Roll No. :
Expt. No. :

[Types] ① primitive / [illegible]
② Reference types

Date
Sheet No. :

primitive types

strings	let name = 'rupa'; // string literal
number	let age = 18; // number literal
Boolean	let isapproved = true;
undefined	let Firstname; (or) let Firstname = "undefined";
null	let Lastname = null;

JS is a Dynamic-typed :-

(inspect to → Console)

```
① >typeof name
   "string"
   >name = 1;
   < 1
   >typeof name
   "number"
```

} Dynamic-typed

```
② typeof age
   "number"
   age = 18.1;
   18.1 ;
   typeof age
   "number"
```

JS → has only typeof for both float & real

```
③ typeof isapproved
   "boolean"
   typeof Firstname
   "undefined"
   typeof lastname
```

Objects

reference types

1. Objects
2. arrays
3. function

Obj: obj in real life has properties & variables

~~let person = {}~~

~~let name = 'ropa';~~
~~let age = 30;~~

instead of declare 2 create var, we ~~restore~~ obj literal as person, to keep clean code

```
let person = {
    name: 'ropa',
    age : 30;
};
Console.log(person);
```

age & name is properties

//o/p:

if I want to change the properties,

best ✓ ① Dot notation   person.name = 'rani';
Console.log(person.name);

② Bracket notation   person ['name'] = 'rani';
Console.log(person, name);

Arrays :

```
let selectedColors = ['red', 'blue'] ;
Console.log(selectedColors);
```

Expt. No. :   Sheet No. :

Add Extra Element in array

```
let selectedColors = ['red', blue'];
selectedColors[2] = 'green';   (or we can use number
Console.log(SelectedColors);
//["red", "blue", green"]          //[ "red", "blue",
                                        as array is obj
                                        in JS
```

```
>typeof selectedColors
<"object);
```

propeties ⟶ in arrays

```
Console.log(SelectedColors.length);   //3
```

functions : fondo building blocks in JS

Keyword : function

```
function greet(){
  Console.log('Hello world');
}
// ↑ no semicolon

greet();
```

(15) parameter

parameter (at tym of declaration)

```
function greet(name){
    Console.log('Hello' + name);
}
greet('John');
greet('nipa');
```

argument (actual value)

multiparameter

```
function greet(name, lastname){
    Console.log('Hello' + name + ' ' + Lastname);
};
greet('John', 'smith');
```

console → return keyword

```
function square(number){
    return square * square;
}
let number = square(2);
Console.log(number);
```

(or)

```
function square(number){
    return square * square;
}
Console.log(square(2));
```

// 2 function calls → ① ②

Cont ⇒ ( )