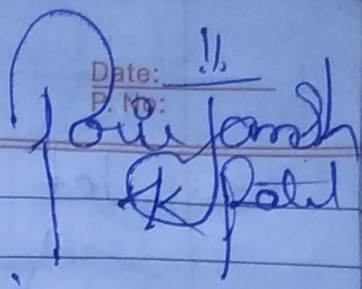
Lecture 19

9/2/23 [JS]

Date: 11



* Object in JS

→ Collection of key-value pair

→ Object have properties and behaviour

Creation →

```
let rectangle = {
property → { "length" : 1,
             "breadth" : 2,
method ← draw : function() {
(behaviour)      console.log("Draw");
                 }
}
```

→ Functions in objects are called method

} Access Values →

```
Console.log(rectangle.length); // 1
```

* Object Creation

pascal notation

(i) Factory Function

Create function which return object.

```
function createRect()
{
    const obj = {};
    return obj;
}

let rect = createRect();
```

camelcase notation

(ii) Constructor Func (1b)

```
function Rectangle(l,b)
{
    this.length = l;
    this.breadth = b;
}

let rect = new
    Rectangle(5,10);
```

* Dynamic Behaviour of Object

add new property. { rect.colour = blue;

Date: 4
P. No.:

let objName = new Rectangle()

constructor — objName() constructor

Rectangle() — constructor — By Default (Function)

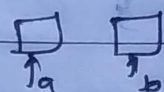
* functions are object

* Types in JS

(i) Primitive → Number, string, Boolean, NULL, Undefined
(ii) Reference → Object, Array, function
(object)

Primitive

```
let a = 10;
let b = a;   [copy]
a++;
```

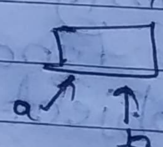


```
console.log(a) // 11
console.log(b) // 10
```

Reference

```
let a = { value: 10 };
let b = a;
a.value++;
Console.log(a.value) // 11
Console.log(b.value) // 11
```

[b = a (reference)] memory



* for-in loop

Ex.

```
const Rectangle = {
    length: 10,
    breadth: 20,
}
```

{ * Used in Objects

```
for(let key in Rectangle)
{ console.log(key, Rectangle[key])
}
```

length 10
breadth 20

* for-of loop

let arr = [10, 20, 40];

for (let i in arr) { * Used in Iterable
{ console.log(i); }

* Check property present in object?

if ('length' in Rectangle)
console.log("Present");

* Object Cloning

(i) Iteration

let duplicate = {};

for (let key in Rectangle)
{
console.log("clone");
duplicate[key] = Rectangle[key]
}

Now duplicate object clones Rectangle object

(ii) Assign

let duplicate = Object.assign({}, Rectangle);

duplicate → destination Object
{} → empty Object
Rectangle → Source Object

(iii) Spread

let duplicate = {...Rectangle};

... → spread operator

* Garbage Collection

Control Automatic by garbage collector.