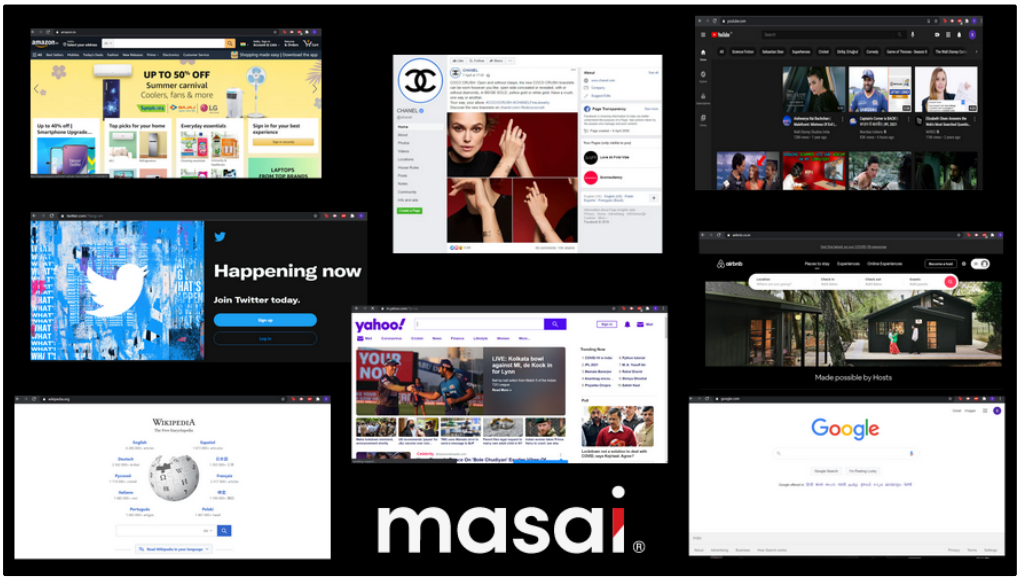
Lecture 1 : Introduction To FSD

Why Web Dev: Browser is the new OS!



Browsers were created to link pages with information (largely text, with some images). But today, the browser is used for everything from watching entertaining videos to learning to code, sharing messages/photos with friends, and even for secure banking:

1. Youtube - is for uploading and sharing all kinds of videos, which could be personal, movie trailers, do it yourself videos, news items, anything that users want
2. Amazon - for online shopping
3. Facebook, Twitter - for social networking with family, friends, workmates
4. Google, Yahoo - these are search engines and provide email services too
5. Wikipedia - online encyclopedia that can be both read and added to by users

What is Web Dev exactly: Key components

Web Development is the building and maintenance of websites. It's what makes a website

What makes a website

1. **look great**
2. **work fast,** and
3. **provide the intended user experience**

Here are the key components of web development:

HTML, CSS, JS

Instructor Task (5 mins): High-level view of HTML CSS JS

HTML, CSS and JS are the three key technologies/programming languages involved in web dev.

- HTML provides the **structure** of the webpage,
- CSS adds **color & design** and
- JavaScript provides **interactivity** to it.

HTML vs CSS vs Javascript

- https://codepen.io/rcyou/pen/QEObEk/
- It may sound CSS is unimportant but check out http://www.csszengarden.com/ .

Data

Data

Role of data on the web.

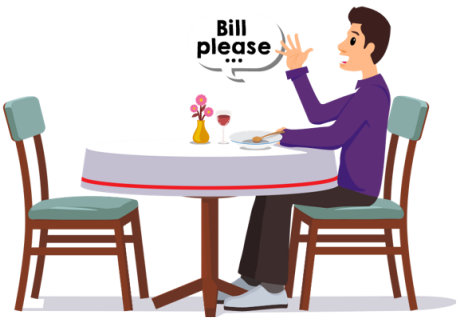
- Data plays a key role in our life. There could be a lot of sources from where data can be generated like Meteorological Data, Natural Data, Financial Data, etc.
- When we use applications like Facebook, Instagram, Twitter, Whatsapp, etc, we use to spend a lot of time. Hence, we are generating the data.
- The data is used by the companies, so they can generate some valuable information like in the case of Amazon, they will generate offers for you based on your shopping history.

Frontend vs Backend vs Database Interaction

- Frontend gives you interactivity through which we can interact with the platform. The platform could be any like Swiggy, Zomato, Amazon, Facebook etc.
- The backend provides processing, What to search, What to save, etc.
- The database stores the data.
- Each of the above operates on data only.

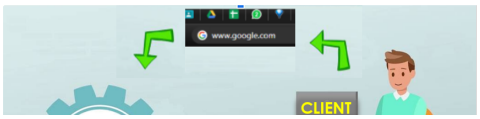
Server / Client Model

When you search for something on Google, your query goes to Google (servers) via the internet. Then a response is sent back by Google, which is shown by your browser (client). This is how most of the web works.



You (Client) are like a customer in the restaurant and the Server is like the waiter taking your order (or, request). The waiter takes your order into the kitchen and comes out with the food (response).

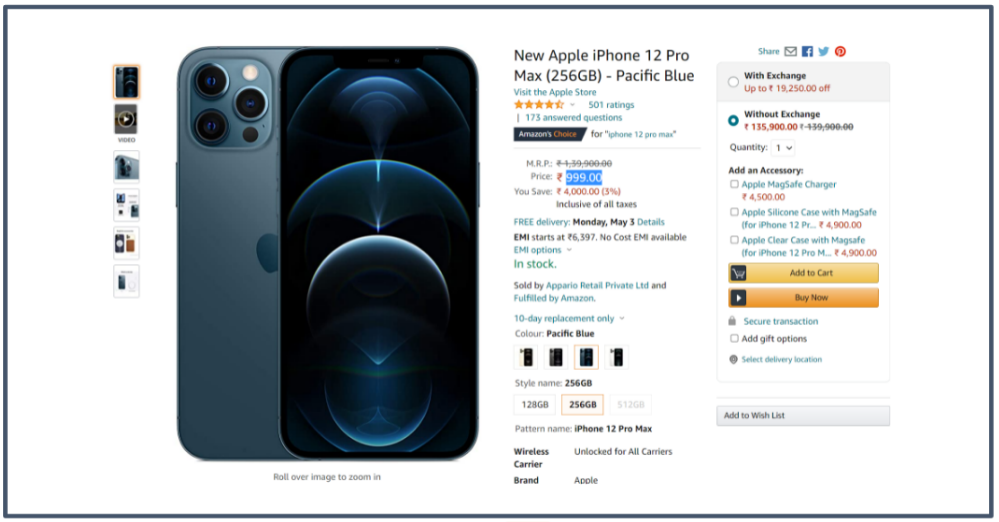
You don't need to go to the chef for the main course, to the bartender for drinks, to the cashier for bill - the waiter handles all those (services) for you!



In a similar way, when you type in a website address, you send a request to the server which then processes that request and sends the appropriate response, like text or video etc. This response is then shown or rendered by your browser (client).

Change the price of any product on Amazon!

If the webpage is rendered on the browser on ***your*** computer - you should be able to change it! Well, you can .. once you know where to look :)



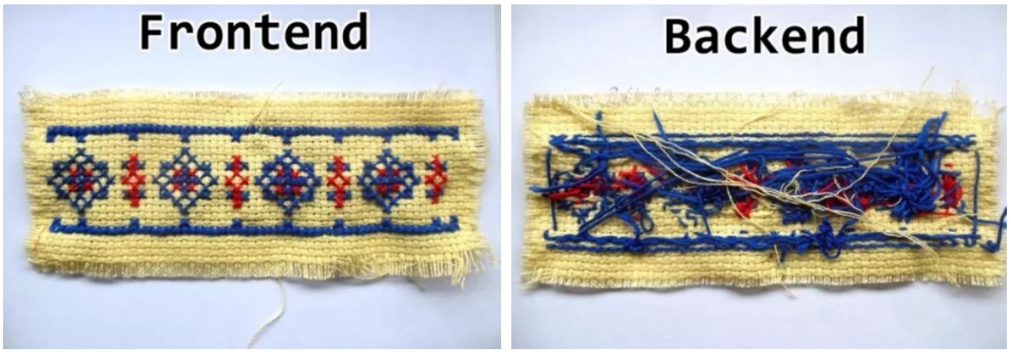
1. Go to any (expensive) product's page - say this iPhone.
2. Now select the price you want to change, right-click and choose Inspect. This will show the code for that element (see Fig 1).
3. Now click on the 3 dots and then choose Edit HTML as shown in Fig 2.
4. Change the price as shown in Fig 3 and voila!

Why the changes disappeared on refreshing?

The page is being sent by Amazon's server. We have only changed the client copy on our computer, not on Amazon's server. Hence when we open the same URL in another tab (or refresh - which resends our request and gets a new response), Amazon's server returns the

request and gets a new response), Amazon's server returns the product page with the original price.

Backend / Frontend / FSD



A good website is a combination of a great user interface and a robust codebase. Frontend and Backend are a lot like different parts of a car. Frontend represents all of the elements a user will see. They are the stylish look of the car and the plush interiors. Backend represents all of the elements of a car you can't see that make it work, like a powerful engine, carbon-fibre chassis.

Ask the students what the below image means, to drive home the point that:

To build a great website, a good-looking frontend is as important as a solid backend.

Internet

Everyone and their mother has literally used the internet in some form or another. So what is it, really?

The Internet is a vast network that connects computers all over the world.

Ratatype

https://www.ratatype.com/groups/4738641/

Intro

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved.

JavaScript enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

Node JS

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command-line tools and for server-side scripting. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server-side and client-side scripts.

Coding Platform

https://replit.com/

Running Javascript Code

```
node filename.js
```

Variables:

We use programming languages like JavaScript to store and manipulate information. Variables in JavaScript are used to store different kinds of data. Think of a variable like a bottle. We use bottles to store water, in much the same way we use variables to store various forms of data in JavaScript.

Example:

```
var number = 200
```

Syntax: You can also declare multiple variables in one statement as seen in the syntax below.

Multiple Variables Example:

```
var a = 100, b = 200, c = 300;
```

The data inside variables is not constant. This means the data inside a variable can be changed.

Example:

```
var a = 200
a = 100
```

In the above example the variable called a first contained the value 200 but a = 100 means that a now contains the value 100.

NOTE: You can name variables whatever you want but try to give them good/descriptive names that tell the reader what the variable is used for

Data Types In Javascript:

There are 7 types of data in JavaScript but we will focus on 2 types of data for now.

Strings:

The first type of data is a String. This is used to store a sequence of characters used to represent text.

Example:

```
var name = "Masai School"
```

Any data within quotes " " is a String in JavaScript.

Numbers:

The second type of data we want to know is a Number, which is used to store any kind of numbers. We have already seen this type of data in the variables example. Numbers can store both Whole Numbers/Integers and Decimals.

Example:

```
var num = 100
var dec = 100.001
```

Checking the type of data: Lets say you have some data but you don't know what type it is. You can user the typeof() inbuilt code to find the type of the data.

Example:

```
var name = "Masai School"
console.log(typeof(name))
```

Output:

```
string
```

Mathematical operators in JavaScript

Common Operators

Javascript supports all the commonly used mathematical operators. Namely `+` `-` `*` `/`.

Example:

```
var a = 2
var b = 3
var c = a + b
var d = a * b
var e = a / b
var f = a - b
```

Output:

```
c = 5
d = 6
e = 0.6666666666666666
f = -1
```

`Modulo` or Remainder Operator:

Many programming languages including JavaScript have a modulo operator `%`. This operator returns the remainder when one variable is divided by another.

Example:

```
var a = 10 % 7
```

Output:

```
a = 3
```

This operator is often useful when you want to check if a number is odd or even.

Example:

```
var a = 10 % 2
var b = 11 % 2
```

Output:

```
a = 0
b = 1
```

Try this out for yourself, any even number `%2` returns 0 while any odd number `%2` returns 1.

`Exponentiation` operator:

This operator is represented by `**`. This returns the value of the first operand raised to the power of the second operand. For example 2^4 = 16.

Example:

```
var a = 2 ** 4
var b = 3 ** 2
var c = 10 ** 1.5
```

Output:

```
a = 16
b = 9
c = 31.622776601683793
```

String concatenation

A special property of `Strings` is that they can be combined or concatenated with one another.

Example:

```
var word1 = "Welcome"
var word2 = "Masai"
var word3 = word1 + " to " + word2 + " school!"
console.log(word3)
```

Output:

```
Welcome to Masai school!
```

Strings can also be combined with other types like `numbers`.

Example:

```
var num1 = 1
var num2 = 2
var output = "1 + 2 = " + (num1 + num2)
console.log(output)
```

Output:

```
1 + 2 = 3
```

Note: Notice the circular brackets between `num1 + num2` this tells javascript that we want to add the two numbers mathematically. Without the brackets the output would be `1 + 2 = 12`.

`Booleans`:

The last data type we are going to learn about is a `Boolean`. This data type has only two values `true` and `false`.

Example:

```
var x = true
var y = false
```

Relational operators

These operators allow you to test the relation between 2 values and returns a `boolean`. JavaScript unlike other languages allows you to compare any type with any other type!

Greater than and greater than equal to

The **greater than** operator `>` allows you to check if one value is greater than the other. It returns `true` if the first value is greater than the second and `false` if the second value is greater.

Example:

```
20 > 10
10 > 20
10 > 10
```

Output:

```
true
false
false
```

The **greater than equal to operator** `>=` also checks if the second value could be equal to the first value.

```
10 > 10
10 >= 10
```

Output:

```
false
true
```

Lesser than and lesser than equal to

The **lesser than** operator `>` allows you to check if one value is lesser than the other. It returns `false` if the first value is greater than the second and `true` if the second value is greater.

