JavaScript Function apply()

Back to JS page

Table of Contents

- JavaScript Function apply()
  - Using apply() to Set this
    - Example 1
  - The Difference Between call() and apply()
    - Example 2
  - The apply() Method with Arguments
    - Example 3
  - Simulate Array Methods with apply()
    - Example 4
  - Document
  - Reference

Using apply() to Set this

The `apply()` method is similar to `call()`. The difference is:

- `call()` expects arguments passed **separately**
- `apply()` expects arguments passed as an **array**

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

<h2>JavaScript Function apply()</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Using apply() to Set this</h4>
<p id="demo"></p>

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

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

const person2 = {
  firstName: "Mary",
  lastName: "Smith"
};

document.getElementById("demo").innerHTML =
  person.fullName.apply(person1) + "<br>" +
  person.fullName.apply(person2);
</script>

</body>
</html>
```

Example 1

Result View Example

The Difference Between call() and apply()

The difference between `call()` and `apply()` is how arguments are passed:

- `call()` : arguments are passed **individually** (comma separated)
- `apply()` : arguments are passed as an **array**

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

<h2>JavaScript Function apply()</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The Difference Between call() and apply()</h4>
<p id="demo"></p>

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

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

// call() - arguments passed individually
let result1 = person.fullName.call(person1, "Oslo", "Norway");

// apply() - arguments passed as an array
let result2 = person.fullName.apply(person1, ["Oslo", "Norway"]);

document.getElementById("demo").innerHTML =
  "call(): " + result1 + "<br>" +
  "apply(): " + result2;
</script>

</body>
</html>
```

Example 2

Result View Example

The apply() Method with Arguments

The `apply()` method accepts an array of arguments. This is useful when you have an array of data and want to pass it to a function:

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

<h2>JavaScript Function apply()</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>The apply() Method with Arguments</h4>
<p id="demo"></p>

<script>
function greet(greeting, punctuation) {
  return greeting + ", " + this.name + punctuation;
}

const person = {
  name: "Alice"
};

const args = ["Hello", "!"];

document.getElementById("demo").innerHTML = greet.apply(person, args);
</script>

</body>
</html>
```

Example 3

Result View Example

Simulate Array Methods with apply()

You can use `apply()` to simulate array methods like `Math.max()` and `Math.min()` on arrays:

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

<h2>JavaScript Function apply()</h2>
<p>From w3schools.com, Experiment by Teeratus_R</p>

<h4>Simulate Array Methods with apply()</h4>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>

<script>
const numbers = [4, 7, 1, 9, 3, 5];

// Math.max normally cannot accept an array directly
let max = Math.max.apply(null, numbers);

// Math.min similarly
let min = Math.min.apply(null, numbers);

document.getElementById("demo1").innerHTML = "Numbers: " + numbers;
document.getElementById("demo2").innerHTML = "Max value: " + max;
document.getElementById("demo3").innerHTML = "Min value: " + min;
</script>

</body>
</html>
```

Example 4

Result View Example

Document

Document in project

You can Download PDF file.

Reference

- W3Schools JavaScript Function apply()