**Метод forEach ()**

Метод forEach () виконує зазначену функцію один раз для кожного елемента в масиві.

Метод forEach () виконує функцію callback один раз для кожного елемента, що знаходиться в масиві в порядку зростання. Вона не буде викликана для віддалених або пропущених елементів масиву. Однак, вона буде викликана для елементів, які присутні в масиві і мають значення undefined.

Функція callback буде викликана з трьома аргументами:

* значення елемента (value);
* індекс елемента (index);
* масив, за яким здійснюється прохід (array).

Якщо в метод forEach () був переданий параметр thisArg, при виклику callback він буде використовуватися в якості значення this. В іншому випадку, в якості значення this буде використовуватися значення undefined. В кінцевому підсумку, значення this, що спостерігається з функції callback, визначається відповідно до звичайних правил визначення this, видимого з функції.

Не існує способу зупинити або перервати цикл forEach () крім як викиданням виключення. Якщо вам необхідно така поведінка, метод forEach() неправильний вибір.

**Fetch**

Метод fetch () - сучасний і дуже потужний спосіб робити мережеві запити і отримувати інформацію з сервера. Він не підтримується старими (можна використовувати Поліфем), але підтримується усіма сучасними браузерами.

Базовий синтаксис:

let promise = fetch(url, [options]);

* url - URL для відправки запиту.
* options - додаткові параметри: метод, заголовки і так далі.

Без options це простий GET-запит, викачує вміст за адресою url.

Браузер відразу ж починає запит і повертає промис, який зовнішній код використовує для отримання результату.

Процес отримання відповіді зазвичай відбувається в два етапи.

По-перше, promise виконується з об'єктом вбудованого класу Response як результат, як тільки сервер надішле заголовки відповіді.

На цьому етапі ми можемо перевірити статус HTTP-запиту і визначити, виконався він успішно, а також подивитися заголовки, але поки без тіла відповіді.

Проміс завершується з помилкою, якщо fetch не зміг виконати HTTP-запит, наприклад при помилку мережі або якщо немає такого сайту. HTTP-статуси 404 і 500 не є помилкою.

По-друге, для отримання тіла відповіді нам потрібно використовувати додатковий виклик методу.

Response надає кілька методів, заснованих на Проміс, для доступу до тіла відповіді в різних форматах:

* response.text () - читає відповідь і повертає як звичайний текст,
* response.json () - декодує відповідь в форматі JSON,
* response.formData () - повертає відповідь як об'єкт FormData,
* response.blob () - повертає об'єкт як Blob (бінарні дані з типом),
* response.arrayBuffer () - повертає відповідь як ArrayBuffer (низькорівневе подання бінарних даних),
* крім цього, response.body - це об'єкт ReadableStream, за допомогою якого можна зчитувати тіло запиту по частинах.

Ми можемо вибрати тільки один метод читання відповіді.

Якщо ми вже отримали відповідь з response.text (), тоді response.json () не спрацює, тому що дані вже були оброблені.

Заголовки відповіді зберігаються в схожому на Map об'єкті response.headers.

Типовий запит за допомогою fetch складається з двох операторів await:

let response = await fetch (url, options); // завершується з заголовками відповіді

let result = await response.json (); // читати тіло відповіді в форматі JSON