Example:

```
20 < 10
10 < 20
10 < 10
```

Output:

```
false
true
false
```

The **lesser than equal to operator** `<=` also checks if the second value could be equal to the first value.

```
10 < 10
10 <= 10
```

Output:

```
false
true
```

Comparison Operators

Equality

The **equality** operator `==` lets you test if two values are equal or not. It accepts 2 inputs of any type and outputs `true` if they are equal and `false` if the are not equal.

Example:

```
1 == 1
1 == 2
"Masai" == "Masai"
"Masai" == "masai"
```

Output:

```
true
false
true
false
```

Inequality Operator

The **inequality** operator `!=` performs the opposite function of the equality operator. It accepts 2 inputs of any type and outputs `false` if they are equal and `true` if the are not equal.

Example:

```
1 != 1
1 != 2
"Masai" != "Masai"
"Masai" != "masai"

1 != '1' // false
1 !== '1' // true
```

Similar to `===`, `!==` will check for type as well.

It is recommended to use `===` and `!==` when it comes to comparison operators

Output:

```
false
true
```

```
false
true
```

Conditional Statements

Conditional Statements

- Conditional statements are used to decide the flow of execution based on different conditions. If a condition is true, you can perform one action and if the condition is false, you can perform another action.
- Through Conditional Statements, we can control which code needs to run or which code will not run.
- Code runs based on certain conditions.
    - For Ex: let's understand with the analogy, the traffic light controls the flow of vehicles on the road. Depending upon the colour of light, the actions happened. If light is green , then it is asignal to move whereas if the light is red then it is a signal to move.
- Based on the comparision , if the comparison is true then it will execute the one block of code otherwise another block of code.

Different Types of Conditional Statements

There are mainly three types of conditional statements in JavaScript.

1. If statement
2. If…Else statement
3. If…Else If…Else statement

if Statement

- It is to specify a block of JavaScript code to be executed if a condition is true.

Syntax

condition

block of code to be executed if the condition is true

- **if()**
  - It takes a Boolean Value or the expression that will give boolean value.
- if() { }
  - { } knows as code block.

a) If with Boolean Value

```
console.log("Code Start")
        if(true) {
          console.log("Inside Code")
        }
console.log("Code End")
```

b) If with Expression

- The decision is based on the value of Expression

For Example :

```
if(5>3){
       console.log("Inside Code");
  }
```

c) If with Variables

- The decision is based on the value of Expression

For Example :

```
var name1 = "rahul";
```

```
var name2 = "rahul";
var check = (name1==name2);

if(check){
   console.log("Both Names are same");
}
```

Code 1 : Check Whether two numbers are equal

```
var a = 2;
var b = 3;
var c = (a==b);

if(c)
{
    console.log("a and b are equal");
}
```

if/else Statement

- The `if...else` is a type of conditional statement that will execute a block of code when the condition in the `if` statement is `truthy`. If the condition is `falsy`, then the `else` the block will be executed.
- Here is a list of `falsy` values:
  - false
  - 0 (zero)
  - 0 (negative zero)
  - 0n (BigInt zero)
  - `""`, `''`, ``` `` ``` (empty string)
  - null
  - undefined
  - NaN (not a number)
- If the condition is true, then one block of code executes.

- Else another block of code executes.

Syntax

condition

block of code to be executed if the condition is true

block of code to be executed if the condition is false

Code 2 : Check which number is greater

```
var a = 3;
var b = 20;

if(a>b)
{
  console.log("a is greater");
}
else
{
  console.log("a is not greater");
}
```

Code 3 : Check Whether two names are equal or not

```
var name1 = "suraj";
var name2 = "suraj";

if(name1==name2)
{
  console.log("Names are Equal");
}
else
{
  console.log("Names are not equal");
}
```

Hotel Bill Discount

Code 4 :

Given total_bill, discount_start_price if you satisfy the condition Print Discount Availaible Otherwise print No Discount

```
var total_bill = 699;
var discount_start_price = 500;

if(total_bill>=discount_start_price){
  console.log("Discount Availaible");
}
else{
  console.log("No discount");
}
```

Else-if Statement

- There will be times where you want to test multiple conditions. That is where the `else if` the block comes in.
- When the `if` statement is `false`, the computer will move onto the `else if` statement. If that is also `false`, then it will move onto the `else` block.

Syntax

condition1

block of code to be executed if condition1 is true

condition2

block of code to be executed if the condition1 is false and condition2 is true

block of code to be executed if the condition1 is false and condition2 is false

Bill and Discount

Problem Statement: According to the total_bill, the discount will be applied.

Aa Total Bill	Discount Applied

Aa Total Bill	Discount Applied
Greater Than 500	10%
Greater Than 1000	20%
Others	No Discount

Code 5 : For a Restaurant, write the program for the following total_bill > 500 Then print 10% discount total_bill > 1000 Then print 20% discount Otherise No discount

```
var total_bill = 799;

if(total_bill > 1000)
{
  console.log("20 % discount");
}
else if(total_bill > 500)
{
  console.log("10 % discount");
}
else
{
  console.log("No discount");
}
```

If-Else-If vs if-if-if :

Code 6 : If-Else-If

- ***My mother told me to get any one of the thing from the market**

1. **If Rice is availaible then print Buy rice**
2. **Else If wheat is availaible then print buy wheat**
3. **Else If apple is availaible then print buy apple****

```
var rice_availaible = false ;
var wheat_availaible = true;
var apple_availaible = true;

if(rice_availaible)
{
```

```
  console.log("Buy rice");
}
else if(wheat_availaible)
{
  console.log("Buy Wheat");
}
else if(apple_availaible)
{
  console.log("Buy apple");
}
else
{
  console.log("Nothing is availaible");
}
```

Code 7 : If - If - If

- ***My mother told me to get all of the thing if available from the market**

1. **If Rice is available then print Buy rice**
2. **If wheat is availaible then print buy wheat**
3. **If apple is availaible then print buy apple****

```
var rice_availaible = true ;
var wheat_availaible = true;
var apple_availaible = false;

if(rice_availaible)
{
  console.log("Buy rice");
}

if(wheat_availaible)
{
  console.log("Buy Wheat");
}

if(apple_availaible)
{
  console.log("Buy apple");
}
```

Code 8 : Solve the Marriage Problem

Legal Age in India Males ----> 21

Females ----> 18

```
var gender = "female";
var age = 21;

if(gender == "male")
{

  if(age>=21)
  {
    console.log("Males : get marry");
  }
  else
  {
    console.log("Males : Can't get marry");
  }

}

else
{
  if(age>=18){
    console.log("Females : get marry");
  }
  else{
    console.log("Females : Can't get marry");
  }
}
```

Code 9 : Given a char , you need to print whether the char is a vowel or not vowels : a, e, i, o, u

```
var char = "z"

if(char == "a")
{
  console.log("vowel");
}
else if(char == "e")
{
  console.log("vowel");
}
else if(char == "i")
```

```
{
  console.log("vowel");
}
else if(char == "o")
{
  console.log("vowel");
}
else if(char == "u")
{
  console.log("vowel");
}
else{
  console.log("Not a vowel");
}
```

Not Operator

- On applying to a boolean value, the *not* operator turns *true* to *false* and *false* to *true*.

Logical Operators

What is Logical Operators ?

Whenever we need to connect two statements.

In the Last class we learn about conditional statements, that says that if one condition is true then do X otherwise do Y.

For Example: In traffic lights , If the lights are green then Move and if the lights are red then Stop.

But In real, there might be multiple condition on which some result depends.

For Example: Suppose I need to submit some documents in Masai, and the documents are pan card and License Id then only I will get the admission.

here, you can observe that I will only get the admission only when I have the PAN Card and License Id (Both are important).

1. **AND Operator**

PAN Card	**License Id**	**Admission**	
True	False	False	I will not get the admission if I have the PAN Card but not License Id.
False	True	False	I will not get the admission if I have not the PAN Card but have License Id.
False	False	False	I will not get the admission if neither I have the PAN Card nor the License Id.
True	True	True	I will not get the admission if I have the PAN Card as well as License Id.

Similarly, we can have multiple condition on which , the result is dependent.

- Our Boolean operators takes the input values as boolean and produce the result in boolean.
- In programming, we use to denote the AND operator like in this way **&&**.
- **Input (Boolean Value) ——&&-—> Output (Boolean Value)**

// PIC of && Table

Code 1 : AND Operator

```
var a = true;
var b = true;

var c = a && b;
console.log(c);

a = true;
b = false;
console.log(a&&b);

a = false;
b = true;
console.log(a&&b);

a = false;
b = false;
console.log(a&&b);
```

Code 2 : AND with numbers

```
var a = 5>3;
var b = 6>3;

var c = a && b;
console.log(c);
```

Code 3 : if/else

```
// Ist Part : Without AND

if(5>3)
{
  if(6>3)
  {
    console.log("Both are true");
  }
}


// IInd Part : With AND

if(5>3 && 6>3)
{
  console.log("Both are true");
}
```

Code 3 : Combination of multiple statements

```
// Try out on Console

(5<4) && (3>1) && (2>1) && (4<1)
```

Code 4 : **[Student Task]** Check whether Rahul passed or not

```
// For English Subject, Check whether Rahul passed or not

var subject = "english";
var passing_marks = 70;

var rahul_marks = 75;
var rahul_subject = "english";

if((rahul_subject == subject) && (rahul_marks >= passing_marks))
```

```
{
  console.log("Rahul Passed");
}
else
{
  console.log("Rahul not passed");
}
```

Code 5 : [Student Task] Marriage Problem

Gender is male and age ≥ 21 : He can marry

Gender is female and age ≥ 18 : She can marry

```
var gender = "male";
var age = 21;

if((gender == "male") && (age >= 21))
{
  console.log("Male : Can Marry");
}
else if((gender == "female") && (age >= 18))
{
  console.log("Female : Can Marry");
}
else
{
  console.log(gender,"Can't get Marry");
}
```

Code 6 : Differentiate between ,(coma) and +

```
var a = 2;
var b = 3;
var c = "hello";

console.log(a,b,c);
console.log(a+b+c);
```

```
// Case 2 : Integers
var a = 2;
var b =. 3;
console.log(a+b);
console.log(a,b);


// Case 3 : Strings
var a = "Hello";
var b = "World";

console.log(a+b);


// Case 4 : Integer with Strings
var a = 2;
var b = "hello";

console.log(a,b);
console.log(a+b);


// Case 5 : "\n"
var a = 2;
var b = "hello";

console.log(a,"\n",b);
```

Code 7 : Mom wants to make Palak Paneer , So he send sunny to the shop to buy palak and paneer.

Since , She asked for palak paneer . In this case, both items palak and paneer is required to make palak paneer, if any of the item is not available in the shop then it is not possible to make palak paneer dish.

```
var palak_availaible = false;
var paneer_availaible = false;

if(paneer_availaible && palak_availaible)
{
  console.log("Today, we will have a party");
}
else
{
```

```
  console.log("No Party");
}
```

OR Operator ||

If any of the statement is true , then the result will be true

For Example : DriveZy is a Renting bike service Startup, If you want to rent a bike then you need to submit any of the Identity Document

Aadhar Card or PAN Card or License or Voter id Card

Aadhar Card	PAN Card	License	Voter ID card	Result
True	True	True	True	**True**
True	True	True	False	**True**
True	True	False	False	**True**
True	False	True	True	**True**
False	False	False	False	**False**

and many more cases are possible

Observation :

1. If any of the case is true then the final result will be true.
2. If all the cases are false, then only the result will be false.

For Example : Masai ask for documents After Msat in the documentation phase, either submit the 12th Mark Sheet or Diploma

12th Marksheet	Diploma	Admission Result

True	True	True
True	False	True
False	True	True
False	False	False

- Show in console

Code 8 : OR Operator

```
var a = true;
var b = true;

var c = a || b;
console.log(c);

a = true;
b = false;
console.log(a||b);

a = false;
b = true;
console.log(a||b);

a = false;
b = false;
console.log(a||b);
```

Code 9 : [Student Task] OR Operator

1. true || false || true
2. false || true || false
3. false || false || true

Code 10 : Mom wants to prepare something for dinner, she decide that either the will make Potato or Paneer , So she send sunny to the shop to buy potato or paneer.

Since , Either she will prepare potato or panner in the dinner . In this case, if any of the item is available in the shop then it is possible to prepare dinner.

```
var potato_availaible = true;
var paneer_availaible = false;

if(potato_availaible || paneer_availaible)
{
  console.log("Dinner : Possible");
}
else
{
  console.log("Dinner : Not Possible");
}
```

Code 11 : Marriage Problem

Male : age≥21

Female : age≥18

```
var gender = "female";
var age = 18;

if((gender == "male" && age>=21) || (gender == "female" && age>=18))
{
  console.log(gender,": Can get Married");
}
else
{
  console.log(gender,": Can't get married");
}
```

Switch Case

Whenever we have multiple options and we have a choice .

For Ex : ATM Machine , we have multiple options of Deposit, Withdraw, Change Pin , others

- Every option is connected to some code
  - Deposit ——————————> Code 1 [To deposit the money]
  - Withdraw ——————————→ Code 2
  - Change Pin ——————————> Code 3
  - Default ——————————————> Code 4

// pic

In switch case, there are multiple cases, and with each case some code is connected.

