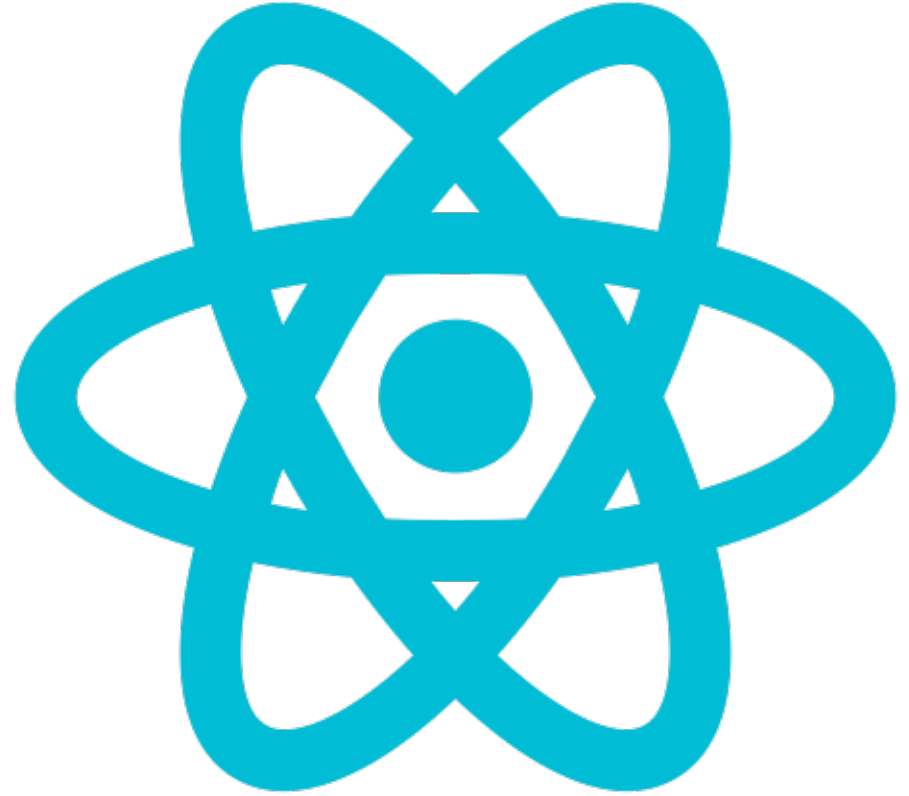
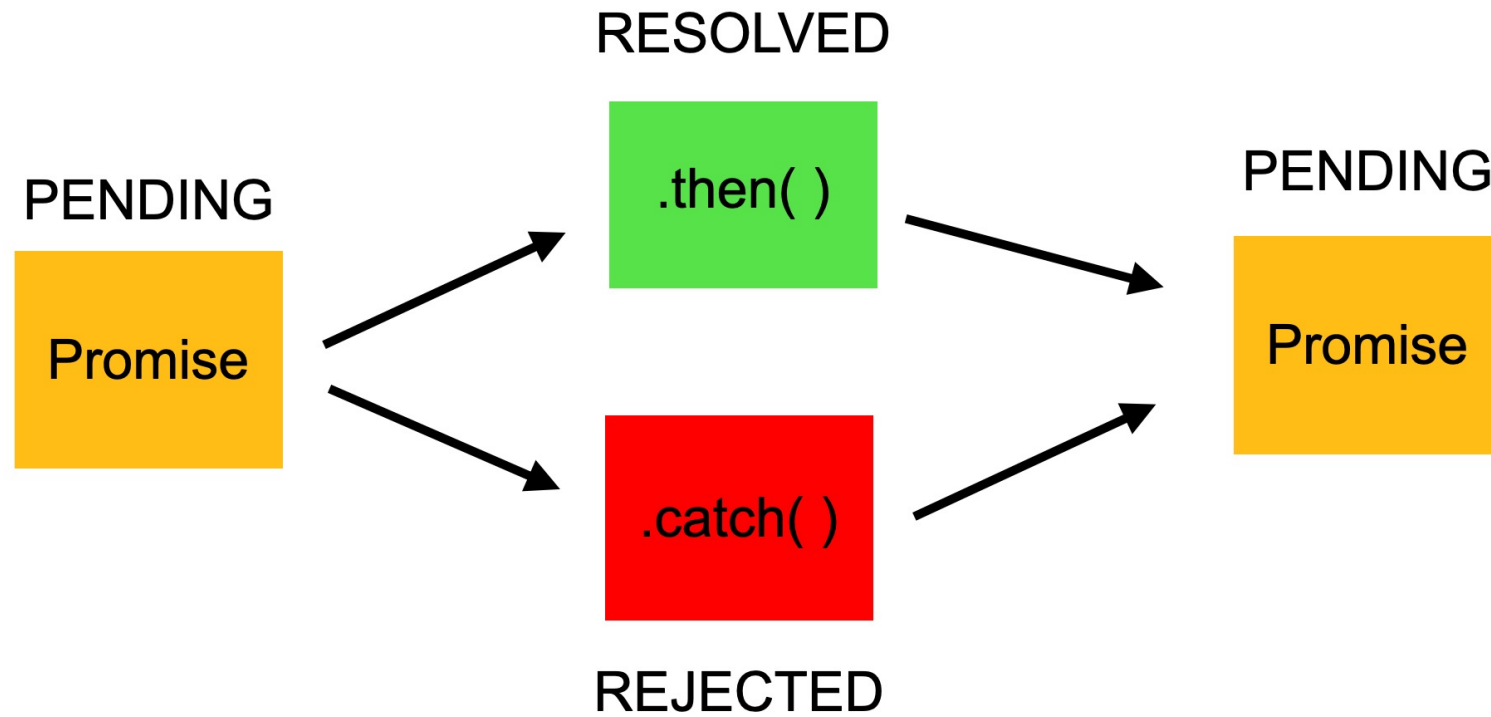


