

Яндекс

Яндекс

Асинхронный код в js

Руслан Муфтиев

В предыдущей серии...



На прошлой лекции "Введение в node.js"

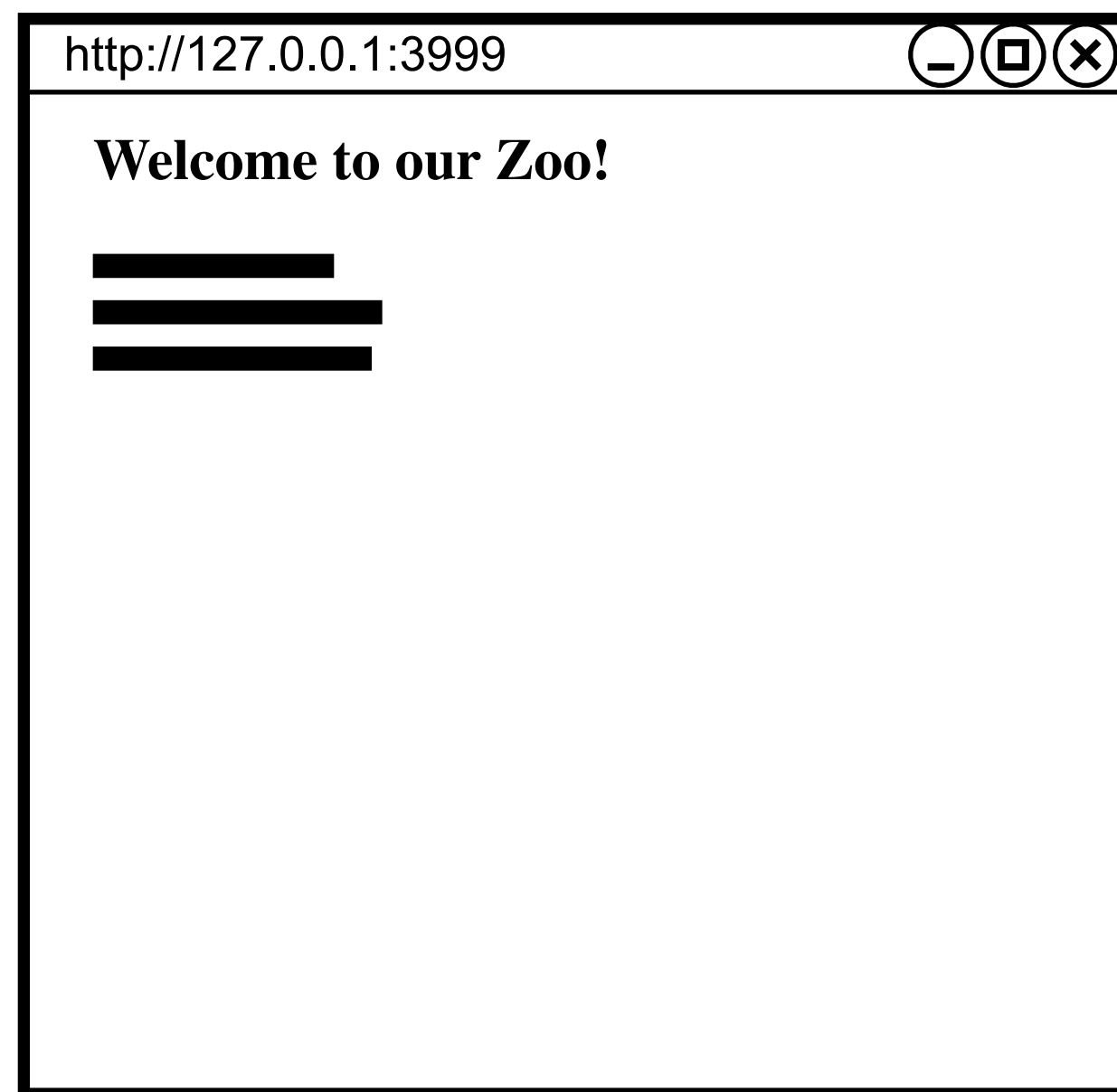
- › Что такое node.js и откуда она появилась
- › Модули в node.js (**module.exports + require**)
- › Работа с файловой системой (модуль **fs**)
- › Немного ES6 (**const, let, arrow functions**, деструктуризация)
- › Обработка ошибок **throw new Error('error message')**
- › Написали простенький сервер (модуль **http**)