Code 12 : Day Schedule

```
var option = 3;

switch(option)
{
  case 1 :
           console.log("Day 1 : Scrum + Coding");

  case 2 :
           console.log("Day 2 : Scrum + Coding + Skillathon");

  case 3 :
           console.log("Day 3 : Scrum + Skillathon + Standups");

  default :
              console.log("Holiday");

}
```

- On choosing the option in above code, it will output the code present corresponding to the given option and also print all the output of all the cases which present below the chosen option.
- To avoid this, we will use break

Code 13 : Day Schedule [with Break]

```
var option = 3;

switch(option)
{
  case 1 :
           console.log("Day 1 : Scrum + Coding");
           break;

  case 2 :
           console.log("Day 2 : Scrum + Coding + Skillathon");
           break;

  case 3 :
           console.log("Day 3 : Scrum + Skillathon + Standups");
           break;

  default :
           console.log("Holiday");
           break;
}
```

While Loop

Increment & Decrement

- The increment and decrement operators in JavaScript will add one (+1) or subtract one (-1).

Aa Title	Increment	Decrement
Untitled	`var a = 5; a = a + 1; console.log(a);`	`var a = 5; a = a - 1; console.log(a);`

- Javascript provides another way of incrementing and decrementing the variable i.e ++/--.

Using ++/-- After the Operand

- When you use the increment/decrement operator after the operand, the value will be returned before the operand is increased/decreased.
- In simple terms, Postpone the operation for later instead of it first printing it.
- This is known as Postfix Increment/Decrement.

Aa Title	Increment(++)	Decrement(--)
Untitled	`var a = 1 console.log(a++); console.log(a).`	`var a = 1; console.log(a--); console.log(a);`

Using ++/-- Before the Operand

- When you use the increment/decrement operator after the operand, the value will be increased/decreased returned before the operand is returned.
- Preponing the operation.
- This is knowns as Prefix Increment & Decrement.

Aa Title	Increment(++)	Decrement(--)
Untitled	`var a = 1; console.log(++a); console.log(a).`	`var a = 1; console.log(++a); console.log(a);`

Examples of Prefix and Postfix

Aa Title	Prefix	Postfix
Untitled	`var a = 10; var c = a++; console.log(a); // 11 console.log(c);  // 10`	`var a = 10; var c = ++a; console.log(a); console.log(c);`

Student Task

Code 1 : Predict the output

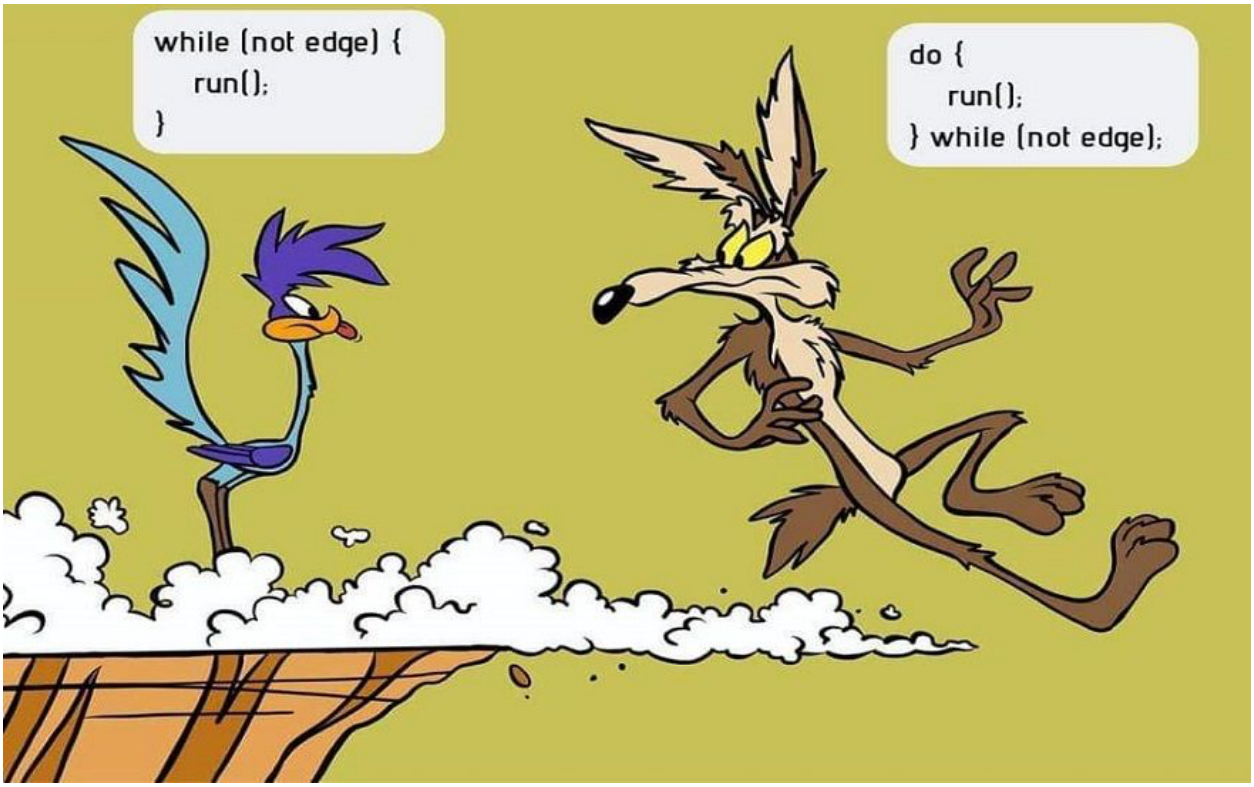
```
var a = 10;
++a;
var b = 10;
b++;
console.log(a)
console.log(b);
```

Student Task

Code 2 : Predict the output

```
var a = 10;
var b = 10;
++a;
b++;
console.log(a);
console.log(b);
```

Loop



Song Library

- On Internet, When we listen to a song. There is an option of listening to the song in the loop, it will play the song again and again when it reaches to end.

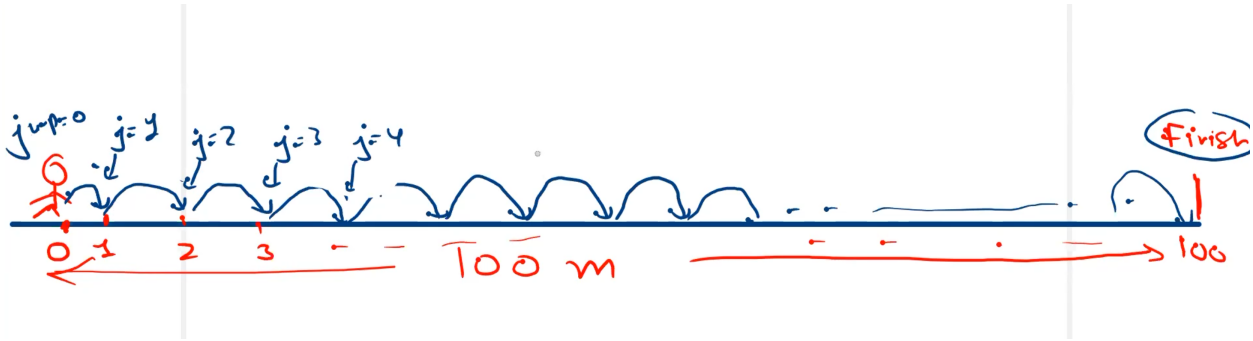
Guests

- There are 10 guests coming to my home, After 2-3 days they decided to leave their home.
- They all have the train on the same day and at the same time.
- I need to drop them at the railway station but I have one bike which can only take one person at a time.
- In this case, I need to drop each guest one by one.
- Taking the First guest to the railway station, dropping them and arrive back and follow the same procedure again and again till the end.

While Loop

- The while loop begins with a condition and it is written similar to an if statement. The inner parenthesis is the condition.
- As long as the condition is true, it will continue to execute the statement(s).
- To stop the loop, the condition must eventually become false.
- A common condition is to have a variable be less than or greater than compared to a number.
- Within the statements, that variable will be incremented or decremented depending on the condition.
- Each time the loop is executed, the variable will change and eventually become larger or less than the number in the condition, stopping the loop

Let's try to understand the Loop Variables : Marathon Analogy



Case 1 : Given a track of 100 m, Hari train himself for a long Marathon of 100 meter. Hari standing at 0th position and he needs to cover 100 meter distance.

Hari will make 1 meter jump at a time.

Lets take Hari position as position = 0

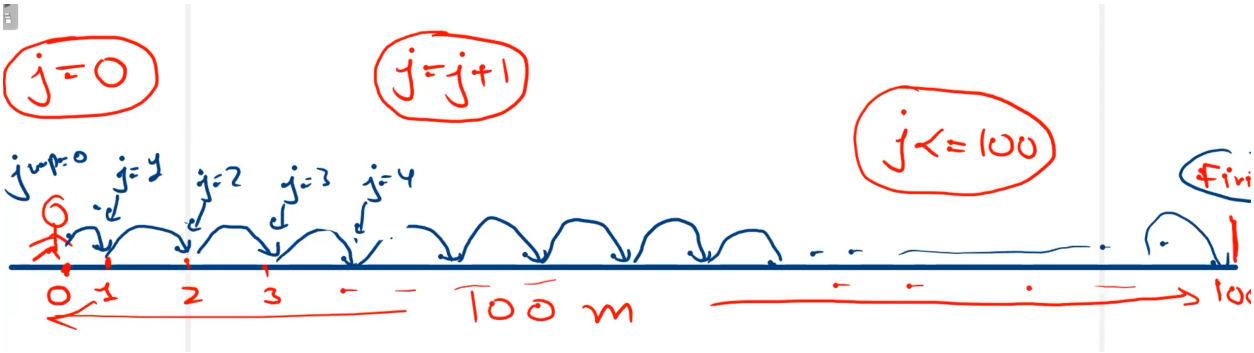
After 1st jump, Hari will be at position = 1 meter

After 2nd jump, Hari will be at position = 2 meter

.....

After 100th jump, Hari will be at position = 100 meter

Observation



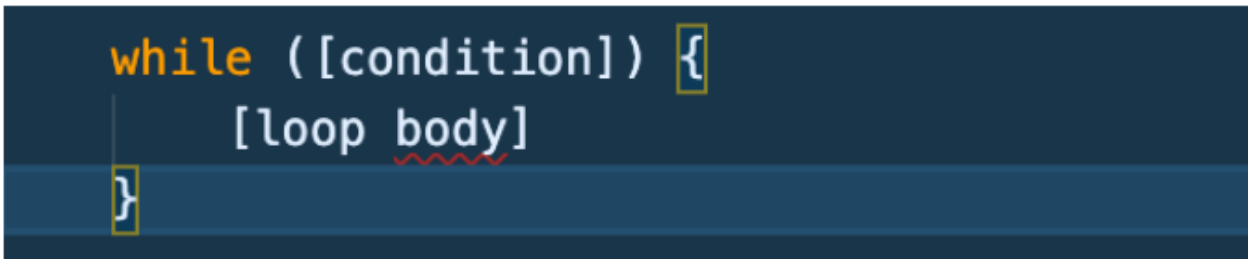
the value of position is going as follows **position** = 0,1,2,3.....100

1. The initial value of **position = 0**
2. The loop is running till **position ≤ 100**
3. At every point, Hari is making a jump of 1

For Loops, 3 things are important :

