Saturday → Js Functions

→ Syntax:

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

→ Used → as → Variable values

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

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

→ Local → Variable declared within Js function
become local to function

→ accessed → from within

→ Created → when → function → starts

→ deleted → when → function → completed

Var iable

Js Nested Functions

* Function within another function → Nested Function
* Function → can have → 1 or more → Inner Functions
* Under → scope of → outer function → called → Parent function

Outer Function → Parent Function

Inner Function → child Function → access (Variable, Parameters) → of Parent function

cannot access variable of inside child function

eg:

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

outer

Inner

07 Tuesday


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

O/P : Call (click it) → O/P → 4.472135954999579

result → hypo(2,4) → a=2, b=4 → return → square(a) = 2²=4, square(b) = 4²=16 → √20

4·4×4·4 → 176 → 320

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.getElementById("inner HTML") = 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 = myfun(2,3);
```

92 ↳ Function Hoisting ↳ defined ↳ using ↳ expression ↳ not hoisted

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

↳ Function → used as → values

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

o/p ← 12

↳ Function used as expression

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

o/p ← 24

↳ typeof ↳ function

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

Object

eg:

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

o/p 3

o/p → replace with → `document.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
5 1 2 3 4
12 6 7 8 9 10 11
19 13 14 15 16 17 18
26 20 21 22 23 24 25
27 28 29 30

09 Thursday

100-266 | Week 15

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 expᵛ/ₙ ~) Constant Value

~) Default Parameter ~) x=y

```
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 Array

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

Sum [29]

sum = sum + arg = 0+4

27 JS ARROW FUNCTION

02 APRIL 2020

093-273 | Week 14

Thursday

* Introduced → ES6
* Allow → us → to write (1) Shorter function Syntax

Before

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

After

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

Move Shorter

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

Function (1) One Argument ← Returns Value

Remove Brackets & Return Keyword

* Arrow Function (1) With Parantheses (1)

```
hello = (val) => "Hi" + val;
```

→ Without Parentheses (1)

```
hello = val => "Hi" + val;
```

(1) No. 4 Binding of this ; 'this' Keywords → always (1) represent → object → defined ← that 'Arrow function' ←

* Eg.:

WITHOUT ARROW FUNC

```
<body>
    <button id="btn">Click</button>
    <p id="p"></p>
<script>
    let hello = "";
    hello = function() {
        document.getElementById("btn").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