

Асинхронность в javascript

Урок 4





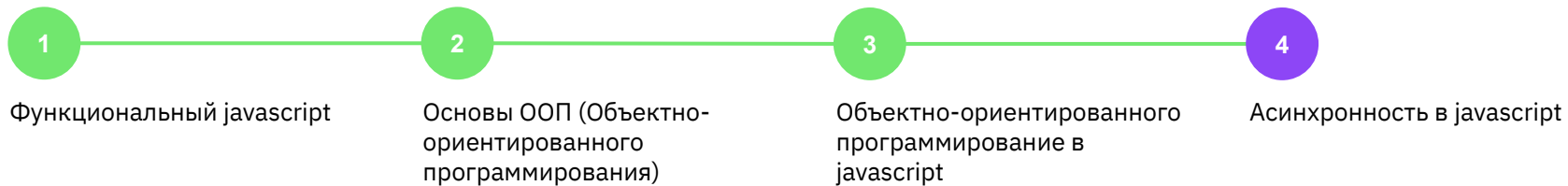
Кадочников Алексей

Frontend-разработчик

- ☀ Веб-разработчик со стажем более 9 лет
- ☀ Преподаватель GeekBrains с 2015 года
- ☀ Автор курсов по Frontend на портале Geekbrains
- ☀ Работал в таких компаниях как VK и Wizard-C







План курса





Что будет на уроке сегодня

-  Асинхронность
-  Формат данных JSON
-  `async/await`
-  “Запланированная асинхронность” - `setTimeout, setInterval`



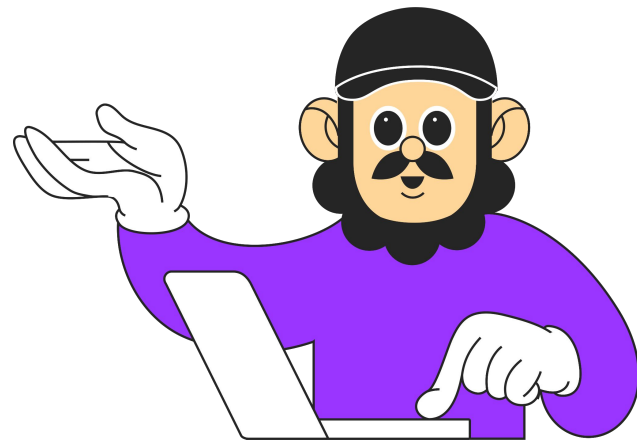
Асинхронность





Асинхронность

Движок JavaScript устроен так, что весь код выполняется в одном потоке. И несмотря на то, что у вас может быть многоядерный процессор, исполнение JavaScript кода на одной странице будет обрабатываться только в одном потоке и в одном ядре процессора.





Асинхронность

- выполнение запросов к серверу,
- ожидание ответных действий пользователя,
- отложенное выполнение кода,
- сложные анимации.

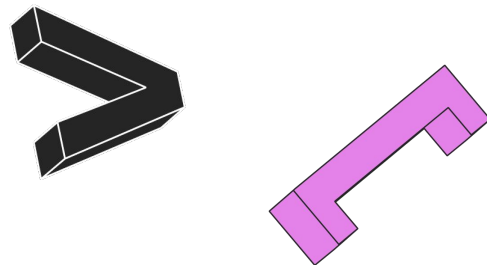




Асинхронность

Для решения этой проблемы придумали механизмы для написания асинхронного кода - это код, обычно состоящий из двух частей:

