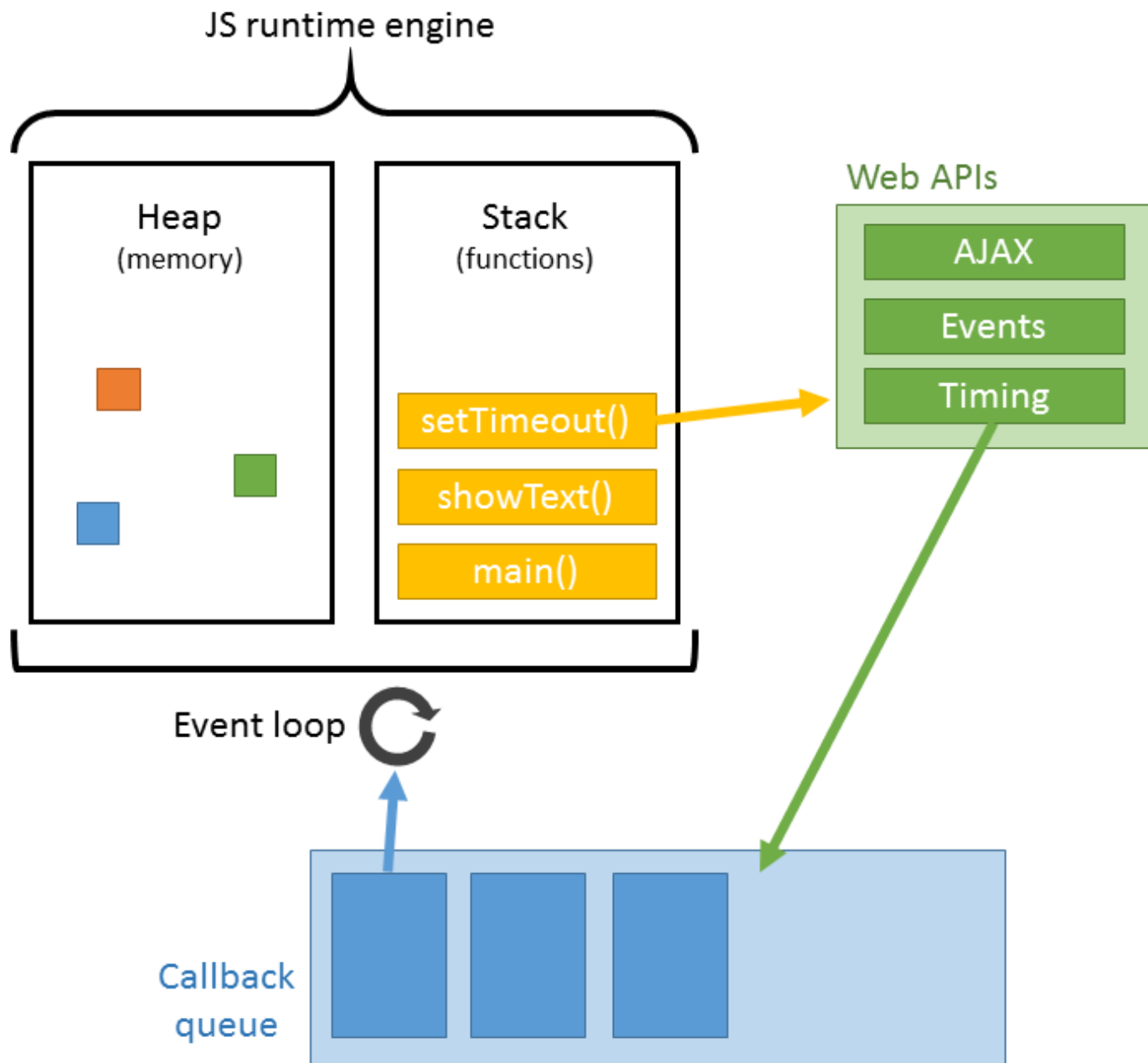


АJAX и асинхронность в JavaScript

JS
COURSE
ORT DNIPRO

ORT**DNIPRO**.ORG/**JS**

1. Event Loop



Вспомним об **Event Loop** в JavaScript

JavaScript – однопоточный язык программирования, в любой момент времени возможно выполнение только одного фрагмента кода. Но есть механизм **callback**'ов...

Подробнее:

<https://developer.mozilla.org/ru/docs/Web/JavaScript/EventLoop>

2. Обект Promise

Объект Promise

```
2
3  ✓ let promise = new Promise((resolve, reject) => {
4
5  ✓   setTimeout( () => {
6     let result = 2 ** 10;
7     resolve(result);
8     //reject('Operation imposible');
9   }, 5000);
10
11 });
12
13 promise.then( result => console.log('Successful, result is', result) );
14
15 promise.catch( error => console.log('Failed, error is', error) );
16
17 promise.finally( () => console.log('Promise Finished'));
18
19 console.log("After Promise");
20
```

Promise – механизм позволяющий писать асинхронный код последовательно (насколько это возможно), избегая вложенности **callback**’ов. **Promise** – объект который принимает функцию, в которой запускается асинхронная операция, при помощи параметров функции есть возможность из асинхронного кода сообщить об успешном или не успешном завершении операции (параметры **resolve**, **reject** – функции вызов которых приведёт к завершению работы **Promise**’а).

