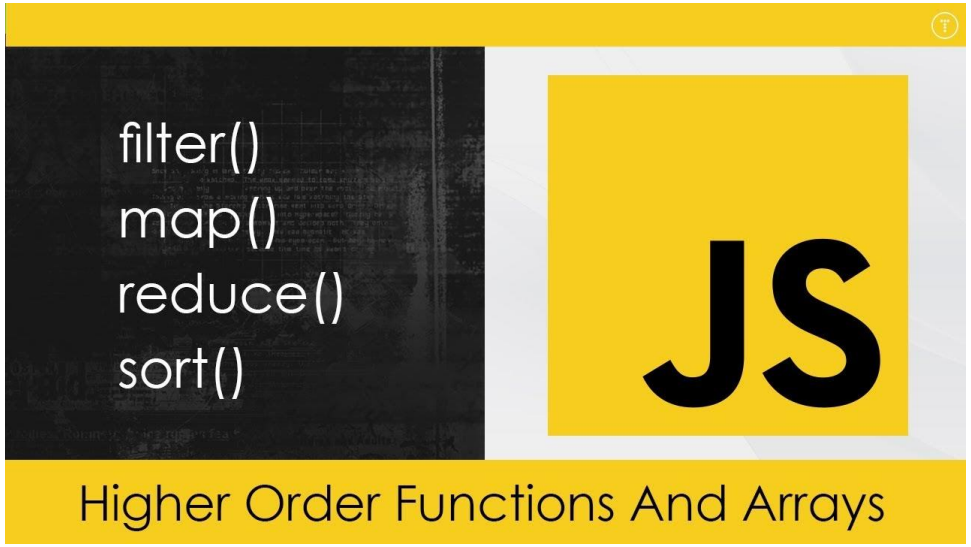
Classes, Arrays and Functions



class

```
class Person {
    constructor (id, name) {//id - is a local variable
        this.id = id; //this.id - is a class field
        this.name = name;
    }

    get Id() {        return this.id;      } //property

    set Id(value) {
        if (value < 10000 || value > 99999) {
          alert("id should be 5 digits.");
          return;
        }
        this.id = value;
    }

    show () { // class methods
        return this.name + " id is " + this.Id;
    }
}
```

```
p1 = new Person(12345,"ana");
alert(p1.show());
p1.Id=77777
alert(p1.show());
//p1.Id(88888) //ERROR!
//alert(p1.show());
```

Inheritance

```
// *** Inheritance ***
class Teacher extends Person {
    constructor (id,name,title) {
        super(id, name) // call the parent constructor
        this.title = title;
    }

    show () { // override the parent show method
        return this.title + " " + this.name + " id is " + this.Id;
    }

    show1 () { // use the parent method
        return this.title + " " + super.show();
    }

    static show_school(){
        alert('Ruppin:)');
    }
}
```

```
t1 = new Teacher(23456,“Avi","dr");
alert(t1.show());
alert(t1.show1());
Teacher.show_school();
```

Function

```
// The "classic" way of defining a function

function f1(a, b) {
return a + b;
}
alert("normal: " + f1(2, 3));
```

Function

```
// this is called a function expression, using an anonymous function
x = function(a, b) {
        return (a + b)
};
alert("anonymous function expression: " + x(2, 3));

// another way to set the function expression by assigning to a non anonymous function
function f1(a, b) {
        return (a + b)
};
y = f1;
alert("non anonymous function expression: " + y(2, 3));
```

Templates

```
// templates are functions
function Person(_name, _age) {
    this.name = _name;
    this.age = _age;
}

p = new Person("Amiti", 0.7);
alert("accessing object fields : " + p.name + " is " + p.age + " years old");
```

Templates

```
// passing a default parameter
function Student(_name, _active = true, _country = "Israel") {
    this.name = _name;
    this.active = _active;
    this.country = _country;
}

s = new Student("Dana");
alert(s.active);
alert(s.country);
```

Templates

```
// using a method in a template
function Cat(_name, _age) {
    this.name = _name;
    this.age = _age;
    this.show = function() {
        return (this.name + " is " + this.age + " years old")
        };
}

c = new Cat("Mitzi", 2);
alert("using a template method: " + c.show());
```

The function is **bound** to the object

The problem

```
// assign the template method to a function expression.
f = c.show;
alert("using an assigned to a template method: " + f());
```

The solution

```
// the assigned variable is now bound to the context of the object
fb = c.show.bind(c);
alert("using bind for the context: " + fb());
```

Arrow functions
Short syntax for a single statement

Syntax : functionName = parameters => single statement;

Classic Syntax	Arrow Function Syntax

function f1(x) {
return x*x;
}
	f1 = x => x * x;

function heikef(x,y) {
return x*x;
}
	heikef = (x,y) => 2 * (x + y);

function myDate() {
return new Date();
}
	myDate = () => new Date();

```
Alert(f1(5));
alert(heikef(2, 3));
alert(myDate());
```

Arrow functions

Short syntax <u>not necessary</u> for a single statement

Syntax : functionName = (parameters…) => {code…}

Classic Syntax	Arrow Function Syntax

function heikef(x,y) {
Code…
}
	heikef = (x,y) => {
	code…
	}

heikef(2, 3);

Arrow functions - continue

```
// arrow functions in a template
function Dog(_name) {
    this.name = _name;
    // define an arrow function in a template
    this.show = () => "the dog name is " + this.name;
}

d = new Dog("Pluto");
alert("using an arrow function in a template: " + d.show());
```

ARROW functions – solve the problem

```
fb = d.show; // assign the function to a new variable.

alert("using an assigned arrow function from a template: " + fb());
// Note that is does work!! arrow functions bind automatically
```

MAP- COMBINE ARRAYS and FUNCTIONS

// map: iterates over all the elements in the array, activates a function for each of them and returns a new array

```
// the classic way
var numbers = [1, 4, 9];
var doubles = [];
for (i in numbers) {
    doubles[i] = numbers[i] * 2;
}
console.log(doubles);
```

MAP- COMBINE ARRAYS and FUNCTIONS

```
var numbers = [1, 4, 9];
var doubles = numbers.map(function(num) {
        return num * 2;
});

console.log(doubles);
```

MAP

```
// map using an arrow function
var numbers = [1, 4, 9];
var doubles = numbers.map(num => num * 2);
console.log(doubles);
```

FIND

```
// find the first number which is greater than 4 and divide by 2

// somearray.find(logicalExpression)

var numbers = [1, 4, 8, 9, 12];
var res = numbers.find(num => num > 4 && num % 3 == 0);
alert(res);
```

FILTER

// filter retrieves all the entries for which the boolean condition is true

// somearray.filter(logicalExpression)

```
var numbers = [1, 4, 8, 9, 12];
var res = numbers.filter(num => num > 4 && num % 2 == 0);
console.log(res);
```

MORE FUNCTION

many more array functions can be found at :

https://www.w3schools.com/jsref/jsref_obj_array.asp