1. Starting Point : position = 0
2. How long jump : Jump of 1 meter
3. Till When : position≤100 meter

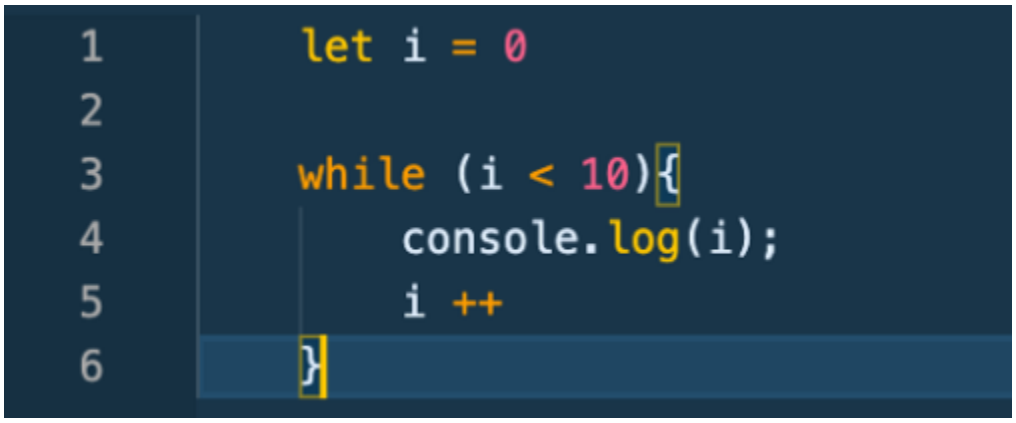
Syntax of While Loop



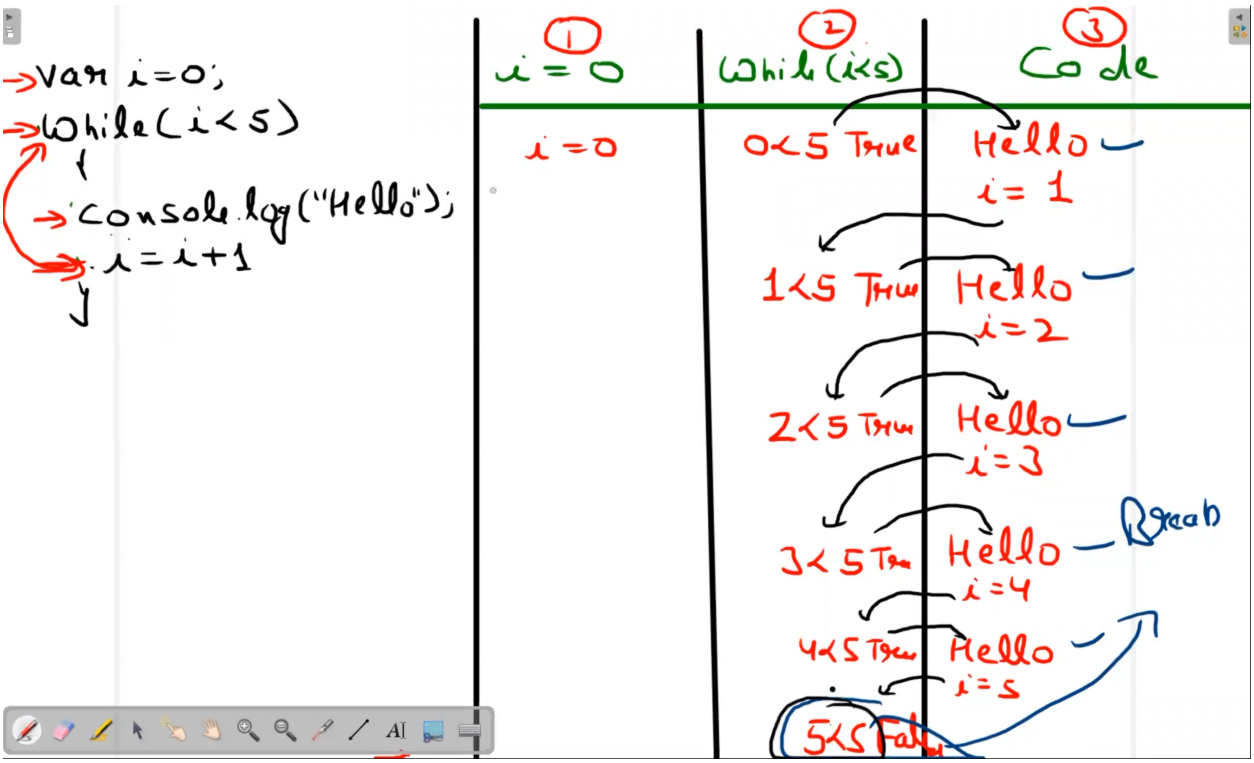

```
Starting Point
While ( Till When )
{
    How long Jump at a time?
```

```
}

------------------------------------------------------------------------

Initialization
While ( Condition )
{
    Increment/Decrement
}
```

```
var position = 0
while(position <= 100)
{
  position = position + 1;
}
```



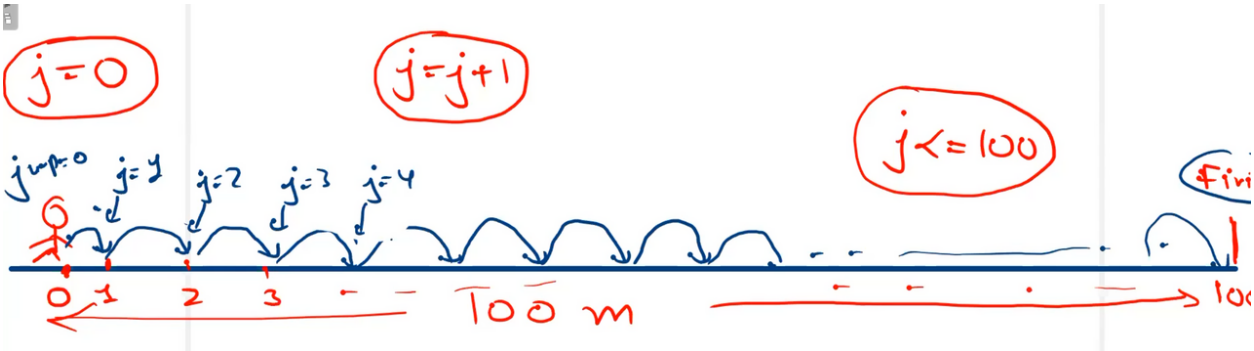
Understanding DRY RUN




```
var i = 0;
while(i<=5)
{
  console.log("hello");
  i = i + 1;
}
```

While Loop Examples

Code 3 : Loop from 1 to 100 [1 meter jump at a time]

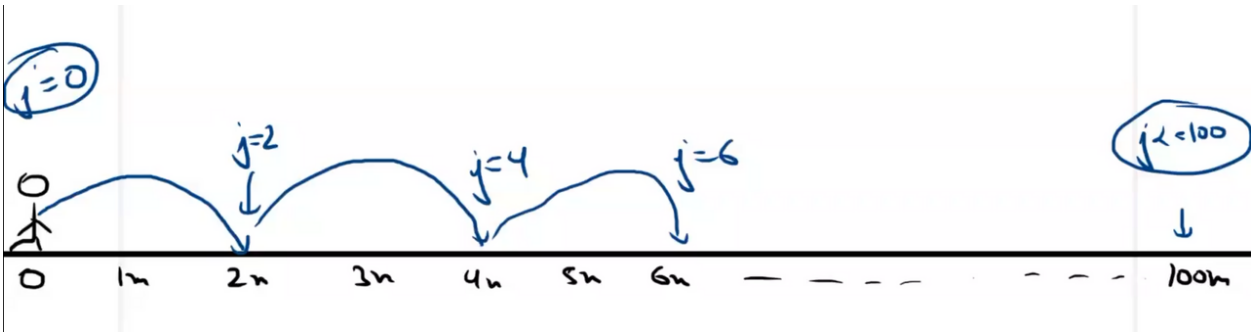


```
var position = 0;

while(position <= 100)
{
  position = position + 1;
  console.log("Current Position ",position);
}
```

Code 4 : Infinite Loop

```
while(true)
{
  console.log("Hello Masai");
}
```

Code 5 : Loop from 1 to 100 [2 meter jump at a time]




```
var position = 0;
while(position<100){
  console.log("Current Position ",position);
  position = position + 2;
}
```

Student Task

Code 6 : Loop from 1 to 100 [15 meter jump at a time]

// PIC of image track

```
var position = 0;
while(position<100){
  console.log("Current Position",position);
  position = position + 15;
}
```

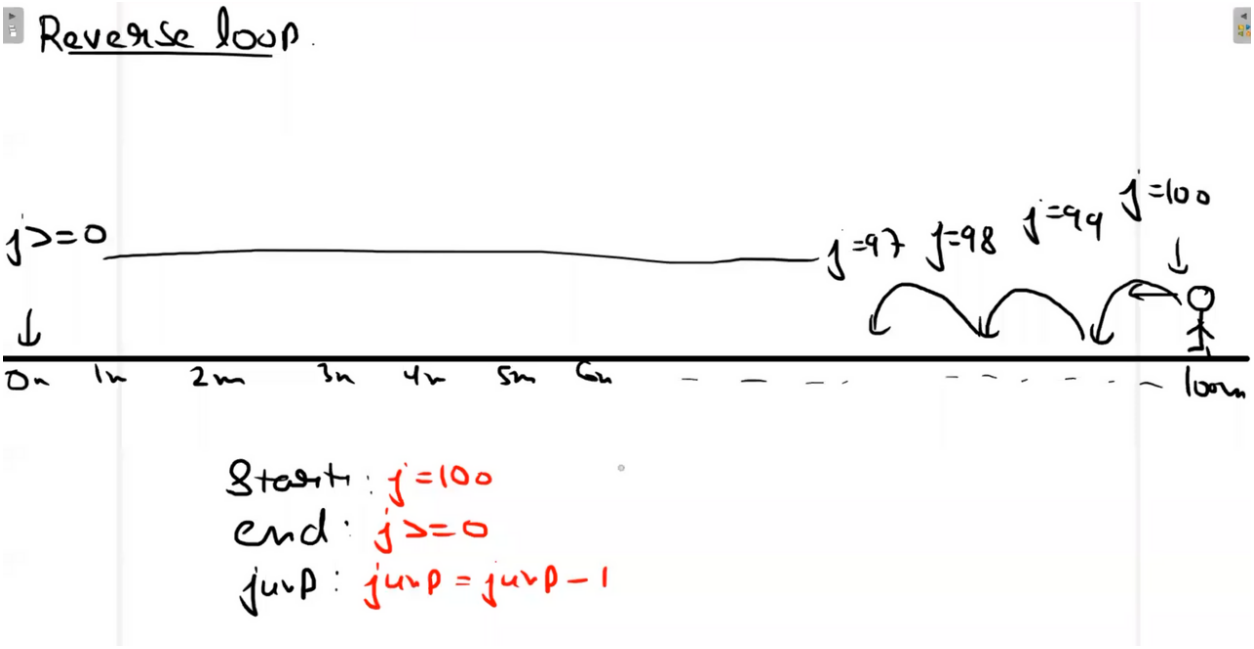
Note : != and < behave differently

Student Task

Code 7 : Loop from 35 to 100 [3 units jump at a time]

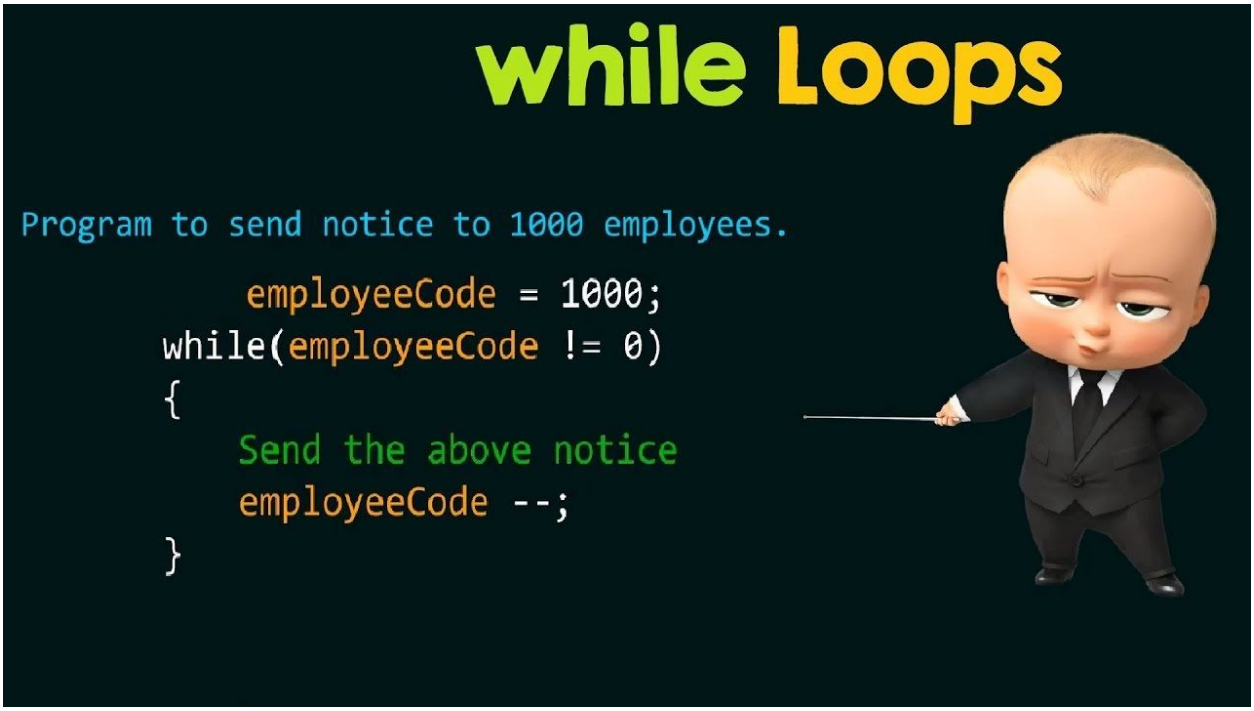
```
var position = 35;
while(position<100){
  console.log("Current Position",position);
  position = position + 3;
}
```

Code 8 : Reverse Loop from 100 to 0 [1 units jump at a time]



```
var position = 100;
while(position>=0){
  console.log("Current Position",position);
  position = position - 1;
}
```

Sending Notice to 1000 Employees



Break

Guest Analogy

- There are 10 guests coming to my home, After 2-3 days they decided to leave their home.
- They all have the train on the same day and at the same time.
- I need to drop them at the railway station but I have one bike which can only take one person at a time.
- In this case, I need to drop each guest one by one.

- Taking the First guest to the railway station, dropping them and arrive back and follow the same procedure again and again till the end.
- Suppose I took the First Guest and dropped him to the Railway station and come back.
- Again I took the Second Guest and follow the same.
- Now, Next I took the third guest to Railway station and found that Train has gone.

So, Will I continue the above procedure or stopped it ?

Obviously, I will stop it and wait for tomorrow.

Code 9 : Loop from 0 to 10 (using break)

```
var guest=0;

while(guest<=10)
{
  console.log("Guest",guest);

  if(guest == 3)
  {
    break;
  }

  guest++;
}
```

Student Task

Code 10 : Predict the output

```
var count=0;
while(true)
{
  console.log("Hello");
  count++;
  ++count;

  if(count>5)
  {
    break;
  }
  count--;
}
```

Continue

Guest Analogy

- There are 10 guests coming to my home, After 2-3 days they decided to leave their home.
- They all have the train on the same day and at the same time.
- I need to drop them at the railway station but I have one bike which can only take one person at a time.
- In this case, I need to drop each guest one by one.
- Taking the First guest to the railway station, dropping them and arrive back and follow the same procedure again and again till the end.
- Suppose I took the First Guest and dropped him to the Railway station and come back.
- Again I took the Second Guest and follow the same.
- Suppose the third guest is Sick, In that case I will skip him.
- and I will continue with the fourth guest and follow the same procedure.

