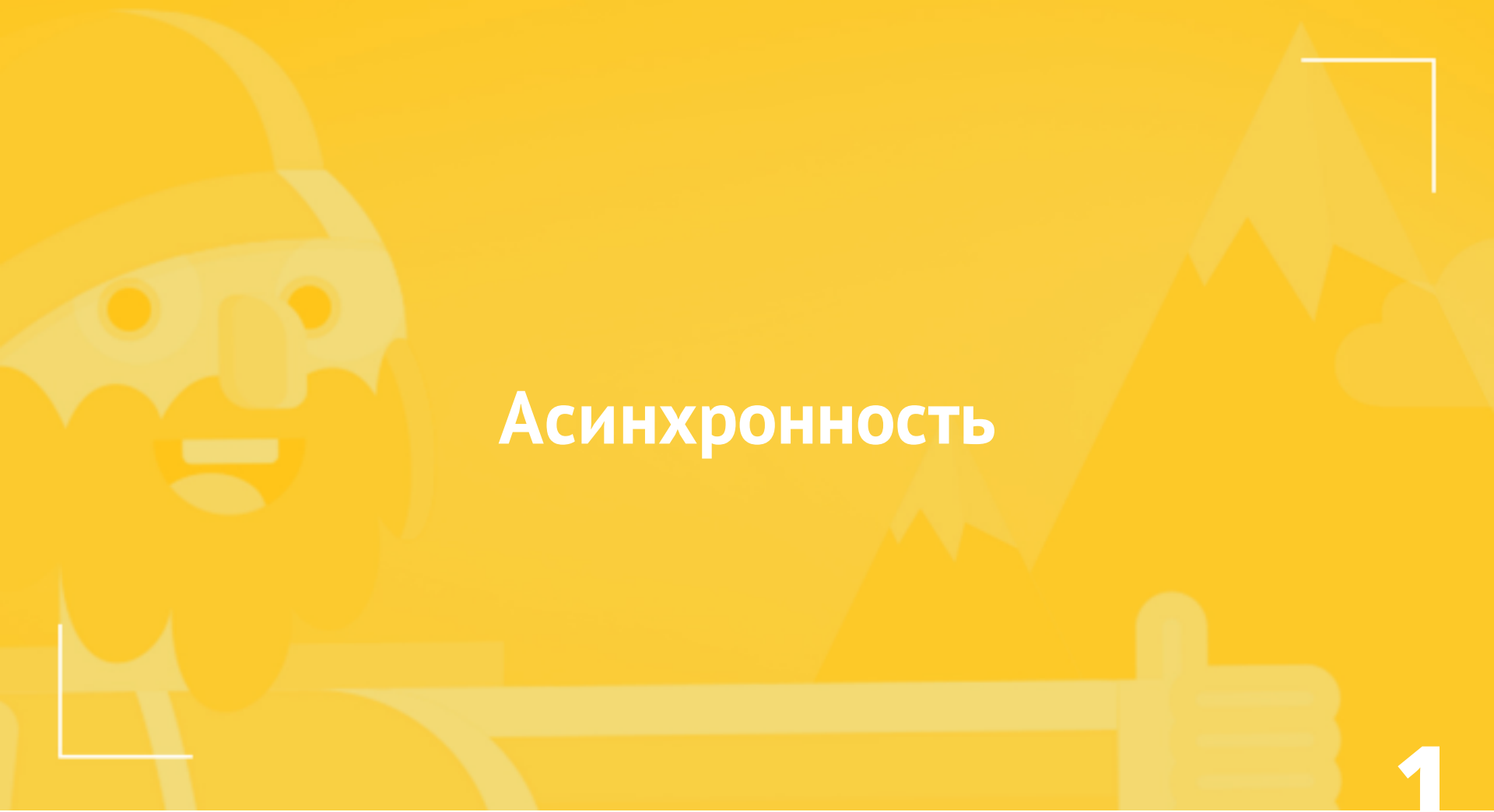


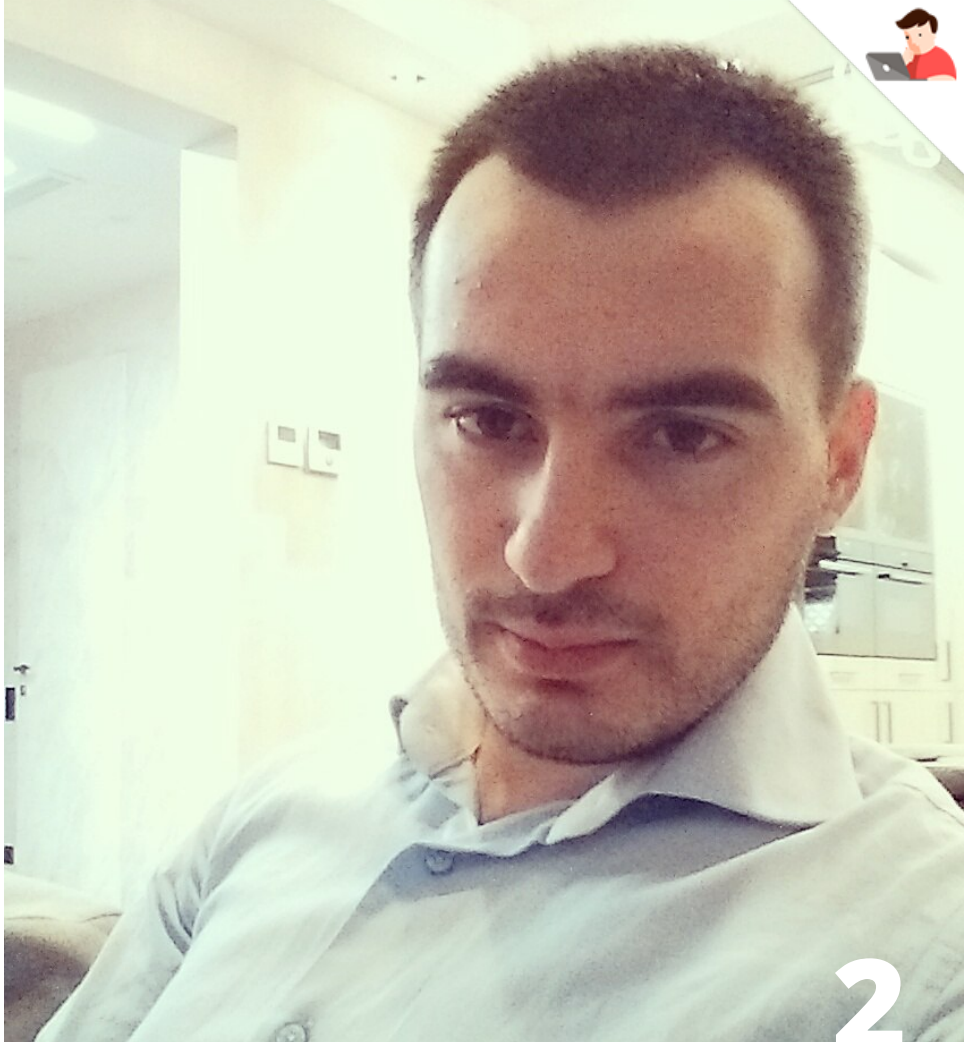
Асинхронность

Ведущий вебинара

Сергей Мелюков

Круг интересов: Backend, Frontend, GameDev, MobileDev

Место работы: Frontend разработчик профессиональных инструментов Avito



Содержание

Таймеры

Таймеры

В JavaScript **нет** встроенного способа **задержать** выполнение скрипта на определенное время, но есть способы запустить код через определенный **промежуток** времени или непрерывно, через **интервал**.

setTimeout(fn, ms)	запустить код один раз через указанное количество миллисекунд
clearTimeout(timeoutId)	отменить назначенный запуск
setInterval(fn, ms)	запускать код с указанной периодичностью
clearInterval(timeoutId)	отменить setInterval

Асинхронность

Асинхронность подразумевает, что можно выполнять **несколько** задач(кусков кода) **одновременно.**

Бытует **неверное** мнение, что JS является асинхронным языком.

