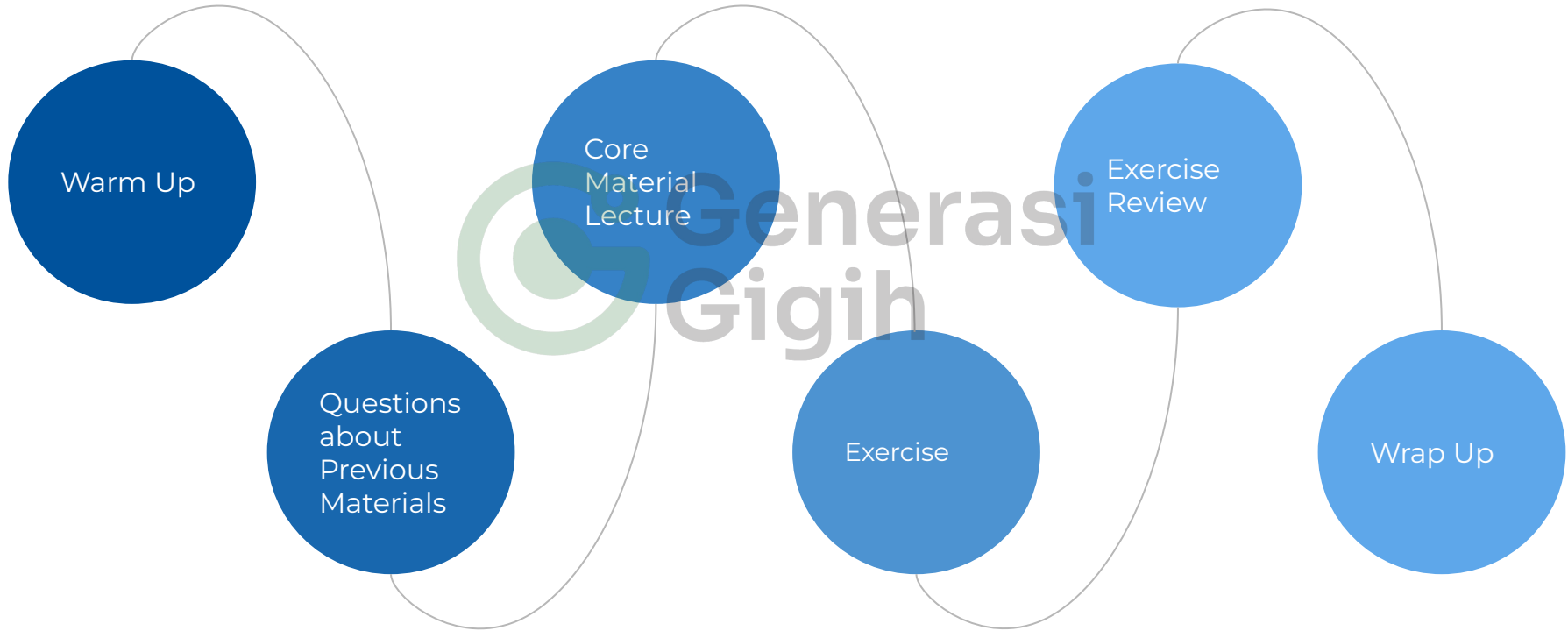


Full Stack Engineer ●●●

Module 1.2: JS Intermediate



Our Agenda



Let's Warm Up!



Let's Discuss



Error Handling

Promise

Event handler

Let's Talk About The Materials

Error handling

- process of dealing with errors that may occur during execution



```
try {  
  let num = 5  
  num.toUpperCase()  
  // throws an error because toUpperCase is not a function for numbers  
} catch (error) {  
  console.log('An error occurred: ' + error.message)  
  throw new Error('There was an error processing your request.')  
}
```

