



# Лекция 8

Взаимодействие с сервером

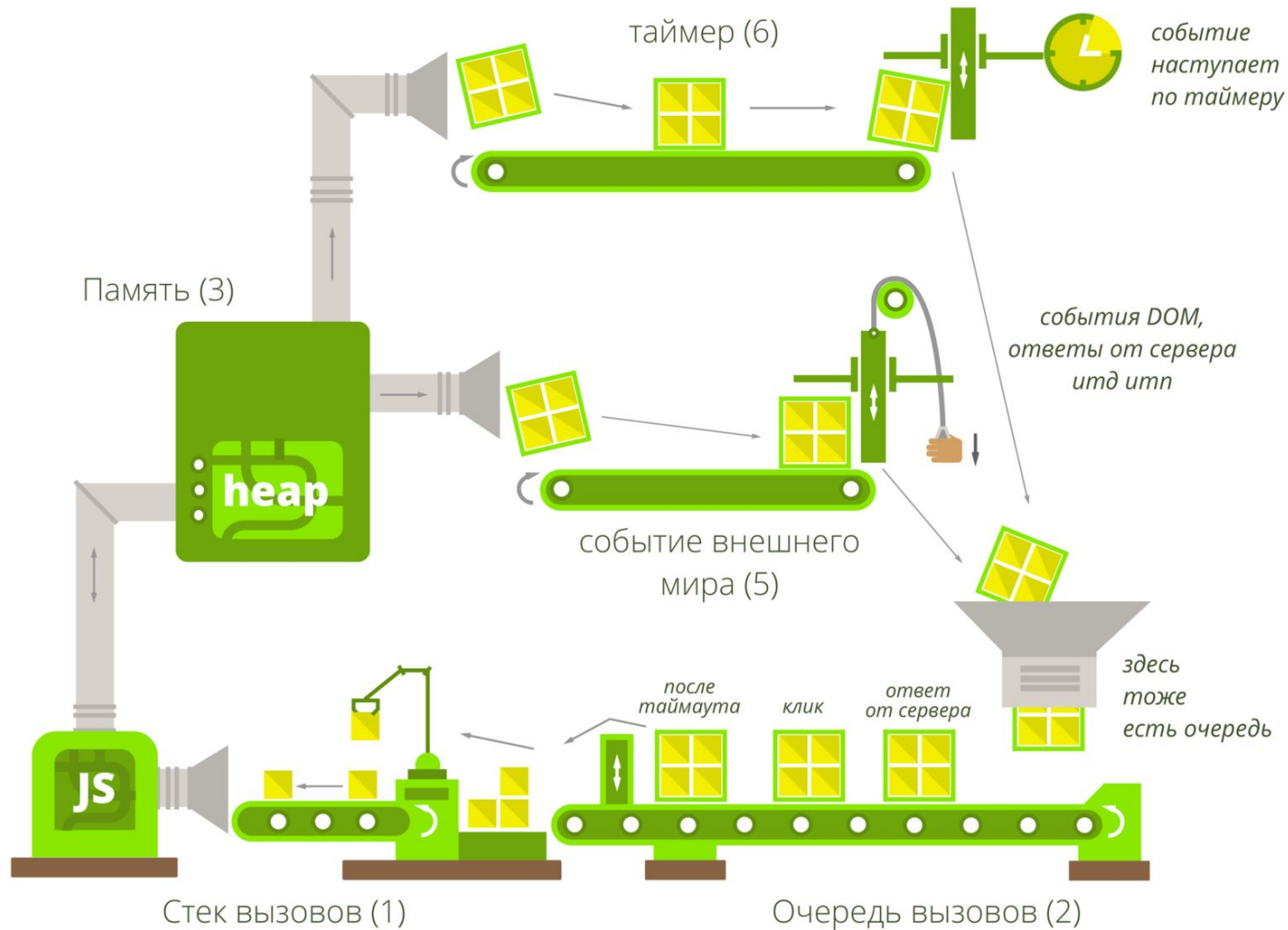
## Что мы научимся делать?

- понимать асинхронность;
- работать с API при помощи XMLHttpRequest;
- работать с разными форматами данных;
- принимать и отправлять данные.

