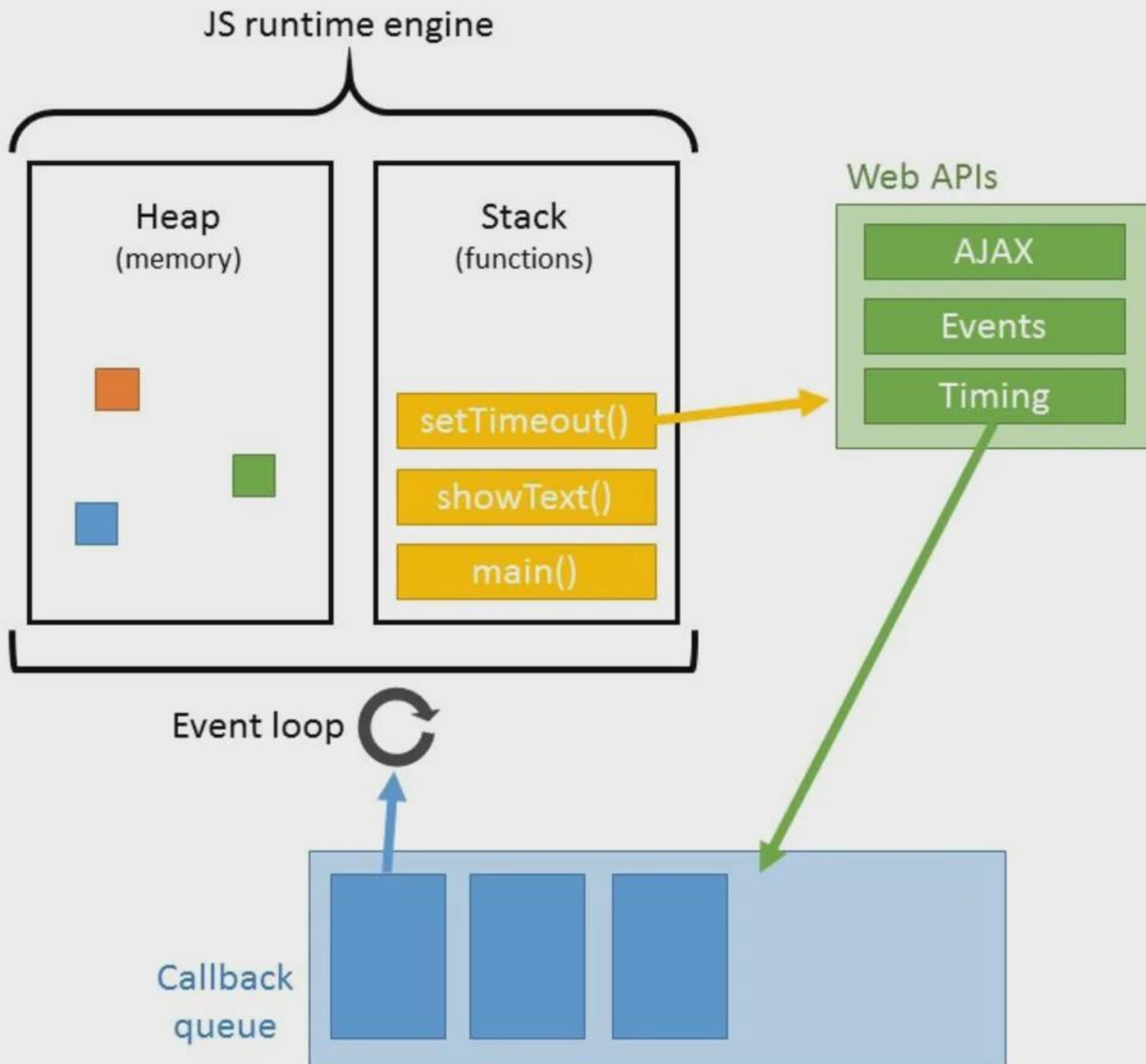


Асинхронность и AJAX



ORTDNIPRO.ORG/JS

1. Цикл событий / Event Loop

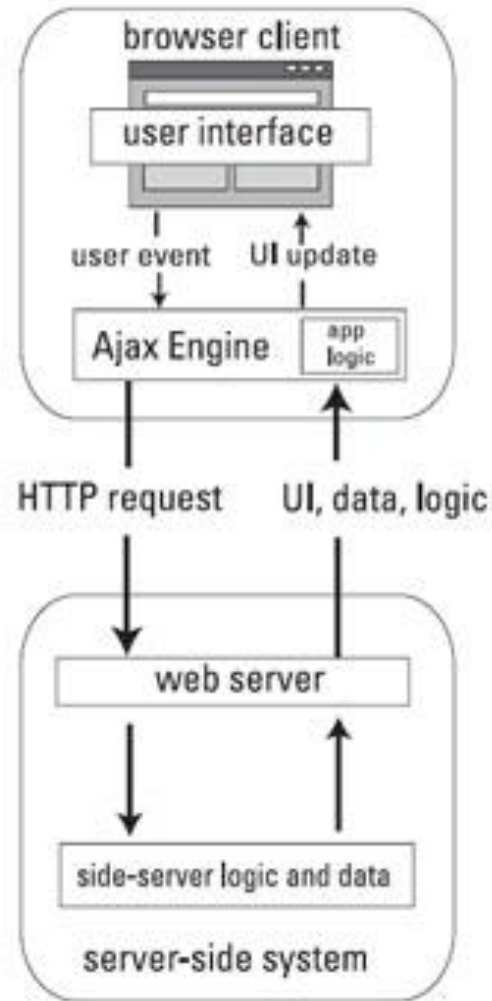
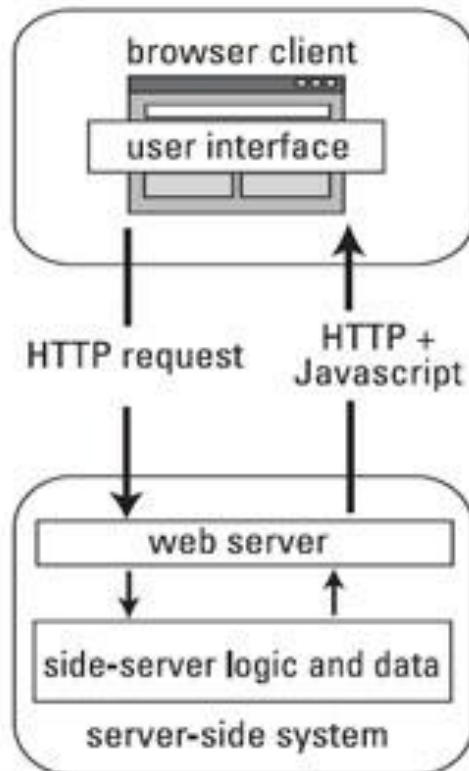


Event Loop

JavaScript однопоточный язык программирования, но тем не менее нам доступны асинхронные инструменты. Доступны они за счёт функционирования механизма **Event Loop** (или **цикла событий**, но **не стоит путать с событиями DOM**).

