МШП 20.35 УНИВЕРСИТЕТ КОД БУДУЩЕГО Минцифры России НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ ЦИФРОВАЯ ЭКОНОМИКА

Fullstack-разработка

NodeJS
Введение

МШП

Цели урока:

1. Познакомимся с новой технологией NodeJS
2. Запустим свой первый сервер

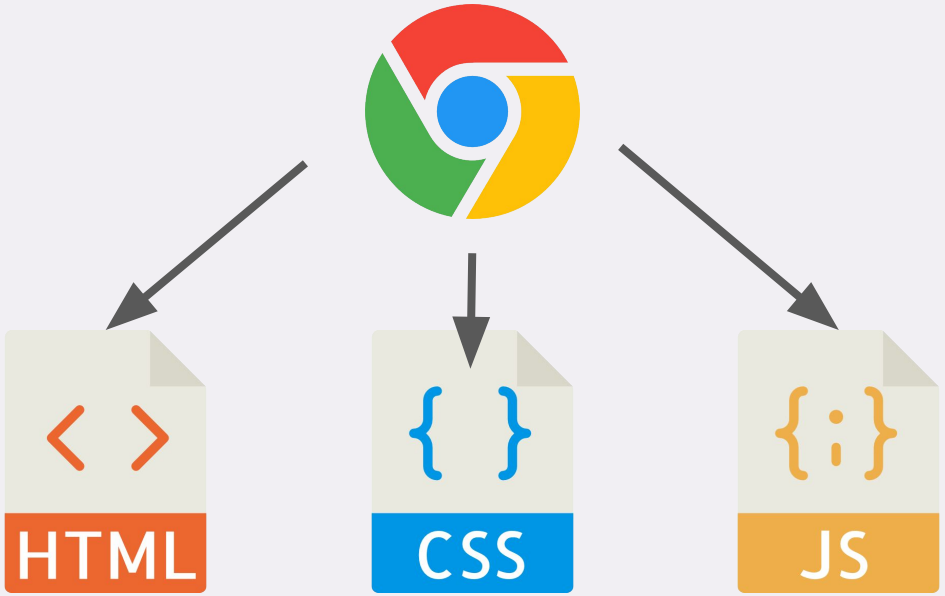
МШП

Теория

NodeJS

МШП

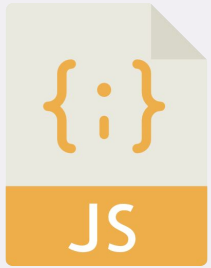
До этого момента мы писали **код, который выполняется в браузере**



МШП

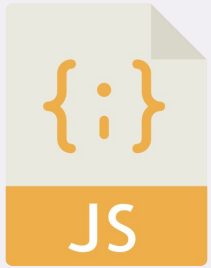
В браузер встроен **движок**,
который превращает
JS-код в машинный код.

Для Google Chrome это движок V8.



МШП

NodeJS – это среда выполнения JS-кода, в основе которой лежит движок V8.



