Synopsis of BEST PRACTICES PART 1:

How Make javaScript Understandable?

- **Use short names for variables and functions that is easy to understand:**

  Example:

  render(); checkEmployee = true;

- **use closures and the module pattern:**

  One of the most widely used design patterns in JavaScript is the *module pattern*. The module pattern makes use of one of the nicer features of JavaScript – closures – in order to give you some control of the privacy of your methods so that third party applications cannot access private data or overwrite it.

  for better variable scope management:

  1- Global variable var:

  Within JavaScript all variables are accessible from the **global scope** except variables that are declared within a function using the var keyword.

```
variable1 = 1; // Global Scope
var variable2 = 2; // Not within a function: Global
Scope
function funcName() {
  variable3 = 3; // No var keyword: Global Scope
  var variable4 = 4; // Local Scope only
}
```

  2- let:

  It works only in block of code scope and outside of block and function can't use and call it.

  Example:

  for(let i =0; i < 2; i++){}

  3- Const:

  This variable is like let only works in block or function that define it but can't change value of that. We can only use it with constant value.

  Example:

  const x = 2; let y = 3; return y += x;

Workaround: use closures and the module pattern:

What is module pattern?

Imagine you're at the fast food and want to order a sandwich and so you if you want firstType of sandwiches of myFastFood's fast food should call it like:

```
constant myFastFood = function(fruit, vegetable, bread){

        sandwichFirst(cheese){
                    let innertial = fruit + vegtable + cheese ;
                      bread.push(innertial);
        }

        sandwichSecond(ketchup){
                    let innertial = fruit + vegtable + cheese ;
                      bread.push(innertial);
        }

        publicAPI = {
              sandwichFirst: sandwichF,
             sandwichSecond: sandwichS
        }
        Return publicAPI;
    }
```

From:	primary:	fxn:	primary for that fxn:

Order : *myFastFood(fruit, vegetable, bread). sandwichF(cheese);*

Or

myFastFood(fruit, vegetable, bread). sandwichS (ketchup);

Revealing Module Pattern:

Keep consistent syntax and mix and match what to make global.

Validate your code:

Analyze source code static and with specific roles (Linting) and Standardisering of Strukturen of codes we can do all of these with some open source programs like **jsLint** and **ESLint**.

Comment as Much as Needed but Not More:

Good code explains itself" is an arrogant myth. of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters.

Avoid Mixing with Other Technologies:

Try to don't mix HTML or CSS with JavaScript for styles use CSS for tags use HTML but for manipulate of them use JS with ids and class's help.

Use Shortcut Notations:

Example:

arrayList Methods can help to make related codes too short like: **forEach, filter,** … .

Modularize code:

Good code should be easy to build upon without rewriting the core.

Enhance Progressively:

Avoid creating a lot of JavaScript dependent code. It's like a chain that with changing one method you should check all other functions and method that related to it.

Allow for Configuration and Translation:

Like a machine for good work all pieces should be on the suitable place to good work

And finding problem as fast as possible so your code should not be scattered throughout your code. This includes labels, CSS classes, IDs and presets.