What are callbacks ?
If we try to basically understand core meaning of callback, then it is something that we are expected to do after some time.

In JS , call back function are expected to be passed during the function call, the function which is accepting the call back as argument is going to execute the call back accordingly.

We have a lot of functions that are already present in JS, which accepts callback functions like sort, forEach, map, reduce etc.

Even we can write our own functions that accepts a callback and uses it inside the implementation.

The best part of callback is , when we are calling the functions at that time we can define what implementation of call back we have to use.
The developer who has implemented the function which accepts callback as argument has now given power to the user to change the implementation of callbacks during the function call.

Callback as a concept exist in mostly languages, in JAVA and python it exists as lambda functions, in C++ it exists as function pointer etc.

What is function expression ?
If we are defining a function and the first word of the line is not function, then it is a function expression.

IIFE: Immediately Invoked Function Expression
Wherever we define the function expression at that moment only we call it and later it doesn't exist.
IIFE is used to avoid naming collisions in our JS files. We generally don't find a lot of IIFE inside production level code, until or unless files are huge.

Nature of JS:

- Javascript is synchronous in nature, that means any piece of code that JS understands that piece of code will for sure execute line by line and if there is a piece of code that is taking a lot of time we have to wait for it.
- Javascript is single threaded.
- But still we have functions like setTimeout working in JS, How ?

- the settimeout function has the capability to wait for some milliseconds and then execute the function that we have passed as a callback after the wait is over.
-
-
-

So, JS is not very powerful, it gets a lot of features from the runtime environment. All the features that the runtime environment provides to JS, these are not native features of JS, that means this can change based on the change in environment.
JS handles the environment based features separately. It doesn’t wait for them to get executed. The only thing JS does when it hits a statement which is using a runtime feature is to just notify the runtime to execute the feature and moves forward with the next line of code. It doesn’t wait for the runtime to complete the task.
So what happens when runtime has completed the task ?
It waits in a queue called as event queue, whose entries only gets executed when there is no piece of code left in the call stack and no global code also left to be executed. This thing of when the queue is ready to execute is checked by event loop.