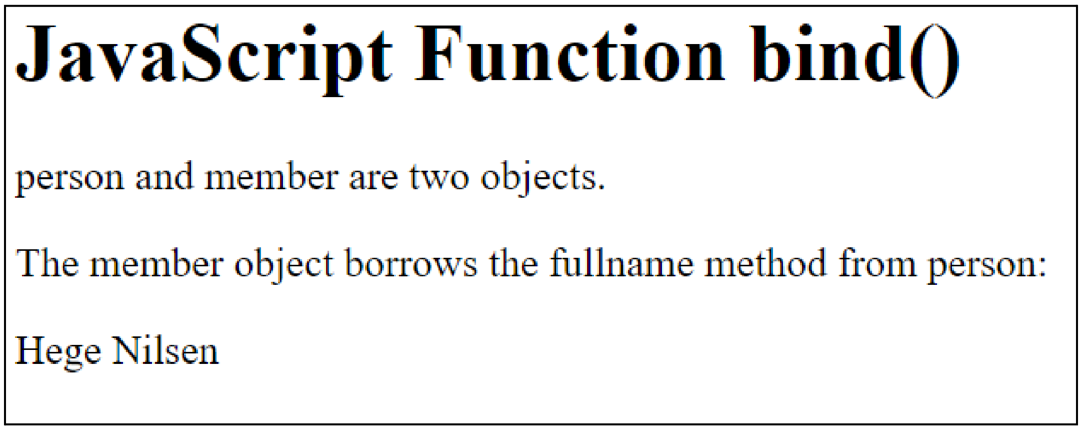
Function Binding and Closure - Practice Code

Problem Statement 1: Function borrowing

With the **bind()** method, create two objects and make an object borrow a method from another object.



JavaScript Function bind()

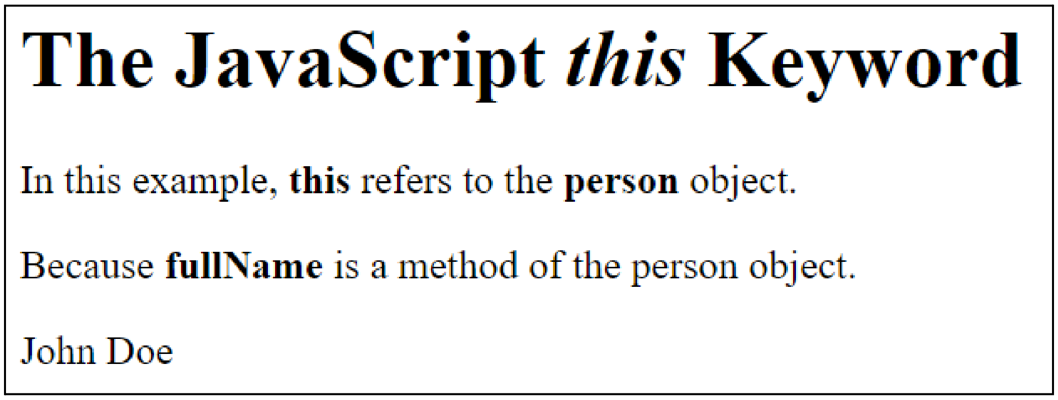
person and member are two objects.

The member object borrows the fullname method from person:

Hege Nilsen

Problem Statement 2: this in method

When used in an object method, **this** refers to the object. Create an object, where you have multiple properties and values, and define a function as one value. With help of **this** keyword access the values of properties in that function. Invoke the function to display the data from the object.