В JS создается лишь *видимость* асинхронности за счет так называемого **цикла** событий.

Влиять на цикл события можно при помощи **setTimeout** и **setInterval**.

Эти методы всего лишь *эмитируют* асинхронность.

Promise и Deferred

Promise и Deferred

Promise(промис) - это способ **организации** кода.

Для создания промиса, необходима **функция**, в которой выполняется какая-то **задача**.

При создании промиса, в эту функцию автоматически передаются два аргумента:

- resolve callback
- reject callback

Задача функции - вызвать одну из функций, исходя из результата своей работы (resolve - успех, reject - неудача)

Promise и Deferred

На *полученный* объект промиса можно “*навесить*” так называемые **then callbacks** - функции, которые будут запущены **автоматически**, в зависимости от *результата* выполнения основной функции промиса. Количество таких **callbacks** - не ограничено.

Промисы можно *сгруппировать* в один объект. В этом случае, можно “навесить” на все сгруппированные промисы один **then callback**, который выполнится, когда **все** или **часть** промисов группы будут выполнены или отклонены

Promise и Deferred

Вызвать **resolve** или **reject** можно только *изнутри* основной функции.

Deferred-объект - это небольшая *надстройка* над промисом, которая позволяет вызвать **resolve** или **reject** из *любой* точки кода, а не только изнутри основной функции промиса.

У классического **Deferred** обычно всего **два** метода (**resolve**, **reject**) и одно свойство **promise**, при помощи которого, можно навешивать **then callbacks** на промис Deferred-объекта.

AJAX

AJAX

AJAX - механизм, встроенный в веб-браузер, позволяющий **посылать** запрос на сервер и обрабатывать **ответ** от сервера *не перезагружая* страницу.

Как результат - создается иллюзия desktop-приложения.

За реализацию **AJAX**, отвечает объект **XMLHttpRequest**

XMLHTTPRequest

Для использования AJAX, необходимо:

- создать новый объект типа XMLHTTPRequest
- вызвать метод open(method, url, async)
- добавить обработчики на все или некоторые события
- вызвать метод send()

Результат запроса будет храниться в свойстве response или responseText.

Если возникла ошибка, то http-код ошибки будет храниться в свойстве status.

Обработчики на события:

onloadstart	загрузка началась
onreadystatechange	смена состояния запроса
onprogress	браузер получил новые данные
onloadend	загрузка закончилась (успешно или с ошибкой)

JSON

JSON (JavaScript Object Notation) - формат данных, совместимый со способом описания обычного javascript-объекта.

По умолчанию, XMLHTTPRequest не преобразует http-ответ в json.

Чтобы сделать это, перед отправкой запроса, необходимо установить свойство responseType в ‘json’.

В таком случае ответ, преобразованный в json, будет храниться в свойство response.

Cross Origin Resource Sharing

По умолчанию, браузер не позволит отправлить ajax-запросы на сервер, хост или порт которого, отличный от того, с которого получена страница.

Но при помощи CORS, эта проблема легко решается.

Ее суть сводится к тому, что сервер, при ответе, отдает специальный заголовок: Access-Control-Allow-Origin: ....

Этот заголовок дает браузеру понять - кто может отправлять данному серверу кросс-доменные ajax-запросы.

JSON-P

JSON-P - один из методов, позволяющий обойти ограничения, накладываемые браузером на кросс-доменные запросы.

Суть метода сводится к тому, что браузер загружает специальный javascript-файл, который, после окончания загрузки, вызывает заранее установленную глобальную функцию и передает ей в качестве аргументов - данные.

Задача разработчика состоит в том, чтобы обработать полученные данные.

Чаще всего, такие javascript-файлы генерируются динамически (с подменой заголовков сервером) и не являются javascript-файлами как таковыми.

AJAX в jQuery

$.ajax(параметры)	функция, формирующая ajax-зарпос. Может принимать большое количество параметров
$.get(url, [обработчик], [тип ответа])	отправить get-зарпос
$.post(url, [данные], [тип ответа])	отправить post-зарпос
$(селектор).load(url)	загрузить содержимое из внешнего источника и поместить внутрь элемента

Время ваших вопросов