CLEAN CODE

You might think that the majority of a developer's work involves writing code. However, in reality, a significant amount of time is spent on reading code(by other team members, by people who are no longer part of your team, even code that you wrote two weeks ago but may not remember much about).

A. Which is clean code

Example 1 =>

```
const x = function (z) {
  const w = "Hello ";
return w + z
};

x("John");
```

Example 2 =>

```
const generateUserGreeting = function (name) {
  const greeting = "Hello ";
  return greeting + name;
};

generateUserGreeting("John");
```

Example 2 represents clean code.

Example 1, Single characters can be used as variable names in the context of a loop or a callback function, but avoid them elsewhere.

B. Naming function and variables

- A good name is descriptive
- Use a consistent vocabulary

```
// Good
function getPlayerScore();
function getPlayerName();
function getPlayerTag();

// Bad
function getUserScore();
function fetchPlayerName();
function retrievePlayer1Tag();
```

Variables should always begin with a noun or an adjective (that is, a noun phrase) and functions with a verb.

```
// Good
const numberOfThings = 10;
const myName = "Thor";
const selected = true;

```

```
// Bad (these start with verbs, could be confused for functio
const getCount = 10;
const isSelected = true;

// Good
function getCount() {
  return numberOfThings;
}

// Bad (it's a noun)
function myName() {
  return "Thor";
}
```

- Use searchable and immediately understandable names

  Sometimes, it can be tempting to use an undeclared variable. Let's take another look at an example:

```
setTimeout(stopTimer, 3600000);
```

  What does the undeclared variable '3600000' mean, and how long is this timeout going to count down before executing 'stopTimer' ?

  Solution, make the code more meaningful by introducing a descriptive variable:

```
const MILLISECONDS_PER_HOUR = 60 * 60 * 1000; // 3,600,000;

setTimeout(stopTimer, MILLISECONDS_PER_HOUR);
```

  Much better, isn't it? You don't need to perform any calculations when reading this code.

C. Indentation and Line length

- Indentation

  The war between coders that use tabs and coders that use spaces to indent their code is essentially a joke by now. What actually matters is consistency.

- **Line length**

  Generally, your code will be easier to read if you manually break lines that are longer than about 80 characters. Many code editors have a line in the display to show when you have crossed this threshold. When manually breaking lines, you should try to break immediately after an operator or comma. There are a few ways to format continuation lines. For example, you can:

```
// This line is a bit too long
let reallyReallyLongLine = something + somethingElse + anotherThing + howManyTacos + oneMoreReallyLongThing;

// You could format it like this
let reallyReallyLongLine =
  something +
  somethingElse +
  anotherThing +
  howManyTacos +
  oneMoreReallyLongThing;
```

```
// Or maybe like this
    let anotherReallyReallyLongLine = something + somethingElse + anotherThing +
                                      howManyTacos + oneMoreReallyLongThing;
```

D. Comments (tips to make good comment)

Tell why, not how. Good comments explain the reasons behind a piece of code.

```
// Bad Example - comment doesn't tell why, only what and how

// This function increments the value of i by 1
function incrementI(i) {
  i = i + 1; // Add one to i
  return i;
}

// Better Example - comment tells a why

// This function increments the value of index to move to the next
function incrementI(i) {
  i = i + 1;
  return i;
}

// Good Example - the code tells all that is needed

function moveToNextElement(index) {
  index = index + 1;
  return index;
}
```

This doesn't mean good code should lack comments. In many situations, well-placed comments are priceless:

```
function calculateBMI(height, weight) {
    // The formula for BMI is weight in kilograms divided by height in meters squared
  const heightInMeters = height / 100;
  const bmi = weight / (heightInMeters * heightInMeters);
  return bmi;
}
```

The comment helps to refresh the reader on how BMI is calculated in plain English.