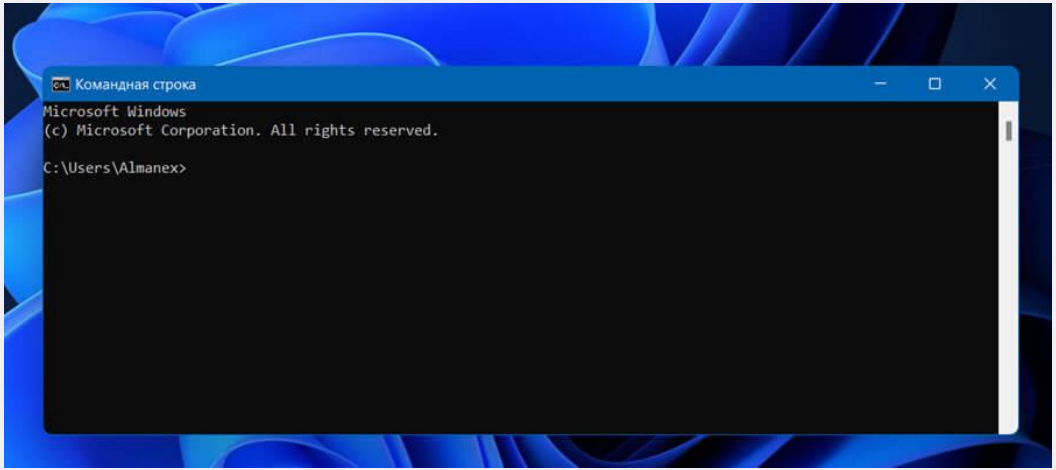
МШП

Возможности NodeJS

- Файловая система 
- Работа с БД 
- Работа с сетью 
- Библиотеки
- Терминал 
- Работа с DOM 

МШП

Терминал



Для запуска JS-кода через NodeJS нам понадобится терминал.

В нём можно запускать **команды.**

МШП

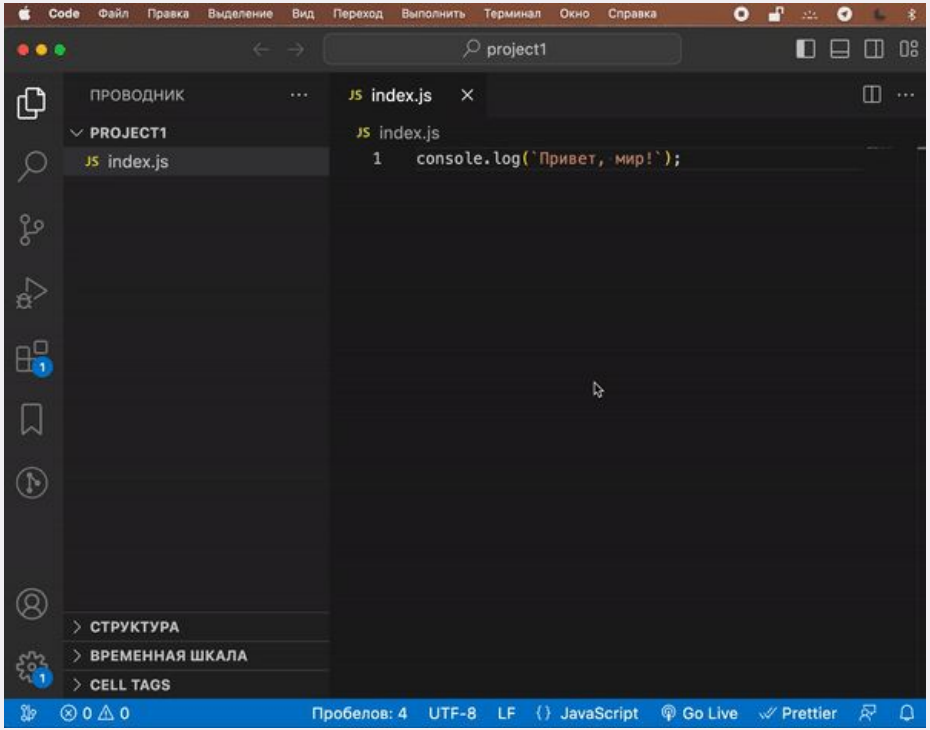
Запуск скрипта

```
node имя_файла
```

1. Наберите в терминале команду **node**
2. Через пробел укажите имя файла с кодом на JS
3. Нажмите Enter

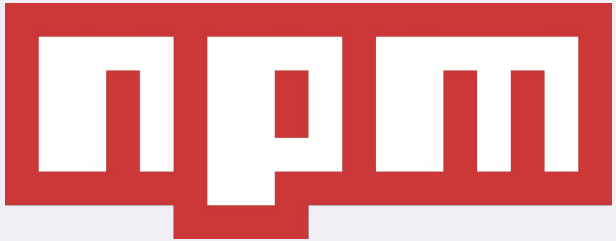
МШП

Работа в терминале



МШП

NPM – это пакетный менеджер для NodeJS. С помощью него мы будем устанавливать **библиотеки**, или, как их часто называют, **пакеты.**



МШП

Пакет Express

В нашем проекте понадобится пакет **express**.

С помощью него мы сделаем из обычного скрипта на JS целый сервер.

Express простой и очень, очень популярный!

МШП

Инициализация проекта

Если в проекте нужен хотя бы один пакет, используй эту команду.

В результате будет создан служебный файл **package.json** с информацией о всех установленных пакетах.

```
npm init -y
```

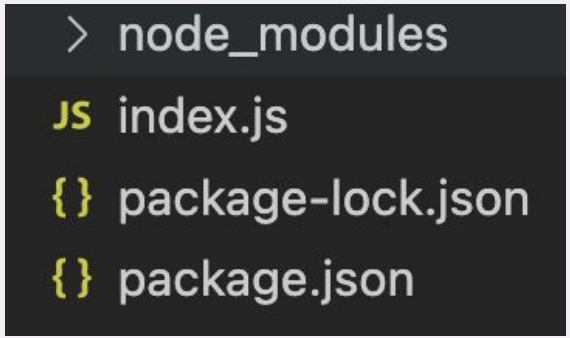
МШП

Установка пакета

Выполни в терминале VSCode команду:

```
npm install express
```

Появится много новых служебных файлов.



МШП

Практика

1. Установка NodeJS
2. Первый скрипт
3. Установка express

МШП

Теория

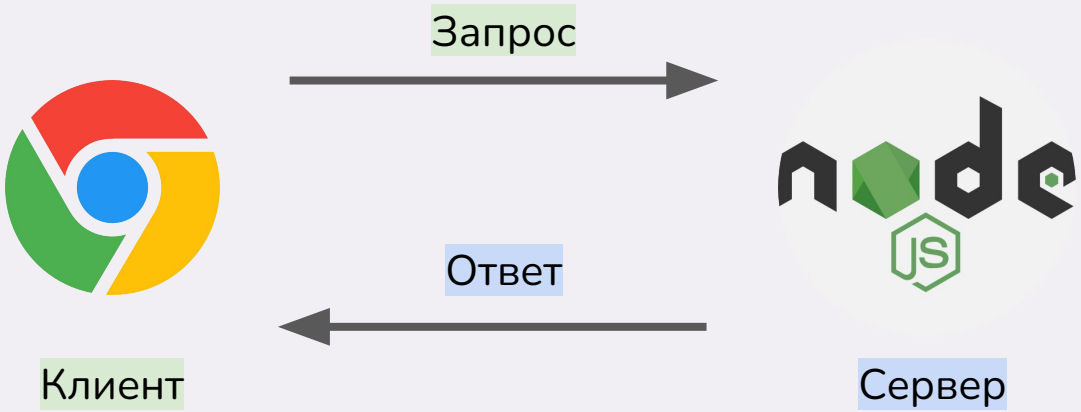
Клиент-серверная архитектура

Когда пользователь заходит в браузер и вбивает адрес сайта, начинается процесс **клиент-серверного взаимодействия**

Клиент-браузер отправляет запрос на получение данных на сервер.

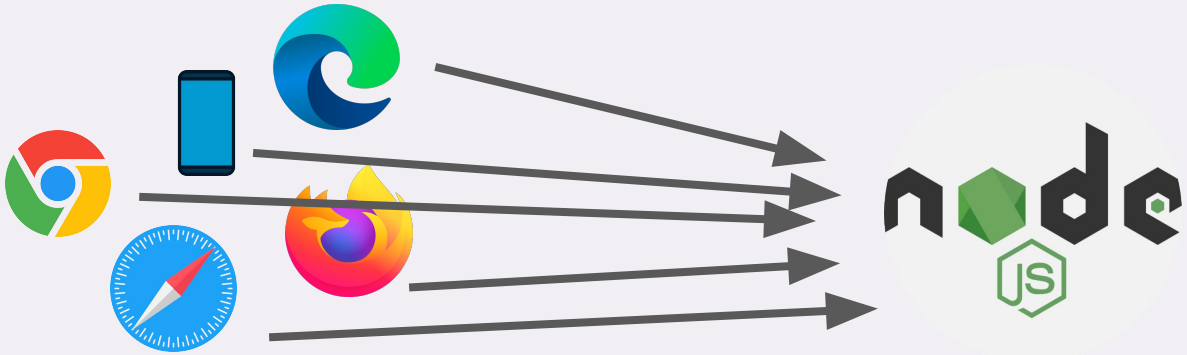
МШП

Сервер в ответ за запрос от клиента отправляет данные.



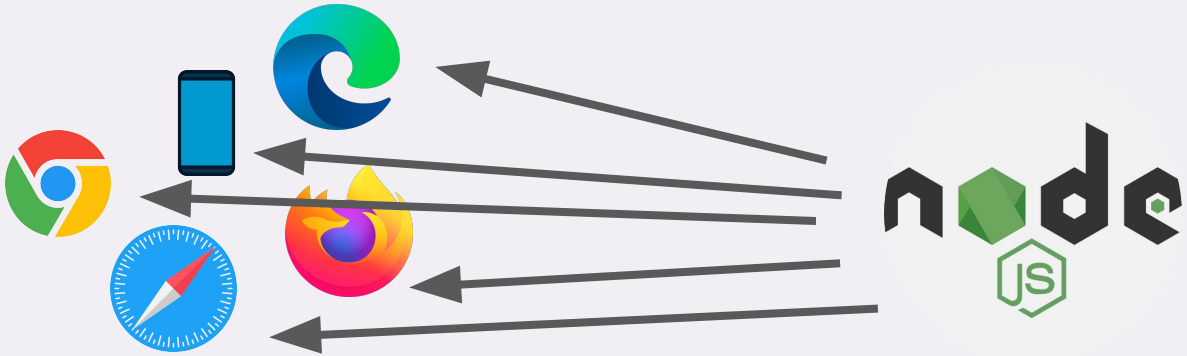
МШП

В реальности на сервер приходят тысячи запросов в секунду от разных клиентов.



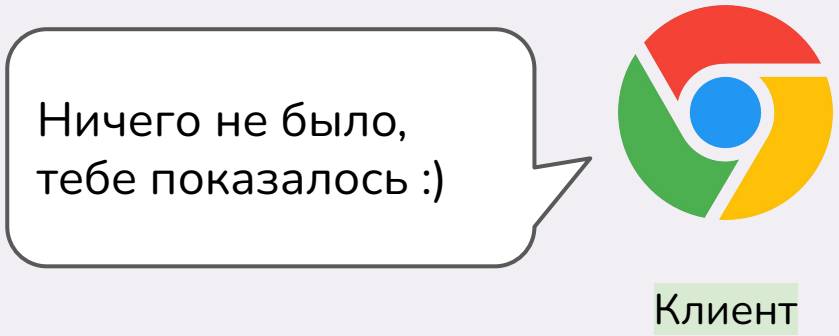
МШП

NodeJS как раз отлично справляется с высокими нагрузками и умеет вовремя отвечать клиентам на их запросы.



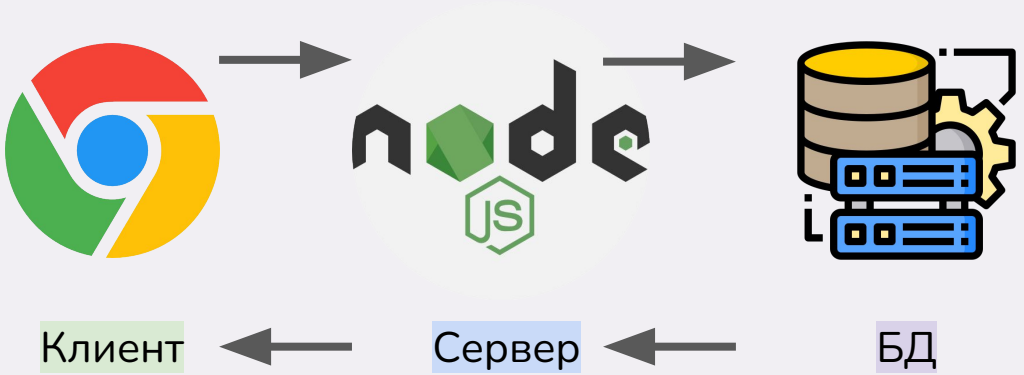
На клиенте мы можем хранить данные в переменных.

