# **Synopsis of BEST PRACTICES PART 1:**

How Make it Understandable?

Use short names for variables and functions that is easy to understand.

Example:

render(); checkEmployee = true;

How JS Language works?

Computers can Understand only binary Language. Everything in that world is only 0 or 1. But it’s hard to convert allcommand to binary So computer engineers invented translation tool. A translation is a special utility on the computer (either an interpreter or a compiler) is used to translate the code you write into commands a computer can understand. Computers have 2 type of this translator:

1. interpreting codes, that translate commands line by line. every time the program is run. It meaning is runtime translate the code
2. compiling the code, that at first should translate code to the

computer language (compiling the code) then run the code

**JS** asserted as an interpreting language but the JavaScript engine actually compiles the program on the fly and then immediately runs the compiled code. So we can say it’s between interpreting and compiling languages.

1. **Oprators:**

Operators are how we perform actions on variables and values Type of values

Math oprators:

+ , - , \* , /  --------------------------------------- 2 + n;

Assignment:

 = -------------------------------------------------- m = 2 + n;

Compound Assignment:

 +=, -=, \*=,  /=  ----------------------- m += 2 --------- m = m + 2;

Increment/Decrement:

++ , --  --------------------------------- m++; ----------- m = m + 1;

Object Property Access:

**.** ----------------------------------------- console**.**log();

Equality:

== , === , != , !==  -------------------------- if( m == 2)

Comparison:

 < , > , <=, >=  ---------------------------------if (m <= 2)

Logical:

 && , ||  ----------------------------------------- if(m || n )

1. **Variables**:

**Variables are containers for carrying values within your script.** You can always give them a new value and restart your script. There is no “permanent” equality to satisfy some condition. The left side of the statement has a completely different purpose than the right side of the statement.

For example: var days = 7;



This is called declaring the variable. It creates a new truck called *days* that can drive around your script and deliver its value OR pick up a new value. The *var* keyword announces that you are creating a new variable. Or, in the analogy we are about to use, creating a new truck. The variable needs a unique name, which is *days* here. This distinguishes this truck from all the other trucks. The assignment operator, or “=” sign, loads the value 7, into the truck.

1. **Statements**:

In a computer language, a group of words, numbers, and operators that performs a specific task is a statement. Statement at the end have (**;)** sign.

Example: m = n + 2;

1. **Expression:**

Epressions are units of code that can be evaluated and resolve to a value. expressions in js can be divided in categories.

**2** -------------------------------------*literal value expression*

b------------------------------------variable expression

b\* 2 -------------------------------arithmetic expression

a = b – 2-------------------------------assignment expression

‘A’ + ‘B’ *-------------------------------String expression*

True/false/ undefined/ i/ this *--------------primery expression*

Functions ---------alert(a) ----------------------call expression

And ….

1. **Value & Types:**

In JavaScript's system, most of this data is neatly separated into things called **values**. Every value has a type, which determines the kind of role it can play. There are six basic types of values:

Numbers ----------------------------------var x = 1;

Strings ------------------------------------var x = “myString”;

Booleans ----------------------------------var x = true;

Objects ------------------------------------ var x = {firstName: "John",  
   lastName : "Doe", id       : 5566,fullName : function() {  
        return this.firstName + " " + this.lastName;  
    }  
};

Functions ------------------------------------- function() {  
        return this.firstName + " " + this.lastName;  
    }

undefined values---------------------------var x;

1. **Conditionals:**

Very often when you write code, you want to perform different actions for different decisions.You can use conditional statements in your code to do this. In JavaScript we have the following conditional statements:

* Use**if**to specify a block of code to be executed, if a specified condition is true
* Use **else** to specify a block of code to be executed, if the same condition is false
* Use **else if** to specify a new condition to test, if the first condition is false
* Use **switch** to specify many alternative blocks of code to be executed

For example:

var myDecision = “I want to go home”;

function myTrip (myDecision) {

**if** (myDecision ===” I want to go home”) {

return “you’re at home way”;

}**else** {

return “you’re going to go to work”;

}

This function meaning is a conditional is like a fork in the road. You go one way if you want to go home and another if you want to go to work

1. **Loops:**

if you want to run the same code over and over again, each time with a different value.

