PHASE 2
WEEK 2

DAY 3

ELBRUS/BOOTCAMP

План

1. SOP (same-origin policy)
2. CORS (cross-origin resource sharing)

SOP

Same-origin policy

SOP

same-origin policy

политика одинакового источника

или “правило ограничения домена”

SOP / Origin

SOP

origin

источник

SOP / Origin

Источник (origin)

Первичные данные для доступа к веб-контенту:

- протокол (схема)
- хост (домен)
- порт соединения

SOP / Origin

Примеры источников:

- http://localhost:3000/users/19/orders

  протокол хост порт

- https://shop.example.com/products/123

  протокол хост

SOP / Same-origin

SOP

same-origin

одинаковый источник

SOP / Same-origin

Источник одинаковый

… если в URL совпадают протокол, хост и порт.

- http://example.com/users/123
- http://example.com/songs/

SOP / Same-origin

Источник одинаковый

- http://example.com/users/123
- http://example.com:80/songs/ — почему?

  Сервер по умолчанию отдаёт контент по HTTP через 80-й порт.
- http://Example.Com/songs/54/lyrics/ru

  Регистр **хоста** игнорируется (но может иметь значение в других частях URL)

SOP / Same-origin policy

SOP

same-origin policy

политика одинакового источника

или “правило ограничения домена”

SOP / Same-origin policy

Политика одинакового источника

Документы из одного источника могут взаимодействовать без ограничений.

Взаимодействие документов из разных источников может быть ограничено.

SOP / Same-origin policy

Для чего это нужно?

Чтобы уменьшить количество возможных векторов атаки, изолируя сторонние документы с потенциально вредоносным кодом.

SOP / Same-origin policy

Как реализована политика SOP?

С помощью **CORS** — это часть политики одинакового источника.

Отвечает за ограничения документов из разных источников.

CORS

CORS

cross-origin resource sharing

совместное использование ресурсов между разными источниками

CORS / Cross-origin

CORS

cross-origin

между разными источниками

CORS / Cross-origin

Источники разные

… если URL отличаются протоколом, хостом или портом.

CORS / Cross-origin

Источники разные

- http://example.com/users
- https://example.com/users — в чём отличие?

  Разные протоколы: http !== https
- http://www.example.com/users

  Разные хосты: example.com !== www.example.com

CORS / Cross-origin

Источники разные:

- http://example.com/users
- http://example.com:3000/users

  Разные порты: 3000 !== 80 (выставлен по умолчанию)

CORS / Resource sharing

CORS

resource sharing

совместное использование ресурсов

CORS / Resource sharing

Как использовать ресурсы из разных источников?

CORS / Resource sharing

CORS-ограничения

… работают только в браузерах.

Браузер проверяет заголовок Access-Control-Allow-Origin в ответе сервера на наличие домена, с которого был сделан запрос.

CORS / Resource sharing

Как разрешить использовать ресурсы?

CORS / Resource sharing

Вариант 1:

Написать свой middleware, устанавливающий заголовки.

```
function allowAccess(req, res, next) {
 res.header('Access-Control-Allow-Origin', '*')
 res.header('Access-Control-Allow-Methods', ['GET', 'HEAD', 'PUT',
'PATCH', 'POST', 'DELETE'])
 next();
}
```

CORS / Resource sharing

Вариант 2:

Установка: `npm i cors`

Использовать пакет `cors` как middleware.

```
const cors = require('cors');

app.use(cors());
```

CORS / Resource sharing

```
const cors = require('cors');

const corsOptions = {
 origin: ['http://localhost:3000', 'http://localhost:4000'],
 optionsSuccessStatus: 200,
};

app.use(cors(corsOptions));
```

CORS / Resource sharing / credentials

По умолчанию fetch-запросы к сторонним источникам не включают заголовки с куками.

Их нужно явно включить через настройку “credentials”.

CORS / Resource sharing / credentials

```
fetch('your-url-here', {
   // Прилагать куки к запросам на сторонний источник
    credentials: 'include',
})
```

CORS / Resource sharing / credentials

На сервере необходимо явно указать, что разрешено принимать куки от сторонних источников.

Это делается через заголовок ‘Access-Control-Allow-Credentials’ — вручную или с помощью настройки ‘credentials’ в конфигурации модуля cors.

CORS / Resource sharing / credentials

```
const cors = require('cors');

const corsOptions = {
 origin: ['http://localhost:3000', 'http://localhost:4000'],
 optionsSuccessStatus: 200,
 credentials: true, // принимать куки от сторонних источников
};

app.use(cors(corsOptions));
```