Welcome to CoGrammar

WD Week 6 Tutorial

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Full Stack Web Development Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Full Stack Web Development Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support
- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: **Feedback on Lectures**

Lecture Overview

- ➔ Recap of Promises
- ➔ Recap of Fetch API

CoGrammar

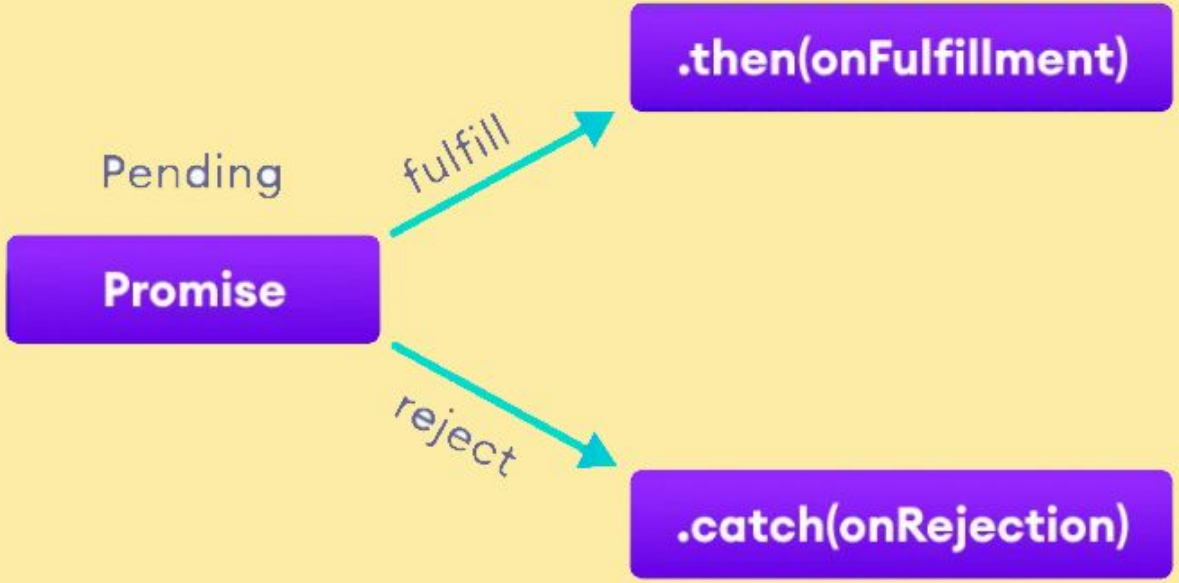
Creating a Promise

- To create a promise object, we use the **Promise()** constructor.
- The **Promise()** constructor takes a function as an argument.
- The function also accepts two functions **resolve()** and **reject()**.
- If the promise returns successfully, the **resolve()** function is called.
- If an error occurs, the **reject()** function is called.

```
let promise = new Promise(function(resolve, reject){
    //do something
});
```

CoGrammar

Creating a Promise



CoGrammar

then() method

```
let countValue = new Promise(function (resolve, reject) {
    resolve("Promise resolved");
  });

  // executes when promise is resolved successfully

  countValue
    .then(function successValue(result) {
      console.log(result);
    })

    .then(function successValue1() {
      console.log("You can call multiple functions this way.");
    });
```

CoGrammar

catch() me if you can...

- The **catch()** method is used with the callback when the promise is **rejected** or if an **error** occurs.

```
let countValue = new Promise(function (resolve, reject) {
    reject('Promise rejected');
});

// executes when promise is resolved successfully
countValue.then(
    function successValue(result) {
        console.log(result);
    },
)

// executes if there is an error
.catch(
    function errorValue(result) {
        console.log(result);
    }
);
```

CoGrammar

OMG, finally()!

- The **finally()** method gets executed when the promise is either resolved successfully or rejected.

```
// returns a promise
let countValue = new Promise(function (resolve, reject) {
    // could be resolved or rejected
    resolve('Promise resolved');
});

// add other blocks of code
countValue.finally(
    function greet() {
        console.log('This code is executed.');
    }
);
```

CoGrammar

Async

- We use the async keyword with a function to represent that the function is an asynchronous function.
- The async function returns a promise.

```
async function name_of_the_function(parameter1, parameter2) {
    // statements
}
```

CoGrammar

Await

```
let promise = new Promise(function (resolve, reject) {
    setTimeout(function () {
    resolve('Promise resolved')}, 4000);
});

// async function
async function asyncFunc() {

    // wait until the promise resolves
    let result = await promise;

    console.log(result);
    console.log('hello');
}

// calling the async function
asyncFunc();
```

CoGrammar

Error Handling

- You can also use the catch() method to catch the error.

```
async function f() {
    console.log('Async function.');
    return Promise.resolve(1000);
}

f().then(function(result) {
    console.log(result)
}).catch(function(err){
    // catch error and do something
});
```

CoGrammar

Error Handling

- The other way you can handle an error is by using try/catch block.

```
let promise = new Promise(function (resolve, reject) {
    setTimeout(function () {
        // resolve('Promise resolved');
        reject("Promise rejected");
    }, 4000);
});
// async function
async function asyncFunc() {
    try {
        // wait until the promise resolves
        let result = await promise;

        console.log(result);
    }
    catch(error) {
        console.log(`Error: ${error}`);
    }
}
```

CoGrammar

fetch() Method

- The fetch() function is used to initiate a request to the specified url.
- It returns a promise that resolves to the Response object representing the response to the request.
- The options parameter is an optional object containing settings for the request such as method, headers, body, etc.

```
fetch(url, options)
  .then((response) => {
    // handle response
  })
  .catch((error) => {
    // handle error
  });
```

CoGrammar

fetch() Method Parameters

- URL (required):
  - Specifies the URL to which the request is made.

- Options (optional): An object containing various settings for the request such as:
  - method: HTTP method (GET, POST, PUT, DELETE, etc.)
  - headers: Headers to include in the request
  - body: Data to send with the request (e.g., JSON, FormData)

Response Handling

- Response Object:
  - Represents the response to the request made by fetch().
  - Contains properties and methods to access response data and metadata.
- .json(): Parses the response body as JSON.

Making a GET Request

```
fetch("https://jsonplaceholder.typicode.com/posts")
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error("Error:", error));
```

CoGrammar

Making a POST Request

```
const postData = {
  username: "example",
  password: "password123",
};

fetch("https://api.example.com/login", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(postData),
})
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error("Error:", error));
```

CoGrammar

Making a DELETE Request

```
fetch("https://api.example.com/user/123", {
  method: "DELETE",
})
  .then((response) => {
    if (response.ok) {
      console.log("User deleted successfully");
    } else {
      console.error("Failed to delete user");
    }
  })
  .catch((error) => console.error("Error:", error));
```

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar