Arrow functions in JavaScript

geeksforgeeks.org/arrow-functions-in-javascript

Arrow functions{()=>} are a clear and concise method of writing normal/regular Javascript functions in a more accurate and shorter way. **Arrow functions** were introduced in the ES6 version. They make our code more structured and readable.

Arrow functions are anonymous functions i.e. they are functions without a name and are not bound by an identifier. Arrow functions do not return any value and can be declared without the function keyword. They are also called **Lambda Functions**.

Advantages of Arrow functions:

- Arrow functions reduce the size of the code
- The return statement and functional braces are optional for single-line functions.
- It increases the readability of the code.
- Arrow functions provide a lexical this binding, which means that they inherit the value of “this” from the enclosing scope. This feature can be advantageous when dealing with event listeners or callback functions where the value of “this” can be uncertain.

Syntax:

```
const gfg= () => {
    console.log("Hi Geek!");
}

gfg();
```

The below examples show the working of the Arrow functions in Javascript.

Arrow functions without parameters:

Javascript

```
const gfg= () => {
console.log("Hi from GeekforGeeks!");
}
gfg();
```

Output

```
Hi from GeekforGeeks!
```

Arrow functions with parameters:

Javascript

```
const gfg= (x,y,z) => {
console.log(x+y+z)
}
gfg(10,20,30);
```

Output

```
60
```

Arrow functions with default parameters:

Javascript

```
const gfg= (x,y,z=30) => {
console.log(x+ " "+ y +" "+ z);
}
gfg(10,20);
```

Output

```
10 20 30
```

Limitations of Arrow functions:

- Arrow functions do not have the prototype property
- Arrow functions cannot be used with the new keyword.
- Arrow functions cannot be used as constructors.
- These functions are anonymous and it is hard to debug the code.

- Arrow functions cannot be used as generator functions that use the yield keyword to return multiple values over time.