Code 11 : Loop from 0 to 10 (using continue)

```
var guest=0;
while(guest<=10)
{
  console.log("Guest",guest);

  if(guest == 3)
  {
    continue;
  }

  guest++;
}
```

Code 12 : Find Sum of 1 to 10

```
**Problem** :
// Sum of 1 to 10
// 1 + 2 + 3....... + 10

var i = 1;
var sum = 0;

while(i<=10)
{
  sum = sum+i;
  i++;
}

console.log(sum);
```

For Loops

For Loops

- Let’s say you want to display Masala Dosa one time then you will print do the console.log("Masala Dosa") one-time.
- Similarly, If it is two then you will write console.log twice
- Let’s say you want to display it 100 times. Without some sort of loop in your code, we would probably have to write the same line of code 100 times.

A *for-loop* can help us to do so by running the same code repeatedly under certain conditions.

Syntax

```
for ([initialization]; [condition]; [iteration]) {
  [loop body]
}
```

1. Initialization : Decides starting point of a loop
2. Condition : Condition is checked before the execution of every iteration. If it evaluates to true, the loop's statement is executed. If it evaluates to false, the loop stops.
3. Iteration : Iteration is used to affect your counter.It can be increment / decrement.
4. Loop Body : The loop body repeats the code as long as the condition part is TRUE.

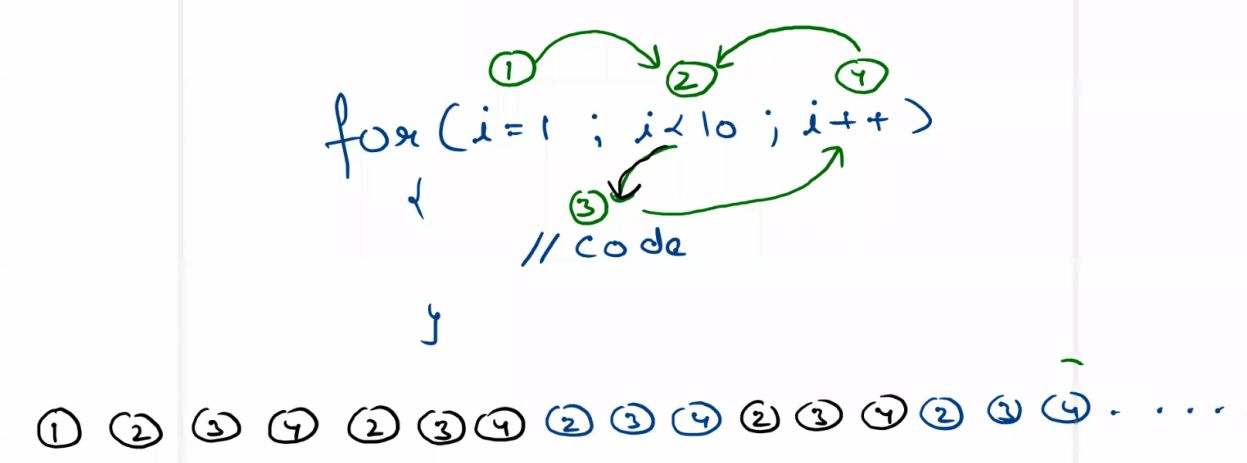
Three Ways of Writing For Loop

1. **for** (*initialize*; *condition*; *increment*)***;***
2. **for** (*initialize*; *condition*; *increment*) ***single statement;***
3. **for** (*initialize*; *condition*; *increment*) { ***multiple; statements;*** }

Comparing For Loop Vs While Loop

```
for ([initialization]; [condition]; [iteration]) {
  [loop]
}

while ([condition]) {
  [loop]
}
```

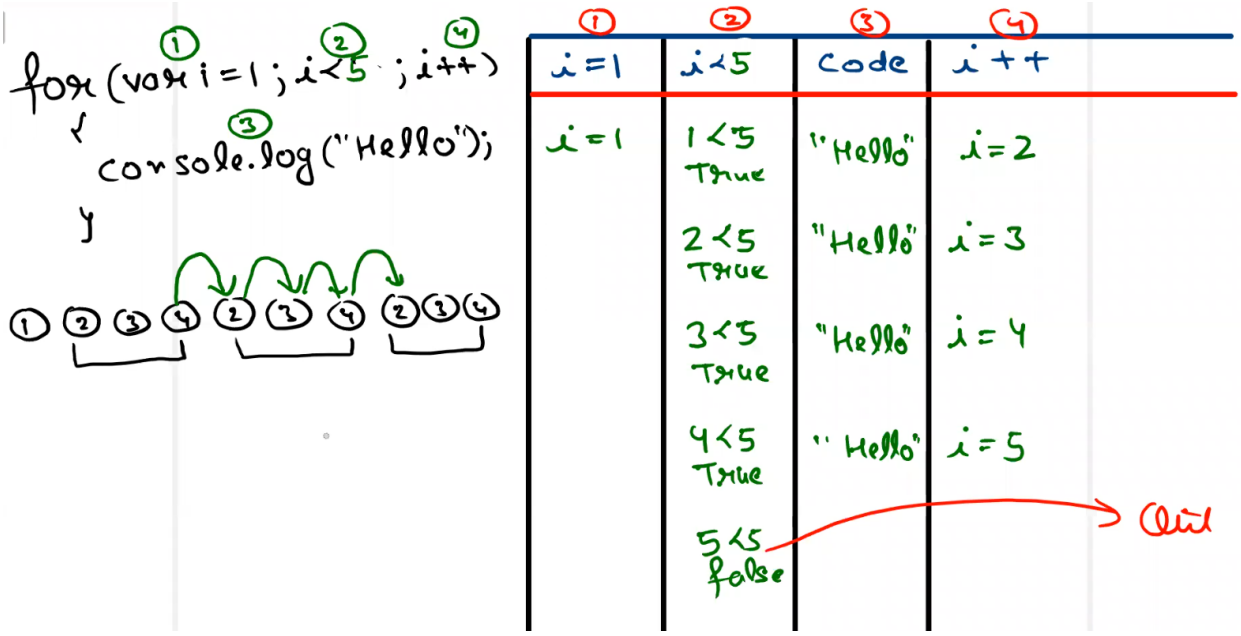
The sequence of Execution of For Loop



1. Initialization -> Condition -> Loop Body -> Iteration -> Condition -> Loop Body -> Iteration and so one
2. Initialization denoted as 1, Condition denoted as 2, Loop Body denotes as 3, Iteration denoted as 4.
3. Sequence of Execution will be : 1 -> 2 -> 3 -> 4 -> 2-> 3 -> 4 -> 2 -> 3 -> 4 and so on

Examples of For Loop with Dry Run

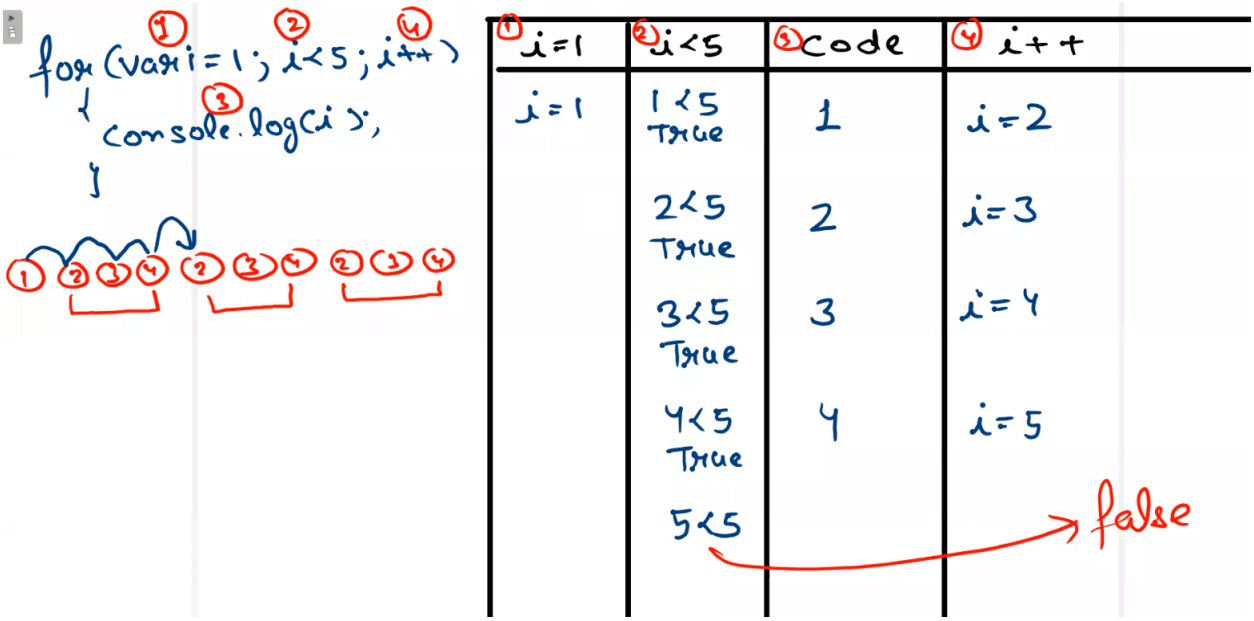
Example 1: Print Hello 5 times.



Code 1 : Print Hello 5 times.

```
for(var i=0; i<5; i++)
{
  console.log("Hello");
}
```

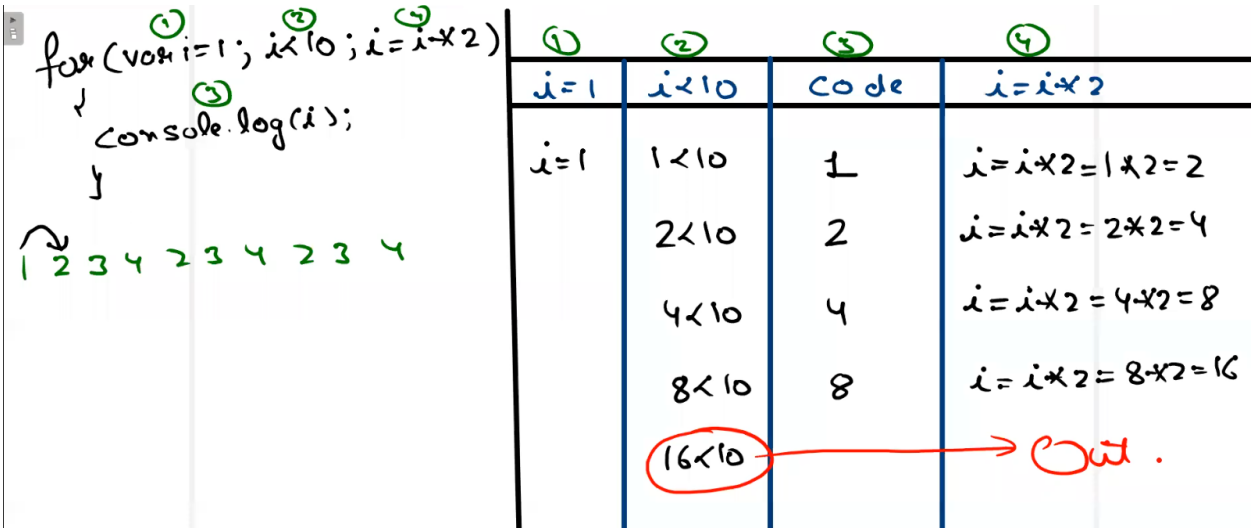
Example 2: Print Values from 1 to 5



Code 2 : Print Values from 1 to 5

```
// Ist way : Output on new line
for(var i = 1; i<=5; i++){
  console.log(i);
}

// IInd way : Output on single line
var bag="";
for(var i = 1; i<=5; i++){
  bag = bag + i + " ";
}
console.log(bag);
```

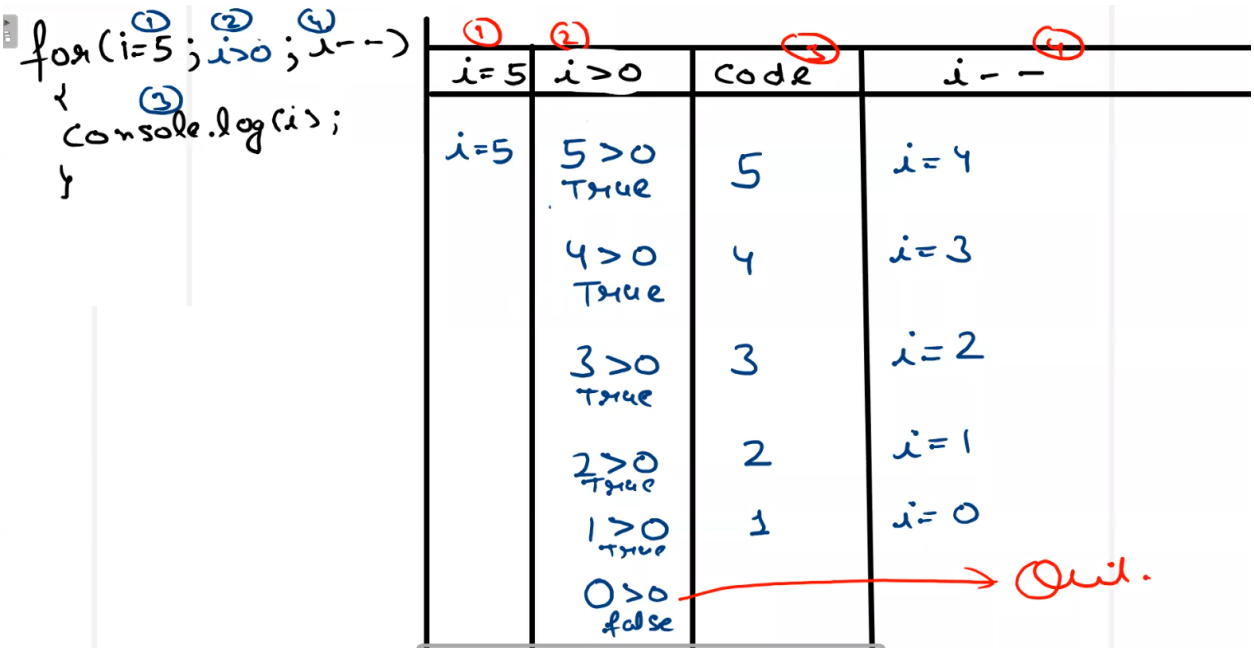
Example 3: Print multiple of 2 values from 1 to 10



