OBJECT COMPUTING

WEBINAR

Calling REST Services with the Fetch API

Mark Volkmann, Partner and Principal Software Engineer
mark@objectcomputing.com

INTRODUCTION

- Fetch standard defines request and response objects and a function for obtaining a response from a request
- Defined by Web Hypertext Application Technology Working Group (WHATWG)
- This screencast presents a recommended way to use the Fetch API that takes advantage of JavaScript Promises and **`async`**/**`await`**

FETCH API BROWSER SUPPORT

- Supported by all modern web browsers
- Internet Explorer is the only popular browser without support
- Polyfill available at https://github.com/github/fetch
  - supports IE 10+

REST SERVICES

- Typically use HTTP verbs in specific way
- **POST** requests **create** a resource
- **GET** requests **retrieve** a resource
- **PUT** requests **update** a resource
- **DELETE** requests **delete** a resource
- **POST** and **PUT** requests often include a **JSON** body

UTILITY FUNCTIONS

- Useful to create a small set of utility functions that hide details of these operations
- Source file **`fetch-util.js`** does this
- Feel free to copy and use in your projects

fetch-util.js ...


```
// Change this to match the URL prefix of your REST services.
// If your project uses REST services with more than one URL prefix,
// drop use URL_PREFIX and just pass full URLs into the functions.
const URL_PREFIX = 'http://localhost:1234/';

// If there are any common options that are
// desired in all HTTP requests, place them here.
const options = {};

const headers = {'Content-Type': 'application/json'};
```

... fetch-util.js ...


```
// Can't name this "delete" because that is a JavaScript keyword.
export async function deleteResource(urlSuffix) {
  const url = URL_PREFIX + urlSuffix;
  return fetch(url, {...options, method: 'DELETE'});
}
```

… fetch-util.js …

```
export async function getJson(urlSuffix) {
  const url = URL_PREFIX + urlSuffix;
  const res = await fetch(url, options);
  if (!res.ok) throw new Error(await res.text());
  return res.json();
}

export async function getText(urlSuffix) {
  const url = URL_PREFIX + urlSuffix;
  const res = await fetch(url, options);
  if (!res.ok) throw new Error(await res.text());
  return res.text();
}
```

method defaults to **'GET'**

... fetch-util.js


```
export function postJson(urlSuffix, obj) {
  const url = URL_PREFIX + urlSuffix;
  const body = JSON.stringify(obj);
  return fetch(url, {...options, method: 'POST', headers, body});
}

export function putJson(urlSuffix, obj) {
  const url = URL_PREFIX + urlSuffix;
  const body = JSON.stringify(obj);
  return fetch(url, {...options, method: 'PUT', headers, body});
}
```

`async` / `await` ...

- Functions that use **`await`** to wait for a Promise to resolve or reject must be marked with **`async`**
- Such functions always return a **`Promise`**

... async / await

- When uses of **await** are wrapped in a **try/catch**, **Promises** that throw will cause the **catch** block to execute

```
async function doStuff() {
  try {
    const result = await someAsyncFn();
    // Do something with result.
  } catch (e) {
    // Handle error.
  }
}
```

```
const doStuff = async () => {
  try {
    const result = await someAsyncFn();
    // Do something with result.
  } catch (e) {
    // Handle error.
  }
};
```

CREATE WITH POST

OBJECT COMPUTING

inside an **async** function

```
try {
  const res = await postJson('some-url-path', someObject);
  if (res.ok) {
    // Handle success.
  } else {
    // Handle error.
  }
} catch (e) {
  // Handle error.
}
```

RETRIEVE WITH GET

OBJECT COMPUTING

inside an **async** function

```
try {
  const resource = await getJson('some-url-path');
  // Do something with resource.
} catch (e) {
  // Handle error.
}
```

UPDATE WITH PUT

OBJECT COMPUTING

inside an **async** function

```
try {
  const res = await putJson('some-url-path', someObject);
  if (res.ok) {
    // Handle success.
  } else {
    // Handle error.
  }
} catch (e) {
  // Handle error.
}
```

DELETE WITH DELETE

OBJECT COMPUTING

inside an **async** function

```
try {
  await deleteResource('some-url-path');
} catch (e) {
  // Handle error.
}
```

COMPLETE EXAMPLE

OBJECT COMPUTING

- https://github.com/mvolkmann/fetch-api-demo
- Demonstrates using **Fetch API**
  from a web app implemented in **Vue** to a
  backend implemented in **Node.js** and **Express**
  that talks to a **PostgreSQL** database

Dogs

Breed

Name

Create Update

Name	Breed	Delete
Dasher	whippet	X
Maisey	treeing walker coonhound	X
Ramsey	native american indian dog	X
Oscar Wilde	german shorthaired pointer	X

Wrap Up

- You now know everything need to call REST services using the FETCH API

LEARN MORE ABOUT OCI EVENTS AND TRAINING

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.

OBJECT
COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com