Подробнее: <https://learn.javascript.ru/promise/>

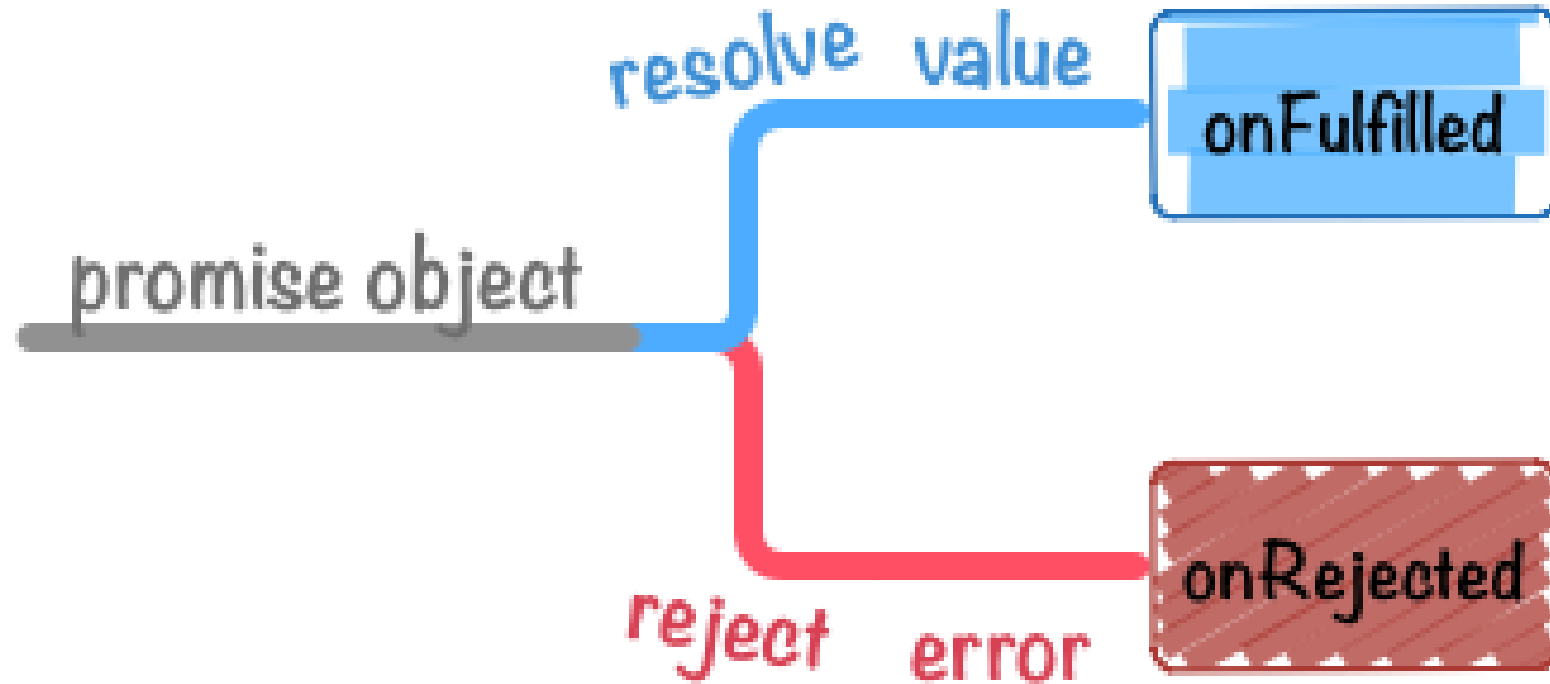
Объект Promise

```
2
3 ✓ let promise = new Promise((resolve, reject) => {
4
5 ✓   setTimeout( () => {
6     let result = 2 ** 10;
7     resolve(result);
8     //reject('Operation imposible');
9   }, 5000);
10
11 });
12
13 promise
14   .then( result => console.log('Successful, result is', result) )
15   .catch( error => console.log('Failed, error is', error) )
16   .finally( () => console.log('Promise Finished') );
17
18 console.log("After Promise");
19
```

У объекта **Promise** есть 3 полезных метода для возможности зарегистрировать функции на случай успешного завершения **Promise**'а, для случая завершения с ошибкой, и для ситуации когда код нужно выполнить как бы **Promise** не завершился (успешно или нет). Эти методы соответственно **.then()**, **.catch()** и **finally()**. Эти методы могут быть вызваны цепочкой т.к. эти методы возвращают ссылку на сами **Promise**. Функция переданная **.then()** может вернуть результат (в т.ч. другой Promise) и цепочка может опять включать **.then()** для его обработки.

Подробнее: <https://learn.javascript.ru/promise/>

Жизненный путь Promise



Жизненный путь **Promise** всегда завершается одним из двух состояний:
Fulfilled – успешное завершение, либо **Rejected** – неудачное завершение.

2. Promise API

Promise API

Благодаря методу **Promise.all()** есть возможность «подождать» **успешного** завершения всех *Promise*'ов, для последующей обработки результатов;

Метод **Promise.allSettled()** – позволяет дождаться всех *Promise*'ов, **независимо** от результата;

Promise.race() позволяет дождаться только первого завершенного *Promise*'а (**успешного или неуспешного**);

Promise.any() – возвращает первый **успешно** завершенный *Promise* из переданной коллекции *Promise*'ов.