2. **AJAX** и функция **fetch()**

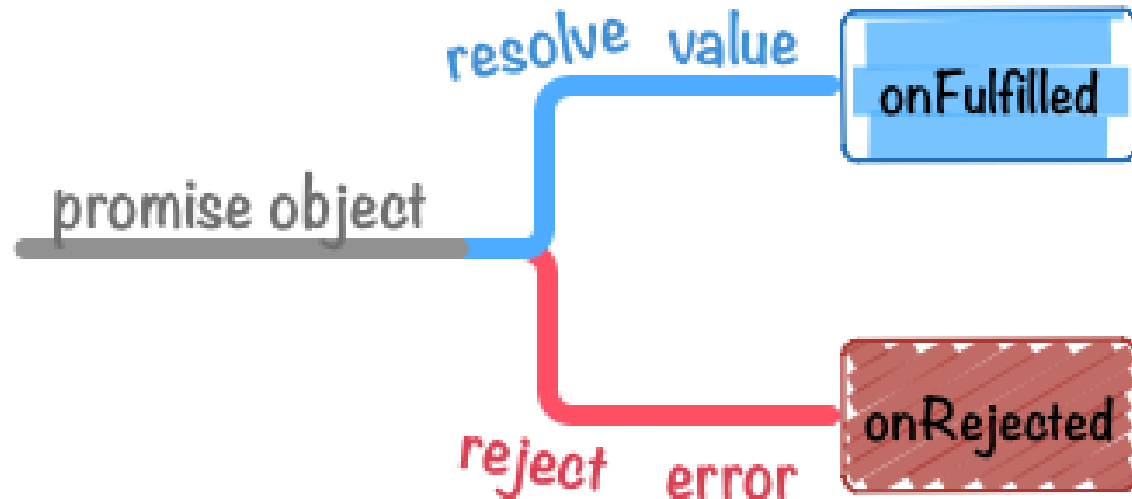
Asynchronous JavaScript And XML



Идея заложенная в **AJAX** – не перезагружая страницу, запросить (или передать) у сервера новые данные и использовать их в документе. Для выполнения запросов нам доступна функция **fetch()**.

3. Операторы `async/await`

Объект Promise



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.

async/await – упрощение кода Promise'ов

```
2  
3 let url = 'https://bank.gov.ua/NBUStatService/  
4 v1/statdirectory/exchange?json';  
5  
6
```

```
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

4. WebAPI

WebAPI построенные на обмене данными в формате JSON

Разработчикам доступно огромное количество сервисов которые предоставляющие доступ к данным в формате **JSON**. Такого рода сервисы носят название **WebAPI**.



```
[{"ccy": "USD", "base_ccy": "UAH", "buy": "28.05000", "sale": "28.25000"},  
{"ccy": "EUR", "base_ccy": "UAH", "buy": "31.95000", "sale": "32.45000"},  
{"ccy": "RUR", "base_ccy": "UAH", "buy": "0.41500", "sale": "0.43500"},  
{"ccy": "BTC", "base_ccy": "USD", "buy": "6143.7724", "sale": "6790.4852"}]
```

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

JSON (JavaScript Object Notation)

JSON - текстовый формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Основан на синтаксисе (правилах записи) массивов в **JavaScript**. Формат поддерживается практически во всех современных языках программирования.