# **Понятие про асинхронность**

## **Асинхронный код –**

код, который выполняется параллельно с выполнением другого кода и не блокирует его.






```
/*
```


В JavaScript можно использовать стандартный таймер, который позволяет задавать вызов функции через заданный период времени.

```
*/
```

```
const timerId = setTimeout(handler, timeout, arguments);
```



В переменную  
записывается  
идентификатор таймера



Функция-обработчик



Задержка **в мс**



Аргументы функции



```
/*
```

```
    setInterval в отличии от setTimeout, запускает выполнение  
    функции не один раз, а регулярно повторяет её через  
    указанный интервал времени.
```

```
*/
```

```
const timerId = setInterval(handler, timeout, arguments);
```



```
const returnData = () => {  
  console.log({  
    name: 'Bruce',  
    age: 23,  
    gender: 'male'  
  });  
}
```

```
const getData = setTimeout(returnData, 1000);
```

Какой результат мы получим?  
И какой будет результат при  
замене **setTimeout** на  
**setInterval**?





```
console.log('1');  
setTimeout(function afterTwoSeconds() {  
    console.log('2');  
}, 2000);  
console.log('3');
```

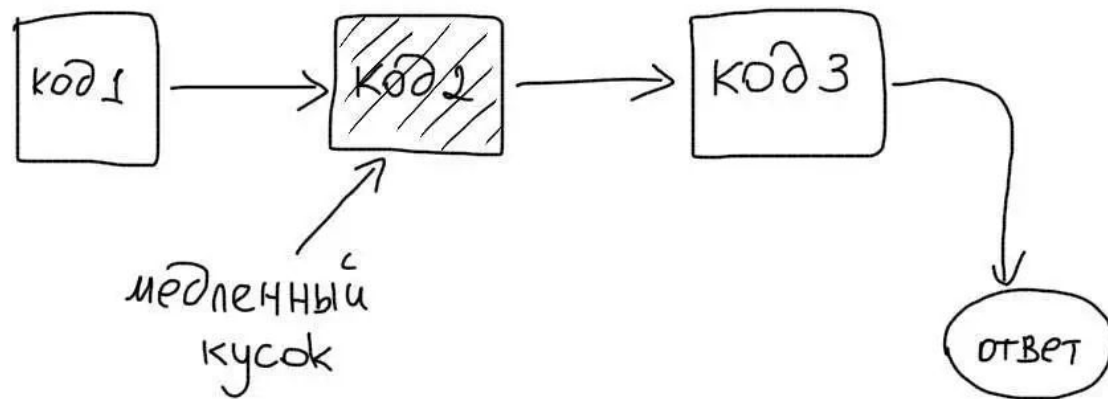
```
// 1
```

```
// 3
```

```
// 2
```

## **Зачем нужна асинхронность?**

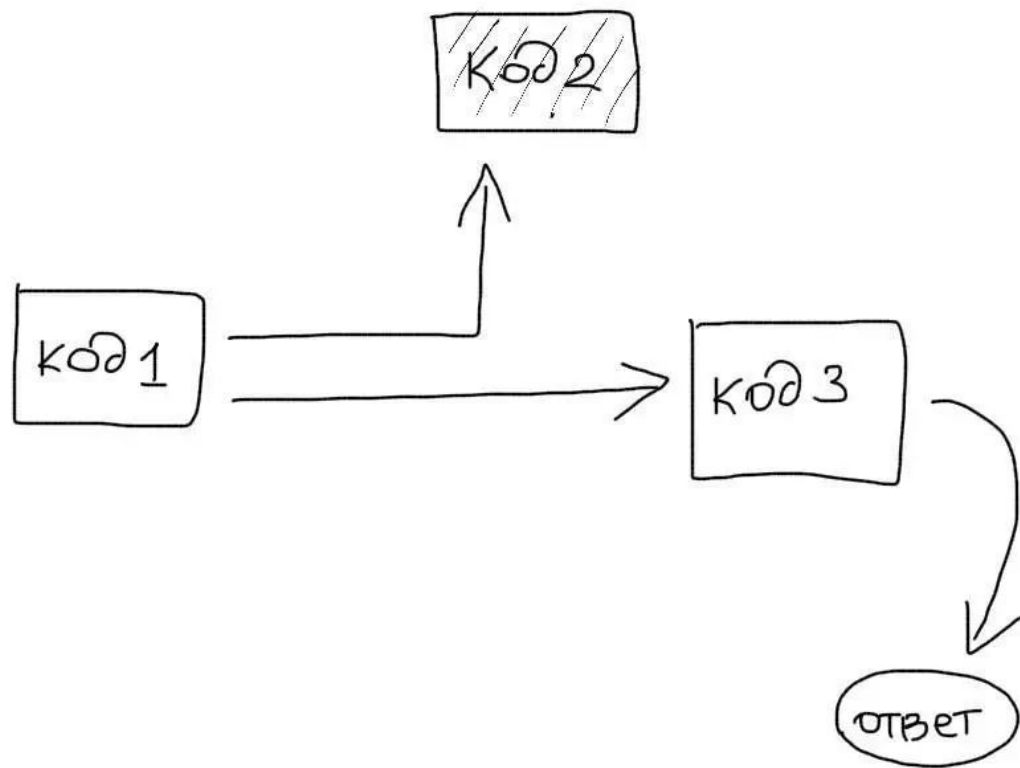
Любое приложение – это набор последовательных инструкций. Чтобы выполнить следующую инструкцию, нужно дождаться окончания выполнения предыдущей. Время выполнения всей программы — это суммарное время выполнения всех инструкций. А это значит, что медленная часть программы будет делать медленной всю программу.



