1/JavaScript Function apply():

With the `apply()` method, you can write a method that can be used on different objects.

The JavaScript apply() Method:

The `apply()` method is similar to the `call()` method (previous chapter).

Example:

```
const person = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}

const person1 = {
  firstName: "Mary",
  lastName: "Doe"
}

// This will return "Mary Doe":
person.fullName.apply(person1);
```

The Difference Between call() and apply()

The difference is:

The `call()` method takes arguments **separately**.

The `apply()` method takes arguments as an **array**.

The apply() Method with Arguments

The `apply()` method accepts arguments in an array:

Example

```
const person = {
  fullName: function(city, country) {
    return this.firstName + " " + this.lastName + "," + city + "," +
country;
  }
}

const person1 = {
  firstName:"John",
  lastName: "Doe"
}

person.fullName.apply(person1, ["Oslo", "Norway"]);
```

2/JavaScript Function call():

Method Reuse:

With the `call()` method, you can write a method that can be used on different objects.

All Functions are Methods

In JavaScript all functions are object methods.

If a function is not a method of a JavaScript object, it is a function of the global object (see previous chapter).

The example below creates an object with 3 properties, firstName, lastName, fullName.

Example

```
const person = {
  firstName:"John",
  lastName: "Doe",
```

```
  fullName: function () {
    return this.firstName + " " + this.lastName;
  }
}

// This will return "John Doe":
person.fullName();
```

In the example above, `this` refers to the **person object**.

this.firstName means the **firstName** property of **this**.

Same as:

this.firstName means the **firstName** property of **person**.

What is **this?

In JavaScript, the `this` keyword refers to an **object**.

Which object depends on how `this` is being invoked (used or called).

The `this` keyword refers to different objects depending on how it is used:

In an object method, `this` refers to the **object**.
Alone, `this` refers to the **global object**.
In a function, `this` refers to the **global object**.
In a function, in strict mode, `this` is `undefined`.

In an event, `this` refers to the **element** that received the event.

Methods like `call()`, `apply()`, and `bind()` can refer `this` to **any object**.

The JavaScript call() Method

The `call()` method is a predefined JavaScript method.

It can be used to invoke (call) a method with an owner object as an argument (parameter).

With `call()`, an object can use a method belonging to another object.

3/JavaScript Function bind():

With the `bind()` method, an object can borrow a method from another object.

The example below creates 2 objects (person and member).

The member object borrows the fullname method from the person object:

Example

```
const person = {
  firstName:"John",
  lastName: "Doe",
  fullName: function () {
    return this.firstName + " " + this.lastName;
  }
}

const member = {
  firstName:"Hege",
```

```
  lastName: "Nilsen",
}

let fullName = person.fullName.bind(member);
```

**Explain the example:

JavaScript Function bind()

This example creates 2 objects (person and member).

The member object borrows the fullname method from person:

Hege Nilsen

4/JavaScript Let:

The `let` keyword was introduced in ES6 (2015)

Variables defined with `let` cannot be **Redeclared**

Variables defined with `let` must be **Declared** before use

Variables defined with `let` have **Block Scope**

5/JavaScript Const.

The `const` keyword was introduced in ES6 (2015)

Variables defined with `const` cannot be **Redeclared**

Variables defined with `const` cannot be **Reassigned**

Variables defined with `const` have **Block Scope**

Difference Between var, let and const

	Scope	Redeclare	Reassign	Hoisted
var	No	Yes	Yes	Yes
let	Yes	No	Yes	No
const	Yes	No	No	No

6/

There are two ways to clone an object in Javascript: Shallow copy: means that only the first level of the object is copied. Deeper levels are referenced. Deep copy: means that all levels of the object are copied.