Code 3 : Print multiple of 2 values from 1 to 10

```
// Ist way : Output on new line
for(var i = 1; i<=10; i=i*2){
  console.log(i);
}

// IInd way : Output on single line
var bag="";
for(var i = 1; i<=10; i=i*2){
  bag = bag + i + " ";
}
console.log(bag);
```

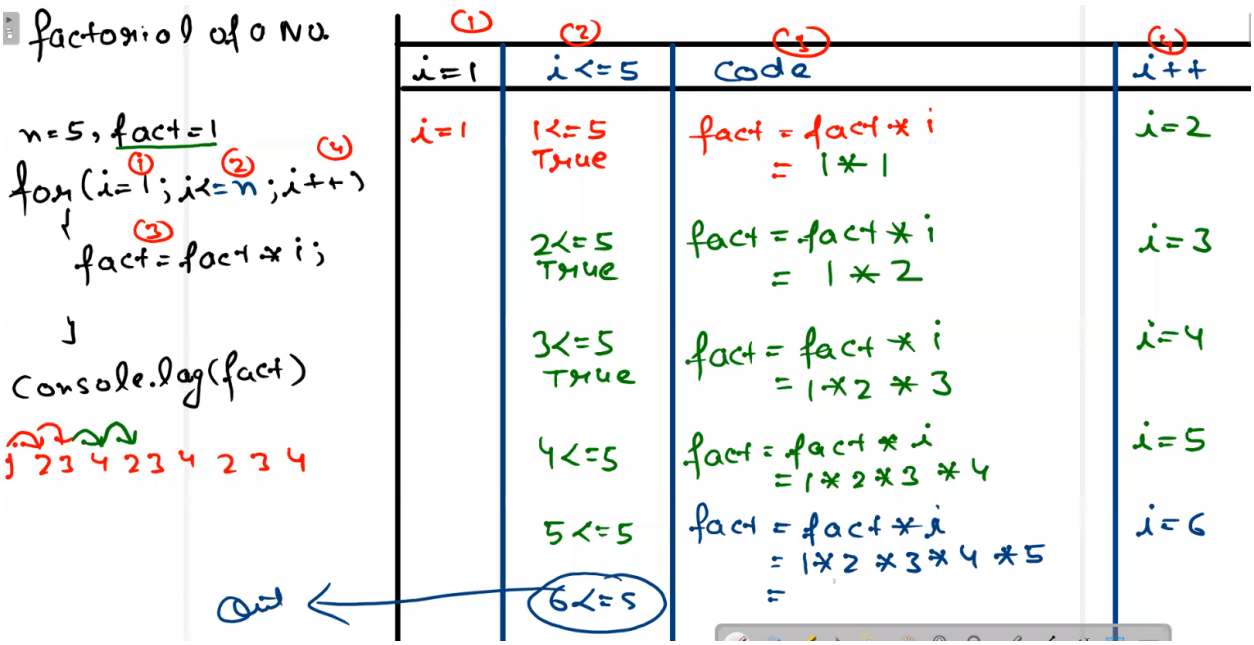
Example 4: Reverse Loop from 5 to 1



Code 4 : Reverse Loop from 5 to 1

```
var bag = "";
for(var i = 5; i>0; i--){
  bag = bag + i + " ";
}
console.log(bag);
```

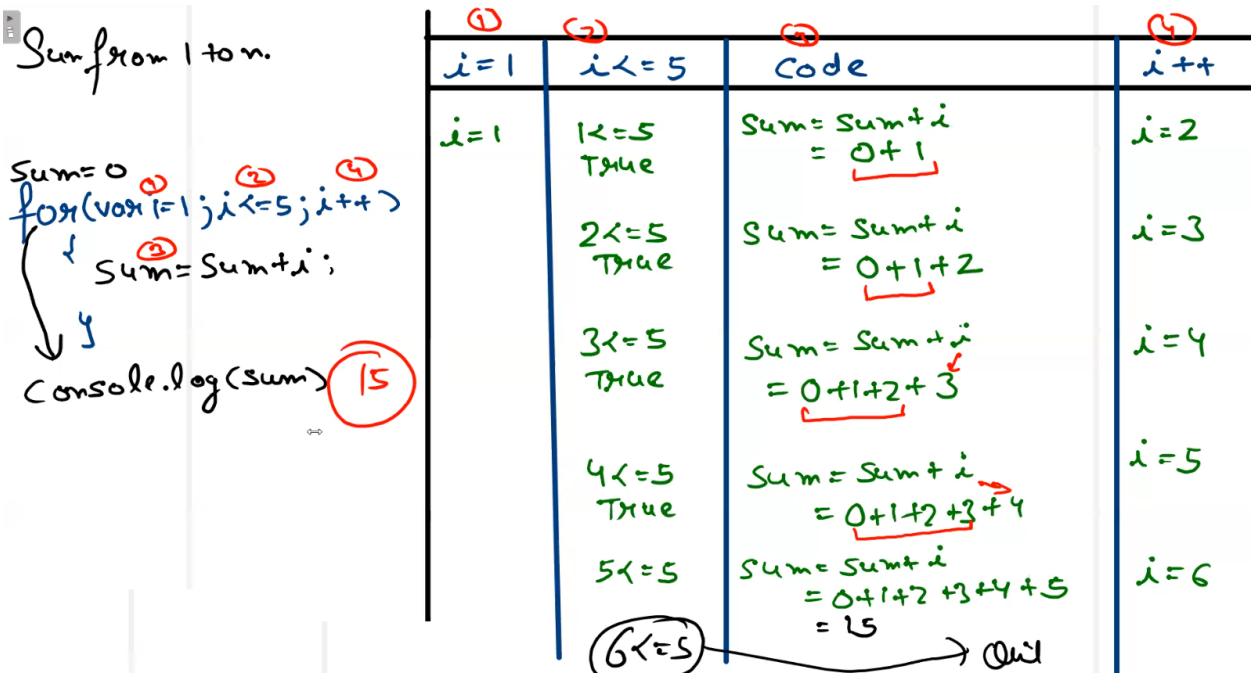
Example 5: Factorial



Code 5 : Factorial

```
var fact = 1;
for(var i=1; i<=5;i++)
{
  fact = fact * i;
  console.log(fact);
}
```

Example 6: Find Sum 1 to N



Code 6 : Find Sum 1 to N

```
var N = 5;
var sum = 0;
for(var i = 1; i<=N; i++){
  sum = sum + i;
}
console.log(sum);
```

Break

Guest Analogy

- There are 10 guests came to my home, After 2-3 days they decided to leave.
- They all have the train on the same day and at the same time.
- I need to drop them at the railway station but I have one bike which can only take one person at a time.
- In this case, I need to drop each guest one by one.

- Taking the First guest to the railway station, dropping them and arrive back and follow the same procedure again and again till the end.
- Suppose I took the First Guest and dropped him to the Railway station and come back.
- Again I took the Second Guest and follow the same.
- Now, Next I took the third guest to Railway station and found that Train has gone.

So, Will I continue the above procedure or stopped it ?

Obviously, I will stop it and wait for tomorrow.

Code 7 : Loop from 1 to 10 (using break). Using console.log before break statement

```
for(var guest=1; guest<=10; guest++)
{
  console.log("guest ",guest,"got the train");

  if(guest == 3){
    break;
  }
}
```

Code 8 : Loop from 1 to 10 (using break). Using console.log after break statement

```
for(var guest=1; guest<=10; guest++)
{
  if(guest == 3){
    break;
  }
  console.log("guest ",guest,"got the train");
}
```

Continue

Guest Analogy

- There are 10 guests coming to my home, After 2-3 days they decided to leave their home.

- They all have the train on the same day and at the same time.
- I need to drop them at the railway station but I have one bike which can only take one person at a time.
- In this case, I need to drop each guest one by one.
- Taking the First guest to the railway station, dropping them and arrive back and follow the same procedure again and again till the end.
- Suppose I took the First Guest and dropped him to the Railway station and come back.
- Again I took the Second Guest and follow the same.
- Suppose the third guest is Sick, In that case I will skip him.
- and I will continue with the fourth guest and follow the same procedure.

Code 9 : Loop from 1 to 10 (using Continue). Using console.log before continue statement

```
for(var guest=1; guest<=10; guest++)
{
  console.log("guest ",guest,"got the train");

  if(guest == 3){
    continue;
  }
}
```

Code 10 : Loop from 1 to 10 (using Continue). Using console.log after continue statement

```
for(var guest=1; guest<=10; guest++)
{
  if(guest == 3){
    continue;
  }
  console.log("guest ",guest,"got the train");
}
```

Code 11 : Predict the output.

```
var count = 1;
for(var i = 1; i<10; i++)
{
  count++;

  if(i==5){
    continue;
  }
}
console.log(count);
```

Code 12 : Predict the output.

```
var count = 1;
for(var i = 1; i<10; i++)
{
   if(i==5){
    continue;
  }
  count++;
}
console.log(count);
```

Nested Loops

Nested Loop

- A nested loop is a loop within a loop.
- Let's try to understand with the analogy
  - Suppose you went to a Gol Gappa Shop and you ate 10 gol-gappas in a sequence. You can consider this as a loop where you have eaten 10 gol gappas in a sequence.
  - In another scenario, Suppose you went to a Gol Gappas Shop with the 5 other family members. Each Member of the family ate 10 gol-gappas. You can consider this as a loop of 10 gol-gappas which run 5 times because of the 5 family members.

Dry Run of problem

Code 1 : print Hello in vertical order using nested loop.

```
for(var i=0;i<2;i++)
{
  for(var j=0; j<3; j++){
    console.log("Hello");
  }
}
```

Code 2 : print Hello in horizontal order using nested loop

```
for(var i=1;i<=2;i++)
{
  var bag="";
  for(var j=1; j<=3; j++){
    bag = bag + "Hello ";
```

```
  }
  console.log(bag);
}
```

Father-Son Marathon

There was a father who trains his son for next olympics for running competition.

Every day Father use to give some target’s to his son

Code 3 : Father gave the son a target , of completing 10 sets . Each set contains 10 Rounds of a field.

```
for(var father=1; father<=8; father++)
{
    for(var son = 1; son<=10; son++)
    {
      console.log("Father count",father,",Son completed ",son);
    }
}
```

Father-Son planting trees

Once upon time, There was a farther and son who were farmers. They once decided that they do plantation of trees in their field. Since Father is very old, he unable to do that much work whereas son is pro-active, that’s why father responsibility is to make sure how many fields are done whereas son has the responsibility of putting the seeds in the field.

Code 4 : Father has 5 fields . Each fields son needs to put 10 seeds

```
// * * * * *
// * * * * *
// * * * * *
// * * * * *
// * * * * *

for(var father=1; father<=5; father++)
{
  var bag = "";
  for(var son=1; son<=10; son++)
  {
    bag = bag + "*";
  }
```

```
  console.log("Field",father,bag);
}
```

Code 5 : Father has 5 fields. Son needs to put the seeds in increasing order.

Field 1 → 1 seed

Field 2 → 2 seed

Field 3 → 3 seed

Field 4 → 4 seed

Field 5 → 5 seed

```
*
* *
* * *
* * * *
* * * * *
```

```
for(var father=1; father<=5; father++)
{

  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag + "* ";
  }
  console.log(bag);
}
```

Code 6 : Print the below pattern

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
for(var father = 1; father<=5; father++)
{
  var bag = "";
```

```
  for(var son = 1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

Code 7 : Father has 5 fields. Son needs to put the seeds in decreasing order.

Field 1 → 5 seed

Field 2 → 4 seed

Field 3 → 3 seed

Field 4 → 2 seed

Field 5 → 1 seed

```
* * * * *
* * * *
* * *
* *
*
```

```
for(var father=5; father>=1; father--)
{
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag +"* ";
  }
  console.log(bag);
}
```

Code 8 : Print the below reverse pattern

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
for(var father=5; father>=1; father--)
{
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

Code 9 : Combining Code 6 and Code 8 form a pyramid.