Наше приложение

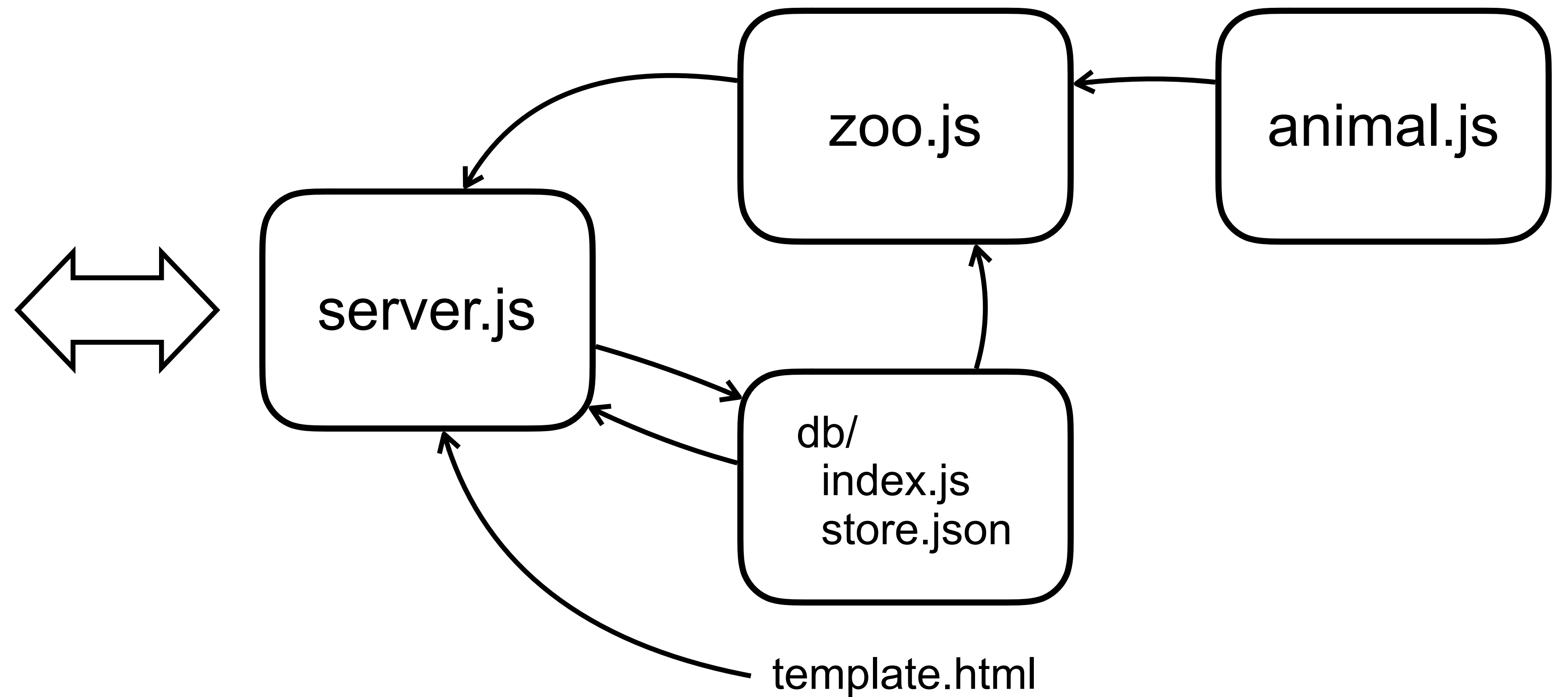


Схема приложения

Browser



Node.js



Пройдемся по коду




Событийное программирование

```
fs.readFile('./store.json', function(error, data) {  
  if (error) {  
    console.error('error message');  
    response.statusCode = 500;  
    response.end('Internal server error');  
  } else {  
    // do your code here  
    response.end('your content');  
  }  
});
```


Событийное программирование

```
server.on('request', function(request, response) {  
  fs.readFile('./store.json', function(error, data) {  
    if (error) {  
      console.error('error message');  
      response.statusCode = 500;  
      response.end('Internal server error');  
      .....  
    } else {  
      // do your code here  
      response.end('your content');  
      .....  
    }  
  });  
});  
});
```



Можем обрабатывать
запросы одновременно?

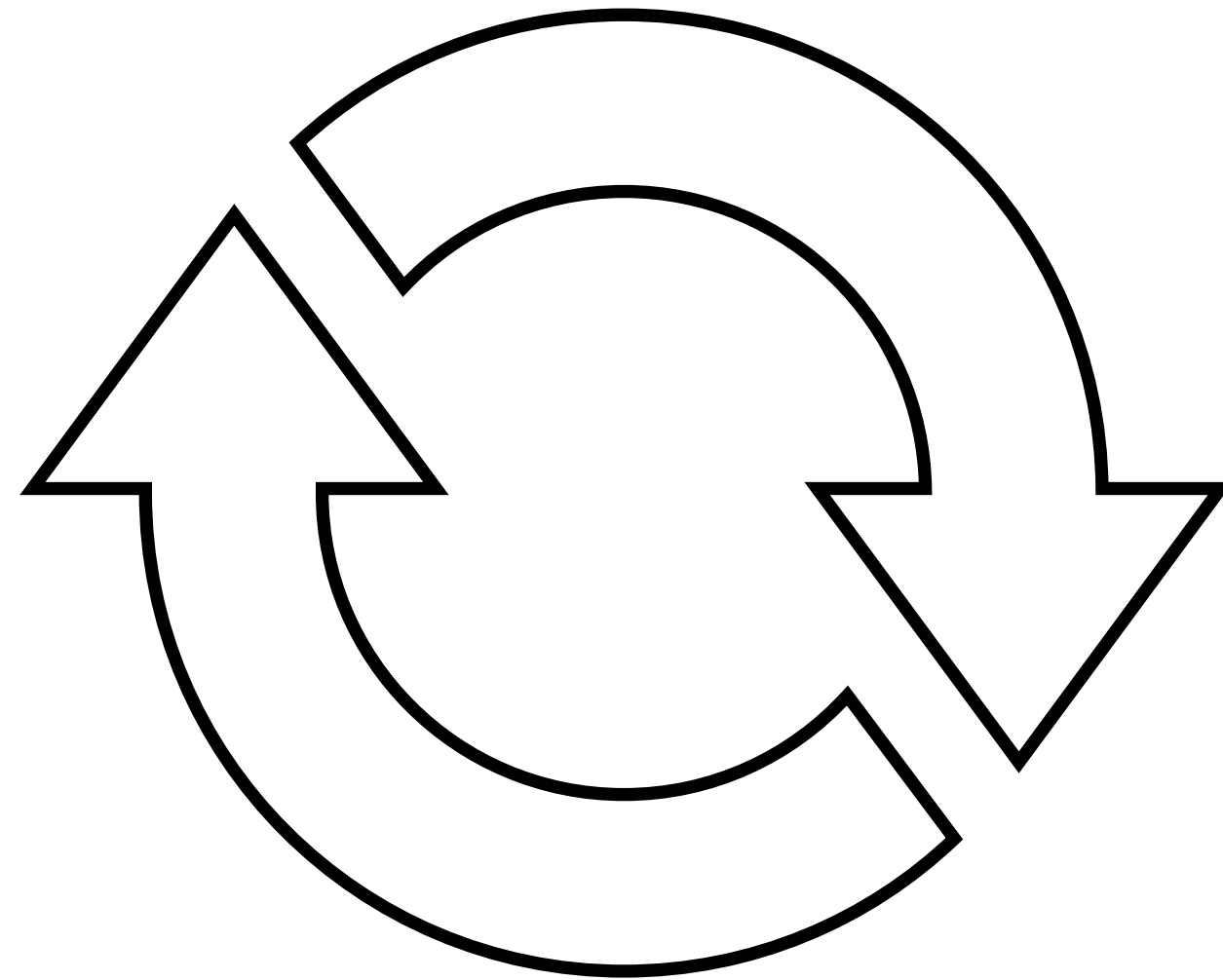
Событийный цикл



Событийный цикл node.js

```
while (есть обработчики) {
```

JavaScript (V8)



I/O, timers

```
}
```

```
const server = http.createServer();
.....

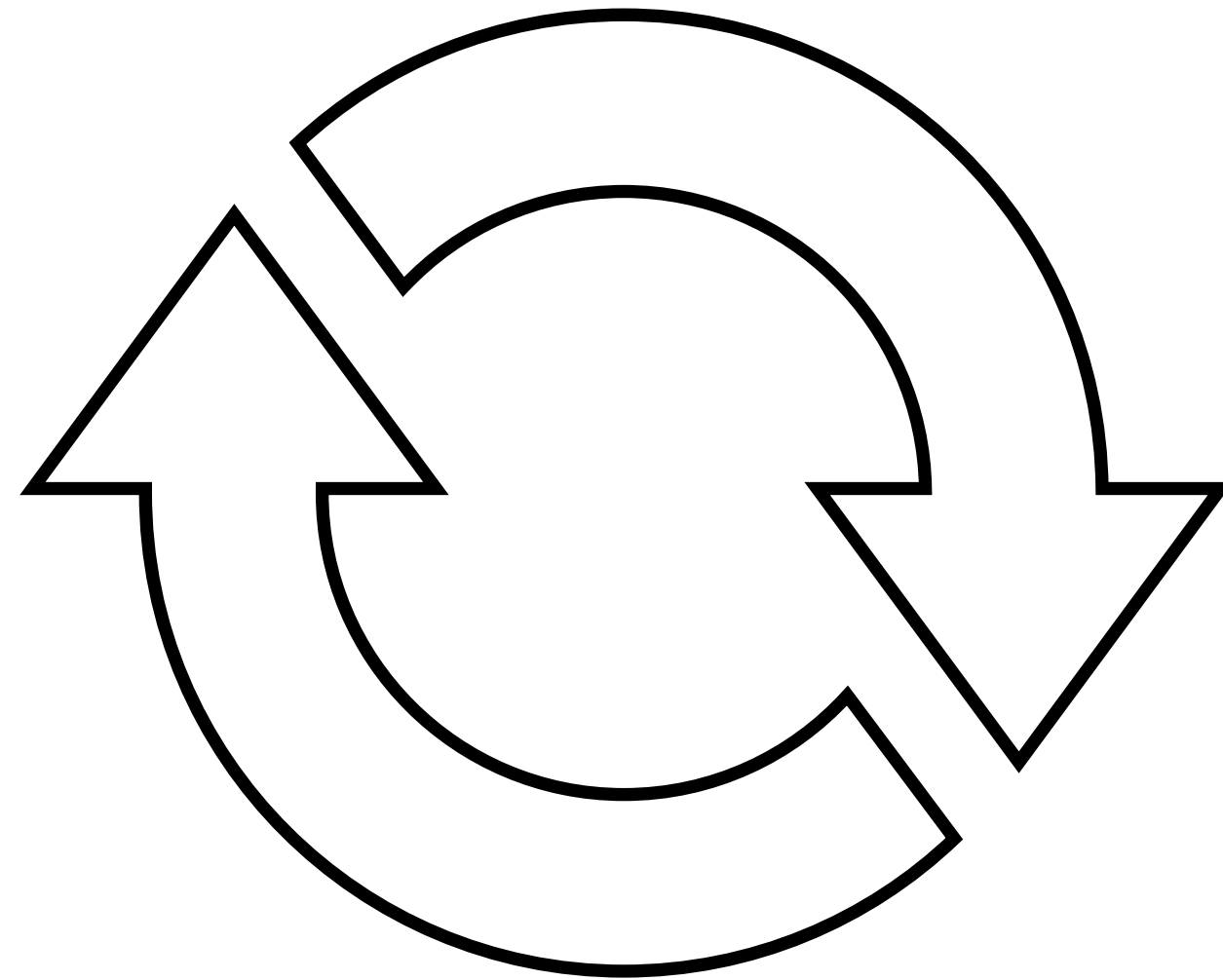
server.on('request', function(request, response) {
.....
  fs.readFile('./store.json', function(error, data) {
    if (error) {
      console.error('error message');
      response.statusCode = 500;
      response.end('Internal server error');
    } else {
      // do your code here
      response.end('your content');
    }
  });
});

server.listen(3999, '127.0.0.1');
.....
```

Событийный цикл node.js (Event Loop)

```
while (есть обработчики) {
```

JavaScript (V8)



I/O, timers

```
}
```

```
const server = http.createServer();

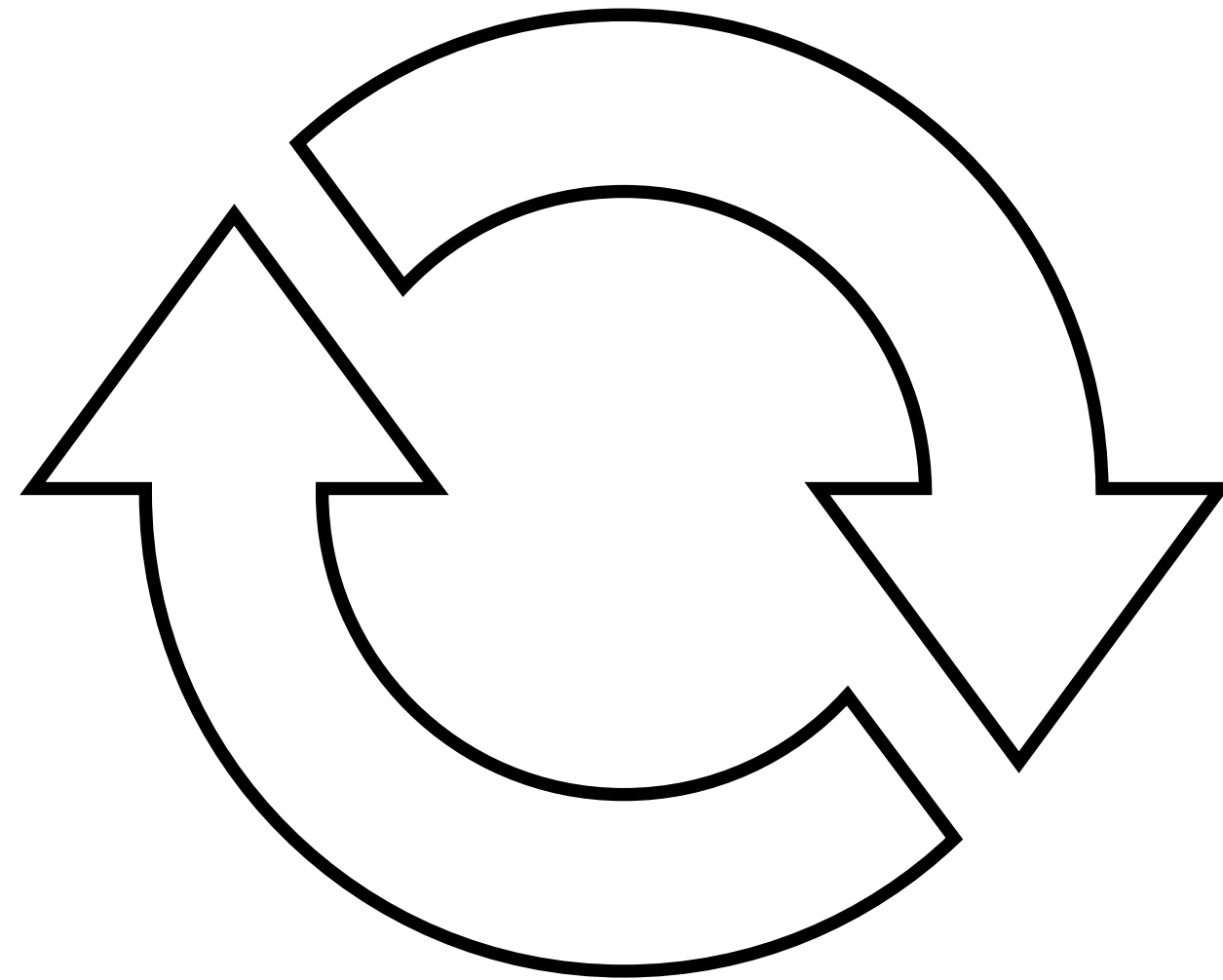
server.on('request', function(request, response) {
  fs.readFile('./store.json', function(error, data) {
    if (error) {
      console.error('error message');
      response.statusCode = 500;
      response.end('Internal server error');
    } else {
      // do your code here
      response.end('your content');
    }
  });
});

server.listen(3999, '127.0.0.1');
```

