8.15 Fetch API

Introduction to Fetch

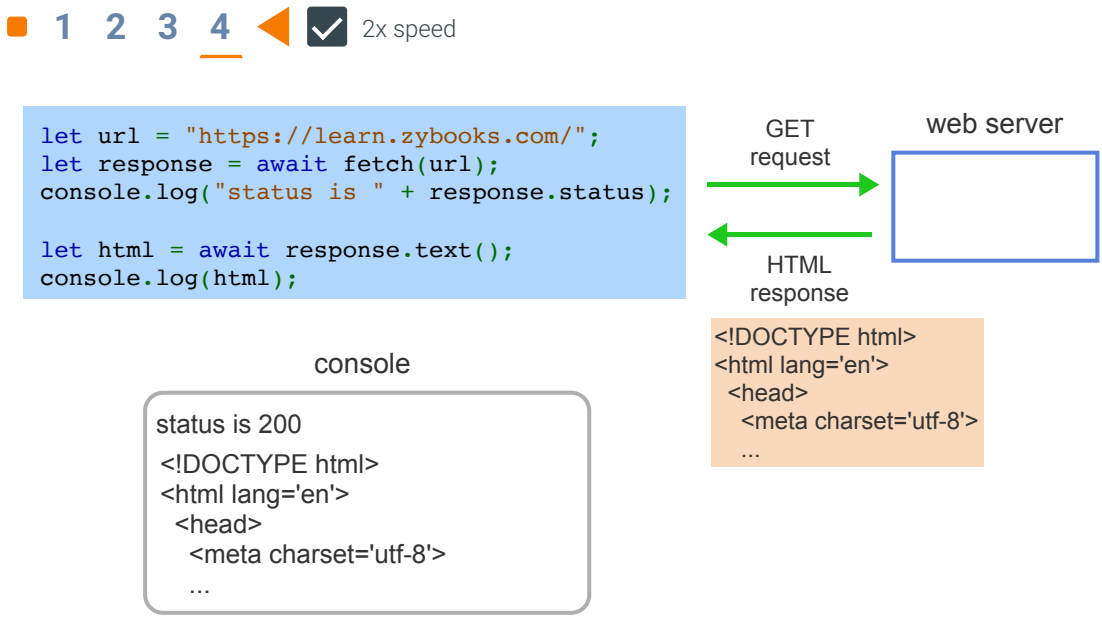
The ***Fetch API*** defines a ***fetch()*** method for sending HTTP requests and receiving HTTP responses. The `fetch()` method is a replacement for using the `XMLHttpRequest` object directly and is generally easier to use.

The `fetch()` method sends a GET request (by default) to the given URL argument and returns a Promise object. The Promise object resolves to a ***Response*** object, which contains information about the HTTP response and methods for retrieving the response body. Some properties and methods of `Response` include:

- The ***response.status*** property is the HTTP response's status code (200, 301, 404, etc.)
- The ***response.ok*** property is true if the HTTP response's status code is 2xx, false otherwise.
- The ***response.headers*** property is an object containing the HTTP response headers.
- The ***response.text()*** method returns a Promise that resolves with the textual body of the HTTP response.

The web browser restricts `fetch()` from sending a cross-origin HTTP request, which is a request made to another domain. Ex: If `fetch()` is called from JavaScript downloaded from abc.com, the browser restricts a cross-origin HTTP request to xyz.com. A web server can implement Cross-Origin Resource Sharing (CORS) to allow cross-origin requests.

PARTICIPATION ACTIVITY 8.15.1: Fetch request for a webpage.



The text() method returns the response body containing HTML as text.

Captions

1. The fetch() method sends an HTTP GET request to the zybooks.com web server. The await operator waits for the HTTP response to return before continuing.
2. The web server responds with the HTML for the URL https://learn.zybooks.com.
3. The status property is the HTTP status code. 200 means success.
4. The text() method returns the response body containing HTML as text.

Feedback?

