

Раздел

[Загрузка документа и ресурсов](#)

Навигация по уроку

Загрузка скриптов

Другие ресурсы

Ошибка в скрипте с другого источника

Итого

Задачи (1)

Комментарии

Поделиться

[Редактировать на GitHub](#)[🏠 → Браузер: документ, события, интерфейсы](#)  
[→ Загрузка документа и ресурсов](#)

2-го октября 2020

## Загрузка ресурсов: onload и onerror

Браузер позволяет отслеживать загрузку сторонних ресурсов: скриптов, ифреймов, изображений и др.

Для этого существуют два события:

- `load` – успешная загрузка,
- `error` – во время загрузки произошла ошибка.

### Загрузка скриптов

Допустим, нам нужно загрузить сторонний скрипт и вызвать функцию, которая объявлена в этом скрипте.

Мы можем загрузить этот скрипт динамически:

```
1 let script = document.createElement('script');
2 script.src = "my.js";
3
4 document.head.append(script);
```

...Но как нам вызвать функцию, которая объявлена внутри того скрипта? Нам нужно подождать, пока скрипт загрузится, и только потом мы можем её вызвать.

#### На заметку:

Для наших собственных скриптов мы можем использовать [JavaScript-модули](#), но они не слишком широко распространены в сторонних библиотеках.

### script.onload

Главный помощник – это событие `load`. Оно срабатывает после того, как скрипт был загружен и выполнен.

Например:

```
1 let script = document.createElement('script');
2
3 // мы можем загрузить любой скрипт с любого домена
4 script.src = "https://cdnjs.cloudflare.com/ajax/libs/lo
5 document.head.append(script);
6
7 script.onload = function() {
8   // в скрипте создаётся вспомогательная функция с имен
9   alert(_); // функция доступна
10 };
```

Таким образом, в обработчике `onload` мы можем использовать переменные, вызывать функции и т.д., которые предоставляет нам сторонний скрипт.

...А что если во время загрузки произошла ошибка? Например, такого скрипта нет (ошибка 404), или сервер был недоступен.

### script.onerror

Раздел

[Загрузка документа и ресурсов](#)

Навигация по уроку

Загрузка скриптов

Другие ресурсы

Ошибка в скрипте с другого источника

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Ошибки, которые возникают во время загрузки скрипта, могут быть отслежены с помощью события `error`.

Например, давайте запросим скрипт, которого не существует:

```
1 let script = document.createElement('script');
2 script.src = "https://example.com/404.js"; // такого файла нет
3 document.head.appendChild(script);
4
5 script.onerror = function() {
6   alert("Error loading " + this.src); // Ошибка загрузки
7 };
```

Обратите внимание, что мы не можем получить описание HTTP-ошибки. Мы не знаем, была ли это ошибка 404 или 500, или какая-то другая. Знаем только, что во время загрузки произошла ошибка.

#### ⚠ Важно:

Обработчики `onload` / `onerror` отслеживают только сам процесс загрузки.

Ошибки обработки и выполнения загруженного скрипта ими не отслеживаются. Чтобы «поймать» ошибки в скрипте, нужно воспользоваться глобальным обработчиком `window.onerror`.

## Другие ресурсы

События `load` и `error` также срабатывают и для других ресурсов, а вообще, для любых ресурсов, у которых есть внешний `src`.

Например:

```
1 let img = document.createElement('img');
2 img.src = "https://js.cx/clipart/train.gif"; // (*)
3
4 img.onload = function() {
5   alert(`Изображение загружено, размеры ${img.width}x${img.height}`);
6 };
7
8 img.onerror = function() {
9   alert("Ошибка во время загрузки изображения");
10 };
```

Однако есть некоторые особенности:

- Большинство ресурсов начинают загружаться после их добавления в документ. За исключением тега `<img>`. Изображения начинают загружаться, когда получают `src (*)`.
- Для `<iframe>` событие `load` срабатывает по окончании загрузки как в случае успеха, так и в случае ошибки.

Такое поведение сложилось по историческим причинам.

## Ошибка в скрипте с другого источника

Есть правило: скрипты с одного сайта не могут получить доступ к содержимому другого сайта. Например, скрипт с `https://facebook.com` не может прочитать почту пользователя на `https://gmail.com`.

Или, если быть более точным, один источник (домен/порт/протокол) не может получить доступ к содержимому с другого источника. Даже поддомен или просто другой порт будут считаться разными источниками, не имеющими доступа друг к другу.

Это правило также касается ресурсов с других доменов.

Если мы используем скрипт с другого домена, и в нем имеется ошибка, мы не сможем узнать детали этой ошибки.

Раздел

[Загрузка документа и ресурсов](#)

Навигация по уроку

Загрузка скриптов

Другие ресурсы

Ошибка в скрипте с другого источника

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Для примера давайте возьмём мини-скрипт `error.js`, который состоит из одного-единственного вызова функции, которой не существует:

```
1 // error.js
2 noSuchFunction();
```

Теперь загрузим этот скрипт с того же сайта, на котором он лежит:

```
1 <script>
2 window.onerror = function(message, url, line, col, erro
3     alert(`${message}\n${url}, ${line}:${col}`);
4 };
5 </script>
6 <script src="/article/onload-onerror/crossorigin/error.
```

Мы видим нормальный отчёт об ошибке:

```
1 Uncaught ReferenceError: noSuchFunction is not defined
2 https://javascript.info/article/onload-onerror/crossori
```

А теперь загрузим этот же скрипт с другого домена:

```
1 <script>
2 window.onerror = function(message, url, line, col, erro
3     alert(`${message}\n${url}, ${line}:${col}`);
4 };
5 </script>
6 <script src="https://cors.javascript.info/article/onloa
```



Отчёт отличается:



```
1 Script error.
2 , 0:0
```

Детали отчёта могут варьироваться в зависимости от браузера, но основная идея остаётся неизменной: любая информация о внутреннем устройстве скрипта, включая стек ошибки, скрыта. Именно потому, что скрипт загружен с другого домена.

