Day 5 - Asynchronous Functions

1. Synchronous as the name suggests is sequential means one thing at a time, one after other, sequential.
   Asynchronous means multiple things happening at the same time, be it in parts by using **parallelising** and **context switching**, etc.
   eg: You making maggie one by one or asking the friend to do you a favour by chopping veggies alongside.
2. Asynchronous Function :
   ex: Filesystem in node.js
3. Code:

```
function findSum(n){
  ans = 0;
  for (let i =1; i<n ; i++){
    ans = ans +1;
  }
  return ans;
}

function findSumTill100(){
  console.log(findSum(100));
}

setTimeout(findSumTill100, 1000)
console.log("Hello World!")
```

Here the function is parsing from findsum to findSumTill100 then the last line where it waits for 1000 ms which is 1 sec to give out the result for the main function (console.log(findSum(100))).

4. Async Functions:
   -fetch
   -setTimeOut
   -fs.readFile
5. Callback as a function is something which is needed by Asynchronous functions more than the synchronous once, because the asynchronous function will not suffice wi/o the callbacks.

```
setTimeout(function(){
console.log("Hello my name is Anish!")},20000);
```

Here 20000 means the “kamla bai” will take 20000ms or 20 sec to get the ketchup from the market.

6. **Promise**
   Clean way to write a code.
   —> Just a wrapper on top of anishReadFile

```
const fs = require("fs").promises;

//my own asynchronous function
function anishReadsFile() {
  console.log("Inside Anish's read file");
  return new Promise(function (resolve) {
    console.log("Inside Promise");
    fs.readFile("a.txt", "utf-8", function (err, data) {
      console.log("before resolve");
      resolve(data);
    });
  });
}

//callback function to call
function onDone(data) {
  console.log(data);
}

anishReadsFile().then(onDone);
```

value.then means the function at the bottom would be called back only when the first few lines of the code are done working.
then means after the completion of one function.

It’s a class which makes the async functions and callbacks slightly more readable.

You need to pass in a function as the first argument which needs to resolve as the First argument.

```
let p = new Promise(function(resolve){

});
console.log(p)
```

```
let p = new Promise(function(resolve){
  resolve("Hi there");
});
p.then(function() {
  console.log(p);
})
```

7. Async Function

```
function anishsAsyncFunction() {
  let p = new Promise(function (resolve) {
    setTimeout(function () {
      resolve("Hi there!");
    }, 3000);
  });
  return p;
}

async function main() {
  let value = await anishsAsyncFunction();
  console.log("Hi there!");
}
main();
console.log("After main");
```

8. Callback —> Promise(then) —> Async/ await syntax do the same thing.
   Async/ await is the cleanest amongst all, since no looping or nesting and only calling the await function inside the async.