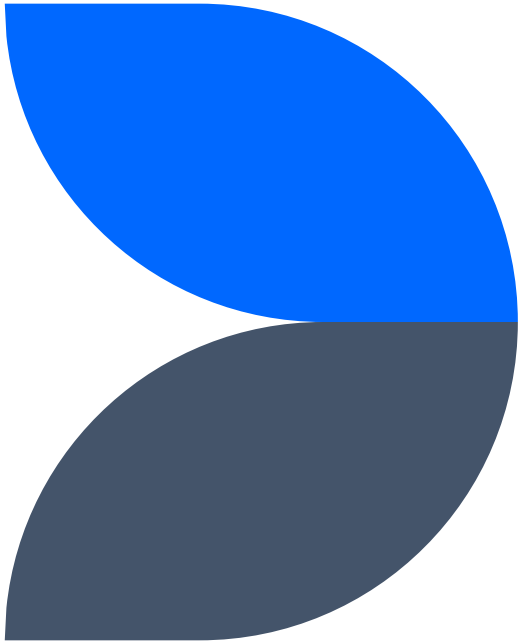
Training

JavaScript



this keyword

this refers to mainly a context, where a piece of code supposed to run.

Example: Function's body

- **this** will have a value, and the value will be determined by how the code is being executed.
- Generally, **this** is used in object methods where it refers to the object.

1. Function's context

Inside a function, the value of this depends on how the function is called.

- The value of this will be the object the function is accessed on,

  user.sayHi()

- For a regular function, the this will be referring to the global value.

this (contd.)

2. Callbacks

In the case of callback, the value of this depends on how the callback is called.
In some API methods, we can supply this value to the method in the arguments.

Example: forEach, bind

3. Arrow functions

In arrow functions, **this** retains the value of the enclosing lexical context's this.

For example, if a regular method is called on an object, it will have the **this** referring to the object, but an arrow method is called it won't be having the value of the object, but it will refer to Window object.

this (contd.)

4. Constructors

Sometimes function are used as constructors, and we create instances/objects from these constructors.

Here **this** inside the function will refer the new objects being created.

Promise

The Promise object represents an eventual completion of an asynchronous operation.

It has 3 states, pending, fulfilled, rejected

- To execute other task after the promise is completed, use then.
- If there is any error has happened, use catch to get the errors.

Promise contd.

- There is a finally block for handling task which needs to be run after all operations

Async / await

Another syntax to write promise code in synchronous fashion.

- To use the await inside a function, the function must be written with the async keyword
- Use try catch block in to handle the error part while using the async await.

Prototype

All JavaScript objects inherit properties and methods from a prototype.

- Let say, we add props/methods to the prototype of a constructor method creates instances from it then we can observe the same props/methods in the instances.
- Prototype props/methods are always live, meaning if we add/update properties and methods to the prototype, all the instances will have the latest changes with them.

Prototypal inheritance

By using the prototype of a particular object, we can inherit properties and methods to another object

Closure

Closure are created every time JS function is created. It is a capability of JS function.
(Exception: Constructor functions)

- When a function is created inside a function and then returned to another scope, then executed this inner function still can access the lexical scope – this is closure