Multipart Bodies

Posting data as `multipart/form-data`

Using FormData API

Browser

```
const form = new FormData();
form.append('my_field', 'my value');
form.append('my_buffer', new Blob([1,2,3]));
form.append('my_file', fileInput.files[0]);

axios.post('https://example.com', form)
```

The same result can be achieved using the internal Axios serializer and corresponding shorthand method:

```
axios.postForm('https://httpbin.org/post', {
  my_field: 'my value',
  my_buffer: new Blob([1,2,3]),
  my_file:  fileInput.files // FileList will be unwrapped as sepate fields
});
```

HTML form can be passes directly as a request payload

Node.js

```
import axios from 'axios';

const form = new FormData();
form.append('my_field', 'my value');
form.append('my_buffer', new Blob(['some content']));
```

Since node.js does not currently support creating a `Blob` from a file, you can use a third-party package for this purpose.

```
import {fileFromPath} from 'formdata-node/file-from-path'

form.append('my_field', 'my value');
form.append('my_file', await fileFromPath('/foo/bar.jpg'));

axios.post('https://example.com', form)
```

For Axios older than `v1.3.0` you must import `form-data` package.

```
const FormData = require('form-data');

const form = new FormData();
form.append('my_field', 'my value');
form.append('my_buffer', new Buffer(10));
form.append('my_file', fs.createReadStream('/foo/bar.jpg'));

axios.post('https://example.com', form)
```

🆕 Automatic serialization

Starting from `v0.27.0`, Axios supports automatic object serialization to a FormData object if the request Content-Type header is set to `multipart/form-data`.

The following request will submit the data in a FormData format (Browser & Node.js):

```
import axios from 'axios';

axios.post('https://httpbin.org/post', {
  user: {
    name: 'Dmitriy'
  },
  file: fs.createReadStream('/foo/bar.jpg')
}, {
  headers: {
    'Content-Type': 'multipart/form-data'
```

Axios FormData serializer supports some special endings to perform the following operations:

- `{}` - serialize the value with JSON.stringify
- `[]` - unwrap the array-like object as separate fields with the same key

  NOTE: unwrap/expand operation will be used by default on arrays and FileList objects

FormData serializer supports additional options via `config.formSerializer: object` property to handle rare cases:

- `visitor: Function` - user-defined visitor function that will be called recursively to serialize the data object to a `FormData` object by following custom rules.
- `dots: boolean = false` - use dot notation instead of brackets to serialize arrays and objects;
- `metaTokens: boolean = true` - add the special ending (e.g `user{}: '{"name": "John"}'`) in the FormData key. The back-end body-parser could potentially use this meta-information to automatically parse the value as JSON.
- `indexes: null|false|true = false` - controls how indexes will be added to unwrapped keys of `flat` array-like objects
  - `null` - don't add brackets (`arr: 1`, `arr: 2`, `arr: 3`)
  - `false` (default) - add empty brackets (`arr[]: 1`, `arr[]: 2`, `arr[]: 3`)
  - `true` - add brackets with indexes (`arr[0]: 1`, `arr[1]: 2`, `arr[2]: 3`)

Let's say we have an object like this one:

```
const obj = {
  x: 1,
  arr: [1, 2, 3],
  arr2: [1, [2], 3],
  users: [{name: 'Peter', surname: 'Griffin'}, {name: 'Thomas', surname: 'Ande
```

The following steps will be executed by the Axios serializer internally:

```
const formData= new FormData();
formData.append('x', '1');
formData.append('arr[]', '1');
formData.append('arr[]', '2');
formData.append('arr[]', '3');
formData.append('arr2[0]', '1');
formData.append('arr2[1][0]', '2');
formData.append('arr2[2]', '3');
formData.append('users[0][name]', 'Peter');
formData.append('users[0][surname]', 'Griffin');
formData.append('users[1][name]', 'Thomas');
formData.append('users[1][surname]', 'Anderson');
formData.append('obj2{}', '[{"x":1}]');
```

```
import axios from 'axios';

axios.post('https://httpbin.org/post', {
  'myObj{}': {x: 1, s: "foo"},
  'files[]': document.querySelector('#fileInput').files
}, {
  headers: {
    'Content-Type': 'multipart/form-data'
  }
}).then(({data})=> console.log(data));
```

Axios supports the following shortcut methods: `postForm`, `putForm`, `patchForm` which are just the corresponding http methods with the content-type header preset to `multipart/form-data`.

`FileList` object can be passed directly:

```
await axios.postForm('https://httpbin.org/post', document.querySelector('#file
```

« Previous
URL-Encoding Bodies

Next »
Notes