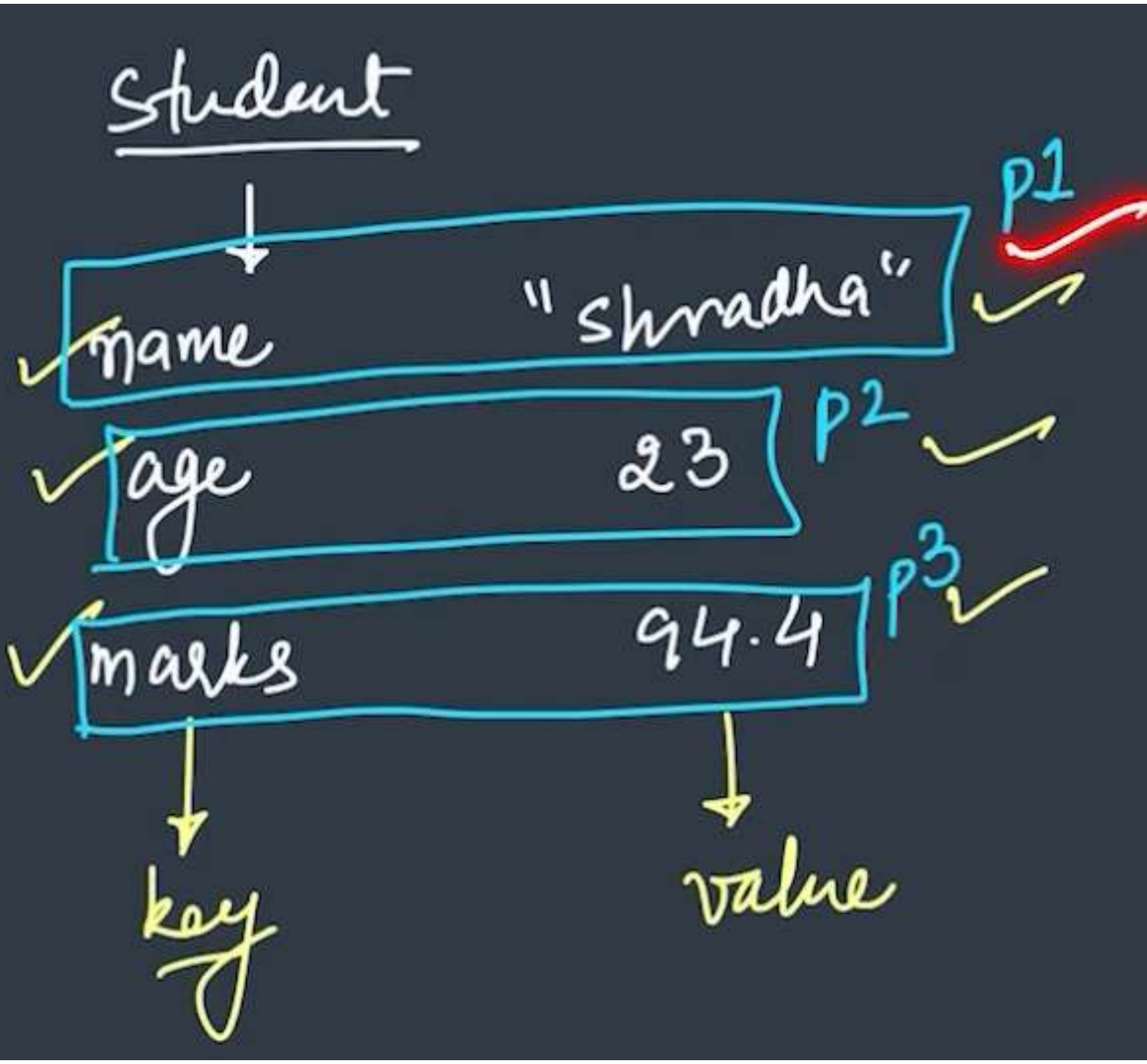
JS Objects Literals

Used to store keyed collections & complex entities.