Alternative syntax

Some developers prefer the alternative Promise syntax shown below when using the Fetch API.

```
let url = "https://learn.zybooks.com/";
fetch(url)
   .then(function(response) {
      console.log("status is " + response.status);
          return response.text();
   })
   .then(function(html) {
      console.log(html);
   })
   .catch(function(error) {
      console.log("Request failed", error)
   });
```

The same code can be written more concisely using arrow functions:

```
fetch(url)
   .then(response => {
      console.log("status is " + response.status);
          return response.text();
   })
   .then(html => console.log(html))
   .catch(error => console.log("Request failed", error));
```

PARTICIPATION ACTIVITY 8.15.2: Fetch.

1) Which HTTP method type does `fetch()` send by default?

- (•) GET
- ( ) POST
- ( ) JSON

Correct

`fetch()` sends GET requests by default.

2) What is missing from the code segment to fetch the given URL?

```
let url =
"https://www.example.com/";
let response = _____;
console.log(response.status);
```

- ( ) `fetch()`
- ( ) `fetch(url)`
- (•) `await fetch(url)`

Correct

The `await` operator is required to wait for the Promise to resolve and for the HTTP response to be processed. If `await` is removed, `response.status` is not defined.

3) Why might the code below output "404" to the console?

```
let url =
"https://www.example.com/trythis";
let response = await fetch(url);
console.log(response.status);
```

- ( ) `fetch()` is not called correctly
- (•) The `url` variable is assigned a URL that points to an invalid resource
- ( ) `response.text()` has not been called

Correct

A web server is likely to return a 404 status code for a URL that points to an invalid resource. The 200 status code usually means the URL points to a valid resource.

4) What is missing to output the HTML returned in the response?

```
let url =
"https://www.example.com/";
let response = await
fetch(url);
let html = _____;
console.log(html);
```

- ( ) `response.text()`
- ( ) `await response`
- (•) `await response.text()`

Correct

The `await` operator is required to wait for the Promise to resolve and for the HTTP response body to be extracted. If `await` is removed, `response.text()` does not return the response body.

5) What happens if the code below is from zybooks.com, and

Correct

example.com does not allow cross-origin requests?

```
let url =
"https://www.example.com/";
let response = await
fetch(url);
console.log(response.status);
```

- `fetch()` throws an exception
- `fetch()` works normally
- `response.status` is assigned 500

`fetch()` throws an exception because the web browser restricts the cross-origin request from zybooks.com to example.com. If example.com implements CORS, the cross-origin request will work normally.

Feedback?

Fetching JSON

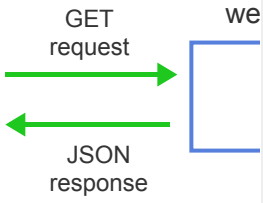
Developers often use the Fetch API to make requests to web APIs that return JSON responses. The ***response.json()*** method returns a Promise that resolves to an object created from parsing the response body as JSON.

The animation below makes a request to the Random User Generator, a web API that generates random user data.

PARTICIPATION ACTIVITY 8.15.3: Fetch request that returns JSON.

1 2 3 4 2x speed

```
let url = "https://randomuser.me/api/?results=3";
let response = await fetch(url);
let users = await response.json();
for (let user of users.results) {
   console.log(`name = ${user.name.first} ${user.name.last}`);
   console.log(`gender = ${user.gender}`);
   console.log(`email = ${user.email}`);
}
```



console

```
name = Judy Welch
gender = female
email = judy.welch@example.com
name = Phillip Brewer
gender = male
email = phillip.brewer@example.com
name = Auguste Riviere
gender = male
email = auguste.riviere@example.com
```

The for-of loop outputs the name, gender, and email address of the three users to the cor

Captions

1. The fetch() method sends an HTTP GET request to the randomuser.me web server.
2. The web server returns JSON that encodes information for 3 randomized users.
3. The json() method parses the JSON response and returns an object containing the JSON data.
4. The for-of loop outputs the name, gender, and email address of the three users to the console.

Feedback?

PARTICIPATION ACTIVITY 8.15.4: Fetch weather data.

The URL `https://wp.zybooks.com/weather.php?zip=XXXXX`, where XXXXX is a five digit ZIP code, returns JSON containing a randomly produced forecast for the given ZIP code. If the ZIP code is not given or is not five digits, the JSON response indicates the ZIP code is not found.

