Briefly explain different types of functions in JavaScript?

Types of functions in JavaScript:

- Lambda or Arrow Function
- First Class Function
- First Order Function
- High Order Function (HOF)
- Unary Function
- Currying Function
- Pure Function

Arrow Function

An **arrow function** is a shorter syntax for a function expression and does not have its own this, arguments, super, or new.target. These functions are best suited for non-method functions, and they cannot be used as constructors.

Example

```
1 const arrowFunction = () ⇒ console.log('Arrow Function');
2
```

First Class Function

First-class functions means when functions in that language are treated like any other variable.

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.

Example

```
1 const handleClick = () ⇒ console.log('Handle Click Events');
2 document.addEventListener('click', handleClick);
```

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.

Example

```
1 const arrowFunction = () ⇒ console.log('Arrow Function');
2
```

High Order Function (HOF)

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

Example

```
1 const regularFunction = () ⇒ console.log('Regular Function');
2
3 const highOrderFunction = (normalFunction) ⇒ normalFunction();
4
5 highOrderFunction(regularFunction);
```

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.

Example

```
1 const unaryFunction = (singleParam) ⇒ console.log(`Hello ${singleParam}`);
```

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
By applying currying, a n-ary function turns it into a unary function.

Currying functions are great to improve code reusability and functional composition.

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

Currying Function Example

```
1 const curryingFunction = (a) ⇒ (b) ⇒ (c) ⇒ a + b + c;
2
3 curryingFunction(1); // b ⇒ c ⇒ 1 + b + c
4 curryingFunction(1)(2); // c ⇒ 3 + c
5 curryingFunction(1)(2)(3); // 6
```

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.

Example

```
// Pure Function
const pureFunction = (a, b) ⇒ a + b;

// Impure Function
const impureFunction = (sideEffects) ⇒ fetch('Fetching data from an API');
```

Did you found it Helpful?

Mohit Sehrawat

Full Stack Developer

Like and **comment** if you found it **helpful**!

FOLLOW FOR MORE!