

Асинхронный код

Очередь событий. Таймеры. Callback. Promise

Синхронный код

```
function grist() {  
    return 'Перемолоть кофейные зерна';  
}  
  
function addWater() {  
    return 'Добавить немного воды';  
}  
  
function toStove() {  
    return 'Поставить на плиту';  
}
```

Синхронный код

```
grist();  
addWater();  
toStove();
```

Синхронный код

```
grist();      // 'Перемолоть кофейные зерна'  
addWater();  
toStove();
```

Синхронный код

```
grist();      // 'Перемолоть кофейные зерна'  
addWater();  // 'Добавить немного воды'  
toStove();
```

Синхронный код

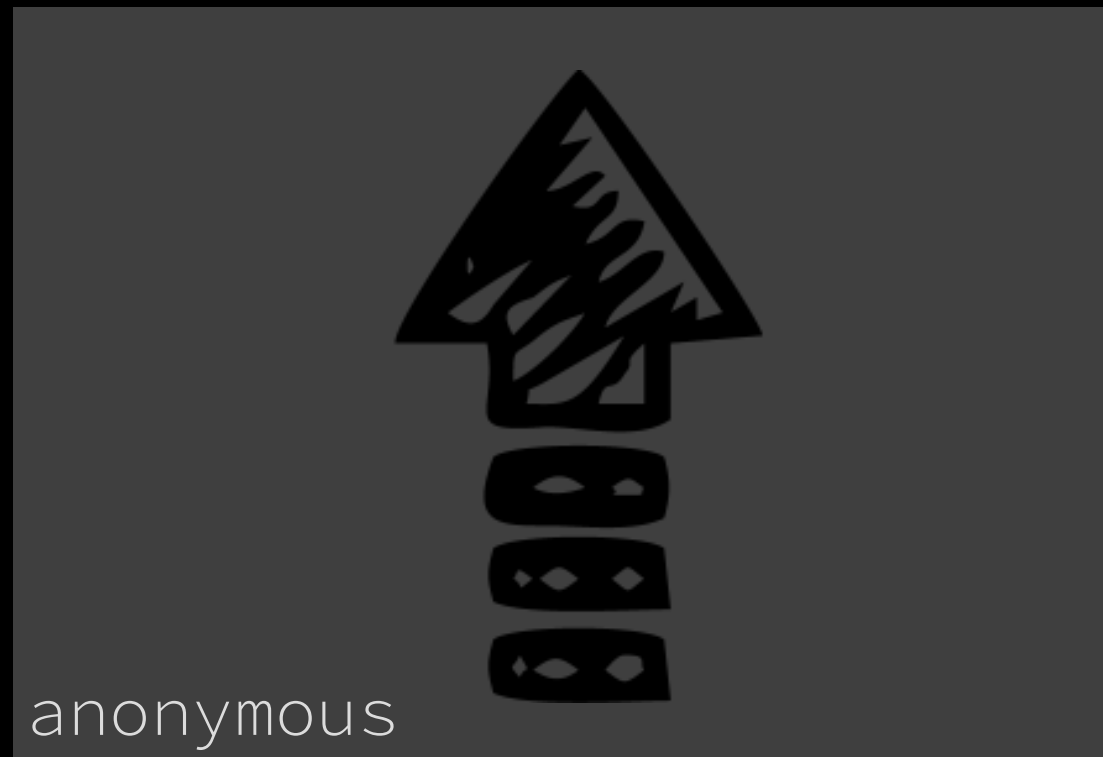
```
grist();      // 'Перемолоть кофейные зерна'  
addWater();   // 'Добавить немного воды'  
toStove();    // 'Поставить на плиту'
```

Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:

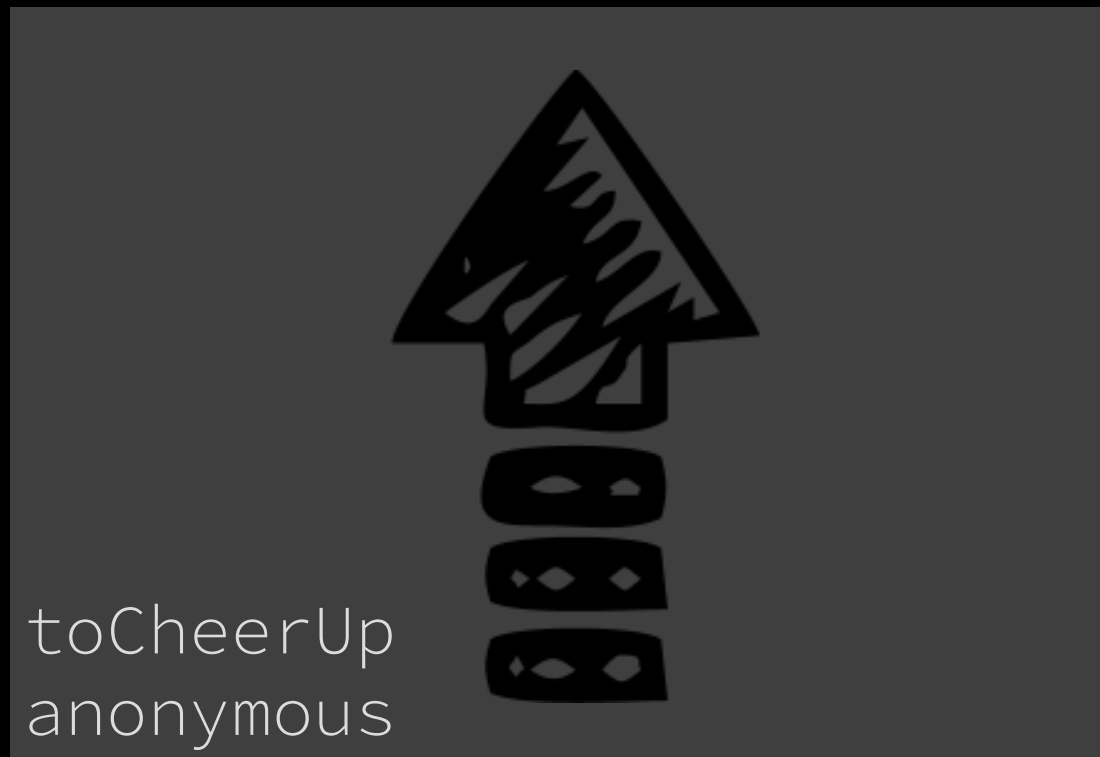


Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:

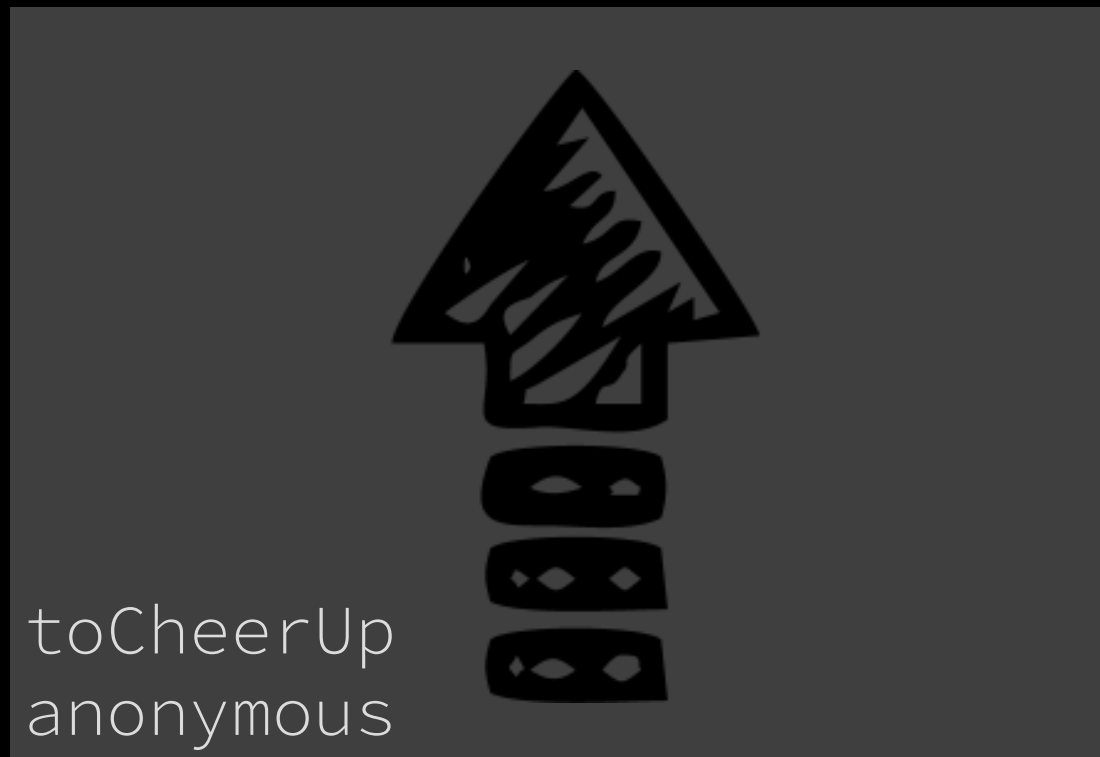


Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:

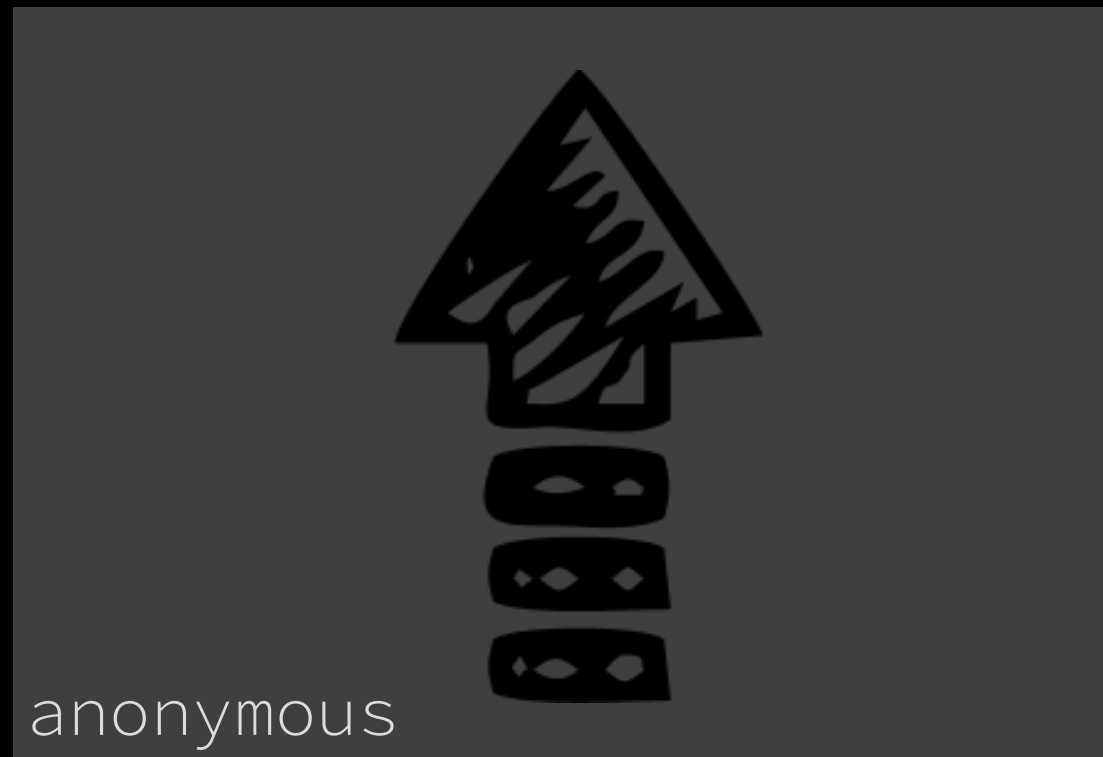


Стек вызовов

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Стек вызовов

Код:

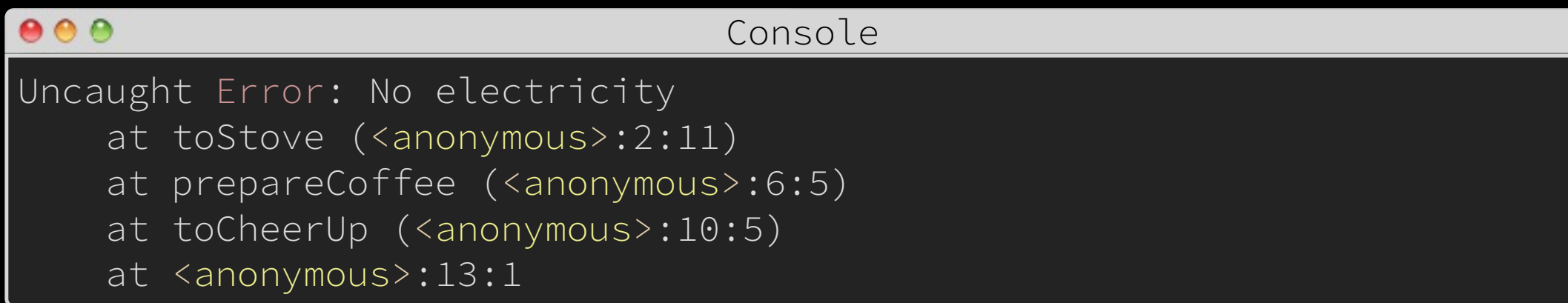
```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Стек вызовов

```
function toStove() {  
    throw new Error('No electricity');  
}  
function prepareCoffee() {  
    toStove();  
}  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```



Console

```
Uncaught Error: No electricity  
    at toStove (<anonymous>:2:11)  
    at prepareCoffee (<anonymous>:6:5)  
    at toCheerUp (<anonymous>:10:5)  
    at <anonymous>:13:1
```

Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Очередь событий:

anonymous



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}
```

```
function toCheerUp() {  
    prepareCoffee();  
}
```

```
toCheerUp();
```

Стек вызовов:



anonymous

Очередь событий:



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:

toCheerUp
anonymous



Очередь событий:



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Очередь событий:



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Очередь событий:



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Очередь событий:



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:

toCheerUp
anonymous



Очередь событий:

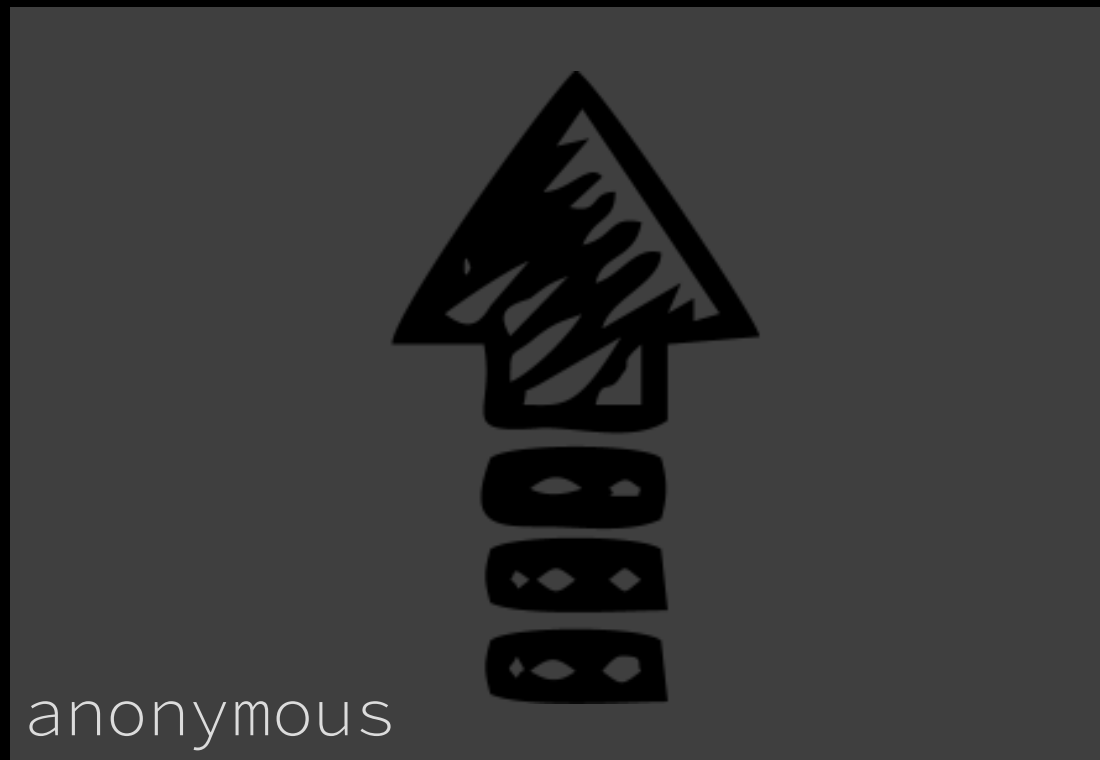


Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Очередь событий:



Очередь событий

Код:

```
function prepareCoffee() {  
    toStove();  
}  
  
function toCheerUp() {  
    prepareCoffee();  
}  
  
toCheerUp();
```

Стек вызовов:



Очередь событий:





Системный таймер

Сальвадор Дали «Постоянство времени»

setTimeout

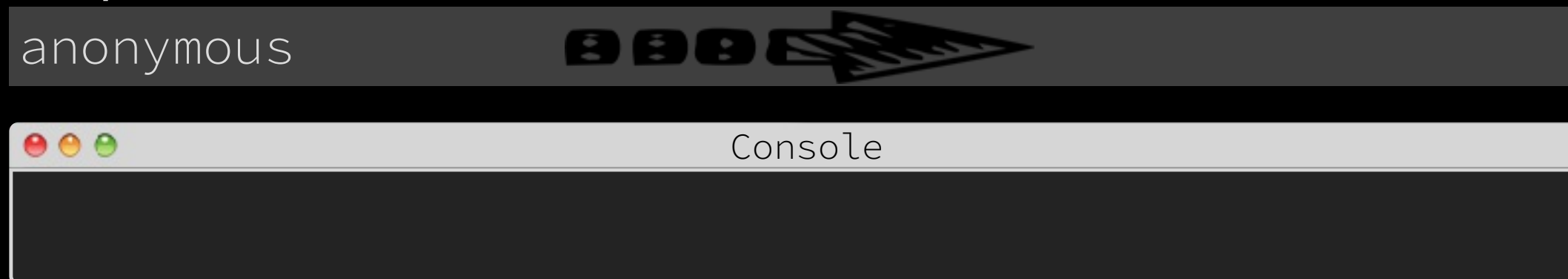
```
setTimeout(func[, delay, param1, param2, ...]);
```

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

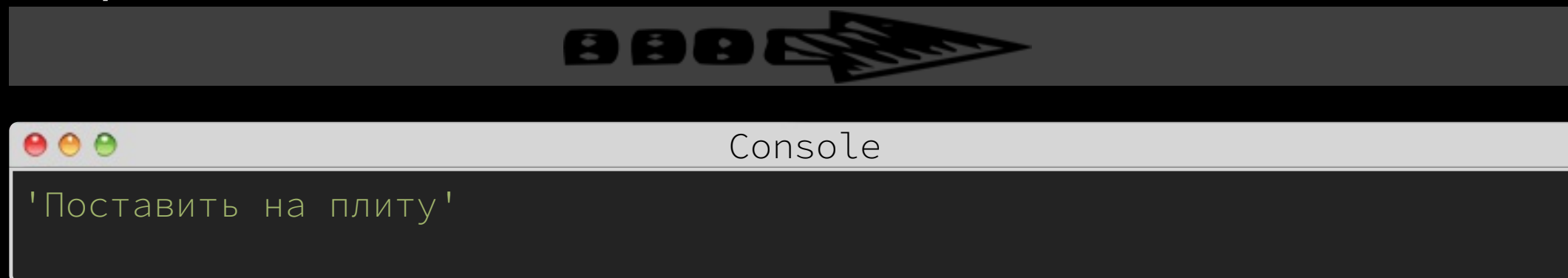
Очередь событий:



Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

Очередь событий:

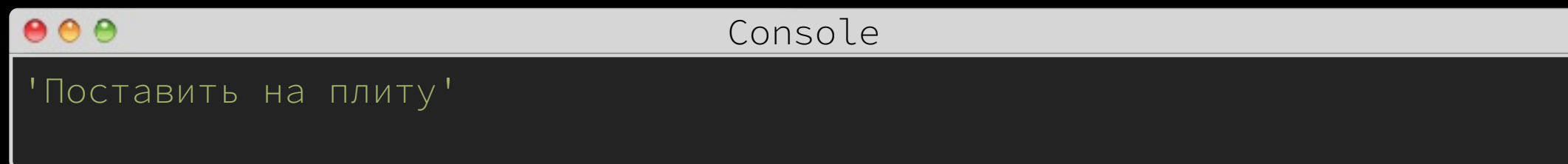


 0 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

Очередь событий:

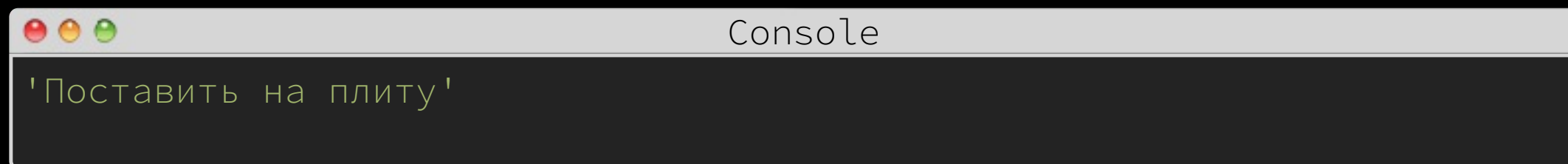


 5 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

Очередь событий:



Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

Очередь событий:

fromStove 

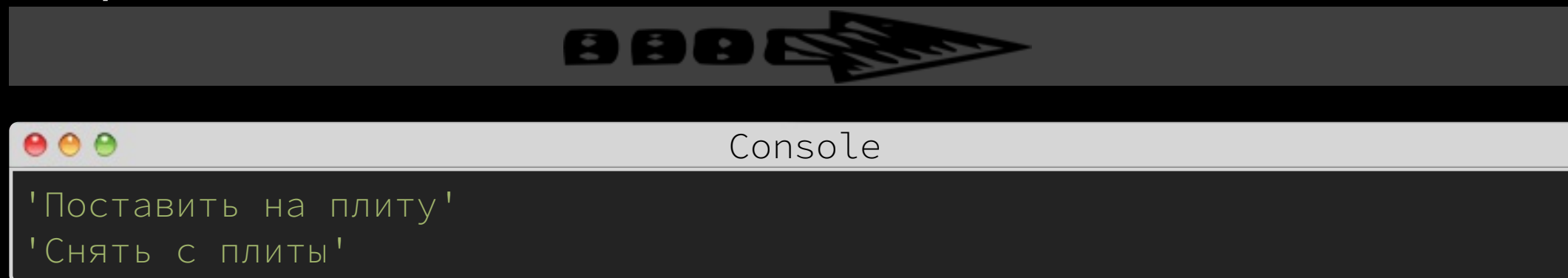
Console

```
'Поставить на плиту'
```


Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
  
function fromStove() {  
    return 'Снять с плиты';  
}  
  
toStove();  
setTimeout(fromStove, 5000);
```

Очередь событий:



setInterval

```
setInterval(func[, delay, param1, param2, ...]);
```

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:

anonymous 



Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:



 0 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:





1 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:



 1 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:

toStir



Console

```
'Поставить на плиту'  
'Помешивать'
```


 2 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:



 2 sec

Код:


```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:

toStir 

 Console

```
'Поставить на плиту'  
'Помешивать'  
'Помешивать'
```

 3 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:



Console

```
'Поставить на плиту'  
'Помешивать'  
'Помешивать'
```

 3 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:

toStir



Console

```
'Поставить на плиту'  
'Помешивать'  
'Помешивать'  
'Помешивать'
```



4 sec

Код:

```
function toStove() {  
    return 'Поставить на плиту';  
}  
function toStir() {  
    return 'Помешивать';  
}  
  
toStove();  
setInterval(toStir, 1000);
```

Очередь событий:



Console

```
'Поставить на плиту'  
'Помешивать'  
'Помешивать'  
'Помешивать'
```

```
var id = setInterval(toStir, 1000);
```

```
clearInterval(id);
```

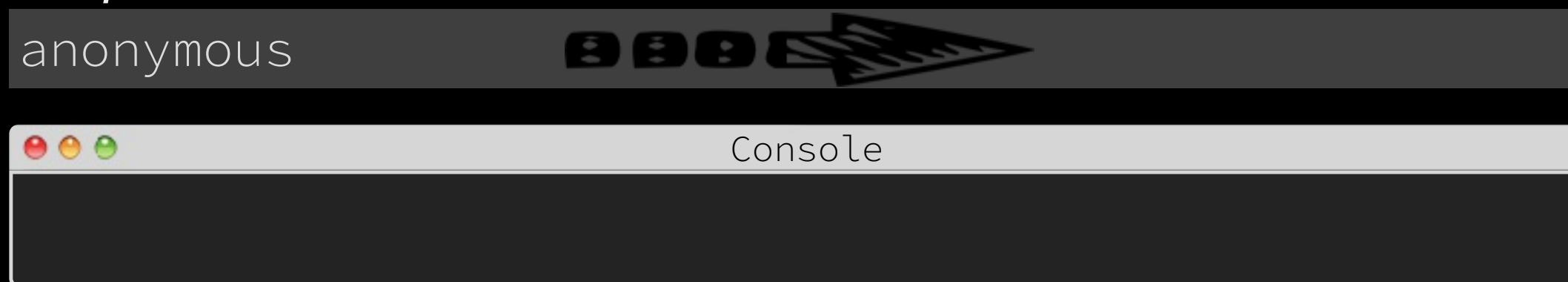
Нулевой интервал