Подробнее: <https://learn.javascript.ru/promise-api>

3. AJAX

Asynchronous JavaScript And XML

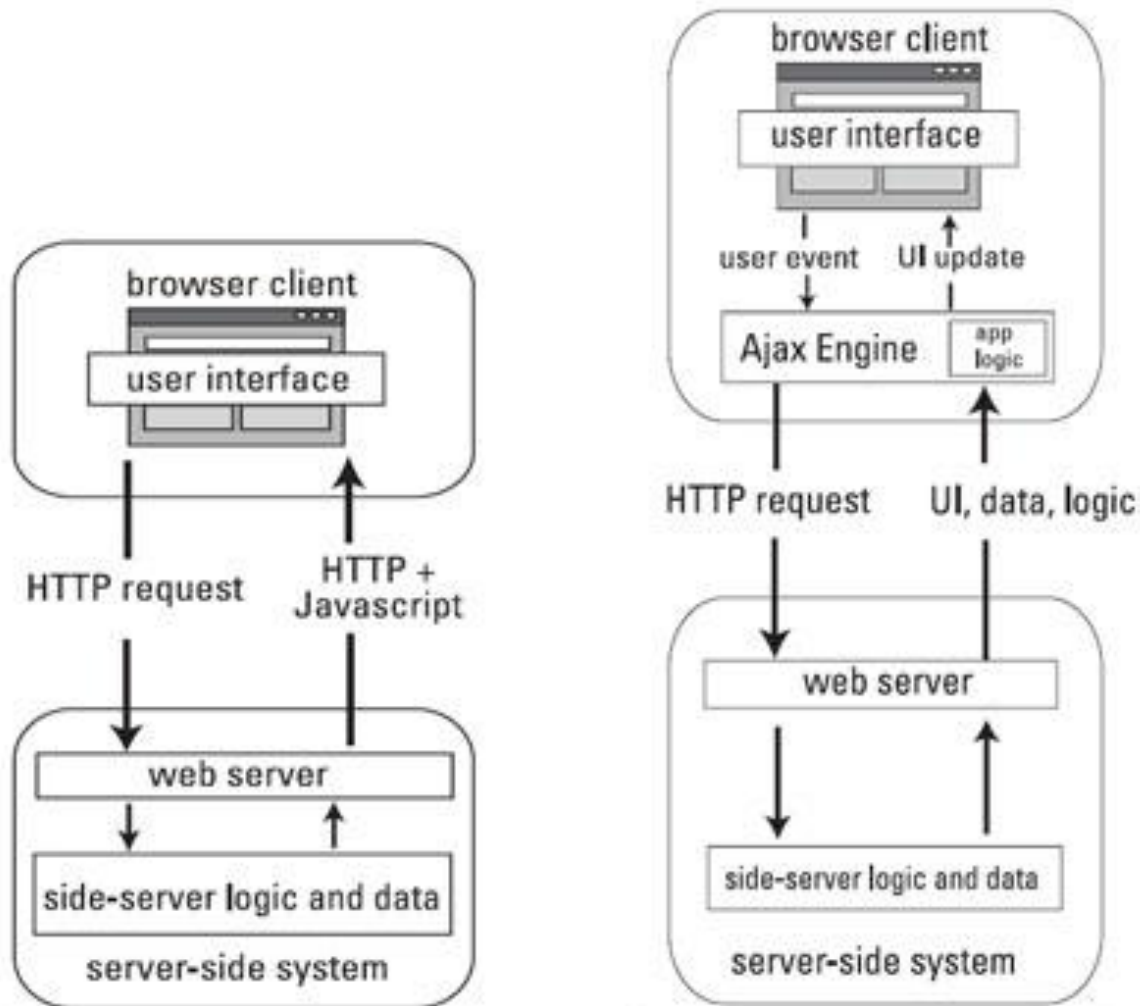
Asynchronous JavaScript And XML



За изменение страницы в браузере пользователя отвечает JavaScript, но до этого момента **JavaScript** изменял страницу только на основе данных полученные еще при загрузке страницы в браузер и/или в зависимости от действий пользователя. Получить какие-то новые (дополнительные) данные **JavaScript** не мог.

С появлением в браузерах специального объекта **XMLHttpRequest** у **JavaScript** появилась возможность делать **HTTP-запросы** (*HTTP-request*) к сайтам, и изменять страницу уже на основе данных которых не было при загрузке странице. Т.е. дополнительно загружать разметку и/или другие данные и вставлять их на страницу.

Asynchronous JavaScript And XML



Идея заложенная в **AJAX** – не перезагружая страницу полностью, запросить у сервера данные и вставить их в дерево документа.

4. fetch()

AJAX на **Promise**'ax

fetch() – Promise «обёртка» для выполнения AJAX-запросов

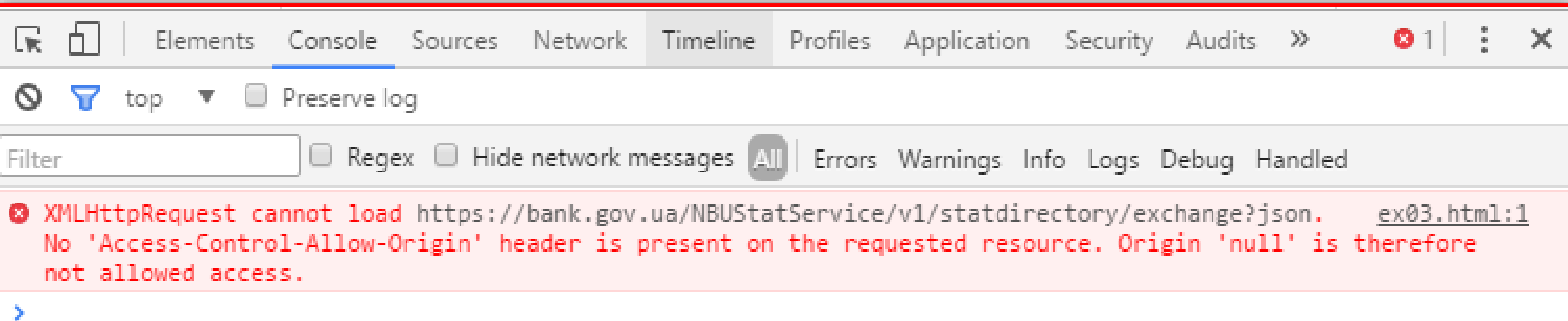
```
2  
3 let url_nbu = 'https://bank.gov.ua/NBUStatService/v1/statdirectory/ovdp?json';  
4  
5 fetch(url_nbu)  
6   .then(result => result.json())  
7   .then(result => console.log(result))  
8   .catch(error => console.log("Error:", error));  
9  
10
```

Функция **fetch()** – выполняет AJAX-запросы, возвращая **Promise**, который завершится с поступлением ответа на запрос или завершится с ошибкой, если запрос будет неудачный.

