deeecode

RECURSION IN JS

JS

swipe left

deeecode

JS

RECURSION IN JS

Recursion in JavaScript, is a concept where a function calls itself, and keeps calling itself until something stops it.

It has the same idea of loops. Recursive functions allow you to execute some lines of code multiple times, until the condition you have specified for it to stop is met.

Here's how to declare and execute a normal function in JavaScript:

```
function printHello() {
  console.log("hello")
}

printHello()
// hello
```

swipe left

JS

RECURSION IN JS

Inside the **printHello()** function definition, we can also call the function like this:

```
function printHello() {
  console.log("hello")
  printHello()
}
```

This is recursion. Now when you call **printHello()**, it will log “hello” to the console, and run **printHello()**, which will, **AGAIN**, log “hello” to the console, and run **printHello()** again and again.

In this code, we didn’t let the recursive function know when to stop, so this will result in an **infinite recursive function** that crashes out application:

swipe left

deeecode

JS

RECURSION IN JS

```
printHello()
// hello
// hello
// hello
// hello
// hello
// hello
```

```
hello
hello
hello
hello
hello
internal/console/constructor.js:294
        if (isStackOverflowError(e))
            ^

RangeError: Maximum call stack size exceeded
```

Because I didn't let the recursion know when to stop, you can see above, hello is logged to the console repeatedly until the application creashes and the error in my console says "**Maximum call stack size exceeded**".

To avoid this, you specify a condition which lets the recursion know when to stop. This condition is called a **BASE CASE**.

Here is an example of a base case:

swipe left

deeecode

JS

RECURSION IN JS

```
let counter = 0

function printHello() {
  console.log("hello")
  counter++

  if(counter > 6) return;

  printHello()
}
```

Here, I've declared a base case that says **if(counter > 6) return**. This means, the recursion will keep happening as long as the **counter** variable is less than or equal to 6. At the point where it becomes more than 6, which is when the **counter** is **7**, the **return** keyword would terminate the recursion.

swipe left

Note that recursion is not a replacement for loops (like for and while) in JavaScript.

But, in some cases, recursions can be more effective and easier to read than loops.

If you enjoyed this post, or have any questions, let me know in the comments 😇