```
function beHappy() {  
    return 'Насладиться результатом';  
}  
  
setTimeout(beHappy, 0);  
  
console.log('Много-много синхронных дел');
```



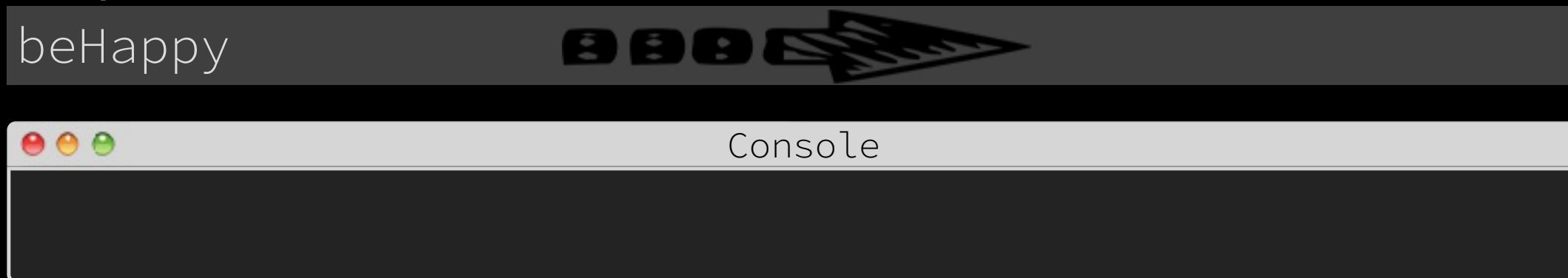
```
function beHappy() {  
    return 'Насладиться результатом';  
}  
  
setTimeout(beHappy, 0);  
  
console.log('Много-много синхронных дел');
```

Очередь событий:



```
function beHappy() {  
    return 'Насладиться результатом';  
}  
  
setTimeout(beHappy, 0);  
  
console.log('Много-много синхронных дел');
```

Очередь событий:



```
function beHappy() {  
    return 'Насладиться результатом';  
}  
  
setTimeout(beHappy, 0);  
  
console.log('Много-много синхронных дел');
```

Очередь событий:



```
function beHappy() {  
    return 'Насладиться результатом';  
}  
  
setTimeout(beHappy, 0);  
  
console.log('Много-много синхронных дел');
```

Очередь событий:

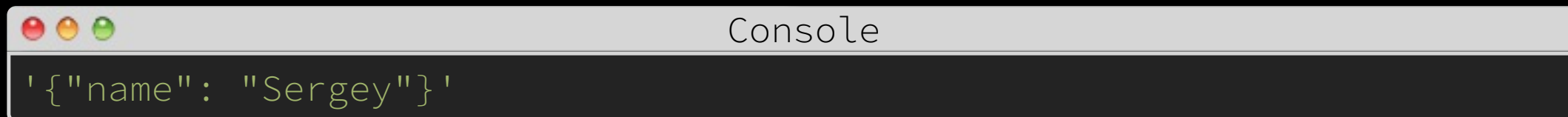
beHappy 

Console

```
'Много-много синхронных дел'  
'Насладиться результатом'
```

Работа с файлами

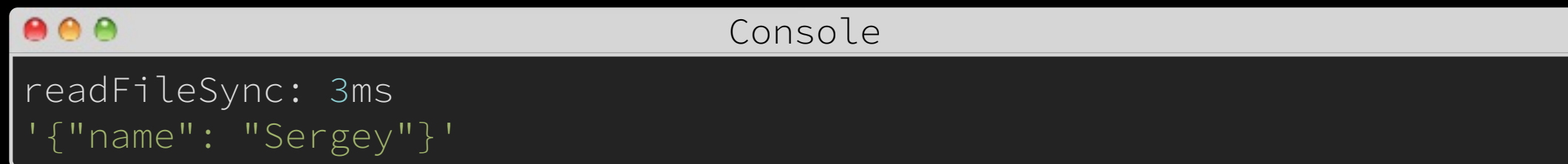
```
var fs = require('fs');  
var fileName = __dirname + '/data.json';  
  
var data = fs.readFileSync(fileName, 'utf-8');  
  
console.log(data);
```



Console

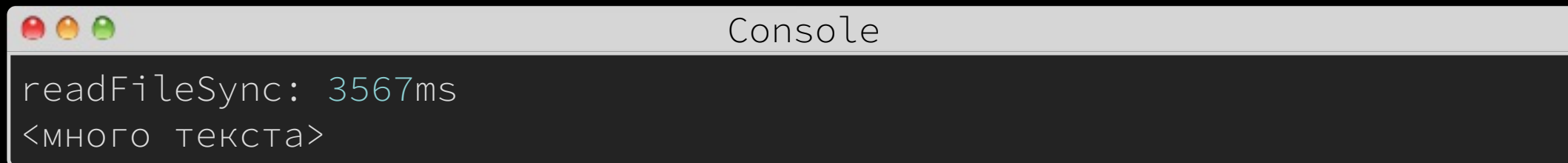
```
' {"name": "Sergey"} '
```

```
var fs = require('fs');  
var fileName = __dirname + '/data.json';  
  
console.time('readFileSync');  
var data = fs.readFileSync(fileName, 'utf-8');  
console.timeEnd('readFileSync');  
  
console.log(data);
```



```
readFileSync: 3ms  
'{"name": "Sergey"}'
```

```
var fs = require('fs');  
var fileName = __dirname + '/bigData.mov';  
  
console.time('readFileSync');  
var data = fs.readFileSync(fileName, 'utf-8');  
console.timeEnd('readFileSync');  
  
console.log(data);
```



readFileSync: 3567ms
<МНОГО ТЕКСТА>

Во время синхронных операций не
обрабатываются другие события: таймеры,
пользовательские события и тп

Пример



Синхронная операция

