MDN web docs
moz://a
Sign in
Search MDN
Technologies ▾
References & Guides ▾
Feedback ▾

Using FormData Objects

English ▾

The `FormData` object lets you compile a set of key/value pairs to send using `XMLHttpRequest`. It is primarily intended for use in sending form data, but can be used independently from forms in order to transmit keyed data. The transmitted data is in the same format that the form's `submit()` method would use to send the data if the form's encoding type were set to `multipart/form-data`.

Creating a FormData object from scratch

You can build a `FormData` object yourself, instantiating it then appending fields to it by calling its `append()` method, like this:

```
var formData = new FormData();

formData.append("username", "Groucho");
formData.append("accountnum", 123456); // number 123456 is immediately converted t

// HTML file input, chosen by user
formData.append("userfile", fileInputElement.files[0]);

// JavaScript file-like object
var content = '<a id="a"><b id="b">hey!</b></a>'; // the body of the new file...
var blob = new Blob([content], { type: "text/xml"});

```

```
formData.append("webmasterfile", blob);

var request = new XMLHttpRequest();
request.open("POST", "http://foo.com/submitform.php");
request.send(formData);
```

Note: The fields "userfile" and "webmasterfile" both contain a file. The number assigned to the field "accountnum" is immediately converted into a string by the `FormData.append()` method (the field's value can be a `Blob`, `File`, or a string: **if the value is neither a Blob nor a File, the value is converted to a string**).

This example builds a `FormData` instance containing values for fields named "username", "accountnum", "userfile" and "webmasterfile", then uses the `XMLHttpRequest` method `send()` to send the form's data. The field "webmasterfile" is a `Blob`. A `Blob` object represents a file-like object of immutable, raw data. Blobs represent data that isn't necessarily in a JavaScript-native format. The `File` interface is based on `Blob`, inheriting blob functionality and expanding it to support files on the user's system. In order to build a `Blob` you can invoke the `Blob() constructor`.

Retrieving a FormData object from an HTML form

To construct a `FormData` object that contains the data from an existing `<form>`, specify that form element when creating the `FormData` object:

Note: FormData will only use input fields that use the name attribute.

```
var formData = new FormData(someFormElement);
```

For example:

```
var formElement = document.querySelector("form");
var request = new XMLHttpRequest();

```

```
request.open("POST", "submitform.php");
request.send(new FormData(formElement));
```

You can also append additional data to the `FormData` object between retrieving it from a form and sending it, like this:

```
var formElement = document.querySelector("form");
var formData = new FormData(formElement);
var request = new XMLHttpRequest();
request.open("POST", "submitform.php");
formData.append("serialnumber", serialNumber++);
request.send(formData);
```

This lets you augment the form's data before sending it along, to include additional information that's not necessarily user-editable.

Sending files using a FormData object

You can also send files using `FormData`. Simply include an `<input>` element of type `file` in your `<form>`:

```
<form enctype="multipart/form-data" method="post" name="fileinfo">
  <label>Your email address:</label>
  <input type="email" autocomplete="on" autofocus name="userid" placeholder="email
  <label>Custom file label:</label>
  <input type="text" name="filelabel" size="12" maxlength="32" /><br />
  <label>File to stash:</label>
  <input type="file" name="file" required />
  <input type="submit" value="Stash the file!" />
</form>
<div></div>
```

Then you can send it using code like the following:

```
var form = document.forms.namedItem("fileinfo");
form.addEventListener('submit', function(ev) {

  var oOutput = document.querySelector("div"),
      oData = new FormData(form);

  oData.append("CustomField", "This is some extra data");

  var oReq = new XMLHttpRequest();
  oReq.open("POST", "stash.php", true);
  oReq.onload = function(oEvent) {
    if (oReq.status == 200) {
      oOutput.innerHTML = "Uploaded!";
    } else {
      oOutput.innerHTML = "Error " + oReq.status + " occurred when trying to uploa
    }
  };

  oReq.send(oData);
  ev.preventDefault();
}, false);
```

Note: If you pass in a reference to the form, the request method specified in the form will be used over the method specified in the open() call.

You can also append a `File` or `Blob` directly to the `FormData` object, like this:

```
data.append("myfile", myBlob, "filename.txt");
```

When using the `append()` method it is possible to use the third optional parameter to pass a filename inside the `Content-Disposition` header that is sent to the server. When no filename is specified (or the parameter isn't supported), the name "blob" is used.

Using a formdata event

A more recent addition to the platform than the `FormData` object is the `formdata` event — this is fired on an `HTMLFormElement` object after the entry list representing the form's data is constructed. This happens when the form is submitted, but can also be triggered by the invocation of a `FormData()` constructor.

This allows a `FormData` object to be quickly obtained in response to a `formdata` event firing, rather than needing to put it together yourself.

Typically this is used as shown in our simple formdata event demo — in the JavaScript we reference a form:

```
const formElem = document.querySelector('form');
```

In our `submit` event handler we use `preventDefault` to stop the default form submission, then invoke a `FormData` constructor to trigger the `formdata` event:

```
formElem.addEventListener('submit', (e) => {
  // on form submission, prevent default
  e.preventDefault();

  // construct a FormData object, which fires the formdata event
  new FormData(formElem);
});
```

When the `formdata` event fires we can access the `FormData` object using `FormDataEvent.formData`, then do what we like with it (below we post it to the server using `XMLHttpRequest`).

```
formElem.addEventListener('formdata', (e) => {
  console.log('formdata fired');

  // Get the form data from the event object
  let data = e.formData;
  for (var value of data.values()) {
    console.log(value);
  }
```

```

  // submit the data via XHR
  let request = new XMLHttpRequest();
  request.open("POST", "/formHandler");
  request.send(data);
});
```

> **Note**: The `formdata` event and `FormDataEvent` object are available in Chrome from version 77 (and other equivalent Chromiums), and Firefox 72 (first available behind the `dom.formdata.event.enabled` pref in Firefox 71).

Submitting forms and uploading files via AJAX *without* `FormData` objects

If you want to know how to serialize and submit a form via AJAX *without* using FormData objects, please read this paragraph.

See also

- Using XMLHttpRequest
- `HTMLFormElement`
- `Blob`
- Typed Arrays

Last modified: Jan 2, 2020, by MDN contributors

Creating a FormData object from scratch

Related Topics

`FormData`

- Constructor

  `FormData()`

- Methods

  `append()`
  `delete()`
  `entries()`
  `get()`
  `getAll()`
  `has()`
  `keys()`
  `set()`
  `values()`

- Related pages for XMLHttpRequest

  `XMLHttpRequest`
  `XMLHttpRequestEventTarget`
  `XMLHttpRequestUpload`

Learn the best of web development

Get the latest and greatest from MDN delivered straight to your inbox.

you@example.com

Sign up now