‘This’ Keyword in Javacript

In JavaScript, the `this` keyword refers to the context in which a function is executed. Its value is not determined by where the function is defined, but by how it is called. This dynamic binding is a core concept in JavaScript and can lead to different values for `this` in various scenarios.

Common Scenarios for `this`:

- **Global Context:** When `this` is used alone in the global scope (outside of any function or object), it refers to the global object. In a web browser, this is the `window` object. In Node.js, it's the `global` object.

JavaScript
```
    console.log(this); // In a browser, this logs the window object
```

- **Object Methods:** When `this` is used inside a method of an object, it refers to the object itself that the method belongs to.

JavaScript
```
    const person = {
        firstName: "John",
        lastName: "Doe",
        fullName: function() {
            return this.firstName + " " + this.lastName; // 'this' refers to
'person'
        }
    };
    console.log(person.fullName()); // Output: John Doe
```

- **Functions (Non-Strict Mode):** In a regular function call (not as a method of an object) and not in strict mode, `this` also refers to the global object.

JavaScript
```
    function showThis() {
        console.log(this); // In a browser, this logs the window object
    }
    showThis();
```

- **Functions (Strict Mode):** When `'use strict'` is enabled, `this` inside a function that is not bound to an object will be `undefined`.

JavaScript
```
'use strict';
function showStrictThis() {
    console.log(this); // Output: undefined
}
showStrictThis();
```

- **Event Handlers:** In the context of an event listener, `this` refers to the DOM element that received the event.

JavaScript
```
document.getElementById('myButton').addEventListener('click', function()
{
    console.log(this); // 'this' refers to the button element
});
```

- `call()`, `apply()`, and `bind()` Methods: These methods explicitly allow you to set the value of `this` for a function call.
  - `call()`: Invokes a function with a specified `this` value and arguments provided individually.
  - `apply()`: Invokes a function with a specified `this` value and arguments provided as an array (or array-like object).
  - `bind()`: Creates a new function that, when called, has its `this` keyword set to the provided value.

JavaScript
```
const anotherPerson = { name: "Jane" };
function greet() {
    console.log(`Hello, ${this.name}`);
}
greet.call(anotherPerson); // Output: Hello, Jane
```