```
var fs = require('fs');  
var fileName = __dirname + '/data.json';  
  
var data = fs.readFile(fileName, 'utf-8');  
  
console.log(data);
```



```
var fs = require('fs');  
var fileName = __dirname + '/data.json';  
  
fs.readFile(fileName, 'utf-8', function (err, data) {  
    console.log(data);  
});
```



Console

```
'{"name": "Sergey"}'
```

callback

callback

```
function cb(err, data) {
```

callback

```
function cb(err, data) {  
  if (err) {  
    console.error(err.stack);  
  }  
  
}
```

callback

```
function cb(err, data) {  
  if (err) {  
    console.error(err.stack);  
  } else {  
    console.log(data);  
  }  
}
```


callback. Достоинства

- Нет накладных расходов
- Не нужно подключать дополнительные библиотеки

callback. Недостатки

- Глубокий уровень вложенности

```
var fs = require('fs');

fs.readFile('data.json', function (err, data) {
  if (err) {
    console.error(err.stack);
  } else {
    console.log(data);
  }
});
```

```
var fs = require('fs');

fs.readFile('data.json', function (err, data) {
  if (err) {
    console.error(err.stack);
  } else {
    fs.readFile('ext.json', function (e, ext) {
      if (e) {
        console.error(e.stack);
      } else {
        console.log(data + ext);
      }
    });
  }
});
```

callback. Недостатки

- Глубокий уровень вложенности
- Обработка ошибок и данных в одном месте
- Необработанные исключения

```
var fs = require('fs');

function readTwoFiles(cb) {
  var tmp;

  fs.readFile('data.json', function (err, data) {
    if (tmp) {cb(err, data + tmp);}
    else { tmp = data; }
  });

  fs.readFile('ext.json', function (err, data) {
    if (tmp) {cb(err, tmp + data);}
    else { tmp = data; }
  });
}
```

```
var fs = require('fs');

function readTwoFiles(cb) {
  var tmp;

  fs.readFile('data.json', function (err, data) {
    if (tmp) {cb(err, data + tmp);}
    else { throw Error('Mu-ha-ha!'); }
  });

  fs.readFile('ext.json', function (err, data) {
    if (tmp) {cb(err, tmp + data);}
    else { tmp = data; }
  });
}
```

callback. Недостатки

- Глубокий уровень вложенности
- Обработка ошибок и данных в одном месте
- Необработанные исключения
- Лишние переменные

callback. Когда использовать

- Нужна высокая производительность
- Код библиотеки

Promises

Promises. Асинхронный код

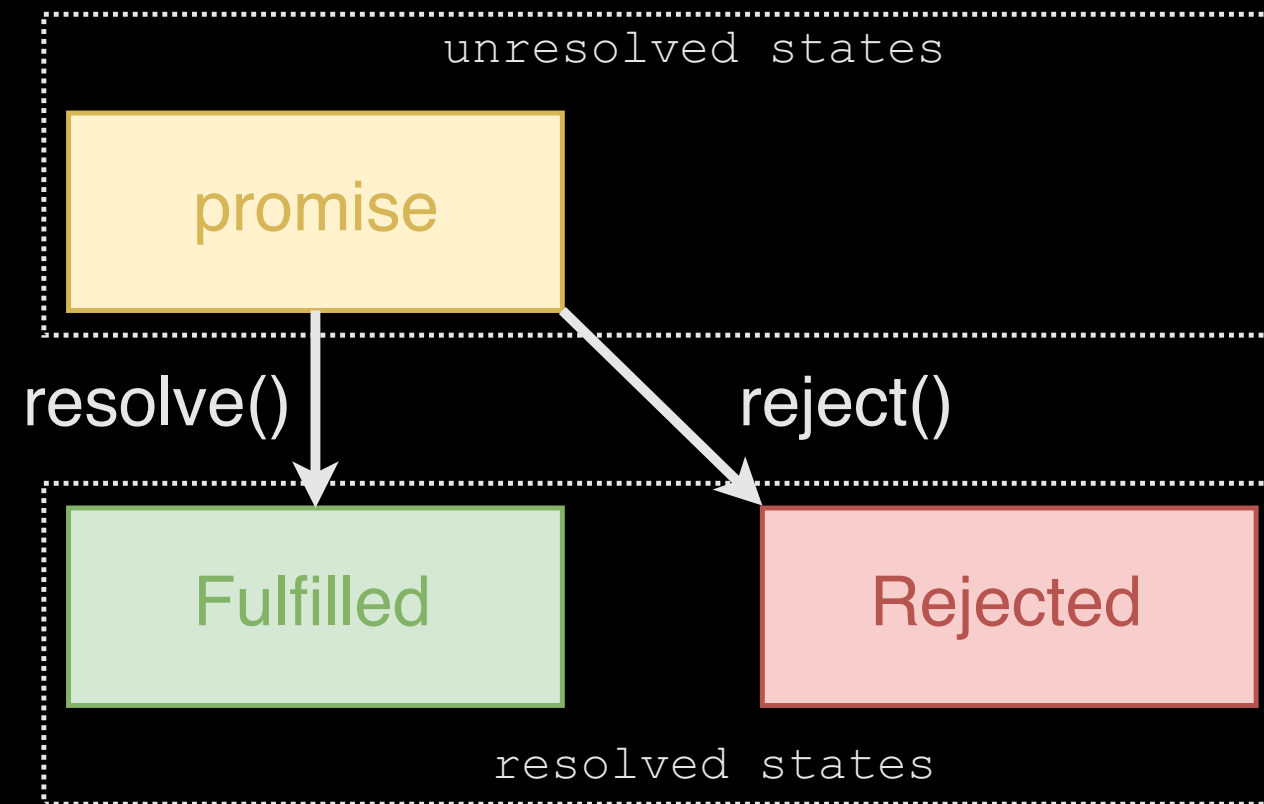
Promises. Асинхронный код

Promises. Асинхронный код

```
var promise = new Promise(function (resolve, reject) {  
    fs.readFile('data.json', function (err, data) {  
        if (err) {  
            reject(err);  
        }  
    });  
});
```

Promises. Асинхронный код

```
var promise = new Promise(function (resolve, reject) {  
    fs.readFile('data.json', function (err, data) {  
        if (err) {  
            reject(err);  
        } else {  
            resolve(data);  
        }  
    });  
});
```



Promises. Обработчики