Successful request	Unsuccessful request

<pre>{
"success": true,
"forecast": [
{ "high": 90, "low": 72, "desc": "sunny" },
{ "high": 92, "low": 73, "desc": "mostly sunny" },
{ "high": 87, "low": 64, "desc": "rain" },
{ "high": 88, "low": 65, "desc": "cloudy" },
{ "high": 90, "low": 68, "desc": "partly cloudy" }
]
}</pre>
	<pre>{
	"success": false,
	"error": "ZIP code not found"
	}</pre>

Enter any ZIP code in the webpage below, and click the Search button. When Search is clicked, the `fetch()` method requests the URL above with the ZIP code entered in the form. The raw JSON response is displayed in the webpage, which is not ideal.

Make the following changes:

1. Replace the call to `response.text()` with the method that parses a JSON response:

```
//let json = await response.text();
let weather = await response.json();
```

2. Replace the code that appends the raw JSON to the `html` string with code that loops through the `weather.forecast` array and produces a numbered

list with each day's forecast:

```
//html += json;
html += "<ol>";
for (let day of weather.forecast) {
   html += "<li>" + day.desc + ": high is " + day.high + ", low
is " + day.low + "</li>";
}
html += "</ol>";
```

3. Render the webpage, and verify the changes you have made work correctly to show the weather for the ZIP code you enter.
4. Finally, modify the code to display an appropriate error message if the response indicates the ZIP code is not found. Hint: Examine the `weather.success` property.
5. Render the webpage, and verify that entering a bad ZIP code like "123" produces an error message.

HTML | JavaScript

```
<!DOCTYPE html>
<html lang="en">
<head>
   <title>Weather Forecast</title>
</head>
<body>
   <p>
      <label for="zip">ZIP code:</label>
      <input type="text" id="zip" maxlength="5">
      <button id="search">Search</button>
   </p>
   <div id="forecast"></div>
</body>
</html>

```

Render webpage Reset code

Your webpage	Expected webpage
ZIP code: Search	ZIP code: Search

▼ View solution

Explain

```
--- START FILE: HTML ---

<!DOCTYPE html>
<html>
<body>
   <p>
      <label for="zip">ZIP code:</label>
      <input type="text" id="zip" maxlength="5">
      <button id="search">Search</button>
   </p>
   <div id="forecast"></div>
</body>
</html>


--- END FILE: HTML ---

--- START FILE: JavaScript ---

async function getForecast() {
   let zipcode = document.getElementById("zip").value;
   let url = "https://wp.zybooks.com/weather.php?zip=" +
zipcode;
   let response = await fetch(url);

   // Verify response code is 2XX
   if (response.ok) {
```

```
      let weather = await response.json();

      let html = "";
      if (weather.success) {
         html += "<h1>Forecast</h1>";
         html += "<ol>";
         for (let day of weather.forecast) {
            html += "<li>" + day.desc + ": high is " +
day.high + ", low is " + day.low + "</li>";
         }
         html += "</ol>";
      }
      else {
         html = "<h1>Error: " + weather.error + "</h1>";
      }

      // Show forecast
      document.getElementById("forecast").innerHTML = html;
   }
   else {
      alert("HTTP error: " + response.status);
   }
}

document.getElementById("search").addEventListener("click",
getForecast);


--- END FILE: JavaScript ---
```

Feedback?

PARTICIPATION ACTIVITY 8.15.5: Fetching JSON.

A web API responds with the JSON below.

```
{
   "success": true,
   "questions": [
      { "text": "Who is often called the father of the computer?",
"answer": "Charles Babbage" },
      { "text": "Who is considered the first programmer?", "answer":
"Ada Lovelace" },
      { "text": "What computer popularized the Graphical User
Interface in 1984?", "answer": "Macintosh" },
      { "text": "What is the name of the first modern-day digital
computer?", "answer": "ENIAC" }
   ]
}
```

1) The `fetch()` method must specify the returned response is expected to contain JSON.

   - ○ True
   - ◉ False

Correct

`fetch()` can request any resource type without specifying the type. The same method call is used to request resources with any response type: `fetch(url)`.

2) The `response.json()` method returns an object with two properties: `success` and `questions`.

   - ◉ True
   - ○ False

Correct

The JSON "success" is converted into a boolean property, and "questions" is converted into an array of objects.

