↪ Js Functions

9 ↪ **Syntax:**

```
function function (para1, para2 ...){
         name
     //code to execute
  }
```

10

11

↪ **Used** → as ↪ Variable values

```
let x = toCelsius(77)
let txt = "Temp:" + x + "C";
```

→

```
let txt = "Temp:" + toCelsius(77) + "C"
```

1

2

↪ **Local** ↪ Variable declared within Js function become 'local' to function

3 → accessed → from within

4

Created ↪ when → function ↪ starts

5 deleted ↪ when → function ↪ completed

~) Js Nested Functions

- Function within another function ~) Nested Function
- Function ~) can have ~) 1 or more ~) Inner Functions
- Under ~) scope of ~) outer functions ~) called ~) Parent Function

Outer Function ~) Parent Function

Inner Function ~) Child Function ~) access {Variable, Parameter} ~) Parent function ← of

cannot access variable ~~variable~~ of inside child function

eg.:

```
<body>
  <script>
      function hypotenuse(a, b) {                 Outer
          function square(x) {                    Inner
              return x*x;
          }
          return Math.sqrt(square(a) + square(b));
      }
      function second() {
```

APRIL						2020
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

```
var result = hypotenuse (2,4) ;
document.write (result) ;
}
</script>
    <button onclick ="secondcs"> Call </button>
<body>
</html>
```

O/P : Call ⟶ O/P → 4.47213595499958

↑ click it

x = 2

result hypo(2,4) → 2, 4 → a=2, b=4 → return → square(a) = $2^2 = 4$

square(b) = $4^2 = 16$

sqrt() $\sqrt{20}$

4·4×4·4
176
176 0
19·36 → 20

Is Functions ~~Closures~~

Function Expression → Stored in variable → used as function

Eg: `Const x = function (a,b) {return a*b};`

Eg:

```
<script>
 Const x = function(a,b) {return a*b};
 document.~~write~~ get-Id().innerHTML = x(4,3) ;
</script>
```

O/p : 12

Anonymous Function → Function without name → No Function name → Invoked using variable name

→ Ends with ; ⟶ part of executable Statement

Function Constructor (pg 123) ⟶

```
const myfun = new Function("a", "b", "return a*b");
let x = myfunc(4,3);
```

Function name

92 ↳ Function Hoisting →

```
myFunc(5);
function myFunc(y) {
    return y*y;
}
```

↳ defined
↳ using
↳ expression → not hoisted

↳ Function → used as → values →

```
function my(a,b) {
    return a*b;
}
let x = my(4,3);
~~docume~~
console.log(x);
```

12 ← O/p

↳ Function used as expression →

```
function ~~myFunc~~ my(a,b) {
    return a*b;
}
let x = my(4,3) * 2;
console.log(x);
```

24 ← O/p

↳ typeof → function

↳ arguments.length → Return → No of Arguments → received → when ← function was invoked

Object

→ Eg:

```
<body>
<p id="elem"></p>
<script>
    function my(a,b,c) {
        return arguments.length;
    }
    document.getEle..ById("elem")
        .innerHTML = my(4,3,2);
</script>
</body>
```

O/p: 3

O/p ← If → replace with → .innerHTML = my.toString();

→ whole function only return

↳ Object Constructor ∴ Function designed to create new objects.
(pg 150) (pg 158)

APRIL 2020						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

→ Arrow Function (refer pg 86)

→ Allow short syntax for writing function expression.

→ No need ┌ keyword – function
return
{ } – Curly Brackets

→ do not have → their → own this.
→ Not well suit. → define → Object Method
→ Not Hoisted
→ Must be defined before use
→ const safer than var → function expr → Constant Value

→ Default Parameter →

x=4

```
function my(x,y){
    if (y === undefined){
        y=2;
    }
    return x*y;
}
```

→ Default Parameter Value →

x=5

```
function my(x, y=10){
    return x+y;
}
```

→ Rest Parameter (...)

Allow → Function → To Treat → Indefinate → no of Argument → as an Array

```
function sum(...args){
    let sum=0;
    for (let arg of args)
        sum+=arg;
    return sum;
}
let a =sum(4,9,16);
```

29 ←

sum [0 4] 13

sum =sum +arg =0+4

JS ARROW FUNCTION

- Introduced → ES6
- Allow → us → to write → Shorter function Syntax
- Before

```
hello = function(){
    return "Hello";
}
```

After

```
hello = () => {
    return "Hello";
}
```

More Shorter

```
hello = () => "Hello";
```

Function → One Statement → Returns → Value

Remove Brackets & Return Keyword

- Arrow Function → With Parantheses → `hello = (val) => "Hi" + val;`
  → Without Parentheses → `hello = val => "Hi" + val;`

- No Binding of this; This Keywords → always → represent → object → that → defined → Arrow function

- Eg.;

WITHOUT ARROW FUNC

```
<body>
  <button id="btn">Click</button>
  <p id="bt"></p>
  <script>
    let hello = "";
    hello = function(){
      document.getElementById("bt").innerHTML
                            += this;
    }
    window.addEventListner("load",
                         hello);
    document.getElementById("btn").addEventListner("click", hello);
  </script>
</body>
```

WITH ARROW FUNC

Replace with Arrow Function

```
hello = () => {
    ——— += this;
}
```

Same line