Javascript functions

Anonymous functions

Pass by value and pass by reference

Anonymous functions

Recursive functions

Default parameters

Advanced functions

Function type

call() method

The call() method is used to invoke a function with a specified this value and arguments provided individually.

thisArg is the value of this provided for the call to the function

arg1, arg2, arg3 are the arguments to be passed to the function

apply() method

apply() method is similar to call() to invoke a function with a specified value, but it takes arguments as array instead of individually.

thisArg is the value of this provided for the call to the function

argsArray is array of arguments to be passed to the function

bind() method

bind() method create a new function and sets value of this keyword to a specified value, along with a given sequence of arguments

thisArgs is value of this to be used when the bound function is invoked

arg1, arg2 arguments to be prefilled

differences and usecases

call() and apply() are similar but differ in accepting arguments.

We can use call() method when we know the number of arguments beforehand.

We can use apply() method when we have array of arguments.

bind() is used to create a new function with a bound this value, which is useful when we need to pass a function as a callback but want to ensure it has the correct this context.

Closures

Immediate invoked function expression

Returning multiple values

Arrow functions

Rest parameters

**Callback**

In javascript callback is a function that is passed as argument to another function and is executed at certain point within that function, either immediately / synchronously after a operation complete / asynchronously. The purpose of a callback is to allow a function to perform additional operations after the main task is completed.

* **Function passed as argument**

A callback function is simply a function that is passed into another function as argument

* **Executed at a specific point**

The callback is invoked within the parent function, either immediately / after some event.

**Types of callbacks**

* **Synchronous callbacks**

These are executed immediately during the execution of the parent function. Common uses in array methods forEach, map, and filter.

* **Asynchronous callback**

These are executed after the completion of asynchronous operation, such as reading a file, making a network request.

**Callback before promise**

Callback in asynchronous code

* Drawback of callbacks

Callback hell

Asynchronous operations that depend on each other can lead to deeply nested callback, making the code hard to read and maintain.

Error handling

Managing errors in callback chains can be cumbersome, requiring respective checks and making the code less readable.

**A modern alternative of callbacks is Promises**

Advantages of promises over callback

* Avoiding callback hell

Promises flatten the structure of asynchronous code, making it more linear and readable.

* Error Handling

Promises provide a unified catch() method that can catch errors from any part of the promise chain, reducing the need for repetitive error checks.

* Composability

Promise can be easily composed with methods like Promise.all, Promise.race enabling more complex asynchronous workflows.

**Callback after promises**

Event after the introduction of promises, callback are still used, especially in

* Legacy code

Older codebase may still use callbacks, and understanding them is important for maintaining / refactoring such code.

* Nodejs

In nodejs, callbacks are still commonly used, particularly in older APIs. Many Nodejs functions follows the error first callback pattern

* Event handling

Callback are heavily used in event handling.

**Transition from callback to promises**

* Modernizing code

Promises can be used to modernizing callback based code by wrapping asynchronous execution in a promise constructor, making the code more maintainable and readable.

* async / await

introduced in ES8 / 2017, async / await syntax builds on promise and further simplifies asynchronous code, making it similar to synchronous code while still handling asynchronous executions processes.