FormData

В этой главе речь пойдёт об отправке HTML-форм: с файлами и без, с дополнительными полями и так далее. Объекты FormData помогут нам с этим. Как вы, наверняка, догадались по его названию, это объект, представляющий данные HTML формы.

Конструктор:

```
let formData = new FormData([form]);
```

Если передать в конструктор элемент HTML-формы `form`, то создаваемый объект автоматически прочитает из неё поля.

Его особенность заключается в том, что методы для работы с сетью, например `fetch`, позволяют указать объект `FormData` в свойстве тела запроса `body`.

Он будет соответствующим образом закодирован и отправлен с заголовком `Content-Type: form/multipart`.

То есть, для сервера это выглядит как обычная отправка формы.

Отправка простой формы

Давайте сначала отправим простую форму.

Как вы видите, код очень компактный:

```
<form id="formElem">
  <input type="text" name="name" value="John">
  <input type="text" name="surname" value="Smith">
  <input type="submit">
</form>

<script>
  formElem.onsubmit = async (e) => {
    e.preventDefault();

    let response = await fetch('/article/formdata/post/
      method: 'POST',
      body: new FormData(formElem)
    });

    let result = await response.json();

    alert(result.message);
  };
</script>
```

John Smith Отправить

В этом примере серверный код не представлен, он за рамками этой статьи, он принимает POST-запрос с данными формы и отвечает сообщением «Пользователь сохранён».

Методы объекта FormData

С помощью указанных ниже методов мы можем изменять поля в объекте `FormData`:

- `formData.append(name, value)` – добавляет к объекту поле с именем `name` и значением `value`,

- `formData.append(name, blob, fileName)` – добавляет поле, как будто в форме имеется элемент `<input type="file">`, третий аргумент `fileName` устанавливает имя файла (не имя поля формы), как будто это имя из файловой системы пользователя,
- `formData.delete(name)` – удаляет поле с заданным именем `name`,
- `formData.get(name)` – получает значение поля с именем `name`,
- `formData.has(name)` – если существует поле с именем `name`, то возвращает `true`, иначе `false`

Технически форма может иметь много полей с одним и тем же именем `name`, поэтому несколько вызовов `append` добавят несколько полей с одинаковыми именами.

Ещё существует метод `set`, его синтаксис такой же, как у `append`. Разница в том, что `.set` удаляет все уже имеющиеся поля с именем `name` и только затем добавляет новое. То есть этот метод гарантирует, что будет существовать только одно поле с именем `name`, в остальном он аналогичен `.append`:

- `formData.set(name, value)`,
- `formData.set(name, blob, fileName)`.

Поля объекта `formData` можно перебирать, используя цикл `for..of`:

```
let formData = new FormData();
formData.append('key1', 'value1');
formData.append('key2', 'value2');

// Список пар ключ/значение
for(let [name, value] of formData) {
  alert(`${name} = ${value}`); // key1=value1, потом ke
}
```

Отправка формы с файлом

Объекты `FormData` всегда отсылаются с заголовком `Content-Type: form/multipart`, этот способ кодировки позволяет отсылать файлы. Таким образом, поля `<input type="file">` тоже отправляются, как это и происходит в случае обычной формы.

Пример такой формы:

```
<form id="formElem">
  <input type="text" name="firstName" value="John">
  Картинка: <input type="file" name="picture" accept="i
  <input type="submit">
</form>

<script>
  formElem.onsubmit = async (e) => {
    e.preventDefault();

    let response = await fetch('/article/formdata/post/
      method: 'POST',
      body: new FormData(formElem)
    });

    let result = await response.json();

    alert(result.message);
  };
</script>
```

John Картинка: Выберите файл Файл не выбран
Отправить

Отправка формы с Blob-данными

Ранее в главе Fetch мы видели, что очень легко отправить динамически сгенерированные бинарные данные в формате `Blob`. Мы можем явно передать её в параметр `body` запроса `fetch`.

Но на практике бывает удобнее отправлять изображение не отдельно, а в составе формы, добавив дополнительные поля для имени и другие метаданные.

Кроме того, серверы часто настроены на приём именно форм, а не просто бинарных данных.

В примере ниже посылается изображение из `<canvas>` и ещё несколько полей, как форма, используя `FormData`:

```
<body style="margin:0">
  <canvas id="canvasElem" width="100" height="80" style

  <input type="button" value="Отправить" onclick="submi

  <script>
    canvasElem.onmousemove = function(e) {
      let ctx = canvasElem.getContext('2d');
      ctx.lineTo(e.clientX, e.clientY);
      ctx.stroke();
    };

    async function submit() {
      let imageBlob = await new Promise(resolve => canv

      let formData = new FormData();
      formData.append("firstName", "John");
      formData.append("image", imageBlob, "image.png");

      let response = await fetch('/article/formdata/pos
        method: 'POST',
        body: formData
      });
      let result = await response.json();
      alert(result.message);
    }

  </script>
</body>
```

Пожалуйста, обратите внимание на то, как добавляется изображение `Blob`:

```
formData.append("image", imageBlob, "image.png");
```

Это как если бы в форме был элемент `<input type="file" name="image">` и пользователь прикрепил бы файл с именем `"image.png"` (3й аргумент) и данными `imageBlob` (2й аргумент) из своей файловой системы.

Сервер прочитает и данные и файл, точно так же, как если бы это была обычная отправка формы.

Итого

Объекты FormData используются, чтобы взять данные из HTML-формы и отправить их с помощью `fetch` или другого метода для работы с сетью.

Мы можем создать такой объект уже с данными, передав в конструктор HTML-форму – `new FormData(form)`, или же можно создать объект вообще без формы и затем добавить к нему поля с помощью методов:

- `formData.append(name, value)`

- `formData.append(name, blob, fileName)`
- `formData.set(name, value)`
- `formData.set(name, blob, fileName)`

Отметим две особенности:

1. Метод `set` удаляет предыдущие поля с таким же именем, а `append` – нет. В этом их единственное отличие.
2. Чтобы послать файл, нужно использовать синтаксис с тремя аргументами, в качестве третьего как раз указывается имя файла, которое обычно, при `<input type="file">`, берётся из файловой системы.

Другие методы:

- `formData.delete(name)`
- `formData.get(name)`
- `formData.has(name)`

Вот и всё!

Комментарии

перед тем как писать...