```
promise.then();
```


Promises. Обработчики

```
promise.then(function (data) {  
    console.log(data)  
});
```

Promises. Обработчики

```
promise.then(function (data) {  
    console.log(data)  
}, function (err) {  
    console.error(err);  
});
```

Promises. Обработчики

```
promise.then(console.log, console.error);
```

Promises. Недостатки

- Дополнительные обёртки
- Немного медленнее, чем callback

Promises. Достоинства

- Контролируем исключения

```
var promise = new Promise(function (resolve, reject) {  
    fs.readFile('data.json', function (err, data) {  
        if (err) {  
            reject(err);  
        } else {  
            throw new Error('Mu-ha-ha!');  
        }  
    });  
});
```

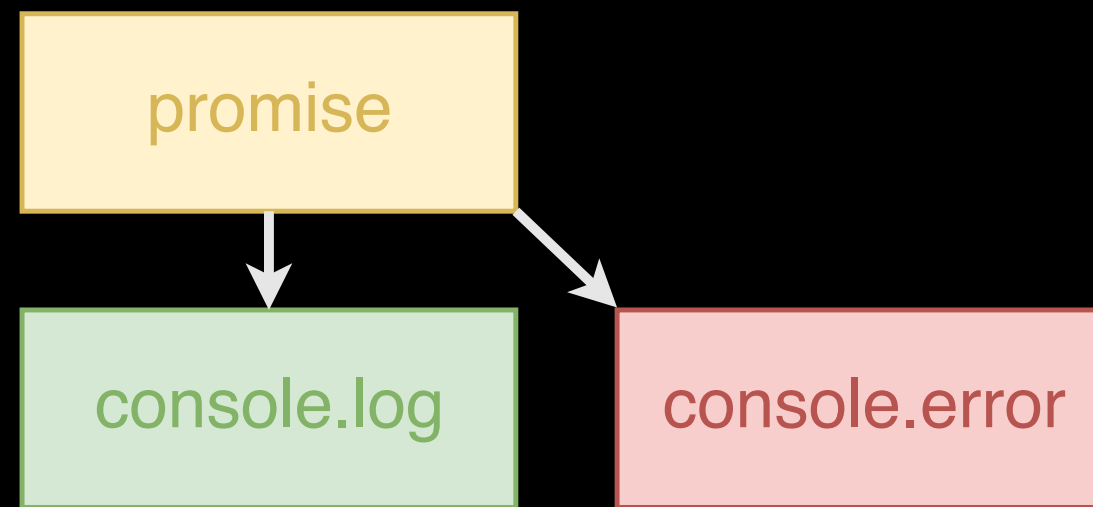
```
promise.then(console.log, console.error);
```



Promises. Достоинства

- Контролируем исключения
- Несколько обработчиков

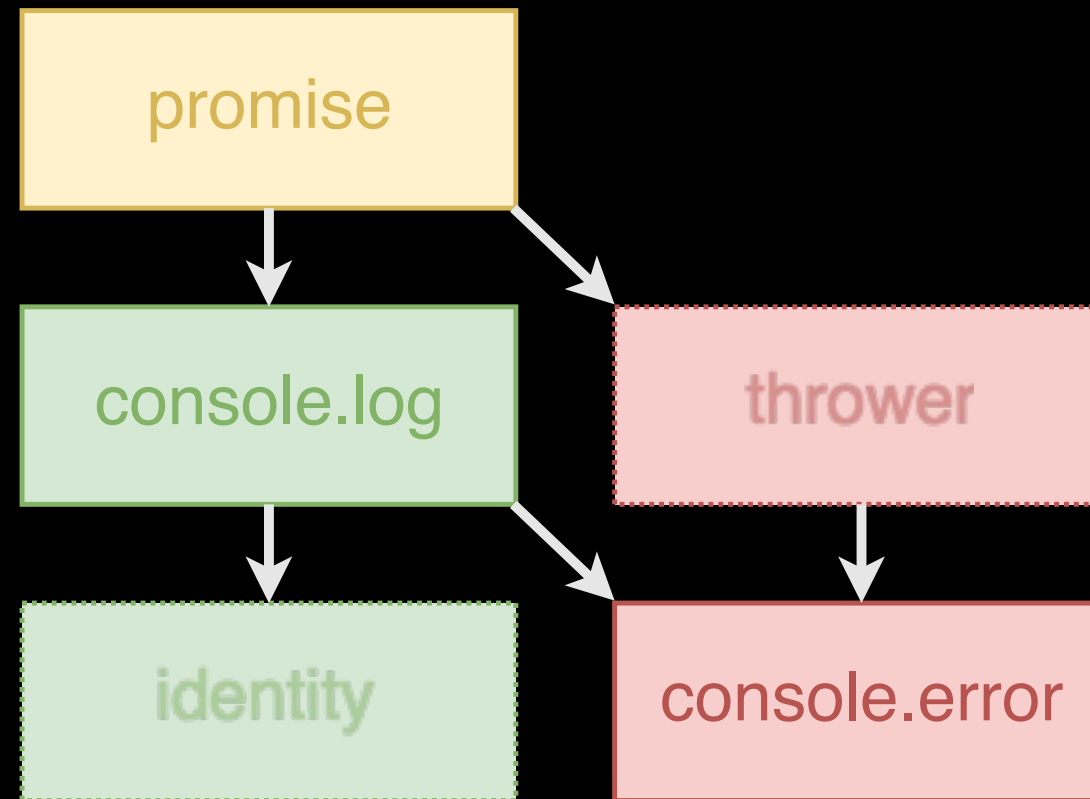
Чейнинг