## Где использовать асинхронность?

Когда речь идет о генерации ответа пользователю от Web приложения, многие операции можно "откладывать". Имеет смысл откладывать тяжелые операции, которые могут отнять кучу времени:

- Отправка почты;
- Обработка изображений и медиа данных;
- Обработка и обновление данных статистики;
- Загрузка файлов;
- Отправка внешних запросов и обработка ответов (например, использование различных API).



**Работа с сервером**

## **API –**

это интерфейс программирования, интерфейс создания приложений. Если говорить более понятным языком, то API – это готовый код для упрощения жизни программисту. API создавался для того, чтобы программист реально мог облегчить задачу написания того или иного приложения благодаря использованию готового кода (например, функций).

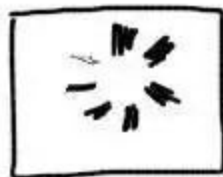
## **AJAX –**

подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных веб-страница не перезагружается полностью, и веб-приложения становятся быстрее и удобнее.



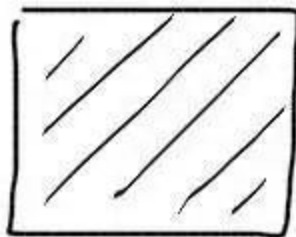
Основная страница

HTML



AJAX

Медленная  
загрузка



# **Форматы передаваемых данных**

# CSV

CSV (от англ. Comma-Separated Values — значения, разделённые запятыми) — текстовый формат, предназначенный для представления табличных данных.

- Каждая строка файла — это одна строка таблицы;
- Разделителем значений колонок является символ запятой (,). Однако на практике часто используются другие разделители;
- Значения, содержащие зарезервированные символы (двойная кавычка, запятая, точка с запятой, новая строка) обрамляются двойными кавычками (").

1997,Ford,E350,"ac,abs,moon",3000.00  
1999,Chevy,"Venture""ExtendedEdition"",,4900.00  
1996,Jeep,GrandCherokee,"MUSTSELL!  
air,moonroof,loaded",4799.00




1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture «Extended Edition»		4900.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

# XML

XML – расширяемый язык разметки, достаточно похожий на HTML, но с более строгим синтаксисом.

В XML поле объекта можно представить в виде отдельного тега, у которого могут быть дополнительные атрибуты. При этом, имена этих тегов и атрибутов придумывает сам разработчик, то есть, тут нет фиксированной грамматики.

Для XML документа создается DOM-дерево и по нему можно двигаться, как по обычному HTML.



```
const text = `
    <book>
        <title>Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
    </book>
</bookstore>`;

const parser = new DOMParser();
const xmlDoc = parser.parseFromString(text, 'text/xml');

console.log(xmlDoc); // XMLDocument { ... }
console.log(xmlDoc.children[0]); // <bookstore>
```



```
<person>
  <firstName>Иван</firstName>
  <lastName>Иванов</lastName>
  <address>
    <streetAddress>Московское ш., 101, кв.101</streetAddress>
    <city>Ленинград</city>
    <postalCode>101101</postalCode>
  </address>
  <phoneNumbers>
    <phoneNumber>812 123-1234</phoneNumber>
    <phoneNumber>916 123-4567</phoneNumber>
  </phoneNumbers>
</person>
```



```
<person firstName="Иван" lastName="Иванов">  
  <address streetAddress="Московское ш., 101, кв.101" city="Ленинград"/>  
  <phoneNumbers>  
    <phoneNumber>812 123-1234</phoneNumber>  
    <phoneNumber>916 123-4567</phoneNumber>  
  </phoneNumbers>  
</person>
```