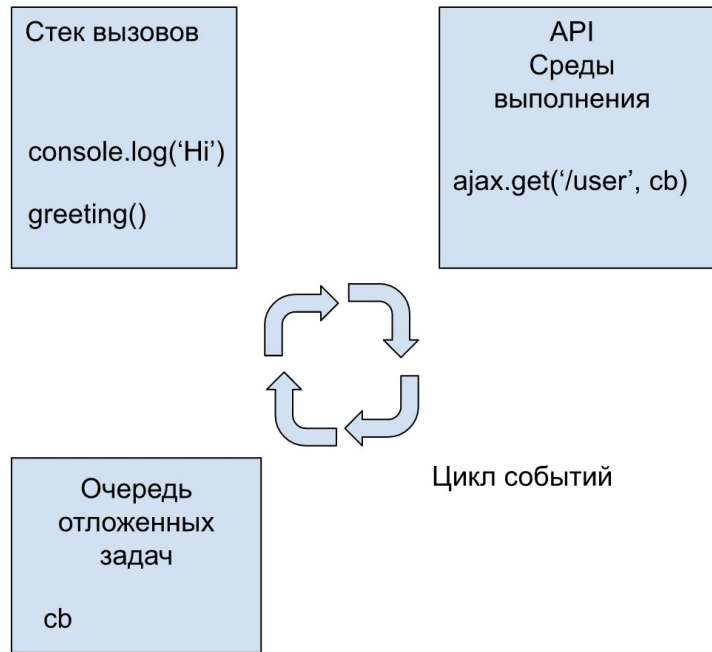
- часть, которая выполняется долго или требует значительных ресурсов,
- часть, которая вызывается по завершению первой части и может обработать результат выполнения первой части или просто сообщает что выполнение первой части кода завершилось.





Event loop

Цикл событий или Event Loop по английски - это механизм, который умеет обрабатывать входящие события (это сам код, и его команды, действия пользователя, взаимодействие с операционной системой), и управлять выполняемыми задачами, исходя из этих событий.





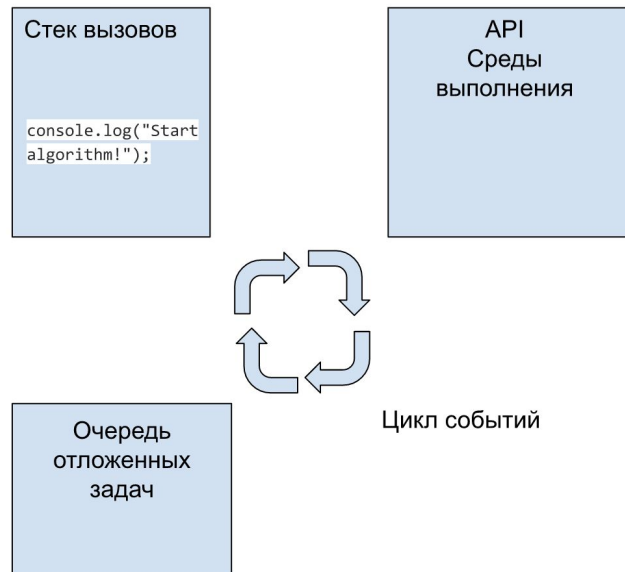
В работе цикла событий участвует еще три механизма:

1. Call Stack (стек вызовов функций) - это механизм, который используется, чтобы сказать процессору какие команды выполнять, по мере исполнения алгоритма. Это основной поток синхронных команд нашего алгоритма.
2. API среды выполнения - это вспомогательный функционал, который предоставляет браузер или другая среда выполнения, так например функция `setTimeout` является не функцией движка JavaScript, а одной из функций среды выполнения.
3. Отложенная очередь задач - это стек задач, который пополняется новыми заданиями, по мере их появления. Так например вызов функции обратного вызова в выполняемой в текущий момент функции будет внесен в этот стек, чтобы цикл событий выполнил эту функцию когда у него будет свободный стек вызовов.



Event loop

Эта команда синхронная, поэтому она выполняется и стек освобождается.





Асинхронность, обусловленная сетевым взаимодействием, AJAX

Сетевые взаимодействия - это наиболее частое применение асинхронности в JavaScript. Все запросы из браузера (или между серверами) работают с помощью протокола HTTP.





HTTP - HyperText Transfer Protocol

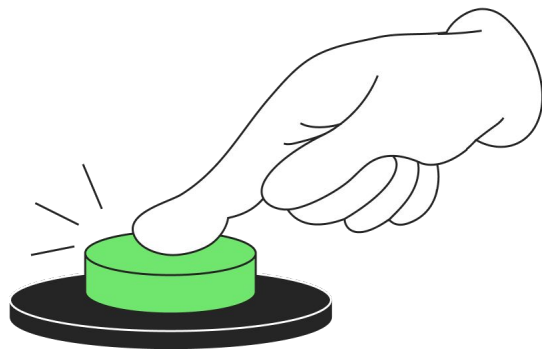
Протокол передачи гипертекста - это протокол передачи данных прикладного уровня, что означает что он использует основной протокол обмена данными в сетях (например TCP/IP), и работает по принципу клиент/сервер.