```
function identity(data) {  
    return data;  
}
```

```
function thrower(err) {  
    throw err;  
}
```

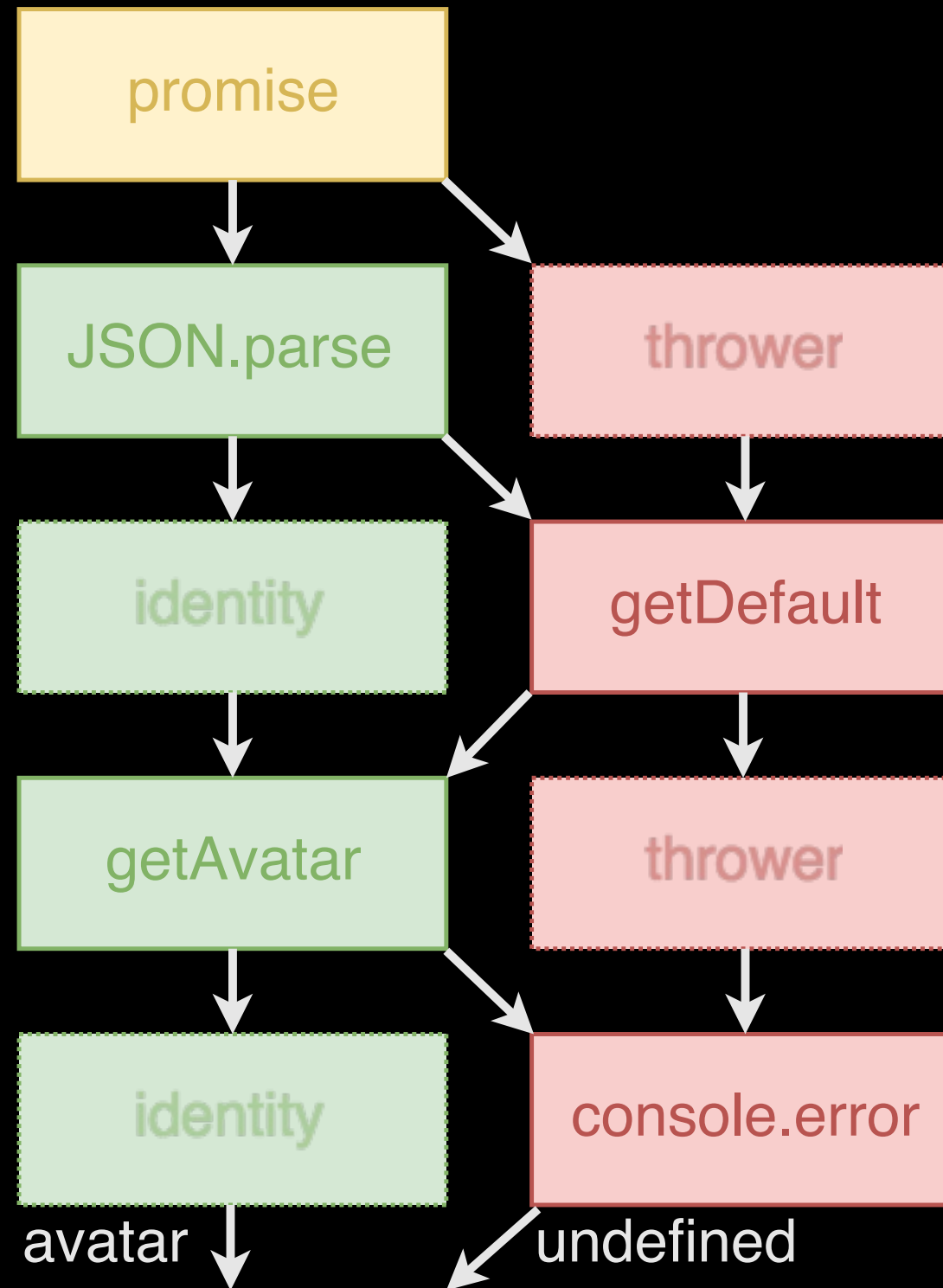
```
promise  
    .then(console.log, thrower)  
    .then(identity, console.error);
```



```
promise
  .then(JSON.parse, thrower)
  .then(identity, getDefault)
  .then(getAvatar, thrower)
  .then(identity, console.error);
```

```
function getDefault() {
  return { name: 'Sergey' };
}
```

```
function getAvatar (data) {
  var name = data.name;
  return request('https://my.avatar/' + name);
}
```



- `then` вернул данные — передаём по цепочке дальше
- `then` вернул промис — передаём результат работы промиса
- произошла ошибка — реджектим промис

```
promise
  .then(JSON.parse, thrower)
  .then(identity, getDefault)
  .then(getAvatar, thrower)
  .then(identity, console.error);
```



```
promise
  .then(JSON.parse, thrower)
  .then(identity, getDefault)
  .then(getAvatar, thrower)
  .then(identity, console.error);
```

```
promise
  .then(JSON.parse)
  .catch(getDefault)
  .then(getAvatar)
  .catch(console.error);
```

```
promise  
  .then(JSON.parse)  
  .then(getAvatar);
```

НЕ ТО ЖЕ СМОЕ, ЧТО

```
promise.then(JSON.parse);  
promise.then(getAvatar);
```

Promise.all

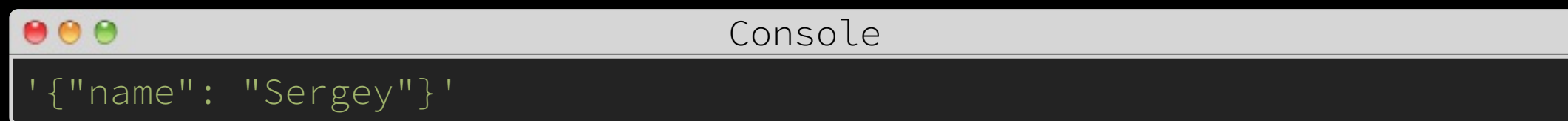
```
function readFile(name) {  
  return new Promise(function (resolve, reject) {  
    fs.readFile(name, function (err, data) {  
      err ? reject(err) : resolve(data);  
    });  
  });  
}
```

Promise.all

```
Promise
  .all([
    readFile('data.json'),
    readFile('ext.json')
  ])
  .then(function (data) {
    console.log(data[0] + data[1])
  });
```

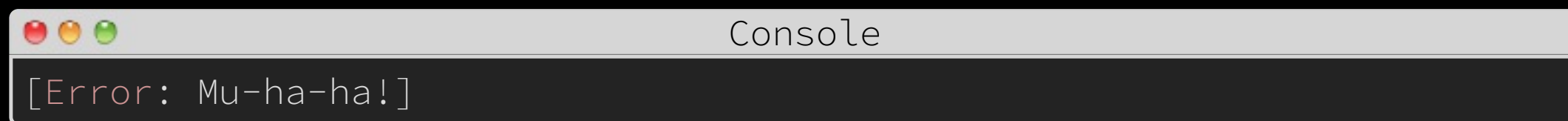
Promise

```
.resolve('{"name": "Sergey"}')  
.then(console.log);
```



Promise

```
.reject(new Error('Mu-ha-ha!'))  
.catch(console.error);
```



Почитать

- Параллельная модель и цикл событий. – [JavaScript | MDN](#)
- `setTimeout` и `setInterval`
- `Promise`