# JSON

Текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования. Для многих языков существует готовый код для создания и обработки данных в формате JSON.



```
{  
  "firstName": "Иван",  
  "lastName": "Иванов",  
  "address": {  
    "streetAddress": "Московское ш., 101, кв.101",  
    "city": "Ленинград",  
    "postalCode": "101101"  
  },  
  "phoneNumbers": [  
    "812 123-1234",  
    "916 123-4567"  
  ]  
}
```

**JSON.parse** – превратит строку с данными в формате JSON в JavaScript-объект/массив/значение.

**JSON.stringify** – преобразует («сериализует») значение в JSON-строку.

# **XMLHttpRequest (XHR)**

**Объект XMLHttpRequest** дает возможность делать HTTP-запросы к серверу без перезагрузки страницы.

Несмотря на слово «XML» в названии, XMLHttpRequest может работать с любыми данными, а не только с XML.



```
const xhr = new XMLHttpRequest();

xhr.open('GET', 'https://api.coindesk.com/v1/bpi/currentprice.json');

xhr.send();

xhr.onreadystatechange = function() {
    if (this.readyState !== 4) return;

    if (this.status !== 200) {
        throw new Error('Запрос не удался');
    } else {
        console.log(xhr.responseText);
    }
}
```

```
xhr.open(method, URL, async, user, password) ;
```

**method** – один из HTTP методов;

**URL** – адрес, по которому осуществляется запрос;

**async** – запрос асинхронный?;

**user, password** – логин и пароль для HTTP-авторизации, если нужны.

**xhr.send([body]);** – открывает соединение и отправляет запрос на сервер. **body** – тело запроса для POST и ему подобных методов.

**xhr.abort()** – прерывает запрос.

**xhr.setRequestHeader(name, value)** – устанавливает заголовок.



**xhr.status** – возвращает HTTP статус ответа (200, 404);

**xhr.statusText** – текстовое описание статуса (OK, Not Found);

**xhr.responseText** – текст ответа ({ "user": "Vasya" }).

**CORS**

```
✖ XMLHttpRequest cannot load http://localhost:5000/testRoute. Response to :8000/#!/preprocess:1
preflight request doesn't pass access control check: No 'Access-Control-Allow-Origin' header is
present on the requested resource. Origin 'http://localhost:8000' is therefore not allowed
access.

✖ ► Possibly unhandled rejection: {"data":null,"status":-1,"config": angular.js:14700
{"method":"POST","transformRequest":[null],"transformResponse":
[null],"jsonpCallbackParam":"callback","url":"http://localhost:5000/testRoute","data":
{"form_data":{}},"headers":{"Accept":"application/json, text/plain, /*/*","Access-Control-Allow-
Origin":"*","Content-Type":"application/json;charset=utf-
8"}}, "statusText":"","xhrStatus":"error"}
```