Зачем нам могут быть нужны детали ошибки?

Существует много сервисов (и мы можем сделать наш собственный), которые обрабатывают глобальные ошибки при помощи `window.onerror`, сохраняют отчёт о них и предоставляют доступ к этому отчёту для анализа. Это здорово, потому что мы можем увидеть реальные ошибки, которые случились у наших пользователей. Но если скрипт – с другого домена, то информации об ошибках в нём почти нет, как мы только что видели.

Похожая кросс-доменная политика (CORS) внедрена и в отношении других ресурсов.

**Чтобы разрешить кросс-доменный доступ, нам нужно поставить тегу `<script>` атрибут `crossorigin`, и, кроме того, удалённый сервер должен поставить специальные заголовки.**

Существует три уровня кросс-доменного доступа:

1. **Атрибут `crossorigin` отсутствует** – доступ запрещён.
2. **`crossorigin="anonymous"`** – доступ разрешён, если сервер отвечает с заголовком `Access-Control-Allow-Origin` со значениями `*` или наш домен. Браузер не отправляет авторизационную информацию и куки на удалённый сервер.
3. **`crossorigin="use-credentials"`** – доступ разрешён, если сервер отвечает с заголовками `Access-Control-Allow-Origin` со значением

наш домен и `Access-Control-Allow-Credentials: true`. Браузер отправляет авторизационную информацию и куки на удалённый сервер.

#### На заметку:

Почитать больше о кросс-доменных доступах вы можете в главе [Fetch: запросы на другие сайты](#). Там описан метод `fetch` для сетевых запросов, но политика там точно такая же.

Такое понятие как «куки» (cookies) не рассматривается в текущей главе, но вы можете почитать о них в главе [Куки, document.cookie](#).

В нашем случае атрибут `crossorigin` отсутствовал. Поэтому кросс-доменный доступ был запрещён. Давайте добавим его.

Мы можем выбрать `"anonymous"` (куки не отправляются, требуется один серверный заголовок) или `"use-credentials"` (куки отправляются, требуются два серверных заголовка) в качестве значения атрибута.

Если куки нас не волнуют, тогда смело выбираем `"anonymous"` :

```
1 <script>
2 window.onerror = function(message, url, line, col, erro
3   alert(`${message}\n${url}, ${line}:${col}`);
4 };
5 </script>
6 <script crossorigin="anonymous" src="https://cors.javas
```

Теперь при условии, что сервер предоставил заголовок `Access-Control-Allow-Origin`, всё хорошо. У нас есть полный отчёт по ошибкам.

## Итого

Изображения `<img>`, внешние стили, скрипты и другие ресурсы предоставляют события `load` и `error` для отслеживания загрузки:

- `load` срабатывает при успешной загрузке,
- `error` срабатывает при ошибке загрузки.

Единственное исключение – это `<iframe>`: по историческим причинам срабатывает всегда `load` вне зависимости от того, как завершилась загрузка, даже если страница не была найдена.

Событие `readystatechange` также работает для ресурсов, но используется редко, потому что события `load/error` проще в использовании.

## Задачи

### Загрузите изображения с колбэком [↗](#)

важность: 4

Обычно изображения начинают загружаться в момент их создания. Когда мы добавляем `<img>` на страницу, пользователь не увидит его тут же. Браузер сначала должен его загрузить.

Чтобы показать изображение сразу, мы можем создать его «заранее»:

```
1 let img = document.createElement('img');
2 img.src = 'my.jpg';
```

Браузер начнёт загружать изображение и положит его в кеш. Позже, когда такое же изображение появится в документе (не важно как), оно будет показано мгновенно.

Создайте функцию `preloadImages(sources, callback)`, которая загружает все изображения из массива `sources` и, когда все они будут

Раздел

[Загрузка документа и ресурсов](#)

Навигация по уроку

Загрузка скриптов

Другие ресурсы

Ошибка в скрипте с другого источника

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)

Раздел

[Загрузка документа и ресурсов](#)

Навигация по уроку

Загрузка скриптов

Другие ресурсы

Ошибка в скрипте с другого источника

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)



загружены, вызывает `callback`.

В данном примере будет показан `alert` после загрузки всех изображений.

```
1 function loaded() {  
2   alert("Изображения загружены")  
3 }  
4  
5 preloadImages(["1.jpg", "2.jpg", "3.jpg"], loaded);
```

В случае ошибки функция должна считать изображение «загруженным».

Другими словами, `callback` выполняется в том случае, когда все изображения либо загружены, либо в процессе их загрузки возникла ошибка.

Такая функция полезна, например, когда нам нужно показать галерею с маленькими скрользящими изображениями, и мы хотим быть уверены, что все из них загружены.

В песочнице подготовлены ссылки к тестовым изображениям, а также код для проверки их загрузки. Код должен выводить `300`.

[Открыть песочницу для задачи.](#)

решение

Проводим [курсы по JavaScript и фреймворкам](#).



## Комментарии

перед тем как писать...

