can be of any name

1) Declare Static Method ⌒ Keyword Static

2) Class ⌒ More than one

1) Static Method ⌒ used ⌒ create → Utility Functions

2) If → declare → more ⌒ One ⌒ with → Similar name, Is invokes ⌒ last one

3) To call Static method within another static method.

4) Cannot → directly call → Static method ⌒ within non static method. In such case

Call ← Static method ← using ← classname or Constructor property

5) Is ⌒ provide Static methods ⌒ belong to class, instance of. To call Static Method → not required ⌒ Instance ; directly → on class ← itself

Eg1) Static Method

```
class Test {
  static display() {return "Invoked";}
}
Console.log ("Invoked");
```

Eg2) More than one Static method (SM)

```
class Test {
    Static display() { return "Invoked"; }
    Static disp() { return "Invoked 2"; }
}
Console.log ("Invoked").
Console.log ("Invoked2");
```

Eg3) More than one SM (same name)

```
class Test {
    Static display() { return "Invoked"; }
    Static display() { return "Invoked2"; }
}
Console.log (Test.display());
```

o/p Invoked 2

Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

JULY 2020

Eg4) Invoke Static Method within Constructor

→ Doubt? → this.constructor.<method name>;

→ Eg:

```
Class Test(){
        constructor(){
            console.log(Test.display());
          console.log(this.constructor.display());
        }
      Static display(){
            return "Invoked!";
        }
    }
  Var test = new Test();
```

Eg5) Invoke Static Method within NonStatic Method

```
Class Test(){
        Static display(){
            return "Invoked!";
        }
        nstatic(){
            console.log(Test.display());
        }
    }
Var test = new Test();
test.nstatic();
```

O/P
Invoked

Console.log(Test.nstatic()); → Type Error: Test.nstatic is not function