JS TIPS BY
EMMANUEL

JAVASCRIPT BEST PRACTICES.

@Ezenagu Emmanuel

01

Following JavaScript best practises can aid in faster page loads and improved performance, as well as improved code readability and ease of maintenance and debugging. Carefully written code can also help to avoid errors and security issues.

02

Avoid Global Variables

- Minimize the use of global variables.

- This includes all data types, objects, and functions.

- Global variables and functions can be overwritten by other scripts.

- Use local variables instead, and learn how to use closures.

03

Always Declare Local Variables

- Local variables should be declared for all variables used in a function.

- If the var, let, or const keyword is not used when declaring a local variable, the local variable will be converted to a global variable.

04

Declarations on Top

All declarations should be placed at the beginning of each script or function as good coding practise.

This will result in:

- A cleaner code
- Give users a single location to look for local variables.
- Make it simpler to avoid undesirable (implied) global variables.
- Reduce the likelihood of unwanted re-declarations.

05

Example;

```
// JS TIPS BY EMMANUEL

// Declare at the beginning
let firstName, lastName, price,
discount, fullPrice;

// Use later
firstName = "John";
lastName = "Doe";

price = 19.90;
discount = 0.10;

fullPrice = price - discount;
```

06

Initialize Variables

When you declare variables, you should always initialise them.

This will:

- Provide cleaner code
- Provide a single location for variable initialization.
- Avoid using undefined values.

07

Declare Arrays with const

- Declaring arrays with const prevents accidental type changes:

08

Example;

```
// JS TIPS FOR BY EMMANUEL

let cities = ["Los Angeles",
"New York", "Chicago"];
cities = 3;     // Changes
array to number
```

```
// JS TIPS FOR BY EMMANUEL

const cities = ["Los
Angeles", "New York",
"Chicago"];
cities = 3;     // Not
possible
```

09

Don't Use new Object()

- Instead of new String(), use "".
- Instead of new Number(), use 0
- Instead of using new Boolean(), use false
- Rather than new Object(), use {}
- Instead of using new Array(), use [].
- Instead of using new RegExp(), use /()/.
- Instead of using new Function(), use function (){}.

10

Example;

```
// JS TIPS FOR BY EMMANUEL

let x1 = "";                  //
new primitive string
let x2 = 0;                   //
new primitive number
let x3 = false;               //
new primitive boolean
const x4 = {};                //
new object
const x5 = [];                //
new array object
const x6 = /()/;              //
new regexp object
const x7 = function(){};      //
new function object
```

Beware of Automatic Type Conversions

- JavaScript is a loosely typed language.
- A variable can hold any type of data.
- A variable's data type can be changed:

12

Example;

```
// JS TIPS FOR BY EMMANUEL

let x = "Hello";     //
typeof x is a string
x = 5;               //
changes typeof x to a number
```

Use === Comparison

- Prior to comparison, the == comparison operator converts (to types that match).

- The === operator requires a value and type comparison:

14

Example;

```
// JS TIPS FOR BY EMMANUEL

0 == "";          // true
1 == "1";         // true
1 == true;        // true

0 === "";         // false
1 === "1";        // false
1 === true;       // false
```

Use Parameter Defaults

- When a function is called with an unspecified argument, the value of the unspecified argument is set to undefined.

- Undefined values can cause your code to fail. Assigning default values to arguments is a good practise.

Example;

```
// JS TIPS FOR BY EMMANUEL


function myFunction(x, y) {
  if (y === undefined) {
    y = 0;
  }
}
```

End Your Switches with Defaults

- Always include a default at the end of your switch statements. Even if you believe it is unnecessary.

Example;

```
switch (new Date().getDay())
{
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
    break;
  default:
    day = "Unknown";
}
```

Avoid Number, String, and Boolean as Objects

• Numbers, strings, and booleans should always be treated as primitive values, not as objects.

• Declaring these types as objects slows execution and has negative side effects:

Example;

```
// JS TIPS FOR BY EMMANUEL

let x = "Peter";
let y = new String("Peter");
(x === y) // is false
because x is a string and y
is an object.
```

Avoid Using eval()

- The eval() function is used to run text as code. But in most cases, it should not be necessary to use it.
- It also poses a security risk because it allows arbitrary code to be executed.

Was this post helpful to you?

Please like, save and share it
to your friends to inspire them!

Follow Me For More!

@Ezenagu Emmanuel