Событийный цикл node.js

```
while (есть обработчики) {
```

JavaScript (V8)



I/O, timers

```
}
```



```
const server = http.createServer();

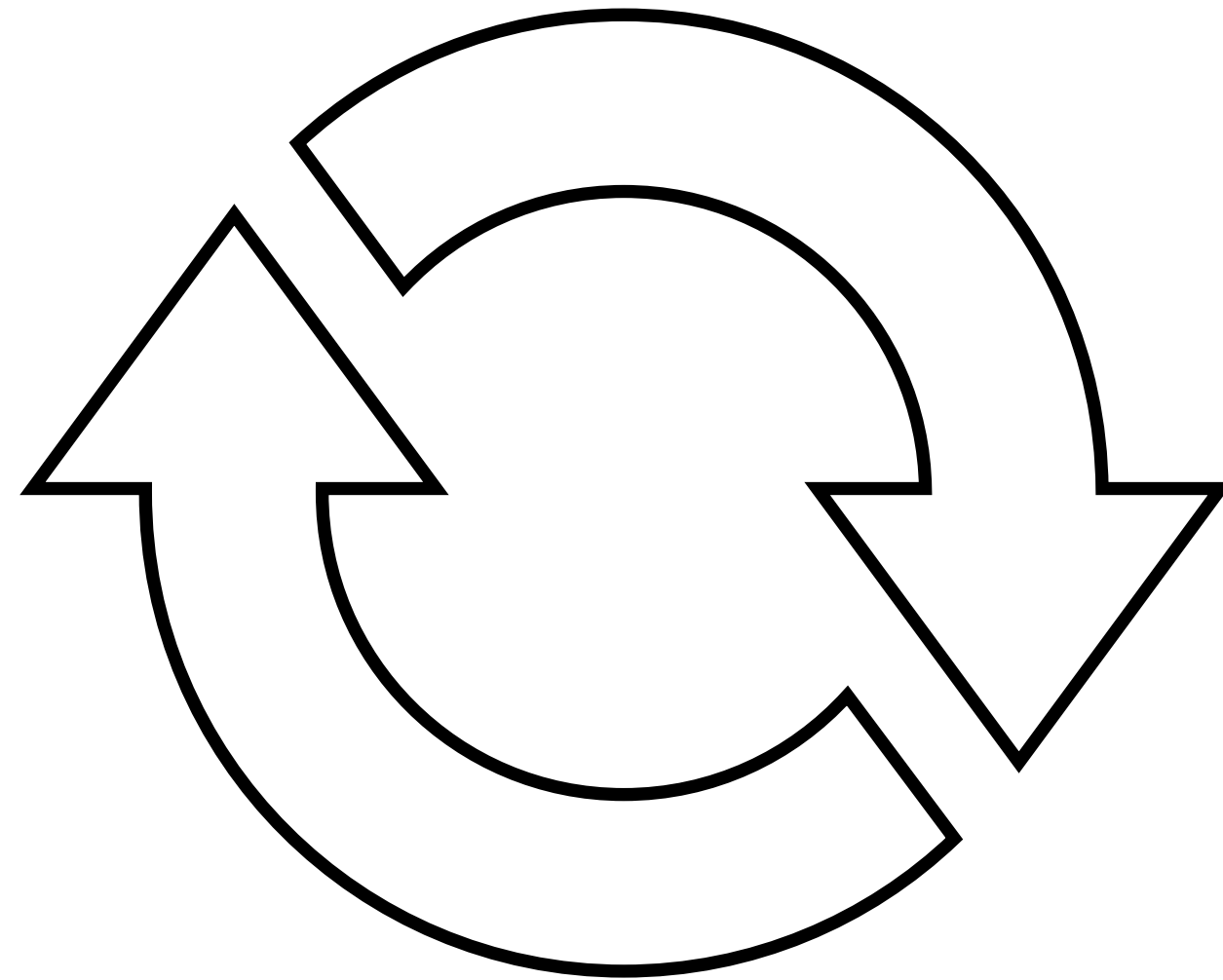
server.on('request', function(request, response) {
  fs.readFile('./store.json', function(error, data) {
    if (error) {
      console.error('error message');
      response.statusCode = 500;
      response.end('Internal server error');
    } else {
      // do your code here
      response.end('your content');
    }
  });
});

server.listen(3999, '127.0.0.1');
```