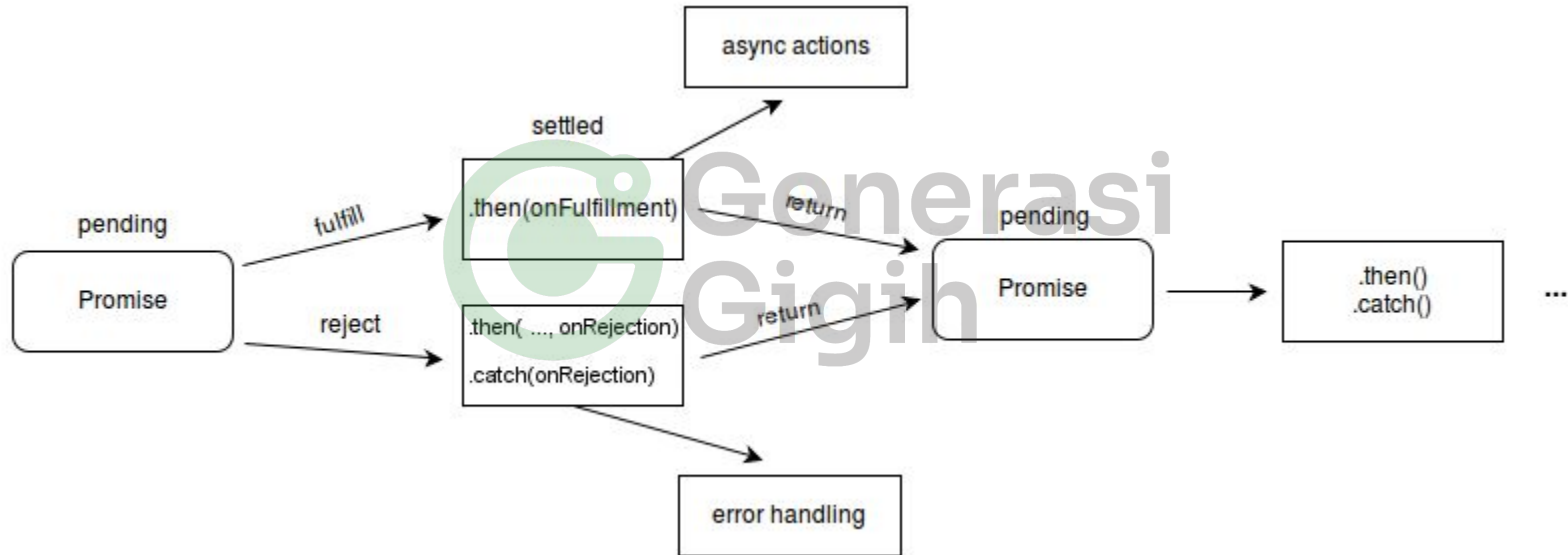
try/catch & throw

- The “try” block contains the code that may throw an error.
- If an error occurs in this block, it will immediately jump to the catch block.
- The catch block takes an error object. It will process the error there
- “throw” is a keyword that is used to manually generate and throw an exception
- Try/catch flow is meant for error handling only. For branches that actually expected, use the if/else or switch instead of using try/catch

Promises

- Object that will return result later
- It allows us to associate “handlers” when the event is success or failure
- One of usage example of Promises is for HTTP call. Promises will prevent our Web to be “blocked”, user still can do things when we call the APIs
- Promises is in one of these states: pending, fulfilled, or rejected
 - When a promise is first created, it is in a pending state.
 - The promise is said to be fulfilled when the asynchronous operation is completed successfully, and the promise object returns the result.
 - The promise is rejected when the asynchronous operation encounters an error, and the promise object returns an error message.
- We can also “chain” promises by returning another promises after first promises

Promises



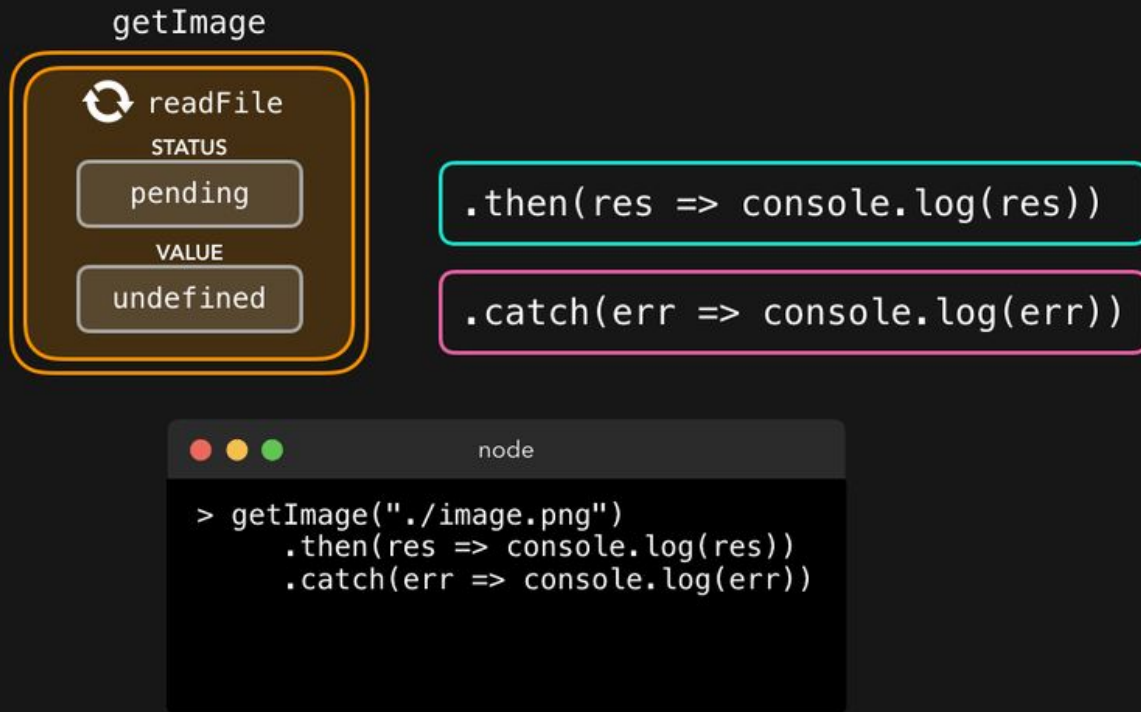
Fulfilled promise



```
node
> getImage("./image.png")
  .then(res => console.log(res))
  .catch(err => console.log(err))
```

