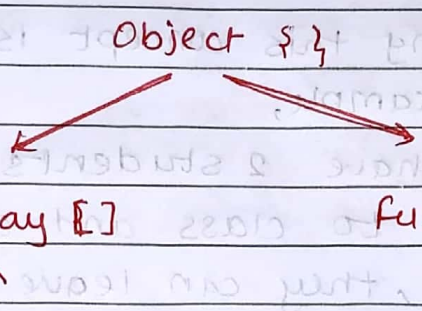
JavaScript = Day 9

PROTOTYPAL INHERITANCE:-

→ Inheritance is an object getting access to the properties and methods of another object.

→ We already disscussed that arrays functions in Javascript are basically objects.



→ so the array object and the function object get access to the properties and methods of object {}.

```
const array = []
array.__proto__
> [concat: f, fill: f, find: f, ...] ⇒ Array.
```

⇒ There are basically the methods we use on array right?

→ Now let's go up the prototype chain, what's on top of Array []? It's object {} (see above diagram).

```
> array.__proto__.__proto__
```

going one chain up (object {})

→ {hasOwnProperty: f, toString: f, valueOf: f, ...}

```
>array.toString()
> ""
```

→ How are we able to use toString method on array? well we just said inheritance is one object (array)

can access method (toString) of another object (object4).

→ So what-ever is on top of the prototype inheritance chain, you'll have access to it.

⇒ Try the same thing with functions and objects.

→ To understand why this concept is important let's take an example,

⇒ Let's say we have 2 students, who say hi when they come to class and once they finish the assignment, they can leave by saying Bye.

```
let student1 = {
        name: 'John',
        assignmentDone: true,
        sayHi: function () {
                console.log('Hi');
        }
        sayBye: function () {
                if (assignmentDone)
                {
                        console.log('Bye');
                }
        }
};
let student2 = {
        name: 'Ria';
}
```

→ And now we don't want to repeat code, so if student2, want to use method of student1, we

~~teamt we~~ learnt we can use bind.

```
const sayBye = student1.sayBye.bind(student2);
```

we want to use sayBye method of student1

we want to use it for student2.

→ But wait, we do have access to sayBye from student1 but sayBye needs assignmentDone variable and student2 does not have it.

→ So we need to find a solution that not just let's student2 have access to sayBye but also assignmentDone.

→ We basically want student2 to 'inherit' all functions and variables [properties] of student1.

Solution :-

student2.__proto__ = student1

> student2.sayBye()
→ Bye

> student2.name
→ Ria

> student2.sayHi()
→ Hi

> student2.assignmentDone
→ time

→ That means whatever properties student2 already has

(name) will be taken from student2. if self, But whatever is new (sayHi, sayBye, assignmentDone) will be inherited (taken) from student1.

Recap

student2.__proto__ = student1

create a prototype chain and

inherit properties not present in student2 from student1.

Excercise

① for (let property in student2)
    console.log(property)

② for (let property in student2)
    if(student2.hasOwnProperty(property)){
        console.log(property);
    }

→ After you execute these, you will know that student2 does not actually have properties of student1 copied, instead we just have a reference to them through prototypal inheritance. (It works up the prototype chain if property present).