⇐ axios



Promise



What is axios

- **Axios** is an HTTP client library, helps you with just that: sending HTTP requests and managing the responses
- It is used to communicate with the backend and it also supports the Promise API that is native to JS ES6.
- Axios is a very popular (over 78k stars on Github) HTTP client, which allows us to make HTTP requests from the browser.
- `npm install axios`
- Not the news website ! <https://www.axios.com/>

Advantages of using Axios over the native Fetch API include:

- Request and response interception
- Streamlined error handling
- Protection against XSRF
- Support for upload progress
- Response timeout
- The ability to cancel requests
- Support for older browsers
- Automatic JSON data transformation

Making Http requests

- For a simple Axios POST request, the object must have a url property. If no method is provided, GET will be used as the default value.

```
// send a POST request
axios({
  method: 'post',
  url: '/login',
  data: {
    firstName: 'Finn',
    lastName: 'Williams'
  }
});
```

Even better - Shorthand methods

- `axios.request(config)`
- `axios.get(url[, config])`
- `axios.delete(url[, config])`
- `axios.head(url[, config])`
- `axios.options(url[, config])`
- `axios.post(url[, data[, config]])`
- `axios.put(url[, data[, config]])`
- `axios.patch(url[, data[, config]])`

Example Get

```
axios.get('https://api.github.com/users/mapbox')  
  .then((response) => {  
    console.log(response.data);  
  });
```

POST JSON with Axios

- Axios automatically serializes JavaScript objects to JSON when passed to the `axios.post` function as the second parameter. This eliminates the need to serialize POST bodies to JSON.
- Axios also sets the Content-Type header to `application/json`. This enables web frameworks to automatically parse the data.
- If you want to send a preserialized JSON string to `axios.post()` as JSON, you'll need to make sure the Content-Type header is set.

Custom headers

- Sending custom headers with Axios is very straightforward. Simply pass an object containing the headers as the last argument. For example:

```
const options = {  
  headers: {'X-Custom-Header': 'value'}  
};  
  
axios.post('/save', { a: 10 }, options);
```

axios.all

- One of Axios' more interesting features is its ability to make multiple requests in parallel by passing an array of arguments to the `axios.all()` method.

```
// execute simultaneous requests
axios.all([
  axios.get('https://api.github.com/users/mapbox'),
  axios.get('https://api.github.com/users/phantomjs')
])
.then(responseArr => {
  //this will be executed only when all requests are complete
  console.log('Date created: ', responseArr[0].data.created_at);
  console.log('Date created: ', responseArr[1].data.created_at);
});
```

Intercepting requests and responses

- HTTP interception is a popular feature of Axios. With this feature, you can examine and change HTTP requests from your program to the server and vice versa, which is very useful for a variety of implicit tasks, such as logging and authentication.
- Also try
`axios.interceptors.response(..`

```
// declare a request interceptor
axios.interceptors.request.use(config => {
  // perform a task before the request is sent
  console.log('Request was sent');

  return config;
}, error => {
  // handle the error
  return Promise.reject(error);
});

// sent a GET request
axios.get('https://api.github.com/users/mapbox')
  .then(response => {
    console.log(response.data.created_at);
  });
```

HTTP response status codes

- HTTP response status codes indicate whether a specific [HTTP](#) request has been successfully completed. Responses are grouped in five classes:
- [Informational responses](#) (100–199)
- [Successful responses](#) (200–299)
- [Redirects](#) (300–399)
- [Client errors](#) (400–499)
- [Server errors](#) (500–599)