Для получения данных от сервера нам нужно отправить на указанный адрес запрос. Запрос должен быть определенного типа (GET, POST, PUT, DELETE и другие) и оформлен специальным образом: содержать необходимые заголовки, из которых сервер получает служебную информацию



GET

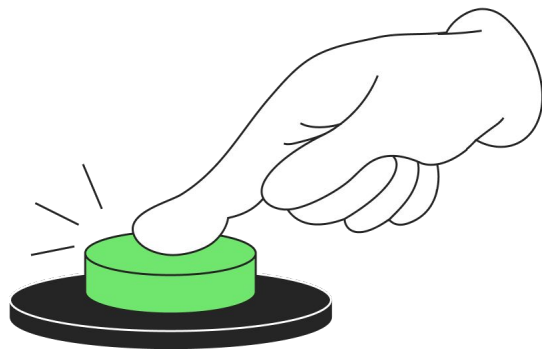
GET запрос нужен для получения данных от сервера. Такой запрос не имеет тела, и все необходимые данные запрашивает с помощью адреса и GET параметров в строке адреса. Параметры в адресе выглядят как <ключ>=<значение>, соединяются с помощью знака & и отделяются от основного адреса знаком ?, например `www.google.com?page=1&amount=20`. GET запрос должен всегда возвращать один и тот же ответ, обращаясь по одному и тому же адресу. С помощью GET запросов можно получать картинки, данные о пользователях и любую другую информацию, хранящуюся на сервере.





POST

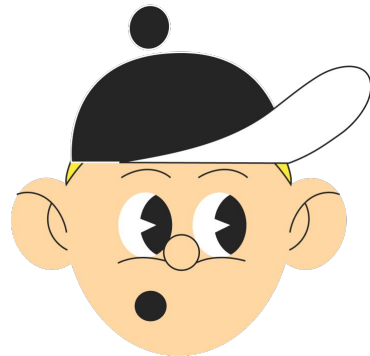
POST запрос нужен для отправки данных на сервер. Чаще всего он используется для отправки данных формы на сервер. POST запрос обязательно передает данные в теле запроса. Сервер может возвращать какие то данные на POST запросы, чаще всего это результат сохранения данных, полученных с этим запросом.





AJAX

Ajax с английского Asynchronous Javascript and XML - технология, которая позволяет сделать асинхронный запрос к серверу для получения данных в виде XML разметки или в каком-нибудь другом виде. Именно эта технология позволила вдохнуть новую жизнь в веб приложения, отправлять данные из форм на сервер без перезагрузки страницы, получать новые данные от сервера и отображать их пользователю в реальном времени.





Объект XMLHttpRequest

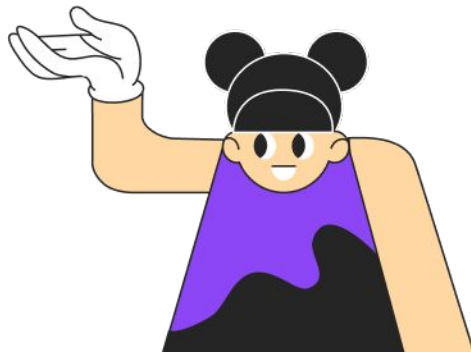
Объект XMLHttpRequest позволяет нам создать объект запроса, который можно отправить на сервер, и обработать ответ. Это базовый механизм, реализуемый движком JavaScript, он не очень удобен для работы, но на нем можно хорошо понять принципы работы с XHR запросами (XHR - сокращение от XMLHttpRequest).





Давайте рассмотрим как отправить POST запрос с помощью объекта XMLHttpRequest.

Для отправки POST запроса нам надо собрать данные для отправки в виде объекта FormData, для этого можно использовать конструктор данного объекта, и дополнить его необходимыми полями.