* **for**- loops through a block of code a number of times
* **for/in**- loops through the properties of an object
* **while**- loops through a block of code while a specified condition is true
* **do/while** - also loops through a block of code while a specified condition is true

Example: for loop:

Let’s look at a basic one:

For(int i = 0; i <5 ; i++){

Console.log(i);

}

The output would be:

0, 1, 2, 3, 4

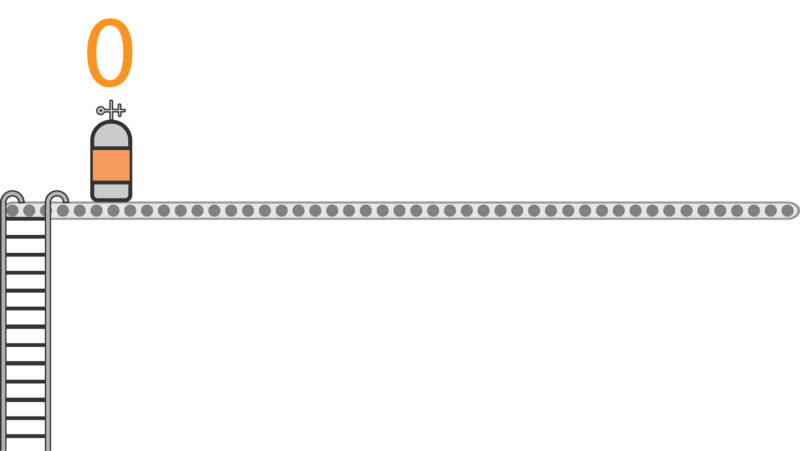
What is the concept of *i*? when does *i* increase?

**Imagine that you are a scuba diver,** and you are planning a day-long trip to a new location. You are checking out a new reef, so you will probably want to take multiple trips to the bottom to make sure you see all the coral and marine life.

Before you start your dive, you need to determine how many oxygen tanks you will need over the course of the day.

let i= 0;

This is the **initialization.**It tells you the value of the first oxygen tank. In this case, you start at tank 0.



i < 5

This is the **condition**. It is kind of like the capacity of the boat. You can only add as many oxygen tanks as your boat will hold.

#### Setting up oxygen tanks (update)

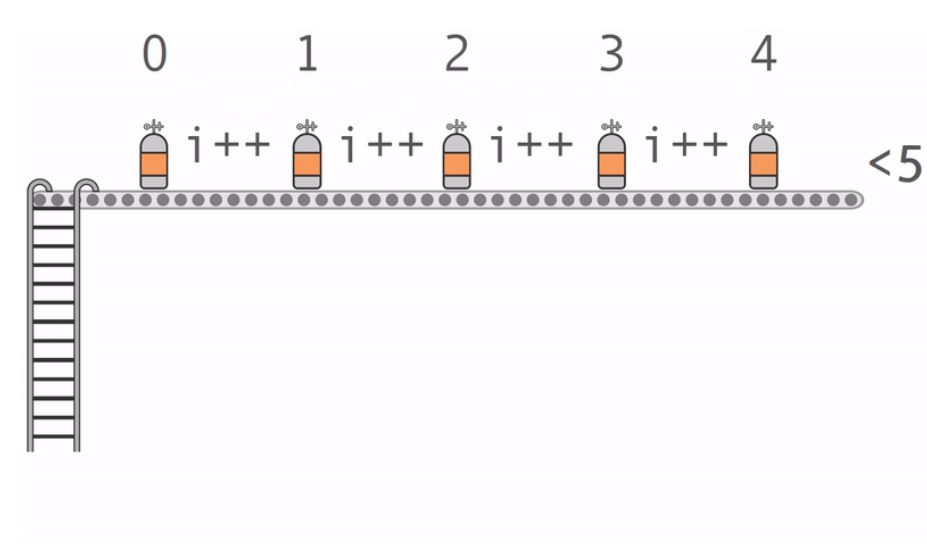
So far, we know that our first oxygen tank has a value of 0, and the last one must be less than 5. But how many tanks do you need to prepare?