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
for(var father = 1; father<=5; father++)
{
  var bag = "";
  for(var son = 1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}

for(var father=4; father>=1; father--)
{
  var bag = "";
  for(var son=1; son<=father; son++)
  {
    bag = bag + son+" ";
  }
  console.log(bag);
}
```

Nested Loop with While

Code 10 : Use While Loop. Print the below pattern

```
**********
**********
**********
**********
**********
**********

var father=1;
while(father<=6)
{

  var son=1;
  var bag = "";
  while(son<=10)
  {
    bag=bag+"*";
    son++;
  }
  console.log(bag);

  father++;
}
```

Break and Continue

Break

Sultan and Bahubali were the good friends , but Sultan was the king whereas Bahubali does n't have any kingdom. Later Sultan gave one part of his empire to Bahubali with a condition that Bahubali will never try to cross the status of Sultan and if he tries that then he will attack on the Prithviraj clan.

Code 11 : Using Break

```
for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {

    if(bahuballi>sultan)
    {
      break;
    }

    console.log("sultan=",sultan," bahuballi=",bahuballi);
```

```
  }
}
```

In the above code, whenever the value of Bahubali become greater than Sultan, At that point the inner loop of bahuballi will break [It means sultan attacked on his clan because Bahubali betray him, by not following his conditions]

Continue

The below code gives the same output as the above code but the only difference is on break is that the execution will stop completely but in case of continue the process will keep on skipping the step and execution will end only once the loop will done.

Code 12 : Using Continue

```
for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {

    if(bahuballi>sultan)
    {
      continue;
    }

    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
}
```

Code 13 : Break vs Continue . Predict the Output

```
**BREAK**

for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {

    if(bahuballi == sultan)
    {
      break;
    }

    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
```

```
}

**CONTINUE**

for(var sultan=1; sultan<=10; sultan++)
{
  for(var bahuballi=1; bahuballi<=10; bahuballi++)
  {
    if(bahuballi == sultan)
    {
      continue;
    }
    console.log("sultan=",sultan," bahuballi=",bahuballi);
  }
}
```

Strings

Strings

- The string is a group of characters
- It can include a-z, A-z, 0-9, and also all special characters like #,@,$, etc
- Each character has an index, Starting from 0 to the length of the string.

Need of Strings

- Lots of information we stored, it actually stored as a string
- For Example, The name of the product, Pincode, and mobile number also, Since we will not perform any mathematical operation on mobile numbers that's why we considered it as a string.

How to declare a String?

```
String s = "Masai School"

There is a total of 12 characters in this string.
```

Code 1: Declare a string variable and print it.

```
var name = "Masai";
console.log(name);

console.log(name[0]);   // M
console.log(name[1]);   // a
console.log(name[2]);   // s
console.log(name[3]);   // a
console.log(name[4]);   // i
console.log(name[5]);   // undefined
```

Code 2: Find the length of the String.

```
var name = "Jantar Mantar";
console.log(name.length);         // 13
```

Real-world use of String

Code 3: Find whether the user enters the valid length password of at least 6 character.

```
var password = "vb";

if(password.length < 6)
{
  console.log("Invalid : Your Password must be atleast 6 characters long");
}
else
{
  console.log("Valid Password");
}
```

Loop in Strings

Code 4: Run loop and print each character of String.

```
var name = "Masai School";
for(var i = 0; i<name.length; i++)
{
  console.log(name[i]);
}
```

Code 5: Run loop on the string and add each character to the third variable and print that variable.

```
var name = "Masai School";
var bag = "";
for(var i = 0; i<name.length; i++)
{
  bag = bag + name[i];
}
console.log(bag);
```

Arrays vs Strings

- We can use an array to store the sequence of characters.

Code 6: Store “Masai” in String and array.

```
var name1 = "Masai";
console.log(name1);
console.log(name1[0]);

var name2 = ["M","a","s","a","i"];
console.log(name2);
console.log(name2[0]);
```

Strings are immutable

- Once the string is declared and initialized, it cannot be updated later.

Code 7: Update Character in String

```
var name = "Masai";
name[0] = "N";
console.log(name);    // Masai
```

- **Let’s use an array to update the string**

```
```

Code 8: Update Character in array

```
var name = ["M","a","s","a","i"];
name[0] = "N";
console.log(name);   // Nasai
```

We can conclude that strings are immutable. Once it is created, it cannot be updated later but in the array it is possible.

Update Strings

- We already know that we can not update the string but we can update the array.

```

```

Code 9: Update String using array and third variable. [First Method]

```
**I Way**

var name = "Masai";
var name2 = []

for(var i=0; i<name.length; i++)
{
  name2.push(name[i]);
}

name2[0] = "N";
var bag = "";
for(var i=0; i<name2.length; i++)
{
  bag = bag + name2[i];
}
console.log(bag);
```

Code 10: Update String using array and third variable. [Second Method]

```
**II Way**

var name = "Masai";
var output = "";

for(var i=0; i<name.length; i++)
{
  if(i==0)
  {
    output = output + "N";
  }
  else
  {
    output = output + name[i];
  }
}

console.log(output);     // Nasai
```

Remove char in Strings

- loop in the given string and don’t add that character which you want to remove otherwise add all.

Code 11: Remove a char from String

```
var name = "Masai";
var output = "";

for(var i=0; i<name.length; i++)
{
  if(name[i] != "s")
  {
    output = output + name[i];
  }
}
console.log(output);
```

Problems in Strings

Code 12: Count the names starting with N or n

```
var names = [”Nobita”, “Naruto”, “Noddy”, “Shinchan”, “Oswald”];
var count = 0;

for(var i=0; i<names.length; i++)
{
  var name = names[i];
  if(name[0] == "N" || name[0] == "n")
  {
    count++;
  }
}
console.log(count);
```

Code 13: Count the names which contain A in them.

```
var names = [”Nobita”, “Naruto”, “Noddy”, “Shinchan”, “Oswald”];
var count = 0;

for(var i=0; i<names.length; i++)
{
  var name = names[i];
  for(var j = 0; j<name.length; j++)
  {
    if(name[j]=='a' || name[j]=='A')
    {
      count++;
      break;
    }
  }
}
console.log(count);


Student Task
```

Array

How do we store information?

- We use variables to store the data.
- For Example: If I want to store the name of a student, In that case, I will use a variable
  - `var name = “Varun”;`

But What if, I want to store the names of 5 students

```
name1 = “Prateek”;
name2 = “Nrupul”;
name3 = “Yogesh”;
name4 = “Aman”;
name5 = “Albert”;
```

To store 5 names, we need to declare 5 variables.

Isn’t possible that one variable will contain all names?

Yes, It is possible with the array.

Arrays

- The array is a contiguous chunk of memory that can store multiple values.

Declaration of an array

- var arr = [];

If I want to store all 5 names in a single variable, then it is possible through an array

`var arr = [”Prateek”, “Nrupul”,”Yogesh”,”Aman”,”Albert”];`

- Each value is stored at some index.

- the array index starts from 0.

Code 1: Declare and print 3 students names using variables

```
var name1 = “Rahul”;
var name2 = “Shubham”;
var name3 = “Rishabh”;
console.log(name1);
console.log(name2);
console.log(name3);
```

Code 2: Declare and Print 3 students names using an array

```
var names = [”Rahul”, “Shubham”, “Rishabh”];
console.log(names[0]);
console.log(names[1]);
console.log(names[2]);
```

Code 3: Perform the following tasks :

1. **Create an array of vegetables**
2. **Store 3 vegetables**
3. **Print all the vegetables**

```
var vegetables = [”Tomato”, “Beans”, “Onion”];
console.log(vegetables[0]);
console.log(vegetables[1]);
console.log(vegetables[2]);
```

Note: Don’t write vegetables[3] that will give undefined.

How to find the length of an array?

- It means How many elements present in the array.
- Use the length function to calculate the length.

Code 4: Find the length of the vegetables array.

```
var vegetables = [”Tomato”, “Beans”, “Onion”];
console.log(vegetables.length);
```

Code 5: Perform the following tasks :

1. **Create an array price.**
2. **Store the prices of 3 products in the array**
3. **Print the price of the last product.**

Not a generic code :

```
var prices = [45, 71, 29];
console.log(prices[2]);
```

Generic Code :

```
var prices = [45, 71, 29];
last_index = prices.length-1;
console.log(prices[last_index]);
```

How to add elements in an array?

- push() function helps to insert the elements in an array.
- push() always inserts at the last.

Code 6: Insert 3 movie names in the arrays .

```
var items2 = [];
items2.push(”Bahuballi”);
items2.push(”Avengers”);
items2.push(”Spider Man”);
```

[Students Task]

Code 7: Perform the following tasks :

1. **Create an array superheroes**
2. **push 3 superheroes in the array**
3. **Print the array**

```
var superheroes = [];
superheroes.push(”bat man”);
superheroes.push(”super man”);
superheroes.push(”iron man”);
console.log(superheroes);
```

How to update the array?

- Suppose I want to change the first index value.
- `superheroes[0] = “Thor”;`

How to print all elements using Loop?

- Write a for loop from starting i = 0 to length of array - 1.
- print all the elements using a loop.

Code 8: print all the elements of the array using a loop.

```
var movies = [];

movies.push(”bat man”);

movies.push(”super man”);

movies.push(”iron man”);

for(var i = 0; i<movies.length; i++)
{
  console.log(movies[i]);
}

Note : Don't run the loop till movies.length, It will show undefined for last index
because last index does n't exist for movies.
```

[Students Task]

Code 9: Perform the following tasks :

1. **Create an array of movies and actors**
2. **Print all the movies names with actors**

```
var movies = ["bahuballi", "Spider Man", "Iron Man", "Super Man"];
var actors = ["Prabhas", "Tom holland", "Robert Downey", "Henry Cavil"];
for(var i=0; i<movies.length; i++){
  console.log(movies[i]," - ",actors[i]);
}

Note : Length of both arrays should be same
```

How to remove elements from an array?

- To remove elements, we have a pop() function
- pop() function that will remove elements from the last.

Code 10: pop the last 2 elements from an array

```
var movies = [];

movies.push(”bat man”);

movies.push(”super man”);

movies.push(”iron man”);

movies.pop();
movies.pop();
console.log(movies);
```

[Students Task]

Code 11: Perform the following tasks :

1. **Create an array of 6 numbers**
2. **print the numbers array**

3. **delete last 3 numbers from that array**
4. **print the numbers array**

```
First Way

var numbers = [2,3,4,5,6,7];
console.log(numbers);
numbers.pop();
numbers.pop();
numbers.pop();
console.log(numbers);
```

```
Second Way

var numbers = [2,3,4,5,6,7];
console.log(numbers);
for(var i=1; i<=3; i++)
{
  numbers.pop();
}
console.log(numbers);
```

Arrays with Loop and Break

Code 12: Print the first 3 items in the array using a loop.

```
First Way

var movies = ["bahuballi", "Spider Man", "Iron Man", "Super Man"];
for(var i = 1; i<=3; i++)
{
  console.log(movies[i]);
}
```

```
Second Way [ Using Break ]

var movies = ["bahuballi", "Spider Man", "Iron Man", "Super Man"];
for(var i = 0; i<movies.length; i++)
{
  if(i==3)
```

```
  {
    break;
  }
  console.log(movies[i]);
}
```

```
```

Arrays with Loop and Continue

Code 12: Print all movies except the third movie.

```
var movies = ["bahuballi", "Spider Man", "Iron Man", "Super Man"];
for(var i = 0; i<movies.length; i++)
{
  if(i==2)
  {
    continue;
  }
  console.log(movies[i]);
}
```

Code 13: Print all movies except the third and fifth movies.

```
var movies = ["bahuballi", "Spider Man", "Iron Man", "Super Man","hulk","thor"];
for(var i = 0; i<movies.length; i++)
{
  if(i==2 || i==4)
  {
    continue;
  }
  console.log(movies[i]);
}
```

```
```

Code 14 : Print the last 3 items of an products array

```
**Approach 1 :**

var products = ["earphone", "headphones", "mic", "ipad", "cell phone", "laptop"];

var start = products.length - 3;
for(var i=start; i<products.length; i++)
{
  console.log(products[i]);
}
```

Above Approach 1 will fail, if suppose the products array is

var products = ["earphone", "headphones"]

```
**Approach 2 :**

var products = ["earphone", "headphones", "mic", "ipad", "cell phone", "laptop"];

var start = 0;
if(products.length>3)
{
  start = products.length - 3;
}

for(var i=start; i<products.length; i++)
{
  console.log(products[i]);
}
```

Code 15 : For Even Array, print the second half of the array

```
var names = ["A","B","C","D","E","F","G","H"];

var start = Math.floor(names.length/2);

for(var i=start; i<names.length; i++)
{
  console.log(names[i]);
}
```

Code 16 : For Even or Odd Array, print the second half of the array

```
var names = ["A","B","C","D","E","F","G","H","K"];

var start=0;

// Handle Even Array
if(names.length % 2 == 0)
{
  start = names.length/2;
}

// Handle Odd Array
else
{
  start = Math.floor(names.length/2);
}

for(var i=start; i<names.length; i++)
{
  console.log(names[i]);
}
```

Code 17 : For Even or Odd Array, print the first half of the array

```
var names = ["A","B","C","D","E","F","G","H","K"];

var start=0;
if(names.length % 2 == 0)
{
  end = names.length/2;
}

else
{
  end = Math.floor(names.length/2);
}

for(var i=0; i<end; i++)
{
  console.log(names[i]);
}
```

Code 18 : Given marks, find the total marks

```
var marks = [10, 15, 19, 20, 21];
var sum=0;
for(var i = 0; i<marks.length; i++)
{
```

```
  sum = sum+marks[i];
}

console.log(sum);
```

Code 19 : Find the sum of all subject marks and average also.

```
var subject_marks = [10, 15, 19, 20, 21];
var sum_marks = 0;

for(var i=0; i<subject_marks.length; i++)
{
  sum_marks = sum_marks + subject_marks[i];
}

var average = Math.floor(sum_marks/subject_marks.length);
console.log("Total sum is ",sum_marks);
console.log("Average is ",average);
```

Code 20 : Given marks, find the maximum marks

```
var marks = [10, 15, 19, 20, 21,45, 31, 18];
var max = -Infinity;

for(var i = 0; i<marks.length; i++)
{
  if(max<marks[i])
  {
    max = marks[i];
  }
}
console.log(max);
```

Objects

Difference between null and undefined

null means no value.

