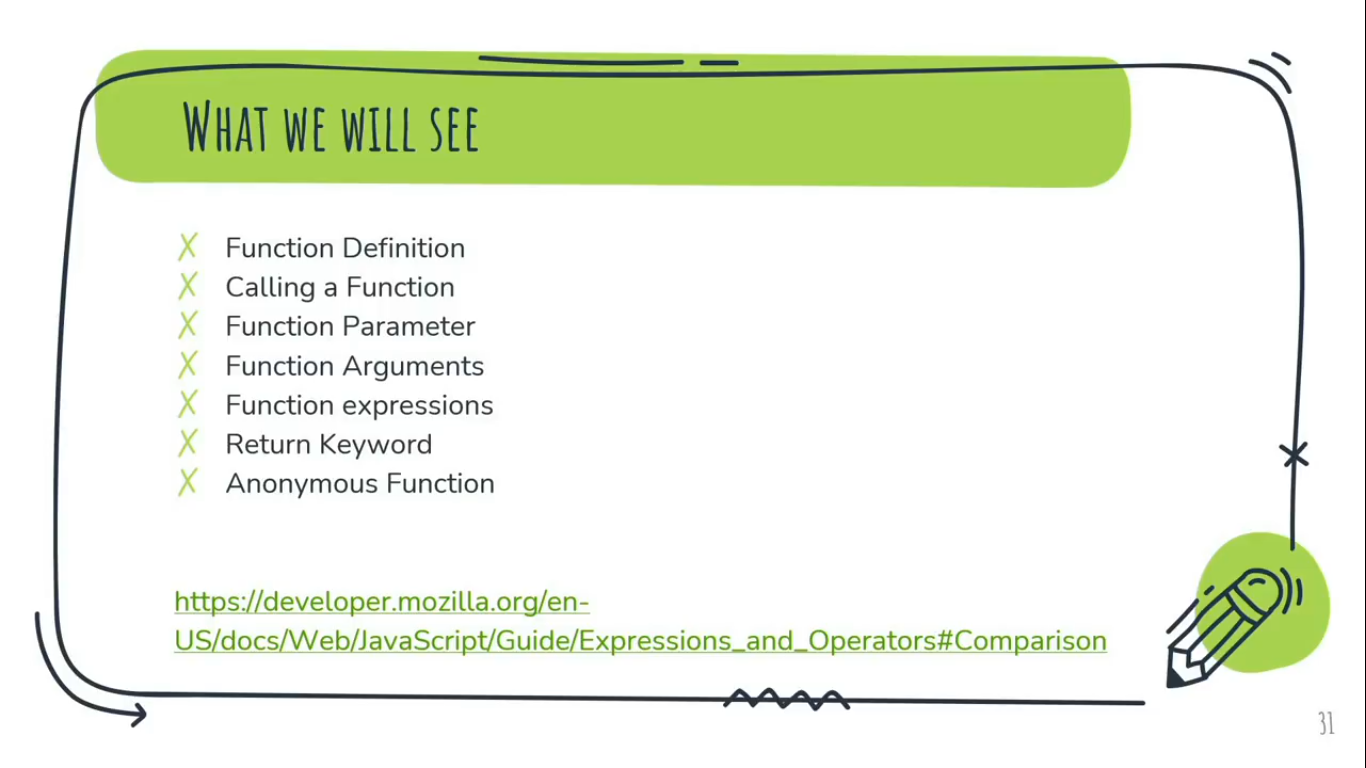
**Functions**

A JavaScript function is a block of code designed to perform a particular task.  
For more Info: [https://developer.mozilla.org/en-us/docs/web/javascript/guide/expressions\_and\_operators#comparison](https://developer.mozilla.org/en-us/docs/web/javascript/guide/expressions_and_operators%23comparison)

**Function Definition:**Before we use a function, we need to define it first.  
A function definition (also called a function declaration, or function statement) consists of the function keyword, followed by:

The name of the function.  
A list of parameters to the function, enclosed in parentheses and separated by commas.   
The JavaScript statements that define the function, enclosed in curly brackets. {...}.  
**Syntax: //Function Defination**  
function functionName(parameter1, parameter2,....)  
{  
//Block of Codes...  
}

**Calling functions:**

Defining a function does not execute it.  
A JavaScript function is executed when “something” invokes it (calls it).

**[I.Q.]** Difference between **Function Parameter** VS **Function Arguments?**

Function Parameter are the names listed in the function’s definition or Value passing while defining function called function parameters.  
Function arguments are the real values passed to the function or Value passing while calling function called function arguments. Arguments can be any value or name of the variable.  
**Ex-1:**  
function add(a, b) //Function Parameters taken as ‘a’, ‘b’   
{

document.write(a+b + '<br>');

}

add(50, 50) //Function Arguments taken as ‘50’, ‘50’

add(100, 100); //Function Arguments taken as ‘100’, ‘100’

**Ex-2:**

function multiply(table){

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

document.write('<br>'+ table + " x " + i + " = " + table\*i);

}

}

multiply(5);

**[I.Q.]** Why Functions?

A function is group of reusable code which can called anywere in our program. This eliminates the need of writing the same code again and again.

* We can reuse code: Define the code once, and use it many times.
* We can use the same code many times with different arguments, to produce different results.

**Function Expressions:**

Function expressions simply means ‘create a function and put it into the variable’ or say declare/define a function and call it by storing/assigning it to the variable. Without return keyword function expression is incomplete.  
**Return Keyword** -  
When JavaScript reaches a return statement, the function will stop executing.  
Function often compute a return value.  
The return value is “returned” back to the ‘function caller’.

**Note:**

* We can only use function expression when function’s block of statements/code are not repeating or not in loop because as JS compute return value using return keyword which instantly inhibit/stop further execution of code.
* In function expression we cannot use multiple function arguments because currently I don’t know how to store multiple function argument in a single variable but we can store multiple arguments in their respective separate variable.

**Ex:**  
function add(a, b) //Function Parameters taken as ‘a’, ‘b’   
{

return a+b; //Return Keyword while using function expression

}

var total = add(50, 50); //Function expression line, where return keyword returned value back to the arguments and store it to the variable name ‘total’.

document.write(total);

**Anonymous Function:**

An unnamed function or a function without a name called anonymous function.

A function expression is similar to and has the same syntax as a function declaration one can define ‘named’ function expressions (where the name of the expression might be used in the call stack for example) or ‘anonymous’ function expressions.

**Ex: // Anonymous Function Expression Example**

var total = function(a, b) //Anonymous Function start from ‘function’

{

return (a+b);

} //Anonymous Function end

document.write(total(50,50));

**CallBack Function:**

Any function is passes or calling as an argument is called a callback function.

A callback is a function that is to be executed after another function has finished executing – hence the name ‘call back’.

**Why Callback function?**

JavaScript is an event driven language. This means that instead of waiting for a response before moving on, JavaScript will keep executing while listening for other events.

Callbacks are a way to make sure certain code doesn’t execute until other code has already finished execution.

Example:

const per1 = (friend, callBackfn) =>{

document.write(`Hey, What's Up ${friend}? <br>`);

callBackfn();

}

const per2 = ()=>{

document.write(`Wait, I call you Back!`);

}

per1(`Chandan`, per2); **//Passing `Chandan`, per2 function as Argument**

Output:-  
Hey, What's Up Chandan?  
Wait, I call you Back!