L02 Readings

This Keyword

- What does this reference to?
- It depends on the call-site of the function that is using **this**.
- By default, **this** refers to the global object.
- For instance if I have a function called **book**:

```
let name="programming";

function book() {
  console.log(this.name);
}

book()
```

- The call-site of the function “book” is a plain function reference – just *book().*
- The executing context here is the global scope and that is what **this** refers to.
- The function will output “programming”.
- This is what we call the default binding of the **this** keyword.
- Now let's take a look at the following code:

```
let name="programming";

function book() {
  console.log(this.name);
}

let obj={
  name:"coding",
  book:book
}

obj.book()
```

- What will the *book()* function output?
- Let’s take a look at the call-site.
- The object “obj” owns the function reference at the time the function is called.
- So the object “obj” is the execution context.
- And so **this** in this case refers to the object “obj”.
- Thus the function will output “coding” instead of “programming”.

- This is called implicit binding of the **this** keyword.
- Now the last case is when you force a function to use a particular object for **this** binding.
- To do that you can use the *call, apply or bind* JavaScript functions.
- Here is an example:

```
let name="programming";

function book() {
  console.log(this.name);
}

let obj={
  name:"coding",
}

book.call(obj);
```

- Here we're explicitly telling the function *book()* to use the object "obj" as a reference for **this.**
- So in this case this refers to the object "obj".
- And the output will be "coding";