Arrow functions in Javascript | JavaScript Tutorial In Hindi #41

In this tutorial, we will learn about how to use the JavaScript **arrow function** to write more concise code for function expressions. Before learning about arrow function expressions, first, let us briefly review traditional JavaScript functions in order to better show the difference of arrow functions from traditional JavaScript functions.

As we have studied in the previous tutorial, the function is a group of reusable code which can be called anywhere in the program. This eliminates the need to rewrite the same code. The function declaration is a named function written with the function keyword. To get more detailed knowledge about functions, check the ***tutorial#10***.

Here is an example of a multiply function that returns the product of two parameters:

```
function mul(a, b) {
return a * b
}//calling function
mul(3,3) //returns 9
```

What are Arrow Functions?

One of the most famous features in modern JavaScript is the arrow function.ES6 arrow functions provide us an alternative way to write a more concise and shorter syntax compared to the traditional function expression. Here is an syntax of arrow functions:

let myfunc = (arg1, arg2, ...argN) => expression

Let's see another example of traditional function expression that adds two numbers:

```
let addition = function(x,y) {
```

```
 return x + y;
}
console.log(addition(10,10)); // returns 20
```

Now lets see an example of arrow function which is equivalent to the above addition() function expression:

```
let addition = (x,y) => x + y;
console.log(addition(10,10)); // 20;
```

In the above example, the arrow function has only one expression x + y so it returns the result of the expression (10+10=20).
Here is an another example of arrow functions that will print " Hello World":

```
greet = () => {
   return "Hello World!";
 }
```

If the function has one statement, and the statement returns a value or string, we can remove the brackets *and* the return keyword:

```
greet = () => "Hello World!";
```

Limitations of Arrow Functions:-

An **arrow function expression** is an alternative to a traditional function expression, but there are some limitations:

- Arrow functions do not have its own bindings to this or super, and should not be used as methods.
- It is not suitable for the call, apply and bind methods, which generally rely on establishing a scope.
- Arrow functions cannot be used as constructors.

Arrow functions are a powerful addition to ES6, but we have to be careful while using them. There are some places where arrow functions are not usable, and this can cause difficulty for us to track errors, especially if we do not understand how they really work. Arrow functions are the best choice when working with closures or callbacks, but it is not a good choice when working with object methods or constructors.

Summary:-

Arrow functions are handy for one-liners. They come in two flavors:

1. Without curly braces: (...args) => expression, where at the right side is an expression. The function evaluates the expression and returns the result.
2. With curly braces: (...args) => { body }, the brackets allow us to write multiple statements inside the function, but in such scenario, we need an explicit return to return something.

Code as described/written in the video

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXC
    <title>Document</title>
</head>
<body>
    <div class="container">
        <h1 id="heading" class='yourhead rhia is'> Welcome to Code With Harry</h1>
        <div id="myfirst" class="child red good" id="first">child 1

            <ul class="this" id='myul'>
                <li class="childul" id='fui'>this</li>
                <li class="childul">is</li>
                <li class="childul">a</li>
                <li class="childul">list </li>
                <li class="childul" id='lui'>of my dreams</li>
        </div>
        <div class="child">child 2</div>
        <div class="child red">child 3</div>
        <div class="child">child 4</div>
        <form action="none.html" method="post">
            <a href="//codewithharry.com">Go to Code With Harry</a>
            <br>
            <br>
            Search this website: <input type="text" name="Hello" id="">
            <button id="btn">Submit form</button>
            <!-- <input type="button" id='btn' value="submit"> -->
        </form>
    </div>
    <br>
    <div class="no">this is a dummy div1</div>
    <div class="no">this is a dummy div2</div>
    <div class="no">this is a dummy div3</div>
```

```
    <div class="container">
    <h1>Student list</h1>
    <ul id="students"></ul>

        <button id="myBtn" class="btn btn-primary">Your Button</button>
        <button class="btn btn-primary">Fetch Data</button>
        <div id="content"></div>
    </div>
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhz
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ
</body>
<!-- <script src="js/tut12.js"></script> -->
<!-- <script src="js/tut14.js"></script> -->
<!-- <script src="js/tut15.js"></script> -->
<!-- <script src="js/tut16.js"></script> -->
<!-- <script src="js/tut17.js"></script> -->
<!-- <script src="js/tut18.js"></script> -->
<!-- <script src="js/tut20.js"></script> -->
<!-- <script src="js/tut21.js"></script> -->
<!-- <script src="js/tut23.js"></script> -->
<!-- <script src="js/tut24.js"></script> -->
<!-- <script src="js/tut25.js"></script> -->
<!-- <script src="js/tut27.js"></script> -->
<!-- <script src="js/tut28.js"></script> -->
<!-- <script src="js/tut30.js"></script> -->
<!-- <script src="js/tut31.js"></script> -->
<!-- <script src="js/tut32.js"></script> -->
<!-- <script src="js/tut34.js"></script> -->
<!-- <script src="js/tut37.js"></script> -->
<!-- <script src="js/tut39.js"></script> -->
<!-- <script src="js/tut39b.js"></script> -->
<!-- <script src="js/tut41.js"></script> -->
<!-- <script src="js/tut43.js"></script> -->
<script src="js/tut44.js"></script>
</html>
```

Code as described/written in the video

Copy

```
console.log('this is tutorial 41');

// ARROW FUNCTIONS

// Creating a regular function
// const harry = function (){
//     console.log("This is the best person ever")
// }

// Converting it to an arrow function
// const harry = ()=>{
//     console.log("This is the best person ever")
// }
// harry();

// function returning something
// const greet = function(){
//     return "Good Morning";
// }

// One liners do not require braces/return
// one line will automatically return
// const greet = () =>  "Good Morning";

// const greet = () =>  ({name: "harry"});

// Single parameters do not need parenthesis
// but you will have to put parenthesis if there are multiple paramteres
const greet = name =>  "Good Morning " + name + ending;



console.log(greet('Harry'))
```

Previous Next

CodeWithHarry