<https://dev.to/lydiahallie/javascript-visualized-promises-async-await-5gke>

Rejected Promise



<https://dev.to/lydiahallie/javascript-visualized-promises-async-await-5gke>

Another Promise Example



```
const wait = time => new Promise((resolve) =>
  setTimeout(resolve, time));

wait(3000).then(() => console.log('World!'));

console.log("Hello!")

// Result:
// Hello!
// World!
```

Async/Await

- Built on top of Promises and provides a way to chain Promises together in a more readable and understandable way.
- The idea is, we can declare a function as "async", and then use the "await" keyword to wait for the result of a Promise before continuing the execution of the function.
- With this, asynchronous code will “looks and behaves” like synchronous code
- But with this we cannot run 2 promises concurrently, so use it with consideration


Async/Await example

```
const wait = time => new Promise(resolve => setTimeout(resolve, time));

async function asyncAwait() {
  console.log("Hello!");
  await wait(3000);
  console.log("World");
}
asyncAwait();

// Result:
// Hello!
// World!
```

Use case: HTTP Client



```
fetch('https://jsonplaceholder.typicode.com/todos/1')  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error(error));
```

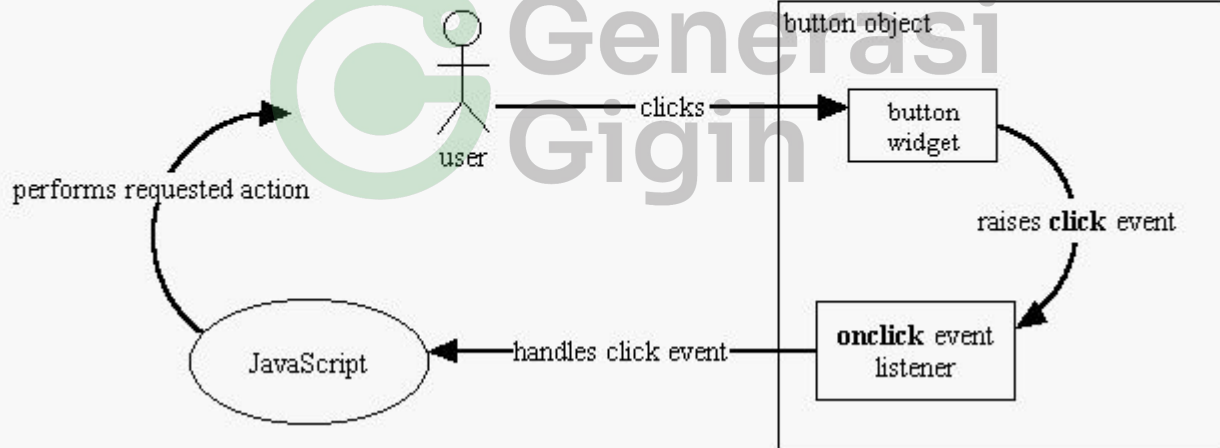
- Calling an API to get JSON data
- Fetch() return a promise, then we read the data using json() method
- In the end, we got the data. In case of error we will catch the error

Event Handler

- Functions that are triggered when a specific event occurs, such as a user clicking on a button or moving their mouse over an element
- Example: `<button onClick="alert('hello')">`
- There are many types of events that you can listen for in JavaScript
 - mouse events (click, mouseover, mouseout, etc.)
 - keyboard events (keydown, keyup, keypress)
 - form events (submit, reset, etc.).
 - Input events (change, etc.)

