May 15, 2020

Serializing form data with the vanilla JS FormData() object

Let’s say you have a form. When it’s submitted, you want to get the values of all of the fields and submit them to an API.

```
<form>
  <label for="title">Title</label>
  <input type="text" name="title" id="title" value="Go to Hogwarts"
>

  <label for="body">Body</label>
  <textarea id="body" name="body">Learn magic, play Quidditch, and
 drink some butter beer.</textarea>

  <input type="hidden" name="userId" value="1">

  <button>Submit</button>
</form>
```

How do you easily get the values of all of the fields?

Today, we’re going to look at an easy, native way to do that: `FormData()`.

The `FormData()` constructor

The `FormData()` constructor accepts one argument: the form to get data from.

```
var form = document.querySelector('form');
var data = new FormData(form);
```

That’s it! The `FormData()` constructor returns a FormData object of key/value pairs from your form fields.

*Form fields **must** have a `name` property, or they'll be skipped. Just an `id` won't work.*

Submitting form data to an API with the `FormData()` method

First, let's set up an event listener to detect when forms are submitted.

```
document.addEventListener('submit', function (event) {

  // Prevent form from submitting to the server
  event.preventDefault();

  // Do some stuff...

});
```

Next, we'll submit the form data to the JSONPlaceholder API (https://jsonplaceholder.typicode.com/) with the `fetch()` method (https://gomakethings.com/how-to-send-data-to-an-api-with-the-vanilla-js-fetch-method/).

To do this, we'll pass our `event.target`, the form that was submitted, into a `new FormData()` method, and use that as the `body` property.

```
document.addEventListener('submit', function (event) {

  event.preventDefault();

  fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: new FormData(event.target),
  }).then(function (response) {
    if (response.ok) {
      return response.json();
    }
    return Promise.reject(response);
  }).then(function (data) {
    console.log(data);
  }).catch(function (error) {
    console.warn(error);
  });
});
```

And that's it! Here's a demo. (https://codepen.io/cferdinandi/pen/RwWYvgY)

Some APIs don't accept FormData objects

If you look at the returned data for the demo, you'll notice that the form data doesn't show up in the returned data object.

The JSON placeholder doesn't accept FormData. It wants a JSON object.

You can convert FormData into an object by looping over the FormData and pushing each item into an object.

To loop over FormData, we have to use the `keys()` method, which returns an *iterator*. We can then loop over each item using a `for...of` loop.

On each loop, we'll use the `get()` method to get the value of the `key` from the `FormData` object, and push it into our object.

```
var serializeForm = function (form) {
  var obj = {};
  var formData = new FormData(form);
  for (var key of formData.keys()) {
    obj[key] = formData.get(key);
  }
  return obj;
};
```

Now, we can submit our API like this.

We'll pass the `event.target` into our `serializeForm()` method, which gets the FormData and convert it into an object for us. Then we'll parse the returned object into a string with `JSON.stringify()`.

```
document.addEventListener('submit', function (event) {

  event.preventDefault();

  fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: JSON.stringify(serializeForm(event.target)),
    headers: {
      'Content-type': 'application/json; charset=UTF-8'
    }
  }).then(function (response) {
    if (response.ok) {
      return response.json();
    }
    return Promise.reject(response);
  }).then(function (data) {
    console.log(data);
  }).catch(function (error) {
    console.warn(error);
  });
});
```

In this updated demo (https://codepen.io/cferdinandi/pen/MWaqLVj), you'll notice that the full set of data gets returned.

UPDATE: A simpler alternative

Jim Winstead tipped me off to a simpler alternative to looping through the object using the `Object.fromEntries()` method.

```
Object.fromEntries(new FormData(event.target));
```

You would use it like this.

```
document.addEventListener('submit', function (event) {

  event.preventDefault();

  fetch('https://jsonplaceholder.typicode.com/posts', {
    method: 'POST',
    body: JSON.stringify(Object.fromEntries(new FormData(event.targ
et))),
    headers: {
      'Content-type': 'application/json; charset=UTF-8'
    }
  }).then(function (response) {
    if (response.ok) {
      return response.json();
    }
    return Promise.reject(response);
  }).then(function (data) {
    console.log(data);
  }).catch(function (error) {
    console.warn(error);
  });
});
```

Here a demo with this method. (https://codepen.io/cferdinandi/pen/rNOQyYP)

In video form

My friend Steve Griffith has a great video on this topic in video form (https://www.youtube.com/watch?v=GWJhE7Licjs), if you're interested in learning more.

Browser compatibility

The `FormData()` constructor works in all modern browsers, and IE10 and above. Unfortunately, *iterators* and the `for...of` method do not work in IE at all, and cannot be polyfilled.

Similarly, the `Object.fromEntries()` method also does not work in IE.

Next week, we'll look at some more backwards compatible ways to serialize form data into arrays, objects, and search parameter strings.