The JavaScript *this* Keyword

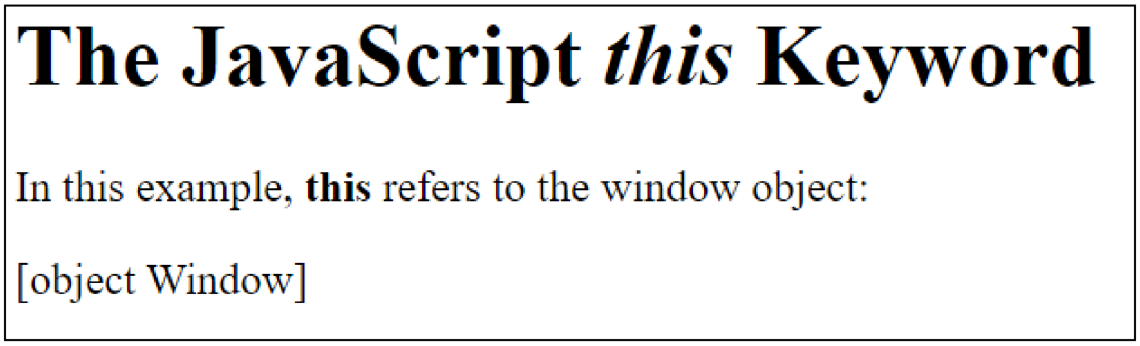
In this example, **this** refers to the **person** object.

Because **fullName** is a method of the person object.

John Doe

Problem Statement 3: this as global object

When used alone, **this** refers to the global object. In a browser window the global object is **[object Window]**, Display the global object in a browser window by assigning **this** as a value of a variable and print the variable.



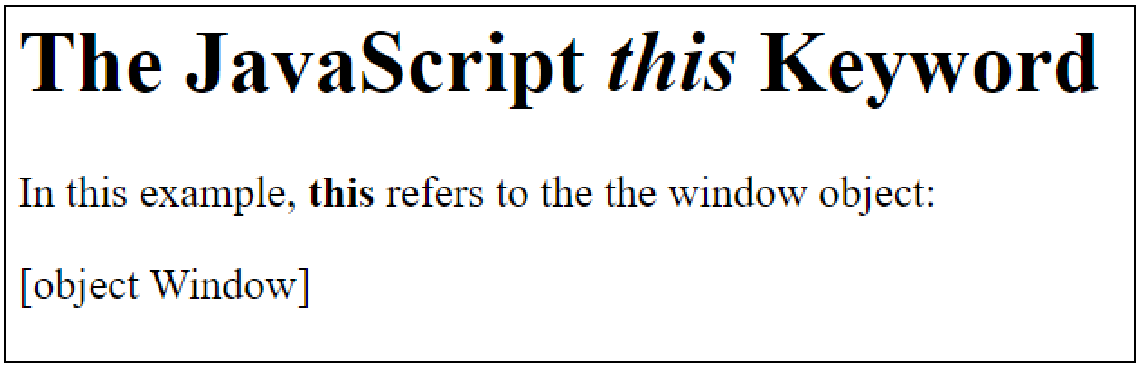
The JavaScript *this* Keyword

In this example, **this** refers to the window object:

[object Window]

Problem Statement 4: this in function (Default)

In a function, the global object is the default binding for **this**. Create a function that returns this. And invoke the function to see what gets printed in the output window.



The JavaScript *this* Keyword

In this example, **this** refers to the the window object:

[object Window]

Solution

Problem Statement 1:

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Function bind()</h1>

<p>person and member are two objects.</p>
<p>The member object borrows the fullname method from person:</p>

<p id="demo"></p>
```

```
<script>
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);

document.getElementById("demo").innerHTML = fullName();
</script>

</body>
</html>
```

Problem Statement 2:

```
<!DOCTYPE html>
<html>
<body>

<h1>The JavaScript <i>this</i> Keyword</h1>
<p>In this example, <b>this</b> refers to the <b>person</b> object.</p>
<p>Because <b>fullName</b> is a method of the person object.</p>

<p id="demo"></p>

<script>
// Create an object:
const person = {
  firstName: "John",
  lastName: "Doe",
  id: 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};

// Display data from the object:
document.getElementById("demo").innerHTML = person.fullName();
</script>

</body>
</html>
```

Problem Statement 3:

```
<!DOCTYPE html>
<html>
<body>

<h1>The JavaScript <i>this</i> Keyword</h1>

<p>In this example, <b>this</b> refers to the window object:</p>

<p id="demo"></p>

<script>
let x = this;
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

Problem Statement 4:

```
<!DOCTYPE html>
<html>
<body>

<h1>The JavaScript <i>this</i> Keyword</h1>

<p>In this example, <b>this</b> refers to the the window object:</p>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = myFunction();

function myFunction() {
  return this;
}
</script>

</body>
</html>
```