Network Calls!

Most websites aren't static and need to get data from some other resource. To do this, those sites need to make network calls. Those can be made with any number of methods, however, we're going to focus on RESTful architectural for our network calls.

In the world of Node and NPM, there are many packages and methods to use to make network calls. 3 very popular methods are the `axios` and `superagent` packages, and the built in `fetch` method. Each has its own quirks. Ultimately, it all boils down to choice (except for a little extra potential work for `fetch`) as to which to use.

Making a network call is relatively simple. All you need is the method with which you'll be making that call (GET, PUT, DELETE, etc), the body (if there is one), and that's it.

Often enough, you may come across promise based syntax for network calls, but most network calling APIs allow for the usage of async await functionality.

External Packages:

Superagent and Axios are two external packages that offer a great way to make network calls. Both work fantastically, and can be used as chained promises OR with async/await.

SuperAgent:

Superagent does not automatically parse the data. Sometimes this can be a desired attribute. Superagent is often typically written with the `.end()` function when chained:

- GET:

```
superagent
  .get(url)
  .end((err, res) => {
  if (err) {
    console.error(err);
    return
  }

  const responseData = JSON.parse(res.text);
  const posts = responseData.data.children.map(obj => obj.data);
  this.setState({ posts });
});
```

Async Await:

```
makeUrlCall = async (url) => {
  try {
    const response = await superagent.get(url)
    const responseData = JSON.parse(response.text)
    const posts = responseData.data.children.map(obj => obj.data);
    this.setState({posts})
  } catch (err) {
    console.error(err)
  }
}
```

- POST (and other methods with bodies):

```
superagent
  .post(postURL)
  .send({ name: 'Manny', species: 'cat' }) // sends a JSON post body
  .set('accept', 'json') // setting header data
  .end((err, res) => {
    // Calling the end function will send the request
  });
```

Superagent sends a body with the dot property `.send()`, where we send our body.

Axios:

Axios does automatically parse the data, so it is often times much easier to work with. Additionally, Axios uses the standard `.then().catch()` chaining when written:

- GET:

```
axios
  .get(`${this.baseURL}r/${this.props.subreddit}.json`)
  .then(response => {
  const posts = response.data.data.children.map(obj => obj.data);
  this.setState({ posts });
})
  .catch(error => {
  console.error(error);
});
```

AsyncAwait

```
makeUrlCall = async (url) => {
  try {
    const response = await axios.get(url)
    const posts = responseData.data.children.map(obj => obj.data);
    this.setState({posts})
  } catch (err) {
    console.error(err)
  }
}
```

- POST (and other methods with bodies)

```
axios.post('/user', {
    firstName: 'Fred',
    lastName: 'Flintstone'
  })
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });
```

Axios takes its second argument as a body, unlike the additional dot property of superagent.

Internal:

Fetch:

Fetch is a javascript interface that accesses the HTTP pipeline. If you use `fetch` you should know that it is a relatively newer method, which may not exist in many browsers that have not been updated.

```
fetch('http://example.com/movies.json')
  .then((response) => {
    return response.json();
  })
  .then((data) => {
    console.log(data);
  });
```