

# functions

Saturday, March 23, 2024 12:43 PM

## function expression.

expression  $\rightarrow$  evaluate then assign.

$a = 5 + 6$   
 $\uparrow$   
expression

data type :-

Primitive

- string
- int / number
- Boolean

Object

- function
- array
- anything with key value pair

$\nabla$  function expression  
Because it is being assigned.

```
let a = function {int a, int b}
      return a + b;
}
```



function,  
it doesn't have a name

how to call it then?

$a(5, 6)$   
 $\rightarrow 11$

we can use  $\textcircled{a}$  as the  
function name

we can also do this

```
let sum = a;
sum(2, 3) // now sum
          as function
          \rightarrow 5
```

## Higher Order function

$\rightarrow$  It can take  
functions as argument  
 $\rightarrow$  It can return function.

YT LINK

[YT LINK for this  
note](#)

It's a function Torjon horse.

meaning, function inside function  
i.e. function taking function as argument.

lets look at some example

```
function car() {
  return "ferrari"
}
```

Basic  
car  $\rightarrow$  It's a function but its like  
engine is off

car()  $\rightarrow$  now, its engine is ignited  
and is in full power running

How higher function looks like lets see! & see side wise comparison of it  
with a regular function.

Higher Function

function name as  
argument

Regular function

regular arg

with a regular function.

Higher function

function doEverything(argument) {

argument();

}

ignite the car

function name as argument

Regular function

function doSomething(argument) {  
console.log(argument);  
}

regular arg  
not function

So, were passing argument function right,  
then,  
we need to define it isn't it?

```
function argument() {  
  return "run run run";  
}
```

notice! we've passed function name in argument section,

I think it's valid because,  
in tutorial I've watched there was expression function.

**#** Check if it's right or wrong. It's for revision  
in tutorial code was something like this

```
const fight = function {  
  console.log("one two one two");  
}
```

```
function doEverything(argument) {  
  argument();  
  argument();  
  argument();  
}
```

→ execute 3 times

// Execute the function  
doEverything(fight);

## # Return a function

```
function ReturnSomething() {  
  return "!";  
}
```

// regular function

## # Higher Order function

```
function ReturnSomething() {  
  return function() {  
    return "!";  
  }  
}
```

console.log(ReturnSomething()); // It will return a function

console.log(ReturnSomething())(); // It will execute returned function

≡ let result = ReturnSomething();

console.log(result()); // It will print "!"