Подробнее: <https://learn.javascript.ru/fetch/>

5. Кросс-доменные запросы

Кросс-доменные запросы



Не все AJAX запросы безопасны, браузер бдит 😊

Сравните результаты
AJAX запросов:

https://courses.dp.ua/ajax/demo_1.php

https://courses.dp.ua/ajax/demo_2.php

Кросс-доменные запросы

Кросс-доменные запросы (т.е. запросы к другому домену, не к тому с которого загружен скрипт) проходят контроль безопасности (**который осуществляет браузер**).

Чтобы страница могла быть доступна через кросс-доменные запросы (читай **AJAX** запросы к страницам других сайтов), страница должна сама сказать об этом, а именно установить в **HTTP** ответе заголовок **Access-Control-Allow-Origin**.

Подробнее: <https://learn.javascript.ru/xhr-crossdomain>

6. API

(Application Programming Interface)

Application Programming Interface

API (интерфейс программирования приложений, интерфейс прикладного программирования)
([англ.](#) *application programming interface*) — По сути набор правил, которые определяют как необходимо общаться со сторонним сайтом/программой/системой если мы хотим запросить у него данные или передать ему данные.

7. **async**/**await**

async/await – упрощение кода Promise'ов

```
2
3 let url = 'https://bank.gov.ua/NBUStatService/
4           v1/statdirectory/exchange?json';
5
6 (async function(){
7     let result = await fetch(url);
8     result     = await result.json();
9
10    console.log(result);
11
12 })();
13
```

async/await – надстройка над **Promise** позволяющая писать код в полностью привычном синхронном стиле, при этом откладывая ожидания завершения операций до тех пор пока её результат действительно понадобится;

async – отмечает функцию как асинхронную (результат такой функции оборачивается в **Promise**);

await – при вызове асинхронных функций указывает, что не нужно ждать результата сейчас

Подробнее: <https://learn.javascript.ru/async-await>

Цикл for-await-of

```
23  
24     let promises = [new Promise(), new Promise(), new Promise(), ];  
25  
26     for await(p of promises){  
27         |     console.log( p.someResultData );  
28     }  
29
```

Цикл **for-await-of** позволяет перебрать итерируемую (перебираемую, массив или псевдомассив) состоящий из объектов типа **Promise**. Цикл будет ожидать когда разрешится каждый из **Promis'ов** и только тогда начинать выполнение каждого шага цикла.

Подробнее: <https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Statements/for-await...of>

8. Полезные API


COVID-19 API



A free API for data on the Coronavirus

<https://covid19api.com/>

API Приватбанка

 Разработчикам

API ПриватБанкаAPI юр.лицAPI LiqPay

Главная

Регистрация


Платежные


Информационные


Публичные


Доступные API ПриватБанка

ПриватБанк стал первым банком в мире, открывшим публичное API (в сентябре 2009 года). Сегодня его используют уже более 4 900 партнеров и не только в Украине.


Платежные


Информационные


Публичные


API для юр.лиц

Архив курсов валют ПриватБанка, НБУ

API позволяет получить информацию о наличных курсах валют ПриватБанка и НБУ на выбранную дату. Архив хранит данные за последние 4 года

Документация

Инфраструктура ПриватБанка. Банкоматы

Информация о размещении банкоматов ПриватБанка

Документация

Отделения

Информация о размещении отделений ПриватБанка

Документация

Курсы валют ПриватБанка

API предоставляет информацию о наличных, безналичных курсах валют ПриватБанка


Документация

Терминалы самообслуживания

Информация о размещении терминалов самообслуживания ПриватБанка

Оплата частями

API позволяет получить информацию о доступной сумме для покупок по программе "Оплата частями"



© 2014-2019 ПриватБанк

<https://api.privatbank.ua/>

25

API Национального Банка Украины



**НАЦІОНАЛЬНИЙ
БАНК УКРАЇНИ**

Валютные API, информация о финансовом рынке и банковском секторе

<https://bank.gov.ua/ua/open-data/api-dev>

Информация по платёжной карте

BINLIST.NET

4571 7360

Enter the first digits of a card number (BIN/IIN)

SCHEME / NETWORK

Visa

TYPE

Debit / Credit

BANK

Jyske Bank, Hjørring

www.jyskebank.dk

+4589893300

BRAND

Visa/Dankort

PREPAID

Yes / No

CARD NUMBER

LENGTH

16

LUHN

Yes / No

COUNTRY

DK Denmark

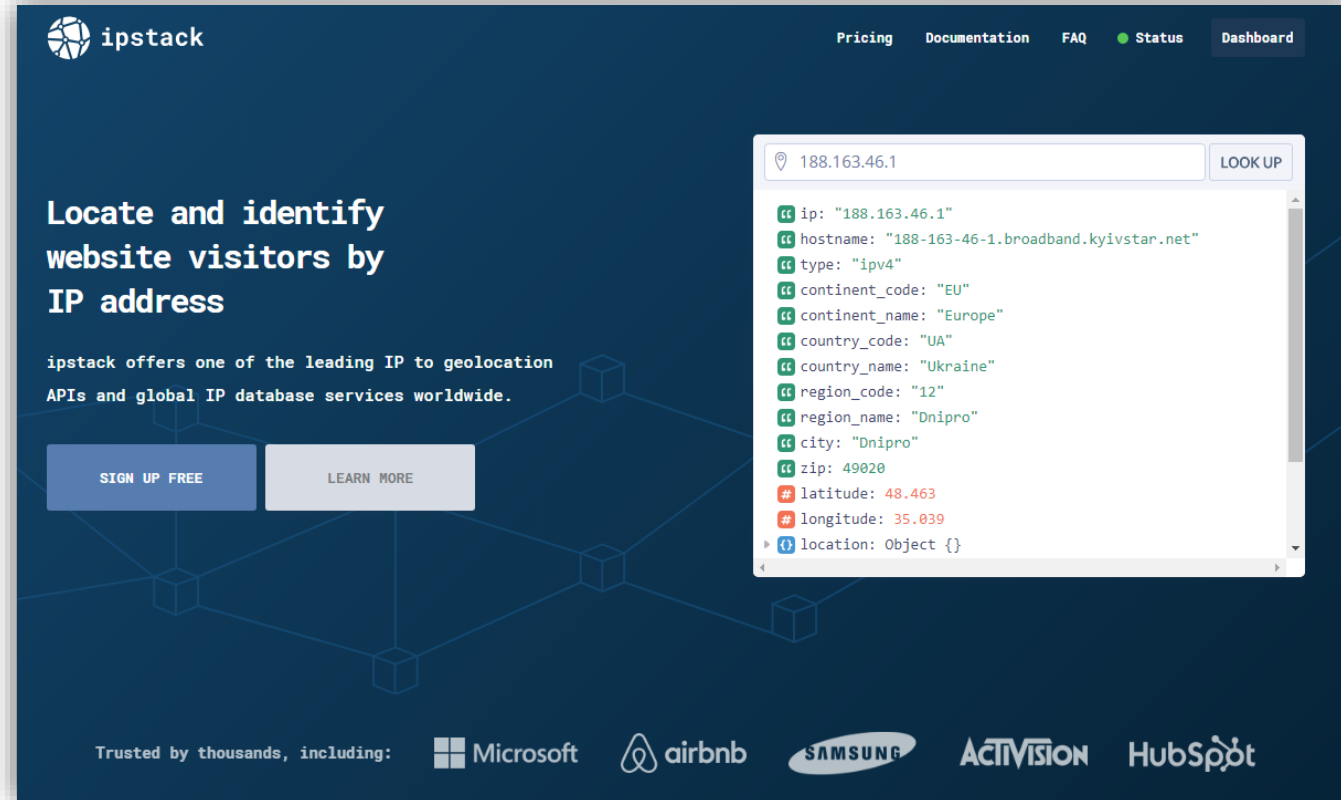
(latitude: 56, longitude: 10)

Сервис позволяют получить информацию в формате **JSON**. Но необходимо зарегистрироваться и получить ключ

<https://binlist.net> | <https://lookup.binlist.net/536354>

Альтернативный сервис: <https://www.bincodes.com/api-bin-checker/>

API WHOIS-сервиса



*Сервис позволяют получить информацию в формате **JSON**. Но необходимо зарегистрироваться и получить ключ*

<https://ipstack.com/>

Weather forecast

[Home](#) / [Weather forecast](#)

Your city name [↗ Current location](#)

Current weather and forecasts in your city

[Main](#) [Daily](#) [Hourly](#) [Chart](#) [Map](#)

Weather in Dnipropetrovsk, UA

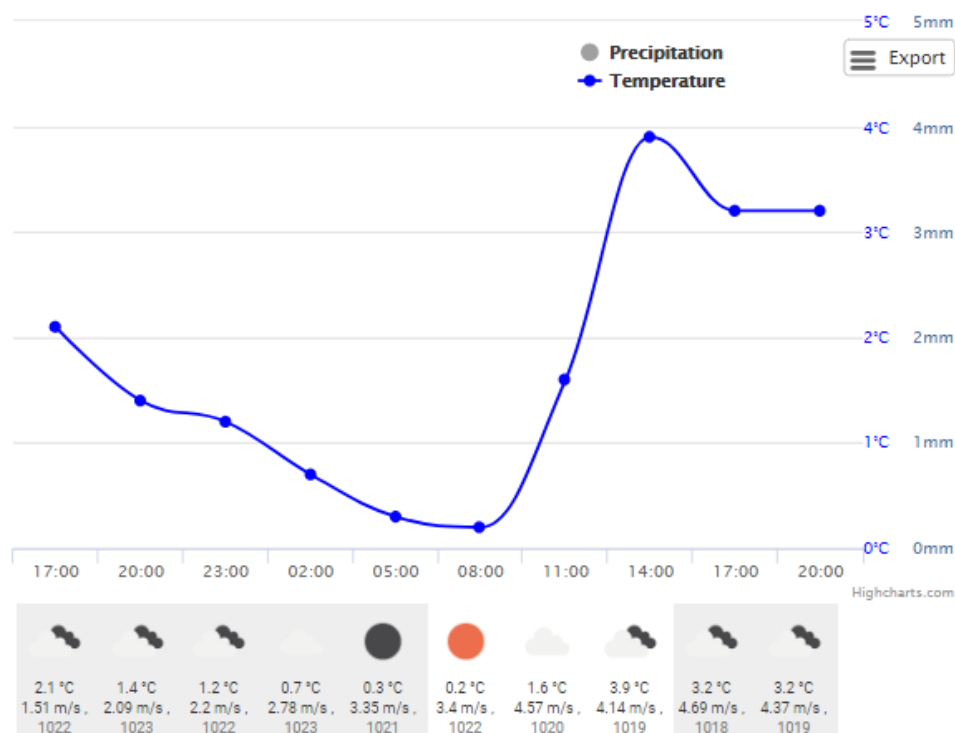
 4 °C

Fog

16:03 Dec 11 Wrong data?

