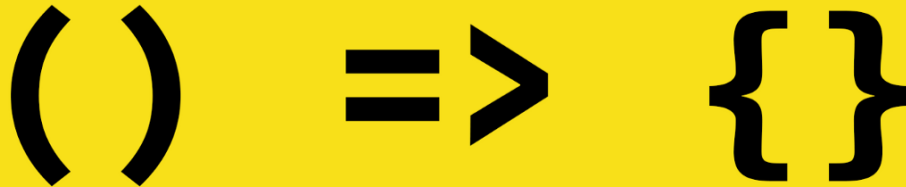


#javascript : Arrow functions or lambda functions



ES6 introduced the concept of fat arrow functions:

- 1) Arrow function as a single line, anonymous expression

```
const double = x => x*2  
console.log(double(2)) //4
```

- 2) Arrow function with multiple parameters

```
const add = (x,y) => x+y  
console.log(add(2,3)) //5
```

- 3) Arrow function with multiple statements in the body

```
const addAndDouble = (x,y) => {  
    const sum = x+y  
    return sum*2  
}  
console.log(addAndDouble(2,3)) //10
```

But what is so special about these arrow functions?

There are differences between *arrow functions* and *traditional functions*, as well as some limitations:

- Arrow functions don't have their own bindings to [this](#), [arguments](#) or [super](#), and should not be used as [methods](#).
- Arrow functions don't have access to the [new.target](#) keyword.
- Arrow functions aren't suitable for [call](#), [apply](#) and [bind](#) methods, which generally rely on establishing a [scope](#).
- Arrow functions cannot be used as [constructors](#).
- Arrow functions cannot use [yield](#), within its body.

Reference:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/Arrow_functions

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

follow and add Rizwan Mushtaq Dhudhaal  in your friend list