Objects

Difference between null and undefined

null means no value.

```
var y = null;
console.log(y);
```

undefined means variable is declared but not assigned

```
var x ;
console.log(x);    // undefined
```

On declaring a variable without giving any value, javascript is assigning undefined to that value since the developer forgets to assign some value.

- It is the responsibility of the developer to assign that value

Null is for developer

- In Myntra, Suppose I am trying to search for some products. In the search bar, if I put any product name then it will return all the products containing the given name.
- It is the responsibility of the developer to handle all those cases when the product doesn’t exist or has some name that the user is trying to search.

Code 1 : Getting Products

```
// Amazon
function getProduct(name)
{
var arr = ["earphone", "headphone", "ipad"];

if(n<0)
{
  return null;
}

  return arr[n];
}

var result = getProduct(-1);

if(result == null){
  console.log("Invalid Input");
}
```

Note : 0 is not equal to null. 0 is also some value.

For Example : If I ask How many mangoes are present in the apple tree, Obviously the answer is null [because apple tree cannot have mangoes].

Array vs Objects(Key-Value Pairs)

Array

```
var subjects = ["maths", "sciene", "english", "Hindi"];
var marks = [40, 50, 80, 20];
```

here, I have two arrays one is containing the subjects and the other contains the marks of that respective subject.

- Suppose If I want to find the marks in English, Then I need to search first in the subjects array for finding the subject index and then using that index I can directly access the marks in the marks array.
- To access the information, the process is complex.

Objects

- It is a data structure that stores the data in a key-value manner.
- It is similar to any other forms which we had filled in our daily life,

  one side which is known as a key, which is telling that what information you want to store and right side acts as a value representing the value of that information.

Storing Information in Arrays vs Objects

Code 1 : Declaring Arrays vs Objects

```
// Arrays
  var user1 = ["Rahul", 25, "male", "Bangalore", "coding"];

// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding"
};
console.log(user2);
```

Note : Key should be unique.

Accessing information in Arrays vs Objects

Code 2 : Accessing the information gender in arrays vs objects

```
// Arrays
  var user1 = ["Rahul", 25, "male", "Bangalore", "coding"];
console.log(user1[2];

// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80]
};

// 1. Bracket Notation
console.log(user["gender"]);
console.log(user["marks"]);
console.log(user["marks"][2]);
console.log(user["marks"].length);

// 2. Dot Notation
console.log(user.gender);
console.log(user.marks);
console.log(user.marks)[2]);
console.log(user.marks.length);
```

In Objects, we can access the information by two ways

1. **Bracket Notation :**

   For Ex : object[”key”]

2. **Dot Notation**

   For Ex : object.key

Adding information in Objects

- There are two ways to add information to an object
  - Bracket Notation : **object[’key’] = value**
  - Dot Notation: **object.key = value**

Code 3: Add the date of birth field in the given object.

```
// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80]
};

// Ist Way
user2['Date_of_Birth'] = "02-Oct-1984";

// IInd Way
user2.Date_of_Birth = "02-Oct-1984";

console.log(user2);
```

Delete Information in Objects

- to delete information use keyword **delete**

  delete object[’key’] ;

  delete object.key;

```
// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80]
};

// Ist way
delete user2["gender"];

// IInd way
delete user2["gender"]

console.log(user2);
```

Object inside Object

- We can also store objects inside objects. Suppose I want to add information i.e Address and Address will contain other subfields i.e State, Country, District, Pincode, etc.
- To access the information, we can use either bracket or dot notation.

```
// Objects
  var user2 = {
    name : "Rahul",
    age  : 25,
    gender: "male",
    city  : "Bangalore",
    hobbies: "coding",
    marks : [25, 100, 80, 90, 80],

    address : {
        state : "Uttarakhand",
        country : "india",
        district : "Dehradun",
        pincode : "249201"
    }
};

//Bracket Notation
console.log(user["address"];
console.log(user["address"]["country"]);
console.log(user["address"]["pincode"]);

// Dot Notation
console.log(user.address);
console.log(user.address.country);
console.log(user.address.pincode);
```

Loops in Objects

- We have a special loop to iterate in objects.
- This special loop is known as, **for-in** loop.

```
var data2 = {
      name : "Kaleen Bhaiyya",
      age : 45,
      gender : "male",
      city : "Mirzapur",
      hobbies : ["Making Guns"]
};


for(var key in data2)
{
  console.log(key," --- ",data2[key]);
}
```

IW Assignment

Problem 1 :

Given an array find the unique items in the array

```
// IW Problem1

var arr = ["Ramesh", "Suresh", "Ramesh", "Kamlesh", "Suresh", "Rupak"];


var party = [];
var present = false;

for(var i = 0; i<arr.length; i++)
{
    for(var j=0; j<party.length; j++)
    {
        if(arr[i] == party[j])
        {
          present= true;
          break;
        }
    }

    if(present == false)
    {
      party.push(arr[i]);
    }
    else
    {
      present = false;
    }

}

console.log(party);
```