Wind	Calm, 1.5 m/s, East (88)
Cloudiness	Overcast clouds
Pressure	1021 hpa
Humidity	100 %
Sunrise	07:21
Sunset	15:45
Geo coords	[48.45, 34.98]

The weather forecast is displayed in accordance with your local time. Please pay attention to it when you will watch the weather in another time zone.



Погодный API сервиса OpenWeather

Сервис требует
регистрации и
использование ключа
при выполнении
запросов

<https://openweathermap.org/>

Current & Forecast weather data collection

Current weather data

[API doc](#) [Subscribe](#)

- Access current weather data for any location including over 200,000 cities
- Current weather is frequently updated based on global models and data from more than 40,000 weather stations
- Data is available in JSON, XML, or HTML format
- Available for Free and all other paid accounts

Hourly forecast ^{NEW}

[API doc](#) [Subscribe](#)

- Hourly forecast is available for 4 days
- Forecast weather data for 96 timestamps
- Higher geographic accuracy
- Forecast is available in JSON and XML
- Available for Developer, Professional and Enterprise accounts

16 day / daily forecast

[API doc](#) [Subscribe](#)

- 16 day forecast is available at any location or city
- 16 day forecast includes daily weather
- Forecast is available in JSON and XML
- Available for all paid accounts

We have combined **Weather services** and **Satellite imagery** in a simple and fast **Agro API**. We have also launched a **Dashboard** for it - it is a **visual service** where you can easily work with satellite imagery and weather data for your fields: satellite, weather and historical data, soil temperature and moisture, accumulated temperature and precipitation, etc.
Learn more on agromonitoring.com

Climate forecast for 30 days ^{NEW}

[API doc](#) [Subscribe](#)

- Forecast weather data for 30 days
- Based on a statistical approach to our **Historical weather data**
- Forecast is available only in JSON format
- The frequency of weather data update is 1 hour
- Available for Developer, Professional and Enterprise accounts

Bulk downloading

[API doc](#) [Subscribe](#)

- We provide number of bulk files with current weather and forecasts
- Current weather bulk is available for 209,000+ cities
- Variety of hourly and daily forecast bulks depends on frequency of data updating
- Available for Professional and Enterprise accounts

5 day / 3 hour forecast

[API doc](#) [Subscribe](#)

- 5 day forecast is available at any location or city
- 5 day forecast includes weather data every 3 hours
- Forecast is available in JSON and XML
- Available for Free and all other paid accounts

Погодный API сервиса OpenWeather

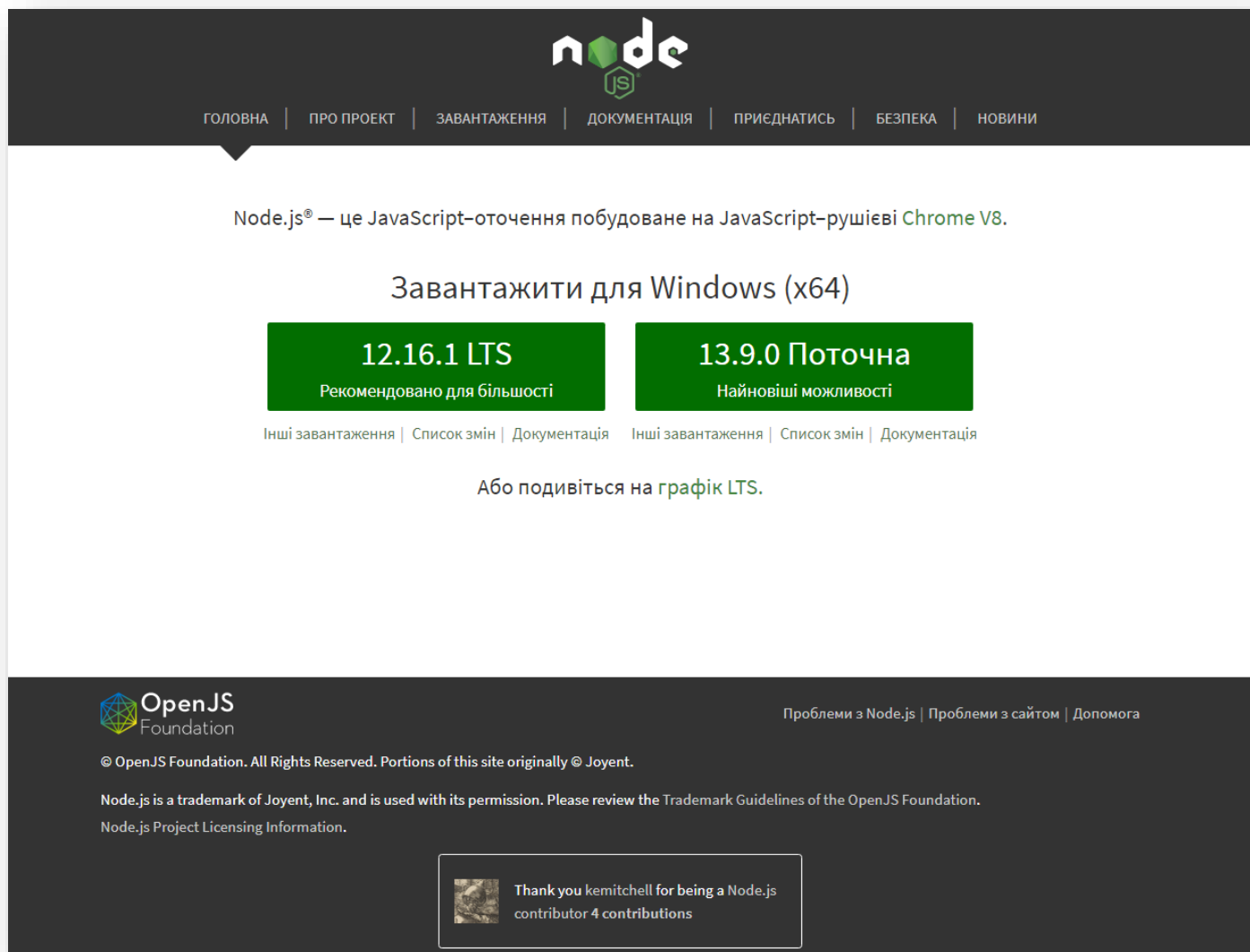
Сервис требует
регистрации и
использование ключа
при выполнении
запросов