Но что, если обновить страницу?
Скрипт запустится заново,
а данные пропадут.



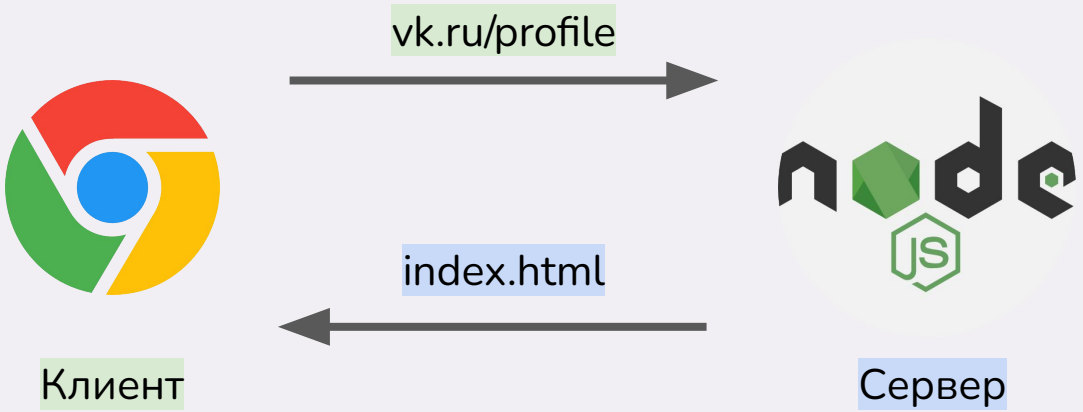
МШП

С помощью **NodeJS** мы будем работать с базой данных. Это позволит **безопасно хранить большое количество данных.**



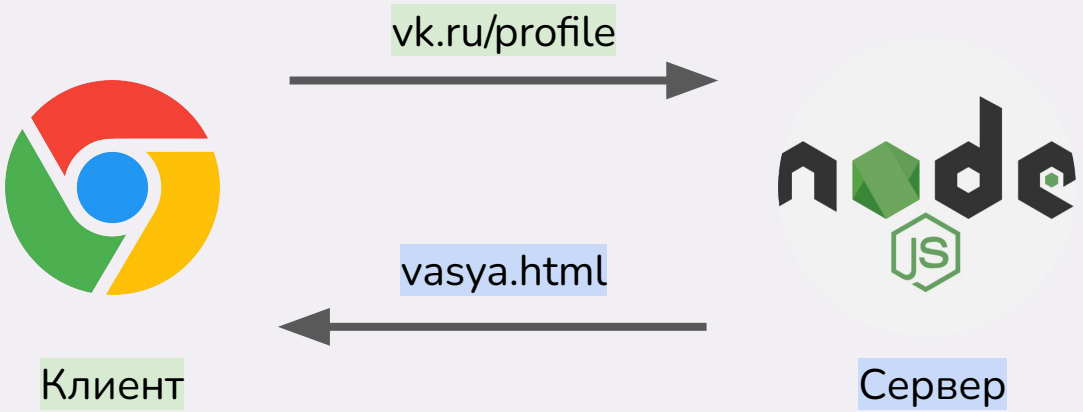
МШП

С помощью HTML и CSS мы создавали **статичные** страницы, которые выглядели для всех одинаково.



С помощью сервера мы сможем подставить в HTML-страницу данные пользователя.

Так что одна и та же страница будет выглядеть по-разному для всех.