Событийный цикл node.js

```
while (есть обработчики) {
```

Javascript (V8)



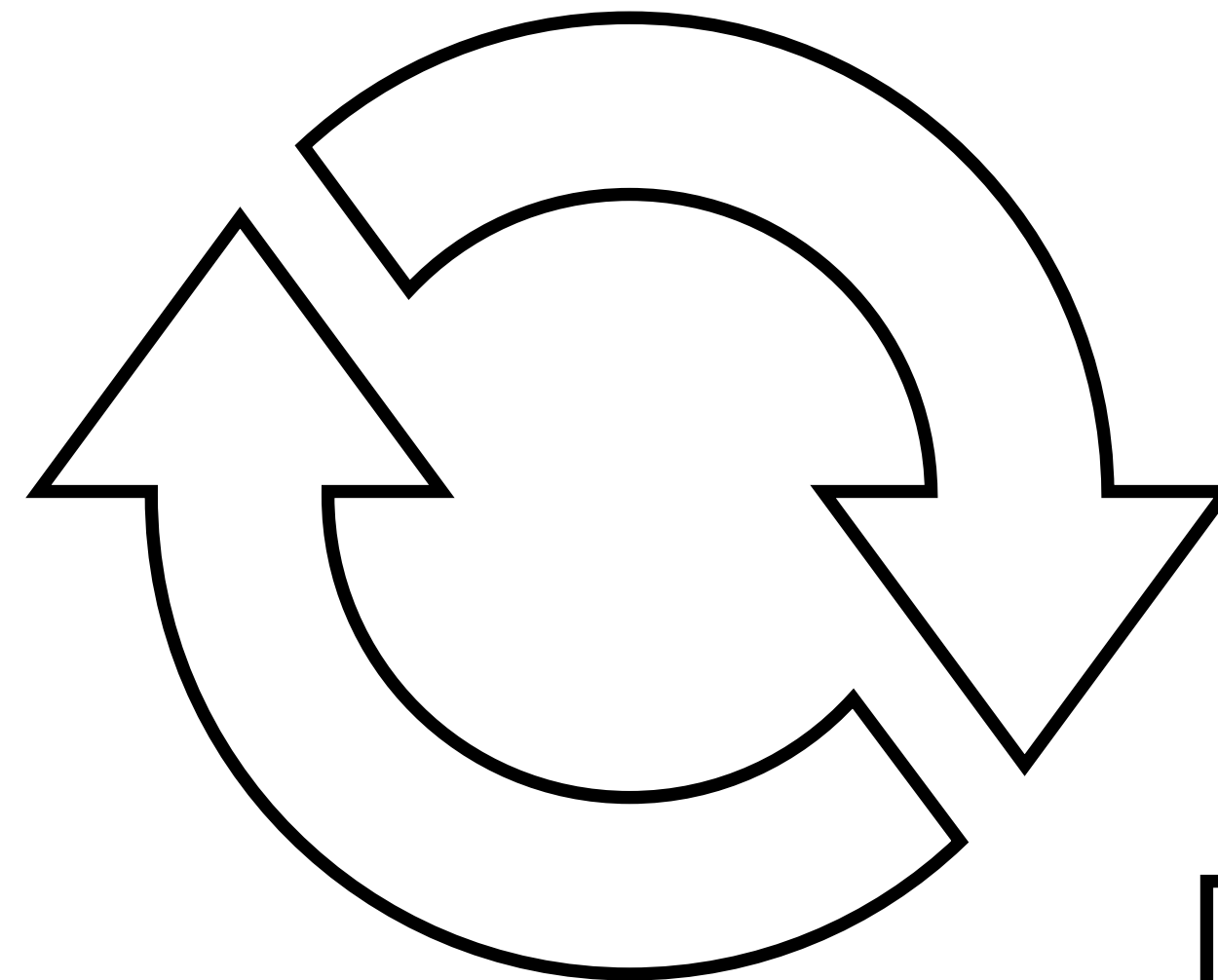
I/O, timers

```
}
```

Событийный цикл node.js

```
while (есть обработчики) {
```

Javascript (V8)



I/O, timers



```
}
```



**Избегайте тяжелых
синхронных операций!**

Попишем код



Promise



Пример "callback hell"

```
server.on('request', function(request, response) {  
  async(arg1, arg2, function cb(err, data) {  
    // some code goes here  
    async2(arg1, arg2, function(err2, data2) {  
      // some other code goes here  
      async3(arg1, arg2, arg3, function(err3, data3) {  
        // some final code goes here  
        response.end('ok');  
      });  
    });  
  });  
});
```

Реализация на Promise

```
server.on('request', function(request, response) {  
  async(arg1, arg2)  
  .then(data) {  
    // some code goes here  
    return async2(arg1, arg2);  
  }).then(data2) {  
    // some other code goes here  
    return async3(arg1, arg2, arg3);  
  }).then(err, data) {  
    // some final code goes here  
    response.end('ok');  
  });  
});
```


Попишем код



Закрепим



ОСНОВНЫЕ ТЕЗИСЫ

- › Научились писать асинхронный код с callback-функциями
- › Разобрали как работает событийно-ориентированный код
- › Событийный цикл (Event Loop)
- › Многопоточные ОС, но однопоточный js
- › Разобрали проблемы с callback-функциями
- › Промисы **new Promise(resolve, reject) => { resolve('ok') };**
- › Переписали приложение на промисы

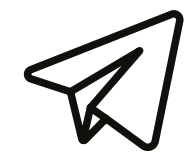
Вопросы

Руслан Муфтиев

Руководитель группы разработчиков
поисковых интерфейсов в Симферополе



muftik@yandex-team.ru



[@muftiev](https://t.me/muftiev)