8.14 async and await

Async function

An async function provides simpler syntax for creating an asynchronous function that returns a Promise. An ***async function*** is a function that is declared with the `async` keyword. The ***async keyword*** must appear in a function declaration before the `function` keyword. An async function fulfills the Promise by returning a value, or rejects the Promise by throwing an exception.

PARTICIPATION ACTIVITY 8.14.1: Async function returns a Promise.

1 2 3 4 5 2x speed

```
async function loadFromCloudAsync(filename) {
   if (filename === "Hello.txt") {
      return "Hello World!";
   }
   else {
      throw Error("File does not exist");
   }
}

function dataLoaded(value) {
   console.log("Data loaded: " + value);
}

function loadFailed(reason) {
   console.error(reason.toString());
}
```

```
loadFromCloudAsync("Hello.txt")
   .then(dataLoaded).catch(loadFailed);
```

Data loaded: Hello World!

```
loadFromCloudAsync("Otherfile.txt")
   .then(dataLoaded).catch(loadFailed);
```

Error: File does not exist

loadFailed() logs the error message to the console.

Captions

1. The async keyword appears before the function declaration, so loadFromCloudAsync() returns a Promise.
2. Calling loadFromCloudAsync() with "Hello.txt" returns a fulfilled Promise with value "Hello World!".
3. dataLoaded() logs the fulfilled value to the console.
4. For all other filenames, loadFromCloudAsync() returns a rejected Promise with an Error reason by throwing the Error.

5. loadFailed() logs the error message to the console.

Feedback?

PARTICIPATION ACTIVITY 8.14.2: Async function.

Complete the async function, which returns a fulfilled Promise that resolves to true if the given number is even or false if the number is odd. If the argument is not a number, the async function rejects the Promise.

```
__A__ function isEvenAsync(num) {
   if (!Number.isInteger(num)) {
      __B__ Error("Argument is not an integer");
   }

   __C__ num % 2 == 0;
}

// Should resolve to false
isEvenAsync(4)
   .then(result => alert("Result: " + result))
   .catch(error => alert(error));

// Should be rejected
isEvenAsync("dog")
   .then(result => alert("Result: " + result))
   .catch(error => alert(error));
```

If unable to drag and drop, refresh the page.

C	`return` The async function fulfills the Promise by returning a value.	Correct
A	`async` An async function must begin with `async` to return a Promise.	Correct
B	`throw` The async function rejects the Promise by throwing an Error.	Correct

Reset

Feedback?

await keyword

The `await` keyword simplifies waiting for a Promise to resolve. The ***await keyword*** is used in an async function to wait for a Promise object to resolve before proceeding to the next statement. Waiting for an asynchronous operation to complete pauses the async function's execution. Then, after the Promise eventually resolves, execution resumes inside the async function at the statement following the `await`.

PARTICIPATION ACTIVITY 8.14.3: await allows an async function to wait for completion of an asynchronous operation.

1 2 3 4 5 6 2x speed

```
function loadData(filename) {
   console.log("Loading data...");
   let promise = loadFromCloudAsync(filename);
   promise.then(function(data) {
      console.log("Data loaded: " + data);
   });
}
```

```
loadData("Hello.txt");
```

Loading data...
Data loaded: Hello World!

```
async function loadDataAsync(filename) {
   console.log("Loading data...");
   let data = await loadFromCloudAsync(filename);
   console.log("Data loaded: " + data);
}
```

```
loadDataAsync("Hello.txt");
```

Loading data...
Data loaded: Hello World!

When the load completes and the Promise resolves, execution resumes in loadDataAsync

Captions

1. loadData() calls loadFromCloudAsync(), which returns a Promise object representing an asynchronous load operation.
2. The then() method supplies a callback for when the Promise is fulfilled.
3. When the Promise is fulfilled, the data is logged to the console.
4. Alternatively, loadData() can be implemented without a callback function by using await. Any function using the await operator must be declared with the async keyword.
5. Execution begins similarly. When the await keyword is reached, execution leaves loadDataAsync() while the load completes.
6. When the load completes and the Promise resolves, execution resumes in loadDataAsync().

Feedback?

PARTICIPATION ACTIVITY 8.14.4: await keyword.

Refer to the animation above.

1) `loadData()` and `loadDataAsync()` allow additional events to be processed while the load completes in the background.

- (•) True
- ( ) False

Correct

Both load functions are equivalent in terms of overall functionality. Using `await` just allows `loadDataAsync()` to be implemented without a callback function, and with a structure more similar to that of synchronous code.

2) `loadData()` and `loadDataAsync()` both risk an uncaught exception if `loadFromCloudAsync()` rejects the Promise.

- (•) True
- ( ) False

Correct

Both functions assume that the Promise will be fulfilled. Better practice would be to handle the rejection cases as well, to avoid an uncaught exception.

3) An exception is thrown if the `async` keyword is removed from `loadDataAsync()`.

- (•) True
- ( ) False

Correct

A JavaScript runtime throws a SyntaxError if `async` is missing from a function declaration that uses `await`.

Feedback?

Handling rejected Promises

An async function uses a try-catch statement to catch the Error object from a rejected Promise.

PARTICIPATION ACTIVITY 8.14.5: Catching the Error from a rejected Promise.

1 2 3 2x speed

```
async function loadDataAsync(filename) {
   console.log("Loading data...");
   let data = await loadFromCloudAsync(filename);
   console.log("Data loaded: " + data);
}
```

```
loadDataAsync("Unknown.txt
Loading data...
Uncaught (in promise) Error
File does not exist
```

```
async function loadDataAsync(filename) {
   console.log("Loading data...");
   try {
      let data = await loadFromCloudAsync(filename);
      console.log("Data loaded: " + data);
```

```
loadDataAsync("Unknown.txt
```

```
    }
    catch (exception) {
       console.log(exception.toString());
    }
}
```

Loading data...
Error: File does not exist

If loadFromCloudAsync() rejects the Promise in the try block, the catch block executes. The async function logs the Error and then terminates normally.

Captions

1. If loadFromCloudAsync() rejects the Promise, the async function terminates immediately because the Error is not caught. The uncaught Error is typically logged to the console.
2. The async function can use a try-catch statement to catch the Error object.
3. If loadFromCloudAsync() rejects the Promise in the try block, the catch block executes. The async function logs the Error and then terminates normally.

Feedback?

PARTICIPATION ACTIVITY 8.14.6: Handling rejected Promises.

Refer to the animation above.

1) Which statement in the top `loadDataAsync()` does NOT execute when the Promise is rejected?

- `console.log("Loading data...");`
- `let data = await loadFromCloudAsync(filename);`
- `console.log("Data loaded: " + data);` (selected)

Correct

An Error is thrown when `loadFromCloudAsync()` rejects the Promise. Since no try-catch exists to catch the Error, the top `loadDataAsync()` terminates immediately.

2) Which variable is assigned the Error object containing the rejected Promise's reason?

- `filename`
- `exception` (selected)
- `data`

Correct

In the bottom `loadDataAsync()`, `exception` is assigned the Error object. Converting `exception` to a string returns the Error object's error message.

3) What does the bottom `loadDataAsync()` function log when the

Correct

The `data` is assigned with "XYZ", so "Data loaded: XYZ" is logged. The catch block does not execute when the

Promise is fulfilled with string "XYZ"?

- (●) "Data loaded: XYZ" only
- ( ) Error message only
- ( ) "Data loaded: XYZ" and error message

Promise is fulfilled.

Feedback?

Exploring further:

- async function (MDN)
- await (MDN)

How was this section? Provide section feedback