```
var y = null;
console.log(y);
```

undefined means variable is declared but not assigned

```
var x ;
console.log(x);    // undefined
```

On declaring a variable without giving any value, javascript is assigning undefined to that value since the developer forgets to assign some value.

- It is the responsibility of the developer to assign that value

Null is for developer

- In Myntra, Suppose I am trying to search for some products. In the search bar, if I put any product name then it will return all the products containing the given name.
- It is the responsibility of the developer to handle all those cases when the product doesn’t exist or has some name that the user is trying to search.

Array vs Objects(Key-Value Pairs)

Array

```
var subjects = ["maths", "sciene", "english", "Hindi"];
var marks = [40, 50, 80, 20];
```

here, I have two arrays one is containing the subjects and the other contains the marks of that respective subject.

- Suppose If I want to find the marks in English, Then I need to search first in the subjects array for finding the subject index and then using that index I can directly access the marks in the marks array.
- To access the information, the process is complex.

Objects

- It is a data structure that stores the data in a key-value manner.
- It is similar to any other forms which we had filled in our daily life,

  one side which is known as a key, which is telling that what information you want to store and right side acts as a value representing the value of that information.

Storing Information in Arrays vs Objects

Code 1 : Declaring Arrays vs Objects

```
// Arrays
  var user1 = ["Rahul", 25, "male", "Bangalore", "coding"];

// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding"
};
console.log(user2);
```

Note : Key should be unique.

Accessing information in Arrays vs Objects

Code 2 : Accessing the information gender in arrays vs objects

```
// Arrays
  var user1 = ["Rahul", 25, "male", "Bangalore", "coding"];
console.log(user1[2];

// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80]
};

// 1. Bracket Notation
console.log(user["gender"]);
console.log(user["marks"]);
console.log(user["marks"][2]);
console.log(user["marks"].length);

// 2. Dot Notation
console.log(user.gender);
console.log(user.marks);
console.log(user.marks)[2]);
console.log(user.marks.length);
```

In Objects, we can access the information by two ways

1. **Bracket Notation :**

   For Ex : object[”key”]

2. **Dot Notation**

   For Ex : object.key

Adding information in Objects

- There are two ways to add information to an object
  - Bracket Notation : **object[’key’] = value**
  - Dot Notation: **object.key = value**

Code 3: Add the date of birth field in the given object.

```
// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80]
};

// Ist Way
user2['Date_of_Birth'] = "02-Oct-1984";

// IInd Way
user2.Date_of_Birth = "02-Oct-1984";

console.log(user2);
```

Delete Information in Objects

- to delete information use keyword **delete**

  delete object[’key’] ;

  delete object.key;

```
// Objects
  var user2 = {
```

```
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80]
};

// Ist way
delete user2["gender"];

// IInd way
delete user2["gender"]

console.log(user2);
```

Object inside Object

- We can also store objects inside objects. Suppose I want to add information i.e Address and Address will contain other subfields i.e State, Country, District, Pincode, etc.
- To access the information, we can use either bracket or dot notation.

```
// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80],

    address : {
        state : "Uttarakhand",
        country : "india",
        district : "Dehradun",
        pincode : "249201"
    }
};

//Bracket Notation
console.log(user["address"];
console.log(user["address"]["country"]);
console.log(user["address"]["pincode"]);

// Dot Notation
console.log(user.address);
```

```
console.log(user.address.country);
console.log(user.address.pincode);
```

Loops in Objects

- We have a special loop to iterate in objects.
- This special loop is known as, **for-in** loop.

```
var data2 = {
      name : "Kaleen Bhaiyya",
      age : 45,
      gender : "male",
      city : "Mirzapur",
      hobbies : ["Making Guns"]
};


for(var key in data2)
{
  console.log(key," --- ",data2[key]);
}
```

IW Assignment

Problem 1 :

Given an array find the unique items in the array

```
// IW Problem1

var arr = ["Ramesh", "Suresh", "Ramesh", "Kamlesh", "Suresh", "Rupak"];


var party = [];
var present = false;

for(var i = 0; i<arr.length; i++)
{
```

```
    for(var j=0; j<party.length; j++)
    {
        if(arr[i] == party[j])
        {
          present= true;
          break;
        }
    }

    if(present == false)
    {
      party.push(arr[i]);
    }
    else
    {
      present = false;
    }

}

console.log(party);
```

Functions

Suppose , we have wrote the code for calculating the sum of two numbers, calculating the difference of two numbers and calculating the multiplication of two numbers in single file.

When I execute the code file, all the three code will execute but What if I want only to run Addition code or subtraction code only.

I need some tool through which I able to control the different block of code.

For Example :

In Amazon , there are different functionalities implemented like Showing products, Adding/Deleting to cart, Orders, perform payments, etc.

- Every functionalities was written separately.

In Instagram, there are different functionalities like posting the image, commenting , chatting, etc.

- For each functionality, their individual code is written.
- Those code execution depends upon the button you are hitting

Thus , we will going to understand HOW TO CONTROL OUR CODE ?

Therefore to achieve that we have something known as functions.

Functions

A ***JavaScript function*** is a block of code designed to perform a particular task.
A ***JavaScript function*** is executed when "something" invokes it (calls it).

- To solve above problem , we will create three functions of names addition, subtraction and multiplication .
- After creating the function, we will put the respective code inside them.
- Now, we can control the code by calling it. It depends on us How we are calling it. Whichever function will get called, it will run.

Code 1 : Write three block of code :

1. **print length of the name**
2. **Sum of two numbers**
3. **Multiplication of two numbers**

```
var name = "Shubham";
console.log(name, name.length);

var a = 3;
var b = 5;
var sum = a + b;
console.log("Sum is ",sum);

var x = 4;
var y = 8;
var multiply = x*y;
console.log(x*y);

If I execute the above code, All the three code will get executed but I don’t want to run
 all the code
```

Code 2 : Using Functions

```
// Printing Name and length of the name
var name = "Varun";


function sheru(){
  var name = "Shiro";
  console.log(name);
  console.log("length ",name.length);
}
```

```
// Sum of two numbers
function sum_of_two_numbers(){
  var a = 2;
  var b = 3;
  console.log("Sum ",a+b);
}



function print_numbers(){
  // printing from 1 to 10
  for(var i = 1; i<10; i++){
    console.log(i);
  }
}
```

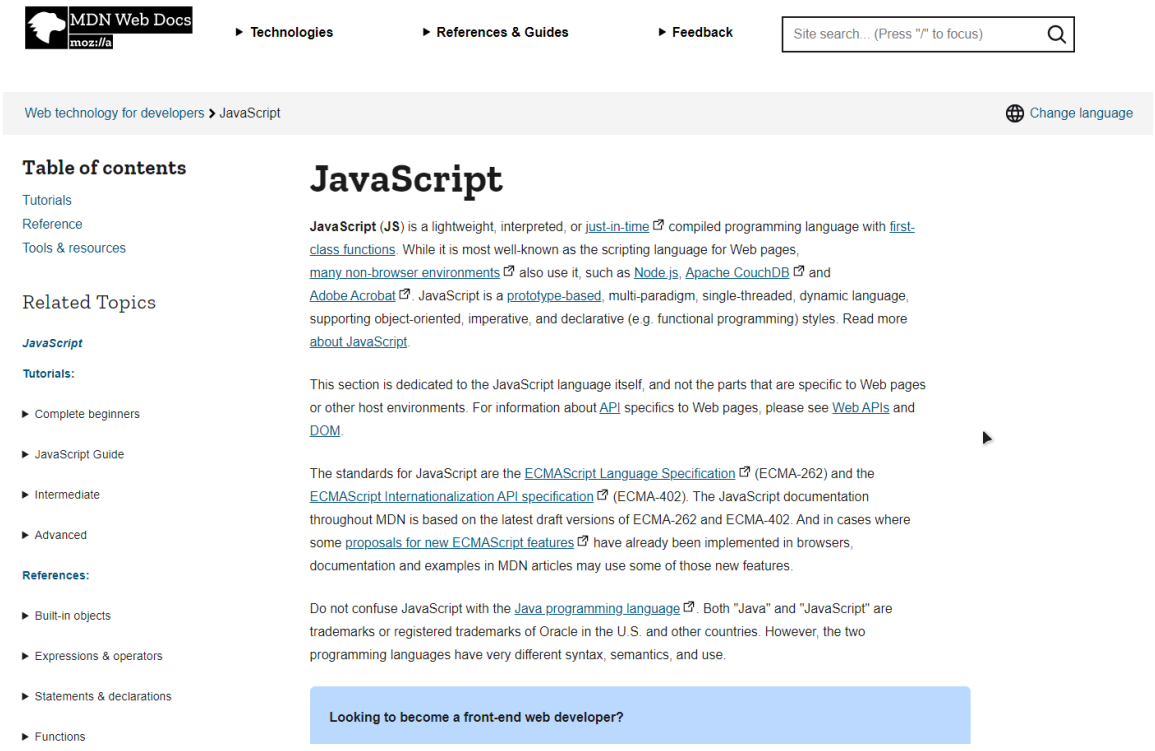
Code 3 : Functions using return values

```
function add(a, b){
  var sum = a + b;
  return sum;
}

function square(x){
  var y = x*x;
  return y;
}

function cube(x){
  var z = x*x*x;
  return z;
  }


var output1 = add(2,3);
console.log("output1 is ", output1);

var output2 = cube(output1);
console.log("output2 is ", output2);

var answer = square(output2);
console.log("answer is ", answer);
```

Code 4 : Local Scope vs Global Scope

```
var outside_child = "chintu"; // Global Variables

function sukhbeer_singh(){
  var sukhbeer_child = "rahul"; // local Variables
  console.log("My child name is ",sukhbeer_child);
}

function kalam_singh(){
  var kalam_child = "rajat"; // local Variables
  console.log("My child name is ",kalam_child);
}

function rajendra_singh(){
  var rajendra_child = "rocky"; // local Variables
  console.log("My child name is ",rajendra_child);
}

sukhbeer_singh();
console.log(outside_child);
```

Inbuilt Function



Code 1 : using toString vs Number()

```
var num1 = 12;
var num2 = 13;

num1 = num1.toString();
num2 = num2.toString();
console.log(num1, num2);


var x = "21";
x = Number(x);
console.log(typeof(x));
```

Code 2 : lastIndexOf

```
var animals = ['Dodo', 'Tiger', 'Penguin', 'Dodo'];
console.log(animals.lastIndexOf("Dodo"));


var people = ["jamna das", "pandu", "kalicharan", "elaichi", "oggy", "pandu"];
console.log(people.lastIndexOf("jerry"));
```

Code 3 : indexOf

```
var animals = ['Dodo', 'Tiger', 'Penguin', 'Dodo'];
console.log(animals.indexOf("Dodo"));


var people = ["jamna das", "pandu", "kalicharan", "elaichi", "oggy", "pandu"];
console.log(people.indexOf("jerry"));
```

Code 4 : shift

```
var people = [1,2,3];
var x = people.shift();

console.log(people);
console.log("removed",x);
```

Code 5 : unshift

```
var arr = [1,2,3];
arr.unshift(4,5,6,7);
console.log(arr);
```

Code 6 : join

```
// Part 1 : Using Bag

var arr = ["M","A","N"];
var bag="";
for(var i=0; i<arr.length; i++)
{
  bag = bag + arr[i];
}

console.log(bag);


// Part 2 : Using Bag
```

```
var bag = arr.join();
console.log(bag);
```

Code 7 : custom join

```
function customJoin(arr, char)
{

  if(char == undefined)
  {
    char = ",";
  }

  var bag="";
  for(var i=0; i<arr.length; i++)
  {
    if(i != arr.length-1)
    {
      bag = bag + arr[i] + char;
    }
    else
    {
      bag = bag + arr[i];
    }
  }

  return bag;
}


var arr = ["M","A","N"];
console.log(customJoin(arr));
```

Code 8 : Slice

```
var animals = ["cat", "dog", "rat", "lion"];
var x = animals.slice(1,3);
console.log(x);
console.log(animals);


var animals = ["cat", "dog", "rat", "lion"];
var manu = animals.splice(2);
console.log("Manu :",manu);
console.log("Original",animals);
```

Code 9 : toLowerCase() vs toUpperCase()

```
var sentence = 'The quick brown fox jumps over the lazy dog.';
console.log(sentence.toLowerCase());


var sentence = 'The quick brown fox jumps over the lazy dog.';
console.log(sentence.toUpperCase());
```

Code 10 : indexOf

```
var paragraph = 'The quick brown fox jumps over the lazy dog. If the dog barked, was it really lazy?';
var searchTerm = 'dog';

var indexOfFirst = paragraph.indexOf(searchTerm);
console.log(indexOfFirst);
```

Code 11 : split()

```
str = 'The quick        brown fox';
var x = str.split(" ");
console.log(x);
```

Code 12 : Custom split mySplit()

```
function mySplit(str)
{
  var output = [];
  var bag = "";

  for(var i=0; i<str.length; i++)
  {

    if(str[i] != " ")
    {
      bag = bag + str[i];
    }
    else
    {
      if(bag != ""){
        output.push(bag);
      }
      bag = "";
```

```
    }

  }

  output.push(bag);

  return output;
}


var str = "The quick        brown fox";
console.log(mySplit(str));
```