|  |  |
| --- | --- |
| **Применение методов GET, POST, PUT, PATCH, DELETE с помощью Fetch API**  GET, POST, PUT, PATCH и DELETE - это пять наиболее распространенных HTTP-методов для получения и отправки данных на сервер.  Используя Fetch API (встроенный в JavaScript интерфейс для выполнения запросов к серверам), рассмотрим применение каждого из методов.  **Метод GET**  Метод GET используется для получения данных с сервера. Это метод предназначен только для чтения, поэтому риск изменения или повреждения данных отсутствует. Вызовем метод GET в нашем API для получения данных.  fetch('https://jsonplaceholder.typicode.com/todos')  .then(response => response.json())  .then(json => console.log(json))  В результате, в консоли браузера мы увидим весь полученный нами массив с множеством данных. Во вкладке Network об успешности запроса свидетельствует статус 200:    В этой же вкладке также можно увидеть полученные данные (таб Response):    Если нам необходим конкретный элемент массива, в запросе можно указать его URI (Uniform Resource Identifier) - это по сути id элемента:  fetch('https://jsonplaceholder.typicode.com/todos/5')  .then(response => response.json())  .then(json => console.log(json)) | **Метод POST**  Метод POST отправляет данные на сервер и создает новый ресурс. Когда этот новый ресурс помещается в родительский объект, служба API автоматически создает с ним связь, назначая свой идентификатор (URI нового ресурса). Простыми словами, этот метод используется для создания новой записи данных.  fetch('https://jsonplaceholder.typicode.com/todos', {  method: 'POST',  body: JSON.stringify({  userId: 1,  title: "buy vegetables",  completed: true  }),  headers: {  "Content-type": "application/json; charset=UTF-8"  }  })  .then(response => response.json())  .then(json => console.log(json))  Обратим внимание на то, что теперь нам нужно было передать метод запроса (method), тело (body) и заголовки (headers). Мы не передавали их для метода GET, потому что для запроса GET эти поля по-умолчанию настроены, однако для всех других типов запросов их нужно указывать.  В теле необходимо присваивать значения в строковой форме. Также обратим внимание, что нам не нужно назначать URI, т.к. API сделает это за нас. Результатом успешного запроса будет статус 201.    Как видно из Response, API присваивает вновь созданному ресурсу идентификатор 201, т.к. изначально количество элементов в массиве было 200. Следующий созданный ресурс будет с идентификатором 202. |
|  |  |
| **Метод PUT**  Метод PUT чаще всего используется для обновления существующего ресурса. Для этого необходим URI ресурса и новая его версия.  fetch('https://jsonplaceholder.typicode.com/todos/5', {  method: 'PUT',  body: JSON.stringify({  userId: 1,  id: 5,  title: "updated resource",  completed: false  }),  headers: {  "Content-type": "application/json; charset=UTF-8"  }  })  .then(response => response.json())  .then(json => console.log(json))  Результатом успешного запроса будет статус 200, а в Response мы увидим уже обновленный ресурс: | **Метод PATCH**  Метод PATCH очень похож на метод PUT, поскольку он также изменяет существующий ресурс. Разница в том, что для метода PUT тело запроса содержит полную новую версию, тогда как для метода PATCH тело запроса должно содержать только конкретные изменения.  fetch('https://jsonplaceholder.typicode.com/todos/2', {  method: 'PATCH',  body: JSON.stringify({  title: "my custom title"  }),  headers: {  "Content-type": "application/json; charset=UTF-8"  }  })  .then(response => response.json())  .then(json => console.log(json))  Как и с PUT, при успехе получим статус 200, а в Response увидим новую версию ресурса: |
| **Метод DELETE**  Метод DELETE используется для удаления ресурса, который указывается с помощью его URI.  fetch('https://jsonplaceholder.typicode.com/todos/3', {  method: 'DELETE'  })  В Response увидим пустой объект. |  |