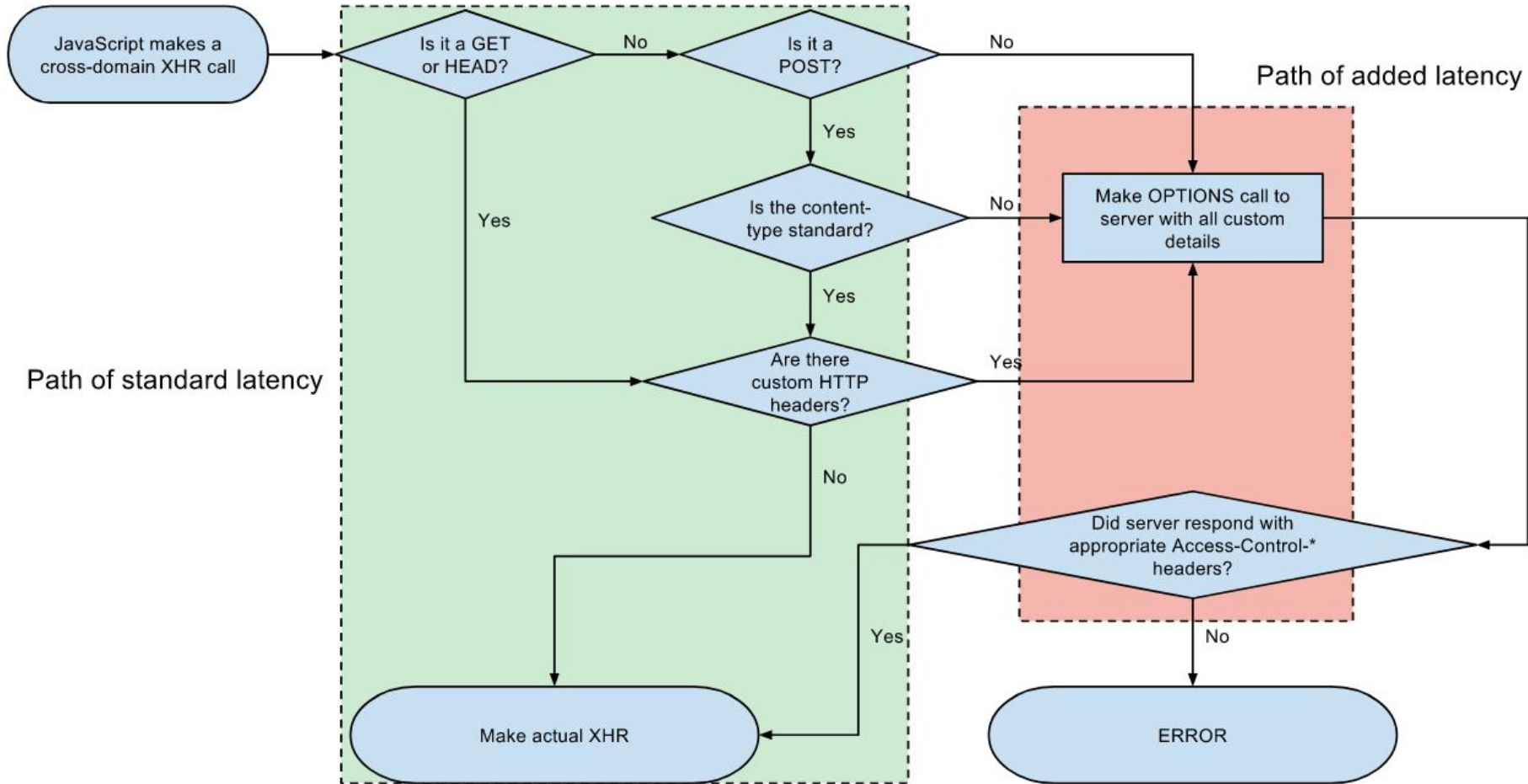
**Cross-origin resource sharing (CORS;** англ. — «совместное использование ресурсов между разными источниками») — технология современных браузеров, которая позволяет предоставить веб-странице доступ к ресурсам другого домена. Раньше для решения этой задачи использовали **JSONP**.

# Кросс-доменные запросы

**Простыми** считаются запросы, если они удовлетворяют следующим двум условиям:

- Простой метод: GET, POST или HEAD;
- Простые заголовки – только из списка:
  - **Accept**
  - **Accept-Language**
  - **Content-Language**
  - **Content-Type** со значением *application/x-www-form-urlencoded*, *multipart/form-data* или *text/plain*.

Все остальные запросы считаются **сложными**.



**Ад колбеков (callback hell)**

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err);
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename);
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err);
        } else {
          console.log(filename + ' : ' + values);
          var aspect = (values.width / values.height);
          widths.forEach(function (width, widthIndex) {
            var height = Math.round(width / aspect);
            console.log('resizing ' + filename + 'to ' + height + 'x' + height);
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function (err) {
              if (err) {
                console.log('Error writing file: ' + err);
              }
            });
          });
        }.bind(this));
      }
    });
  }
});
```

## Как выбраться из ада обратных вызовов?

- Не используйте большую вложенность;
- Используйте именованные функции;
- Используйте модульность;
- Обработывайте каждую ошибку.

Читай больше на  
<http://callbackhell.ru>



## Самостоятельная работа

Написать страницу, которая подтягивает данные о текущей погоде в Запорожье, используя **Dark Sky API**.

- Необходимо выводить на странице такие данные: температуру воздуха, атмосферное давление, описание погоды словами (summary) и иконку;
- Все данные должны быть на русском языке;
- Все данные должны быть в системе Си (si);
- Иконки придется искать самим.