@CODE.CLASH

JavaScript Closures

NEXT

Hey Everyone 👋

Closure is one of important concept in JavaScript. It is widely discussed and still confused concept. Let's understand what the closure is.

Before you learn about closures, you need to understand two concepts:

- Nested Function
- Returning a function

NEXT →

Nested Function

- In JavaScript, a function can also contain another function. This is called a nested function.

```
// outer function
function greet(name) {
  // inner function
  function displayName() {
    console.log('Hi' + ' ' + name);
  }

  // calling inner function
  displayName();
}

// calling outer function
greet('Imtiyaz'); // Hi Imtiyaz
```

NEXT →

Returning a Function

- In JavaScript, you can also return a function within a function.

```
function greet(name) {
  function displayName() {
    console.log('Hi' + ' ' + name);
  }
  // returning a function
  return displayName;
}

const g1 = greet('Imtiyaz');

console.log(g1); // returns the function definition

g1(); // calling the function
```

Output

```
function displayName() {
    console.log('Hi' + ' ' + name);
}
Hi Imtiyaz
```

NEXT →

Javascript Closure

- Closure means that an inner function always has access to the variable of its outer function, even after the outer function has returned.

```
function OuterFunction() {
  // variable defined outside the inner function
  let name = 'Imtiyaz';

  function InnerFunction() {
    // accessing OuterFunction name variable
    return 'Hi' + ' ' + name;
  }

  return InnerFunction;
}

var innerFunc = OuterFunction();

console.log(innerFunc);   // returns the function definition
console.log(innerFunc()); // returns the value
```

Output

```
function InnerFunction() {
  // accessing OuterFunction name variable
    return 'Hi' + ' ' + name;
}
Hi Imtiyaz
```

NEXT →

How it works :

1. In the above example, when OuterFunction() function is called, it returns the function definition of InnerFunction.
2. Here, innerFunc is a reference to the InnerFunction() function. beacuse of our first point.
3. When innerFunc() is called, it still has access to the OuterFunction() function.
4. When we run console.log(innerFunc), it returns the function definition.
5. When we run console.log(innerFunc()), it returns InnerFunction() value. because of our Second point.

NEXT →

Let's have a look at another example.

```
function calculate(x) { // OuterFunction

  function multiply(y) {  // InnerFunction
    return x * y;
  }
  return multiply; // return innerFunction
}

const multiply3 = calculate(3); // closures
const multiply4 = calculate(4); // closures

console.log(multiply3); // returns calculate function definition
console.log(multiply3()); // NaN

console.log(multiply3(6)); // 18
console.log(multiply4(2)); // 8
```

NEXT →

How it works :

- the calculate() function takes a single argument x and returns the function definition of the multiply() function.
- The multiply() function takes a single argument y and returns x * y.
- Both multiply3 and multiply4 are closures.
- The calculate() function is called passing a parameter x.
- When multiply3(6) and multiply4(2) are called, the multipy() function has access to the passed x argument of the outer calculate() function.

NEXT →

Best Of Luck :)

NEXT →