function binding

📌 To execute function with an object

Types

1. static
2. dynamic

static function binding

```
"use stri

function isEven(inp) {
    let out = inp % 2 === 0
    return out
}

let out = isEven(10)
console.log(out)


function sayName() {
    console.log(`im ${this.name}`)
}

// sayName() // error

let p1 = {
    name: 'Nag',
    sayName: sayName // static function binding
}
p1.sayName();


let p2 = {
    name: 'Ria',
    // sayName:sayName,
    sayName: function () {
        console.log(`my name is ${this.name}`)
    }
}
p2.sayName()
```

E.g

```
function doTraining(sub, duration, location) {
    console.log(`trainer, ${this.name} : teaching ${sub} (${duration}) in ${location}`)
}

doTraining("MERN-stack",60,'chennai') // error
```

```
let trainer = { name: 'Nag',doTraining:doTraining }
trainer.doTraining("MERN-stack",60,'chennai')
```

```
let trainer = { name: 'Nag' }
trainer.doTraining = doTraining
trainer.doTraining("MERN-stack",60,'chennai')
```

```
let trainer = { name: "Nag" }
Object.preventExtensions(trainer)

trainer.doTraining = doTraining // error
```

dynamic function binding

```
doTraining.call(trainer, "MERN-stack", 60, 'chennai')
doTraining.apply(trainer, ["MERN-stack", 60, 'chennai'])
let newF=doTraining.bind(trainer,'MERN-stack')
newF(10,'chennai')
newF(60,'bengalore')
```

---

summary

```
function func() {
    console.log(this)
}

func();

let a = { name: 'A' }
a.func = func  // static function binding
a.func()

let b = { name: "B" }
Object.preventExtensions(b)
// b.func = func // error
func.call(b) // dynamic function binding
```

```
//-------------------------------------------------
// Quiz-1
//-------------------------------------------------

// let pName="global";
let p = {
    pName: 'kishore',
    sayName: function () {
        // let pName="local"
        // console.log('im ' + pName)
        console.log('im ' + p.pName)
        console.log('im ' + this.pName)
    }
}
```

```
// p.sayName();

//------------------------------------------------



//------------------------------------------------
// Quiz-2
//------------------------------------------------

let tnr1 = {
    name: 'Nag',
    doTeach: function () {
        console.log(this.name + " teaching javascript")
        let self=this;
        const doLearn = function () {
            console.log(this.name + " learning javascript from trainer "+self.name)
        }
        return doLearn;
    }
}

//------------------------------------------------
// day-1
//------------------------------------------------
let learnFunc = tnr1.doTeach();
let s1 = { name: 'Sean' }
// let s2 = { name: 'Clayton' }
learnFunc.call(s1)
// learnFunc.call(s2)


//------------------------------------------------
// day-2
//------------------------------------------------

let otherTnr={name:'Kishore'}
let doLearn=tnr1.doTeach.call(otherTnr)
let s={name:'Ranjani'}
doLearn.call(s)


//------------------------------------------------
```

Ex.

```
function Trainer(name) {
    this.name = name
}
function Student(name) {
    this.name = name;
}
function fullstackTng() {
    console.log(this.name + " giving fullstack tng..")
    let notes = "NOTES"
    let self = this;
```

```
    let learn = function () {
        console.log(`${this.name} learning fullstack using ${notes},from trainer - ${self.name}`)
    }
    return learn;
}
function doTraining() {
    let tnr = new Trainer("Nag")
    let s1 = new Student("s1")
    let s2 = new Student("s2")
    let learn = fullstackTng.call(tnr)
    learn.call(s1)
    learn.call(s2)
}
doTraining();
```