Sunil Vishwakarma
@linkinsunil

Types of Functions

in JavaScript

first class function

higher order function

first order function

unary function

pure function

currying function

Explained ⟶

if this helped you, please like and share it

Sunil Vishwakarma
@linkinsunil

First Class Function

In Javascript, functions are first class objects. First-class functions means when functions in that language are treated like any other variable.

In such a language, a function

- can be passed as an argument to other functions,
- can be returned by another function and
- can be assigned as a value to a variable.

For example, in the below example, handler function assigned to a listener.

```
const handler = () =>
console.log("Follow Sunil Vishwakarma");

document.addEventListener("click", handler);
```

swipe ⟶

Higher Order Function

Higher-order function is a function that accepts another function as an argument or returns a function as a return value or both.

```
const firstOrderFunc = () ⇒
  console.log("Follow Sunil Vishwakarma");

const higherOrder = (ReturnFirstOrderFunc) ⇒
ReturnFirstOrderFunc();

higherOrder(firstOrderFunc);
```

First Order Function

First-order function is a function that doesn't accept another function as an argument and doesn't return a function as its return value.

```
const firstOrder = () ⇒
console.log("Follow Sunil Vishwakarma");
```

Unary Function

Unary function (i.e. monadic) is a function that accepts exactly one argument. It stands for a single argument accepted by a function.

Let us take an example of unary function,

```
const unaryFunction = (a) ⇒ console.log(a + 10);
// Add 10 to the given argument and display the value
```

swipe ⟶

Pure Function

A Pure function is a function where the return value is only determined by its arguments without any side effects. i.e, If you call a function with the same arguments 'n' number of times and 'n' number of places in the application then it will always return the same value.

Let's take an example to see the difference between pure and impure functions,

```
//Impure
let numberArray = [];
const impureAddNumber = (number) => numberArray.push(number);
//Pure
const pureAddNumber = (number) => (argNumberArray) =>
  argNumberArray.concat([number]);

//Display the results
console.log(impureAddNumber(6)); // returns 1
console.log(numberArray); // returns [6]
console.log(pureAddNumber(7)(numberArray)); // returns [6, 7]
console.log(numberArray); // returns [6]
```

Currying Function

Currying is the process of taking a function with multiple arguments and turning it into a sequence of functions each with only a single argument.

Currying is named after a mathematician Haskell Curry. By applying currying, a n-ary function turns it into a unary function.

Let's take an example of n-ary function and how it turns into a currying function,

```
const multiArgFunction = (a, b, c) ⇒ a + b + c;
console.log(multiArgFunction(1, 2, 3)); // 6

const curryUnaryFunction = (a) ⇒ (b) ⇒ (c) ⇒ a + b + c;

curryUnaryFunction(1);
// returns a function: b ⇒ c ⇒  1 + b + c

curryUnaryFunction(1)(2);
// returns a function: c ⇒ 3 + c

curryUnaryFunction(1)(2)(3);
// returns the number 6
```

swipe ⟶

I create such content daily

Connect With Me ⚡

Sunil Vishwakarma

@linkinsunil

Share with your network