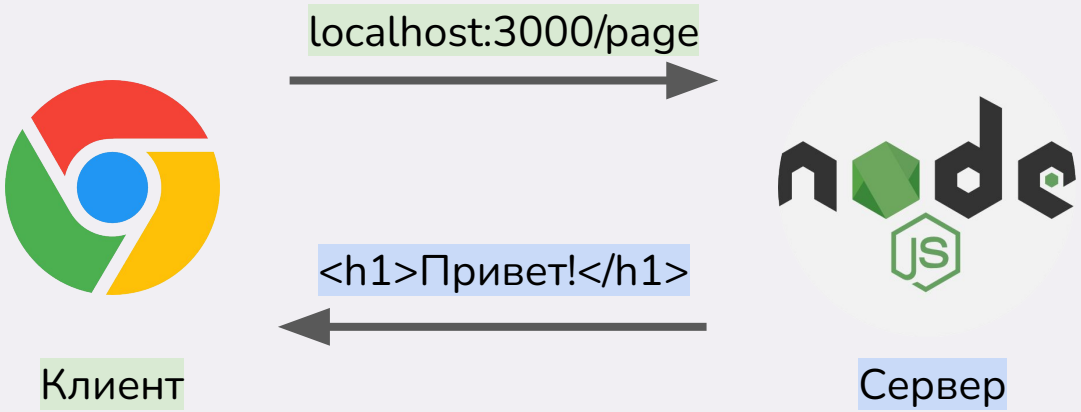


Задачи сервера

- **Быстро** отвечать за запросы клиентов
- **Безопасно** хранить данные в базе данных
- **Создавать HTML-страницы** с данными

МШП

Сегодня мы решим простую задачу: научим сервер принимать наши запросы и в ответ отдавать HTML-код.



МШП

Ранее мы установили npm-пакет **Express**. На нём очень-очень просто создавать серверы любой сложности!

Давайте перейдём в редактор кода и напишем наш первый сервер.

МШП

Практика

Запуск сервера

МШП

Теория

Express

МШП

Код сервера

```
let express = require(`express`);
let app = express();
let port = 3000;
// Роут
app.get(`/`, function (req, res) {
    res.send(`Привет, мир!`);
});
// Запуск сервера
app.listen(port, function () {
    console.log(`Сервер запущен:
                http://localhost:${port}
    `);
});
```

МШП

Подключение

```
let express = require(`express`);
let app = express();
```

Помните, мы установили npm-пакет express?

С помощью метода **require** мы подключаем пакет express в JS код.

Затем создаём сервер в переменной **app**.

МШП

Порт

```
let port = 3000;
app.listen(port, function () {
    // http://localhost:3000
    console.log(`Сервер запущен`);
});
```

Сервер запускается локально,
по умолчанию его доменное имя **localhost.**

Серверов может быть **несколько**, поэтому необходимо указать **свободный порт** – номер, который сервер займёт в домене.

МШП

Роуты

```
app.get(`/`, function (req, res) {
    res.send(`Привет, мир!`);
});
```

Роут – это путь, по которому клиент приходит со своим запросом.

Слеш обозначает **главный роут**. Он сработает, если перейти на http://localhost:3000.

В этот момент выполнится **функция-колбек,** которая отправит ответ клиенту через функцию **res.send.**

МШП

Роуты

http://localhost:3000/page1

```
app.get(`/page1`, function (req, res) {
    res.send(`<p>Страница 1</p>`);
});
```

http://localhost:3000/page2

```
app.get(`/page2`, function (req, res) {
    res.send(`<h1>Страница 2</h1>`);
});
```

Обычно в проектах много **роутов**.

Каждый из них обозначает
отдельную страницу проекта.

МШП

Роуты

```
app.get(`/page1`, function (req, res) {
    res.send(`Привет, мир!`);
});

app.get(`/page2`, function (req, res) {
    res.send(`<h1>Привет, мир!</h1>`);
});
```

Ответ сервера – это HTML-строка.

Именно её пользователь увидит в браузере, когда зайдёт на **роут**.

МШП

Практика

Счётчик просмотров

МШП

Цели урока:

- ✓ Познакомились с новой технологией NodeJS
- ✓ Запустили свой первый сервер

МШП

Домашнее задание