Each project has its own peculiarities determined by the range of tasks to be solved by using it, the set of technologies used in the work, and by several other factors. However, in practically all cases, the approach used by developers is the key to success. There are a number of practices that will remain relevant regardless of the type of project being developed or technologies being used. They are equally important for both experienced professionals and beginners. Here are some non-negotiable standards (tips) to implement on the web technologies.

* **Split your code into small/helper functions**, don’t write big logic in single function.
* **Use linting** make sure you have to configure JSX linting in your project and every developer follow the same rules.
* Define a main function with proper name and divide them into different helper functions for different logic and call them inside main function.
* **Don’t declare a variable** for single use, declare a variable when you are using it multiple areas.
* Don’t declare an **functions/variables** that has already been declared in an outer scope.
* **Easy, readable and meaningful** variable/function name should be defined with the camel casing.
* Add detailed comments to complexity functionality, so that other developers get easily understand your logic.
* Don’t try to update the object by **assigning values directly** to it. In Javascript there is a method Object.assign() which is used for cloning an object.
* Don’t use any **vulnerable function**s. For example, If javascript code uses the **eval()** function. This function evaluates a string and execute it as javascript code. If the input string is controlled by the user, this could lead to XSS (cross-site scripting) vulnerabilities.
* Don’t **defined** too many **variables** inside a function are not accessible from outside the function (**Global Variables**).
* Long-lived variable names should be descriptive whereas the name of the temporary variable should be small like ‘i’,’j’,’k’ used in loops.
* Camelcaps method should be used and in multiple words name first letter of each word should be capital except first. For example ‘employeeName’.
* Do follow all the standard rules for Clean and valid code.
* Don’t mix up with other technologies.
* Avoid heavy nesting and loops inside loops by that code gets unreadable after a certain level of nesting.
* Optimize the loops as much as possible so that the performance of the loop increases.The first step in optimizing the amount of work in a loop is to minimize the number of object members and array item lookups.
* Don’t use multiple ES versions.
* To avoid making a choice every time, Use **Single Quotes** and stick with it while you are representing a string.
* Use **switch** instead of **if when** you are comparing multiple possible conditions of an expression and the expression itself is non-trivial.
* break is optional and is used to prevent "*falling*" through all the other case statements.As return ends the function execution , never break **return** statements.
* Don't use **var** keyword for single use of variables because it is less error-prone.
* Don’t declare any unused **variables/Classes/Functions.**
* Define **class/Constructor** names that start with a Capital **letter**.
* Define function names that start with **small letter.**
* Each line of a code should end with a **semicolon.**
* Explain about each class with at least **2 lines of comments**.
* Use **===** instead of **==** for condition comparisons in the code.
* Each line should contain intent of **4 spaces/1 tab**.
* Don’t represent any consoles and Debuggers in **qa / staging / prod** environments.

There are a lot of things you should consider while committing or pushing a code into the Gitlab according to the web technologies.Here, are some Do’s and Don'ts you have to check before pushing the code.

## The Don’ts :

1. No consoles and No Debuggers .
2. No unused packages/Libraries should be installed.
3. No lintIn errors should be represented.
4. No local API urls should be represented.
5. No alert boxes.

The Do's :

1. Use feature branches, no direct commits on master.
2. Commit messages with a specific reference.
3. Checkout the particular branch you want to commit before pushing it.
4. Pull the complete code into your local system before committing.