3) According to the code below, `trivia.questions[1].answer` is Charles Babbage.

```
let trivia = await
response.json();
```

   - ○ True
   - ◉ False

Correct

`trivia.questions[1].answer` is Ada Lovelace. Charles Babbage is the answer to the question at index 0.

Feedback?

POST request

The `fetch()` method has an optional second parameter, an object that specifies options to modify the HTTP request. Some common `fetch()` options:

- The ***method*** option indicates the HTTP method. Ex: GET, POST, PUT, and DELETE.
- The ***headers*** option specifies various HTTP request headers. Ex: Content-Type and User-Agent.
- The ***body*** option specifies the HTTP request body, which could be form data, a JSON-encoded string, or binary data.

The animation below POSTs form data using `fetch()` and the `FormData` object. The JavaScript object ***FormData*** stores key/value pairs from a form submission.

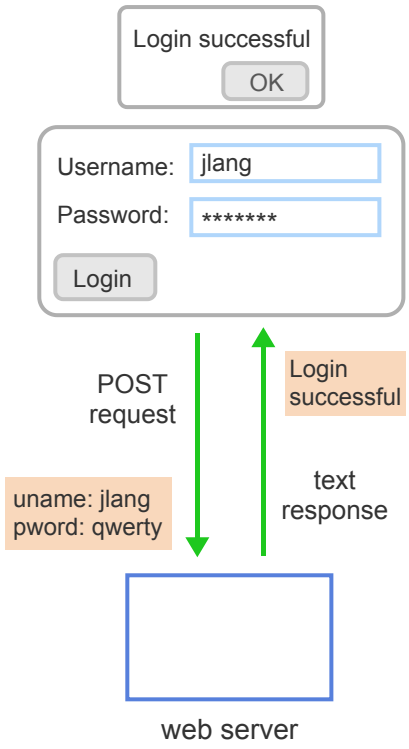
PARTICIPATION ACTIVITY 8.15.6: POSTing form data.

1 2 3 4 5 2x speed

```
<form>
   <p>
     <label for="uname">Username:</label>
     <input type="text" id="uname" name="uname">
   </p>
   <p>
     <label for="pword">Password:</label>
     <input type="password" id="pword" name="pword">
   </p>
   <p>
     <input type="submit" value="Login">
   </p>
</form>
```

```
let form = document.querySelector("form");
form.addEventListener("submit", async function(e) {
   e.preventDefault();
   let response = await fetch("login", {
      method: "POST",
      body: new FormData(form)
   });

   let result = await response.text();
   alert(result);
});
```



The alert() method displays the text response in a dialog box.

Captions

1. The HTML form allows the user to submit a username and password.
2. When the form is submitted, e.preventDefault() prevents the browser from reloading the webpage.
3. fetch() creates a POST request, placing the form data in the request body. The form data is sent to the web server.
4. If the username and password are correct, the server responds with the text "Login successful".
5. The alert() method displays the text response in a dialog box.

Feedback?

PARTICIPATION ACTIVITY 8.15.7: POST JSON data.

Complete the `postSong()` function, which sends a JSON-encoded song in a POST request to a web API. The web API responds with JSON indicating if the operation is successful.

```
__A__ function postSong() {
   let song = {
      title: "Georgia on My Mind",
      artist: "Ray Charles",
      releaseDate: "1930-11-15"
   };

   let response = await fetch("api/songs", {
      method: __B__,
      headers: {
         "Content-Type": __C__
      },
      body: __D__
   });

   let result = __E__;
   console.log(result.status);
}
```

If unable to drag and drop, refresh the page.

C	`"application/json"` The MIME type for sending JSON is `application/json`. The MIME type tells the web server what type of data to expect in the HTTP request body.	**Correct**
A	`async` Only async functions may use the `await` operator.	**Correct**
B	`"POST"` The HTTP request uses the POST method to send the JSON-encoded song.	**Correct**
E	`await response.json()` The `json()` method is required to transform a JSON response into a JavaScript object.	**Correct**
D	`JSON.stringify(song)` The `JSON.stringify()` method converts a JavaScript object into a JSON-encoded string.	**Correct**

Reset

Feedback?

Exploring further:

- Fetch Standard (WHATWG)
- Fetch API (MDN)

How was this section?

Provide section feedback