Event Handler

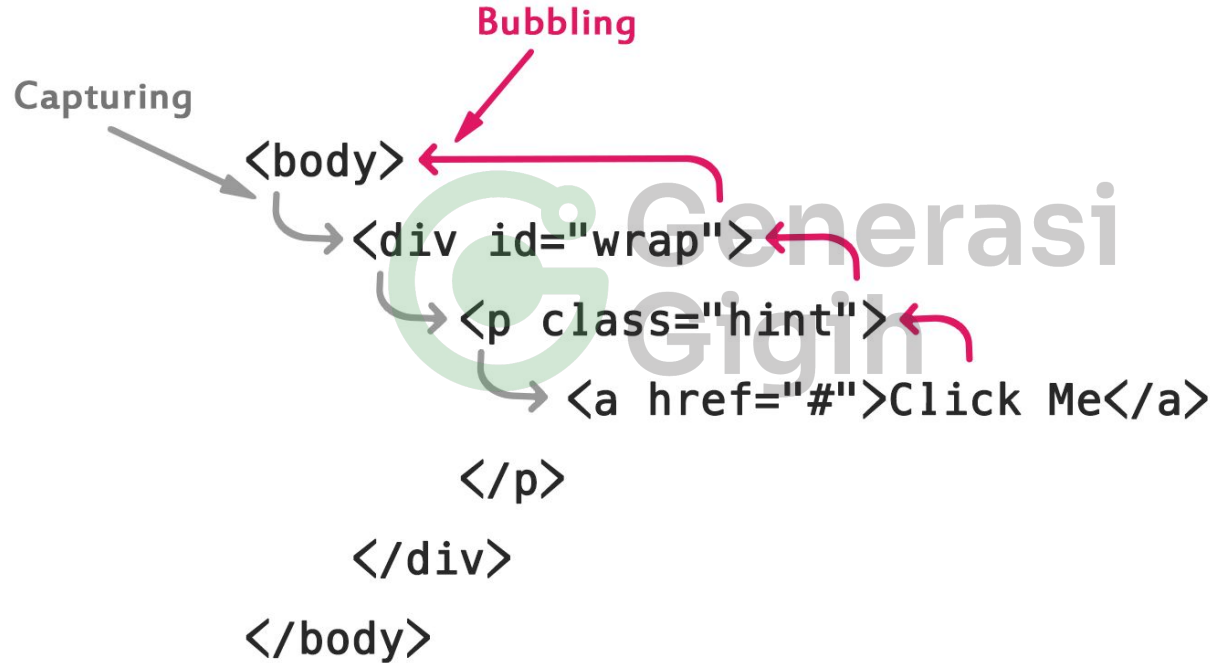
```
<button value="Click Me" onclick="alert('Thank you')" />
```



[Source](#)

Event Handler

[Source](#)



Event Handler Example

<https://jsfiddle.net/y1g8o0x2/>



```
<html>
  <body onClick="console.log('body')">
    <div onClick="console.log('div')">
      <p onClick="console.log('paragraph')">This is a paragraph</p>
    </div>
  </body>
</html>
```

Discussion/Hands On Assignment/Study Case about the material

Lets assume that we have a Song shema with this format

```
{ title: "song title", artists: [{ name: "artist name 1"}], duration: 200 }
```

Create a function that



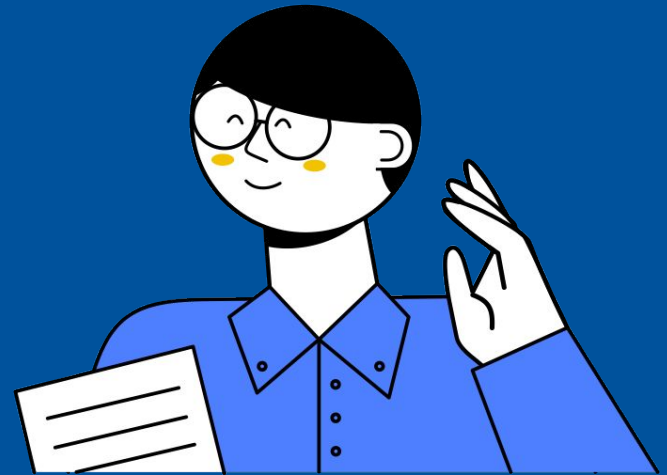
1. Accept a promise of Song list
2. Print an error message if the promise rejected.
3. Print all songs list if the promise is fulfilled

In 2 version (Promise and async/await)



Showcase Time!

Q&A!



Finally, Let's Wrap Up!