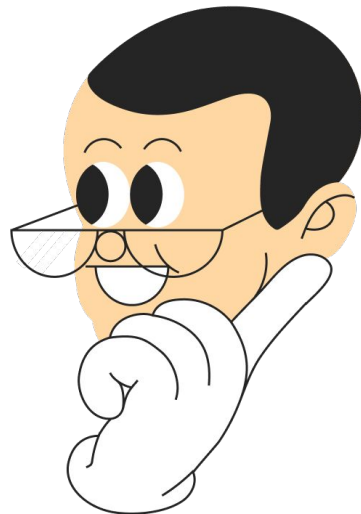
Формат данных JSON





Формат данных JSON

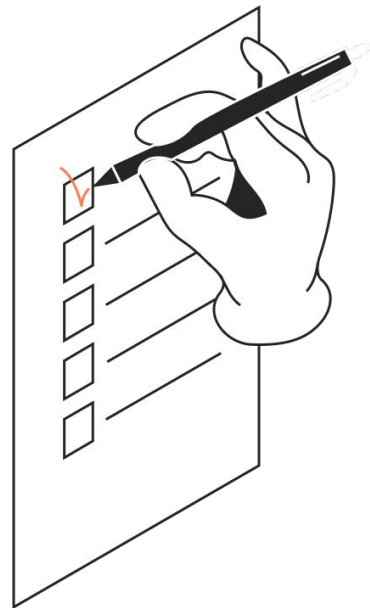
В мире Web есть множества форматов данных с помощью которых мы можем кодировать информацию для отправки на сервер или для получения данных с сервера: XML, простой текст, JSON, двоичные данные. Раньше был достаточно популярным формат XML, но в скором времени его сменил очень простой и удобный формат JSON.





Формат данных JSON

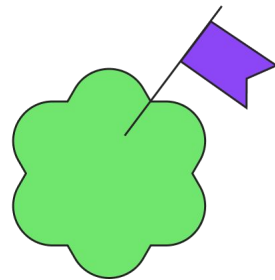
- Все ключи JSON объекта должны быть взяты в кавычки,
- Значения ключей - это примитивные типы, массив или объект. В нем не может быть функций,
- В JSON объекте не допускаются комментарии,
- В конце каждой группы ключей или элементов массивов нельзя ставить запятую - будет ошибка объекта.





JSON.stringify

Метод `stringify` позволяет преобразовать JavaScript объект в строковое представление в JSON формате. Этот очень полезный метод применяется перед отправкой данных на сервер. Также этот метод позволяет сохранить данные в виде объекта из JavaScript в текстовый файл





JSON.parse

Метод `parse` делает обратную процедуру, принимая строку в качестве аргумента, метод пытается распарсить её как строку формата JSON и превратить в объект JavaScript (или массив).





async/await





async/await

Функционал `async/await` для работы с асинхронным кодом (а именно для работы с обещаниями) появился в JS с приходом стандарта ES7, и пока еще не слишком поддерживается браузерами, но есть полифилы для работы с ними.



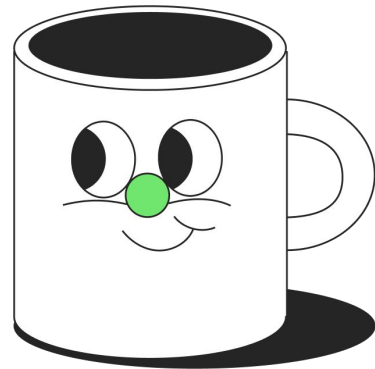


async/await

Суть подхода `async/await` - это писать асинхронный код так, будто он выполняется синхронно, но при этом не блокирует основной поток выполнения.




Подход состоит из применения двух операторов:


- **`async`** - пишется перед функцией и превращает любую функцию в обещание, а также позволяет использовать второй оператор `await` внутри себя.
- **`await`** - оператор пишется перед вызовом асинхронной функции, что заставляет код остановиться в этом месте, пока асинхронная функция не вернет результат.





Итоги

-  Узнали что такое асинхронность
-  Ознакомились с форматом данных JSON
-  Разобрали `async/await`

Спасибо 
за внимание