9f118bfa230072d3603183e520cea4af

<https://openweathermap.org/>

9. Node.js

JavaScript вне браузера



Node.JS

Node.JS – «виртуальная машина» способная выполнять JavaScript-код, которую можно установить на компьютере, и которая построена на базе «куска» браузера «Chrome» отвечающего за обработку JavaScript. **Node.JS** позволил превратить JavaScript в язык общего пользования, поставив его в один ряд с Python, Ruby, Java, C# и другими.

<https://nodejs.org/>

Node.JS

Node.JS — «чистый» **ECMAScript**. И самый «свежий» **ECMAScript**, для разработчика, т.к. применяя **Node.JS** не нужно думать о совместимости с браузерами.

Если в браузере инструментарий взаимодействия с пользователем обеспечивается объектом **globalThis** (*window*), то в его **Node.JS** версии такого функционала нет, в нём «из коробки» нет ничего кроме чистого синтаксиса **JavaScript/ECMAScript-2015/2016/2017/2018/2019/2020/...**

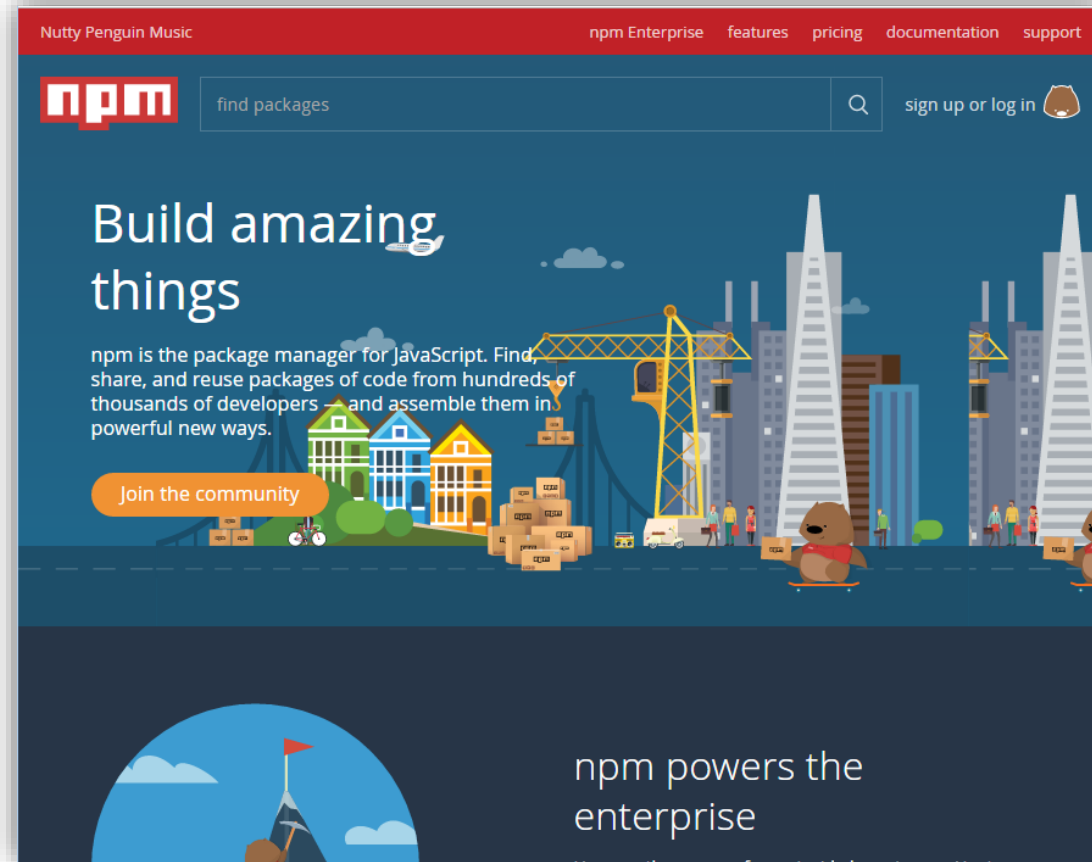
Но, в **Node.JS** есть **модули** (*пакеты*) для решения тех или иных задач (*по аналогии с подключаемыми файлами в других языках*), и модулей для него существует большое количество.