```
[ { "name": "Jane",  "age": 23 },  
  { "name": "Max",   "age": 16 },  
  { "name": "Maria", "age": 34 },  
  { "name": "Alex",  "age": 20 },  
  { "name": "Cate",  "age": 45 } ]
```

<http://www.json.org/json-ru.html>

Для работы с форматом **JSON** у нас есть два метода: **JSON.stringify(data)** – который преобразует структуру данных в строковое представление, и метод **JSON.parse(str)** который делает обратное действие.

API Национального Банка Украины

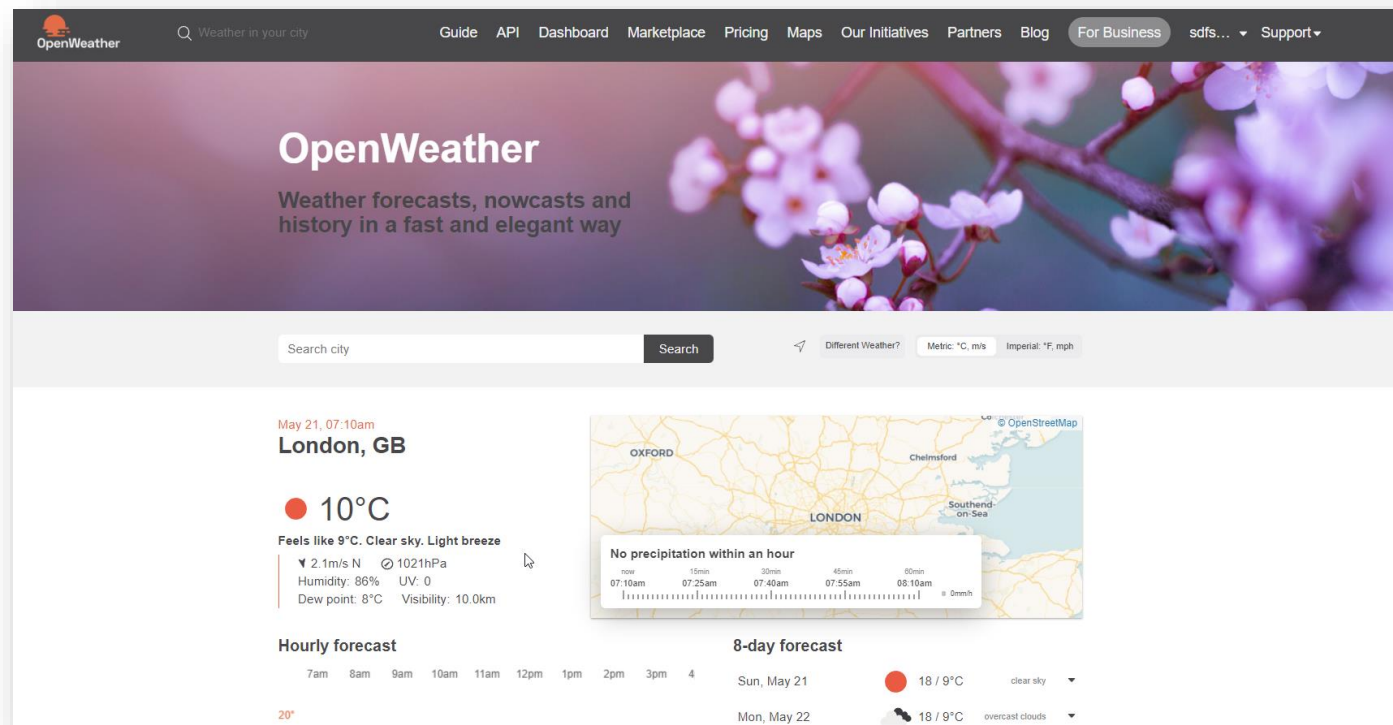


**НАЦІОНАЛЬНИЙ
БАНК УКРАЇНИ**

Валютные API, информация о финансовом рынке и банковском секторе

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

OpenWeather **API** / Прогноз погоды



<https://openweathermap.org/>

API KEY: 132eed1df60cf0306c08b2cf220716d0

5. Немного практики

Fake API + Разработка

Example Code

```
fetch('https://dummyjson.com/products/1')  
  .then(res => res.json())  
  .then(json => console.log(json))
```

Try it now 🚀

Copy Code

```
{  
  "id": 1,  
  "title": "iPhone 9",  
  "description": "An apple mobile which is nothing like apple",  
  "price": 549,  
  "discountPercentage": 12.96,  
  "rating": 4.69,  
  "stock": 94,  
  "brand": "Apple",  
  "category": "smartphones",  
  "thumbnail": "https://i.dummyjson.com/data/products/1/thumbnail.jpg",  
  "images": [  
    "https://i.dummyjson.com/data/products/1/1.jpg",  
    "https://i.dummyjson.com/data/products/1/2.jpg",  
    "https://i.dummyjson.com/data/products/1/3.jpg",  
    "https://i.dummyjson.com/data/products/1/4.jpg",  
    "https://i.dummyjson.com/data/products/1/thumbnail.jpg"  
  ]  
}
```

Yay! 🎉

<https://dummyjson.com/>

Вы можете воспользоваться шаблоном
в репозитории [./src/prices-template/](#)

На следующем занятии

На следующем занятии

**Принципы и подходы ООП в JavaScript и
всё, что с этим связано...**

Домашнее задание
/сделать

Домашнее задание #D.1

«Задача банкомата» Написать скрипт, который спрашивает у пользователя сумму, а в ответ сообщает купюры каких номинала, и в каком количестве необходимо выдать, а также суммарное количество купюр. При этом суммарное количество купюр было минимально возможным. Помните, что у нас в стране купюры номинала 1, 2, 5, 10, 20, 50, 100, 200, 500 и 1000 гривен. Количество купюр каждого номинала, которые есть в банкомате, задайте при помощи объекта (для возможности быстрого редактирования). Банкомат не может выдать **больше 40 купюр**.