property => (key, value) pair

Objects are a collection of properties


student
name
"shradha"
p1
age
23
p2
marks
94.4
p3
key
value

JS Objects Literals

```
let delhi = {
    latitude: "28.7041° N",
    longitude: "77.1025° E"
};
```

```
const student = {
    name: "shradha",
    age: 23,
    marks: 94.4,
    city: "Delhi"
};
```

Thread / Twitter Post

Create an object literal for the properties of thread/ twitter post which includes -

- username
- content
- likes
- reposts
- tags

Get Values

```
let student = {
    name : "shradha",
    marks : 94.4
};
```

```
student["name"]

student.name
```

Add/ Update Value

- Change the city to "Mumbai"
- Add a new property, gender: "Female"
- Change the marks to "A"

```
const student = {
    name: "shradha",
    age: 23,
    marks: 94.4,
    city: "Delhi"
};
```

```
> student
<· ▶ {name: 'shradha', age: 23, marks: 94.4, city: 'Delhi'}
> student.city
<· 'Delhi'
> student.city = "Mumbai";
<· 'Mumbai'
> student.city
<· 'Mumbai'
> student
<· ▶ {name: 'shradha', age: 23, marks: 94.4, city: 'Mumbai'}
>
```

```
> student.gender
<· undefined
> student.gender = "female";
<· 'female'
> student.gender
<· 'female'
> student
<· ▶ {name: 'shradha', age: 23, marks: 94.4, city: 'Mumbai', gender: 'female'}
>
```

Object of Objects

Storing information of multiple students

```
const classInfo = {
    aman : {
        grade: "A+",
        city: "Delhi"
    },
    shradha : {
        grade: "A",
        city: "Pune"
    },
    karan : {
        grade: "O",
        city: "Mumbai"
    }
};
```

Array of Objects

Storing information of multiple students

```
const classInfo = [
    {
        name: "aman",
        grade: "A+",
        city: "Delhi"
    },
    {
        name: "shradha",
        grade: "A+",
        city: "Pune"
    },
    {
        name: "karan",
        grade: "O",
        city: "Mumbai"
    }
];
```

Math Object

Properties

Math.PI

Math.E

Methods

Math.abs(n)

Math.pow(a, b)

Math.floor(n)

Math.ceil(n)

Math.random( )

```
> Math.floor(5)
<· 5
> Math.floor(5.5)
<· 5
> Math.floor(5.9999999999)
<· 5
> Math.floor(-5)
<· -5
> Math.floor(-5.5)
<· -6
>
```

Math Object

a **b

num round off <=

Properties

Math.PI

Math.E

Methods

Math.abs(n)

Math.pow(a, b)

nearest smallest Int value ← Math.floor(n)

Math.ceil(n)

0 to 1 ← Math.random( )

Random Integers

From 1 to 10

Step1 :

```
let num = Math.random();
```

```
0.46747741318127045
```

Step2 :

```
num = num * 10;
4.674774131812704
```

Step3 :

```
num = Math.floor(num);
4
```

Step4 :

```
num = num + 1;
5
```

```
> Math.floor( Math.random() * 10 );
<· 0
>
```

Random Integers

From 1 to 10

```
let random = Math.floor(Math.random() * 10) + 1;
undefined
```

```
random
4
```

Qs

max

Generate a random number between 1 and 100.

Math.floor(Math.random() * 100) + 1

Generate a random number between 1 and 5.

Guessing Game

User enters a max number & then tries to guess a random generated number between 1 to max.

```
user = prompt() // 10
```

app.js > ...

```javascript
const max = prompt("enter the max number");

const random = Math.floor(Math.random() * max) + 1;

let guess = prompt("guess the number");

while(true) {
    if(guess == "quit") {
        console.log("user quit");
        break;
    }

    if(guess == random) {
        console.log("you are right! congrats!!");
        break;
    } else {
        guess = prompt("your guess was wrong. please try again");
    }
}
```