> Promise.allSettled(promises) (recently added method) – waits for all promises to settle and returns their results as an array of objects with: status: "fulfilled" or "rejected" value (if fulfilled) or reason (if rejected).

> Promise.race(promises) – waits for the first promise to settle, and its result/error becomes the outcome.

> Promise.any(promises) (recently added method) – waits for the first promise to fulfill, and its result becomes the outcome. If all of the given promises are rejected, AggregateError becomes the error of Promise.any.

> Promise.resolve(value) – makes a resolved promise with the given value.

> Promise.reject(error) – makes a rejected promise with the given error. Of all these, Promise.all is probably the most common in practice.

Watch Live On Youtube below:

Episode 25 : `this` keyword in JavaScript

> In JavaScript, the this keyword refers to an object, which object depends on how this is being invoked (used or called).

`this` in global space

Anything defined globally is said to be in a global space.

```
console.log(this); // refers to global object i.e. window in case of
browser
// 💡 global object differs based on runtime environment,
```

`this` inside a function

```
function x() {
    // the below value depends on strict/non-strict mode
    console.log(this);
    // in strict mode - undefined
    // in non-strict mode - refers to global window object
}
x();
// 💡 Notes:

// On the first go feels like `this` keyword in global space and inside
function behaves same but in reality it's different.

// The moment you make JS run in strict mode by using: "use strict" at the
top, `this` keyword inside function returns `undefined` whereas global
space will still refers to global window object
```

`this substitution` -> According to `this` substitution, if the value of `this` keyword is `null/undefined`, it will be replaced by globalObject only in non-strict mode. This is the reason why `this` refers to global window object inside function in non-strict mode.

💡 So to summarize, the value of `this` keyword inside function is `undefined`, but because of `this substitution` in non-strict mode `this` keyword refers to `globalWindowObject` and in strict mode it will still be `undefined`

`this` keyword value depends on how the `function` is called. For eg:
In strict mode:

```
x(); // undefined
window.x(); // global window object
```

`this` inside a object's method

```
// `x` key below is a method as per terminology
const obj = {
    a: 10,
    x: function () {
        console.log(this); // {a: 10, x: f()}
        console.log(this.a); // 10
    }
}
obj.x(); // value of `this` is referring to current object i.e. `obj`
```

`call`, `apply` & `bind` methods

> For detail around call, apply and bind method. Refer here.

```
const student = {
    name: 'Alok',
    printName: function () {
        console.log(this.name);
    }
}
student.printName(); // Alok

const student2 = {
    name: 'Kajal',
}
student2.printName(); // throw error

// ❓ how to re-use printName method from `student` object
student.printName.call(student2); // Kajal
// Above `call` method is taking the value of `this` keyword
// So, Inside `printName` method value of `this` is now `student2` object

// So, call, bind and apply is used to set the value of this keyword.
```

`this` inside arrow function

Arrow function doesn't have their own `this` value, they take the value from enclosing lexical context.

```
const obj = {
    a: 10,
    x: () => {
        console.log(this); // window object
        // Above the value of `this` won't be obj anymore instead it will
be enclosing lexical context i.e. window object in current scenario.
    }
}
obj.x();

const obj2 = {
    a: 10,
    x: function () {
        const y = () => {
            console.log(this);
            // Above the value of `this` will be obj2 as function y's
enclosing lexical context is function `x`.
        };
        y();
    }
}
obj2.x();
```

`this` inside DOM

> It refers to HTML element.

```
<button onclick="alert(this)">Click Me</button>
<!-- [object HTMLButtonElement] Button element -->
```

Watch Live On Youtube below: