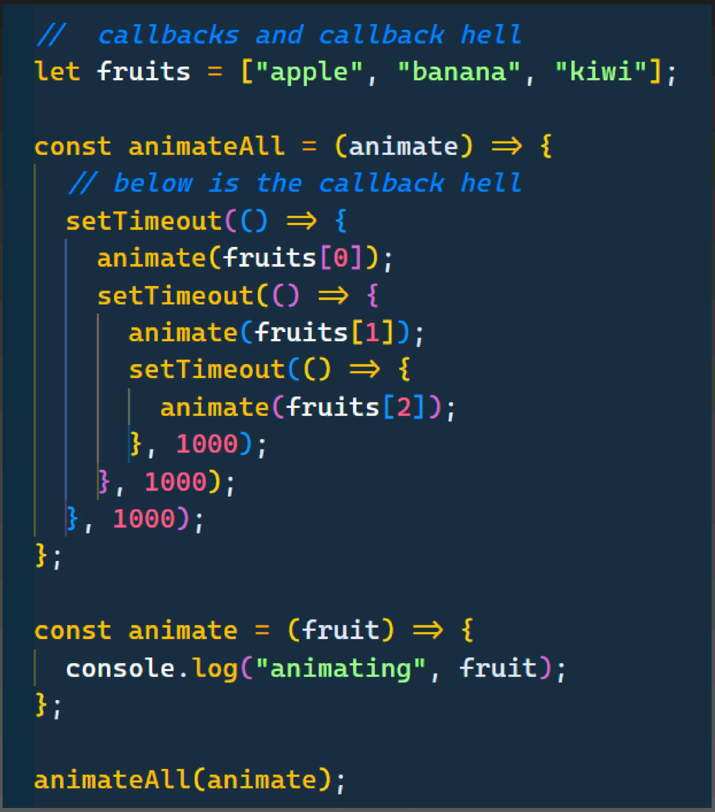
001

CALLBAKCS

A JavaScript callback is a function which is to be executed after another function has finished execution.



```javascript
//  callbacks and callback hell
let fruits = ["apple", "banana", "kiwi"];

const animateAll = (animate) => {
  // below is the callback hell
  setTimeout(() => {
    animate(fruits[0]);
    setTimeout(() => {
      animate(fruits[1]);
      setTimeout(() => {
        animate(fruits[2]);
      }, 1000);
    }, 1000);
  }, 1000);
};

const animate = (fruit) => {
  console.log("animating", fruit);
};

animateAll(animate);
```

Simply said:- Any function that is passed as an argument to another function so that it can be executed in that other function is called as a callback function. This results in callback hell.