9. NPM

(Node Package Manager)

NPM – Node Package Manager

Всемирная библиотека пакетов (модулей) для Node.JS



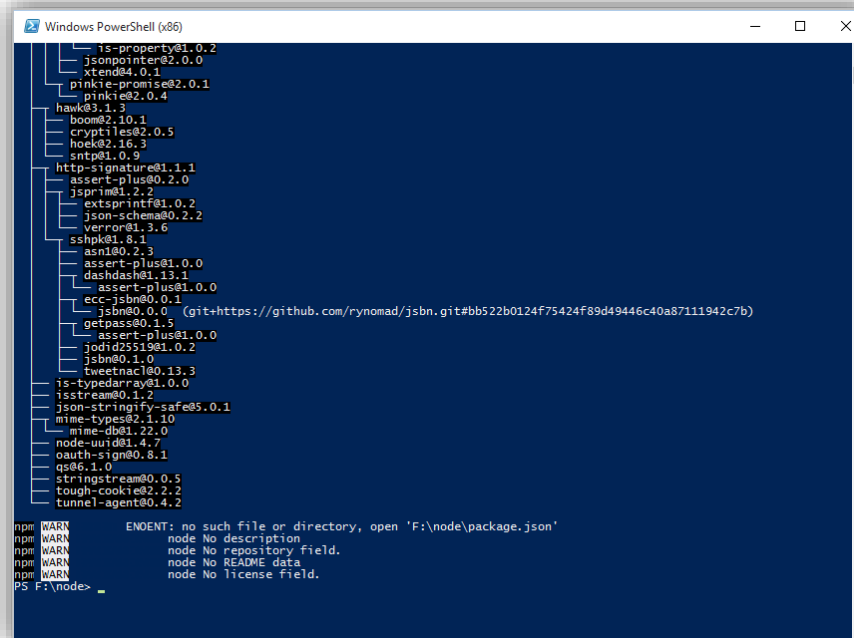
<https://www.npmjs.com/>

NPM – Node Package Manager

Когда нам нужен пакет, то пишет в консоли:

npm install *имя_модуля*

И система управления пакетами (NPM) установит в текущую папку требуемый пакет (модуль) и все зависимые пакеты также.



```
Windows PowerShell (x86)
is-property@1.0.2
jsonpointer@2.0.0
xtend@4.0.1
pinkie-promise@2.0.1
pinkie@2.0.4
hawk@3.1.3
boom@2.10.1
cryptiles@2.0.5
hoek@2.16.3
smtp@1.0.9
http-signature@1.1.1
assert-plus@0.2.0
jsprim@1.2.2
extsprintf@1.0.2
json-schema@0.2.2
verror@1.3.6
sshpk@1.8.1
asn1@0.2.3
assert-plus@1.0.0
dashdash@1.13.1
assert-plus@1.0.0
ecc-jsbn@0.0.4
jsbn@0.0.0 (git+https://github.com/rynomad/jsbn.git#bb522b0124f75424f89d49446c40a87111942c7b)
getpass@0.1.5
assert-plus@1.0.0
jodid25519@1.0.2
jsbn@0.1.0
tweetnacl@0.13.3
is-typedarray@1.0.0
isstream@0.1.2
json-stringify-safe@5.0.1
mime-types@2.1.10
mime-db@1.22.0
node-gui@1.4.7
oauth-sign@0.8.1
qs@6.1.0
stringstream@0.5.1
tough-cookie@2.2.2
tunnel-agent@0.4.2
npm WARN ENOENT: no such file or directory, open 'F:\node\package.json'
npm WARN node No description
npm WARN node No repository field.
npm WARN node No README data
npm WARN node No license field.
PS F:\node>
```

Имя	Дата изменения
.bin	24.04.2016 14:28
ansi-regex	24.04.2016 14:28
ansi-styles	24.04.2016 14:28
asn1	24.04.2016 14:28
assert-plus	24.04.2016 14:28
async	24.04.2016 14:28
aws4	24.04.2016 14:28
aws-sign2	24.04.2016 14:28
bl	24.04.2016 14:28
boom	24.04.2016 14:28
caseless	24.04.2016 14:28
chalk	24.04.2016 14:28
combined-stream	24.04.2016 14:28
commander	24.04.2016 14:28
core-util-is	24.04.2016 14:28
cryptiles	24.04.2016 14:28
dashdash	24.04.2016 14:28
delayed-stream	24.04.2016 14:28
ecc-jsbn	24.04.2016 14:28

10. package.json

package.json – перечень зависимостей

```
npm init
```

Файл *package.json* – хранит информацию о приложении, в частности о необходимых пакетах.

```
npm install some_pack --save
```

Команда `npm init` – позволяет создать файл *package.json*. Использование ключа `--save` добавляет устанавливаемый пакет в файл *package.json*.

```
npm install
```

Использование команды `npm install` позволяет установить все пакеты из *package.json*, если они еще не установлены.

Будет полезным

Lodash – библиотека для работы с данными



<https://lodash.com/>

**К следующему занятию будет
полезно почитать о...**

К следующему занятию...

1. Работа с DOM

(работа с разметкой документа)

2. Операторы Export/Import

(также известны как «ES Modules»)

Домашнее задание
/сделать

Домашнее задание #F.1



Ваша задача, вести список 5-ти ближайших, к пользователю, банкоматов (с **адресом**, и **расстоянием в метрах до пользователя**), отсортированных по расстоянию до пользователя.

<https://api.privatbank.ua/#p24/atm> - документация по «банкоматному» API Приватбанка.

