Agenda

- JSON
- Array
- ~~axios~~

Object

- Everything in JS is an Object, even functions are also objects
- Object is a collection of properties (data members or fields) and methods
- Objects are dynamic, meaning you can add, modify, or delete properties after they are created.
- Properties can be accessed/added using dot operator or [] subscript.
- To create an object (instance) in JS, we can use
    1. { } (Object Literal)
    2. new Object()
    3. constructor function

1. { } Using Object Literal

- Create object using JSON syntax.

```
 // create object & props in JS -- (using {})
let i = {
name: "Pencil",
company: "Natraj",
price: 5.0,
};

console.log("type of i = " + typeof(i));
console.log("i name = ", i.name);
console.log("i company = ", i.company);
console.log("i price = ", i["price"]);
```

2. Using new Object().

- Create object using new Object()
- Properties can be added dynamically or accessed using dot/[] operator.

```
// create object & props in JS -- (using .)
let p = new Object();
p.name = "James Bond";
p.age = 65;
p.addr = "London";
console.log("type of p = " + typeof(p));
console.log("p name = ", p.name);
console.log("p age = ", p.age);
console.log("p addr = ", p.addr);
```

```
// create object & props in JS -- (using [])
let m = new Object();
m["model"] = "iPhone 14 Pro";
m["company"] = "Apple";
m.price = 89000.0;
console.log("type of m = " + typeof (m));
console.log("m model = ", m["model"]);
console.log("m company = ", m.company);
console.log("m price = ", m["price"]);
```

3. Using contructor function

- Create object using constructor function i.e. new FnName(args).
- Properties are added into fn using "this" pointer (which refers to newly created object).
- To add common properties & methods for all objects use prototype

```
// using construction function
// Starting a constructor function name with a capital letter is a naming
convention, not a technical requirement enforced by the JavaScript engine.
// Helps to Distinguish from regular functions
function Car(model, company, price) {
this.model = model;
this.company = company;
this.price = price;
}

// add properties and methods for all objects using "prototype".
Car.prototype.color = "red";
Car.prototype.printInfo = function () {
console.log("car model = ", this.model);
console.log("car company = ", this.company);
console.log("car price = ", this.price);
console.log("car color = ", this.color);
};

// Create Car object
let c1 = new Car("i10", "Hundai", 800000.0);
c1.printInfo();
// Create another Car object
let c2 = new Car("Punch", "Tata", 900000.0);
c2.printInfo();
```


Array

- An array in JavaScript is a special variable that can hold multiple values.
- It allow you to store, access, and manipulate lists of data efficiently.
- Arrays are objects.
- Array object have properties (e.g. length) and methods (e.g. push(), pop(), etc).

- Array indices are in range 0 to n-1. Each element accessed using subscript.
- All array elements can be of same type (common use case) or of mixed types.
- Array declaration can be done in two ways:

1. Using Square Brackets (Recommended)
2. Using the Array Constructor

```
let fruits = ["Apple", "Banana", "Mango"];
let fruits = new Array("Apple", "Banana", "Mango");
```

- ☐ are preferred because they are more concise.

Iterating Array

- Array can be iterated using
    1. for-loop

```
for (let i = 0; i < fruits.length; i++) {
    console.log("fruits[" + i + "] = " + fruits[i]);
}
```

    2. for-of loop

```
for (let f of fruits) {
    console.log("element: ", f);
}
```

Array of Objects

- Array of objects can be created using JSON syntax.

```
// Array of Objects -- using JSON syntax
 // create array
 let mobiles = [
  {
    model: "iPhone 14 Pro",
    company: "Apple",
    price: 89000.0,
  },
  {
    model: "Google Pixel 9",
    company: "Google",
    price: 92000.0,
  },
  {
```

```
    model: "Galaxy S25",
    company: "Samsung",
    price: 120000.0,
  },
];

for (let i = 0; i < mobiles.length; i++) {
    console.log(mobiles[i].model + " | " + mobiles[i].company + " | " +
mobiles[i].price);
}
```

Array Operations

1. push() - to push elements at the end.
2. splice(index, 0, newelems) - to insert elements at given index.
3. pop() - to delete the last element.
4. splice(index, delcount) - to delete num of elements from given index.
5. at() - to access element at given index like []. at() can be used with -ve index to access elements from last e.g. at(-1) is last elem, at(-2) is second last elem, ...

```
//push
let names = ["Nilesh", "Rajiv", "Rohan"];
console.log("names : " + names);
names.push("Nitin");
names.push("Prashant");
console.log("names (after pushing 2 items) : " + names);

//splice - insert ops
names.splice(1, 0, "Sarang", "Vishal");
// from 1st index, delete 0 elements, and insert "Sarang", "Vishal"
console.log("names (after splicing/insert 2 items at index 1) : " + names);

//pop
let n1 = names.pop();
console.log("popped item : " + n1);
let n2 = names.pop();
console.log("popped item : " + n2);
console.log("names (after popping 2 items) : " + names);

//splice - delete ops
let n3 = names.splice(2, 1);
// if delete count (arg2) missing all elements after that pos are deleted
console.log("spliced/deleted item (at 2nd pos) : " + n3);
console.log("names (after splice/delete 2nd pos) : " + names);

//at
console.log("names[3] = ", names[3]);
console.log("names.at(3) = ", names.at(3));
console.log("names[-1] = ", names[-1]); // undefined
console.log("names.at(-1) = ", names.at(-1)); // access last element
```

```
// forward traversal
for (let i = 0; i < names.length; i++) {
    console.log("fwd : " + names[i]);
}
// reverse traversal
for (let i = 0; i < names.length; i++) {
    console.log(-i)
    console.log("rev : " + names.at(-i - 1));
```