Output

```
animating apple
animating banana
animating kiwi
```

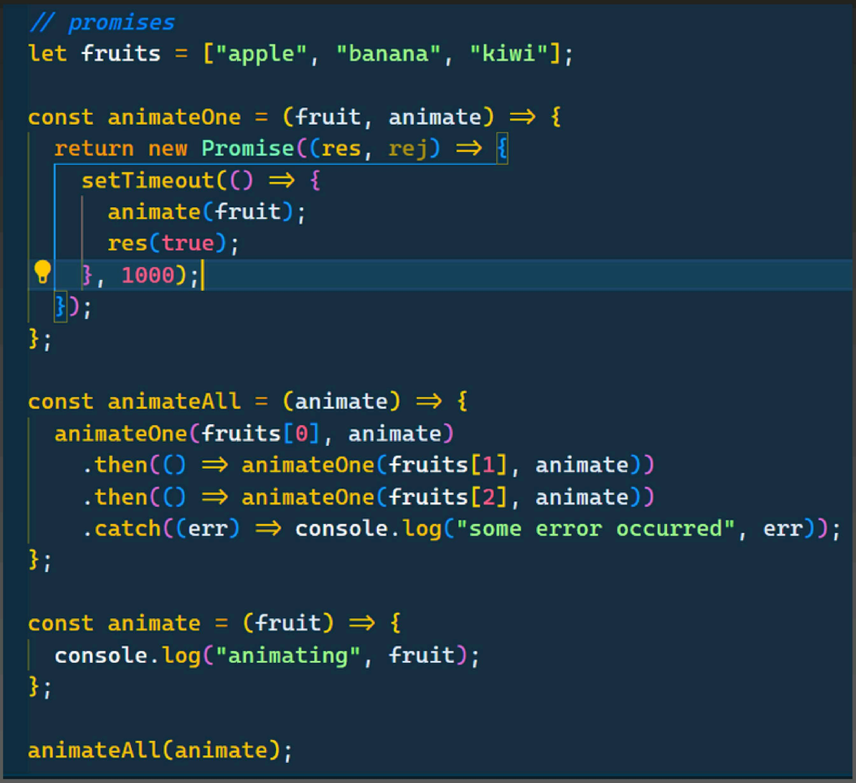
@subhajit-adhikary

JS

PROMISES

A promise is an object that will produce a single value sometime in the future. If the promise is successful, it will produce a resolved value, but if something goes wrong then it will produce a reason why the promise failed.

Simply said:- It behaves very much similar to real life promises.



```javascript
// promises
let fruits = ["apple", "banana", "kiwi"];

const animateOne = (fruit, animate) => {
  return new Promise((res, rej) => {
    setTimeout(() => {
      animate(fruit);
      res(true);
    }, 1000);
  });
};

const animateAll = (animate) => {
  animateOne(fruits[0], animate)
    .then(() => animateOne(fruits[1], animate))
    .then(() => animateOne(fruits[2], animate))
    .catch((err) => console.log("some error occurred", err));
};

const animate = (fruit) => {
  console.log("animating", fruit);
};

animateAll(animate);
```

Output

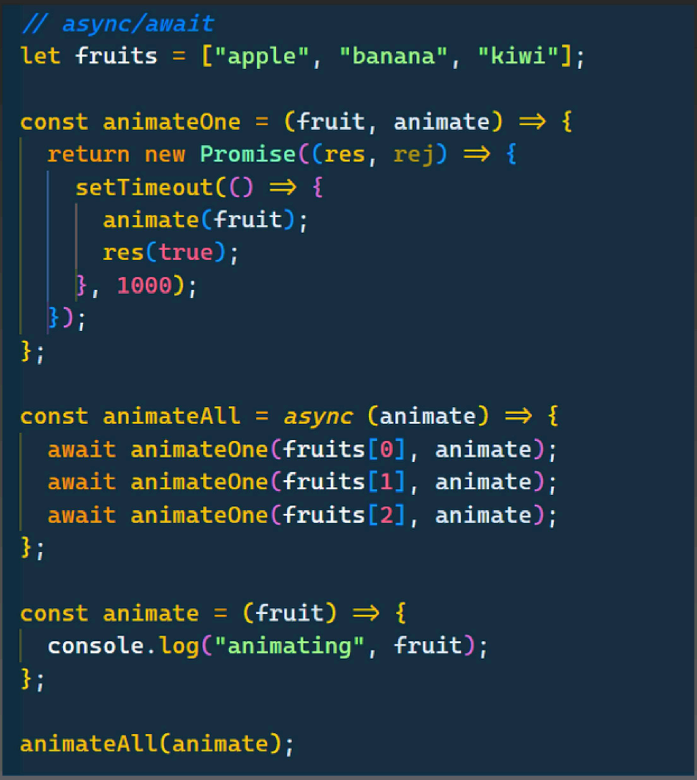
```
animating apple
animating banana
animating kiwi
```

@subhajit-adhikary

003

ASYNC/AWAIT

Async/Await makes it easier to write promises. The keyword 'async' before a function makes the function return a promise, always. And the keyword await is used inside async functions, which makes the program wait until the Promise resolves.



```javascript
// async/await
let fruits = ["apple", "banana", "kiwi"];

const animateOne = (fruit, animate) => {
  return new Promise((res, rej) => {
    setTimeout(() => {
      animate(fruit);
      res(true);
    }, 1000);
  });
};

const animateAll = async (animate) => {
  await animateOne(fruits[0], animate);
  await animateOne(fruits[1], animate);
  await animateOne(fruits[2], animate);
};

const animate = (fruit) => {
  console.log("animating", fruit);
};

animateAll(animate);
```

Output

```
animating apple
animating banana
animating kiwi
```

@subhajit-adhikary