The last part, called the **update**, tells us how many tanks we need to line up.

i++

This is shorthand for: i = i+1

It means that every time we finish a loop, we will add 1 to i. Since i starts at 0, here is what that looks like.



We continue to add oxygen tanks until we hit the limit. When we add one tank at a time, the last value that fulfills the **condition** is 4.

For(int i=0; i<5; i++){

console.log("The current value is: " + i);

}

You will usually find that your for loops have many lines of code. But for the time being, we are just sticking to the one line so that you can trace the path of i.

1. **Functions:**

Let's think about the general concept of cooking with a recipe first. Using a recipe means that:

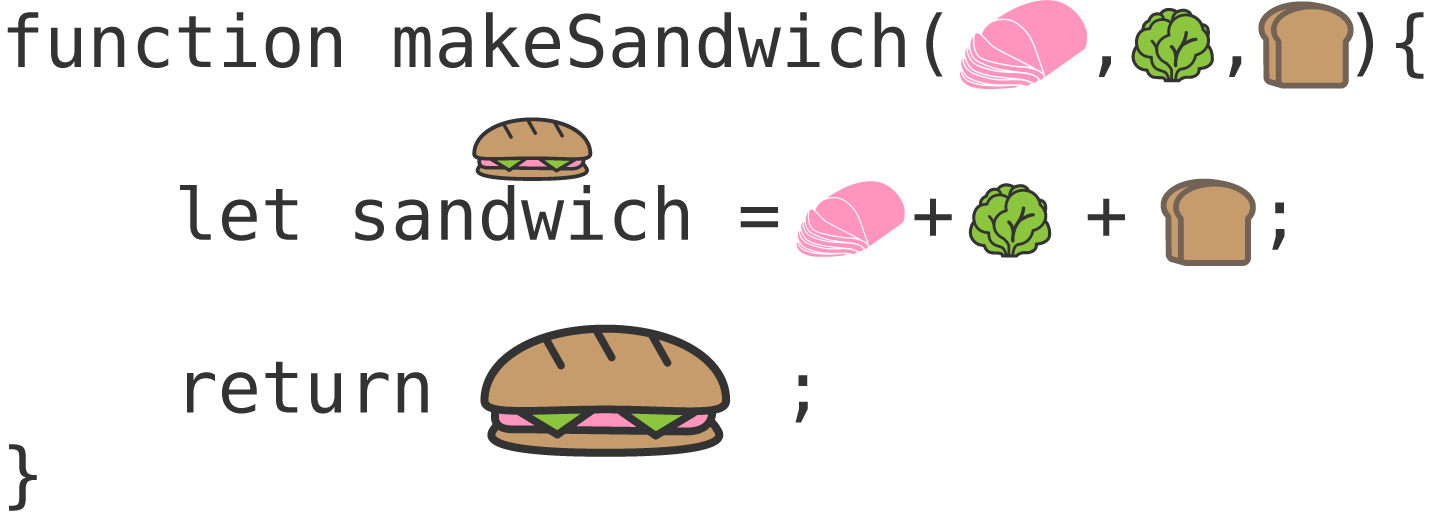
1. You start with a specific set of ingredients
2. You perform a specific procedure using those ingredients
3. You will get a reliable product at the end

A function is also a reusable recipe that performs the same set of actions over and over again on a set of ingredients.

Those ingredients are called **parameters**. You can see them in the diagram below.

Some functions **return**a value. That means that they give you a new value that you can then use throughout your script. Other functions do not return a value. Instead, they might change a value that already exists in your script. Think of it like chopping onions. There is no "new" product, just the same product in a new format.

Check out the example below to see the general layout of a function. In this example, we are **declaring** a make Sandwich function. It does return a value- a full sandwich. In other words, we are documenting the recipe on how to make a sandwich.



1. **Codes** **comment:**

In computer programming, a comment is a programmer-readable explanation or annotation in the source code of a computer program. They are added with the purpose of making the source code easier for humans to understand, and are generally ignored by compilers and interpreters.

In JavaScript, there are two types of comments possible:

a single-line comment start with // before text

a multiline comment. Start with /\* before text and \*/ end of text