in this article, we are going to have another introduction to it when we get into react.

14. What does 'truthy'/'falsey' mean? What is an example of a 'truthy'/falsey' statement?

Truthy and Falsy are values that are considered false or true when evaluated in a boolean context:

Falsy Values:	**Truthy values:**
False	if(true)
Null	if({}) /Object
Undefined	if([]) /Array
0	if(42) and if(-43)
NaN	if(“brain”)
‘’ //Empty strings	if(new Date())
document.all	if(3.50) and if(-5.30)
	if(-infinity) and if(-infinity)