

# **АСИНХРОННЫЙ КОД**

## **СТЕК ВЫЗОВОВ. ОЧЕРЕДЬ СОБЫТИЙ.**

## **EVENT LOOP.**

**Игорь Савичев**

# **СИНХРОННЫЙ КОД**

# СИНХРОННЫЙ КОД

```
function prepareBeans() {
    console.log('Перемолоть кофейные зерна');
}

function addWater() {
    console.log('Добавить немнога воды');
}

function onStove() {
    console.log('Поставить на плиту');
}
```

# СИНХРОННЫЙ КОД

```
prepareBeans();  
addWater();  
onStove();
```

# СИНХРОННЫЙ КОД

```
prepareBeans() ; // 'Перемолоть кофейные зерна'  
addWater();  
onStove();
```

# СИНХРОННЫЙ КОД

```
prepareBeans();    // 'Перемолоть кофейные зерна'  
addWater();       // 'Добавить немного воды'  
onStove();
```

# СИНХРОННЫЙ КОД

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

# **СТЕК ВЫЗОВОВ**

# JAVASCRIPT

- 1 поток => 1 операция за раз
- 1 стек вызовов

```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

# Immediately Invoked Function Expression

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

    function wakeUp() {
        prepareCoffee();
    }

    wakeUp();
})();
```

```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



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

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

```
wakeUp();
```

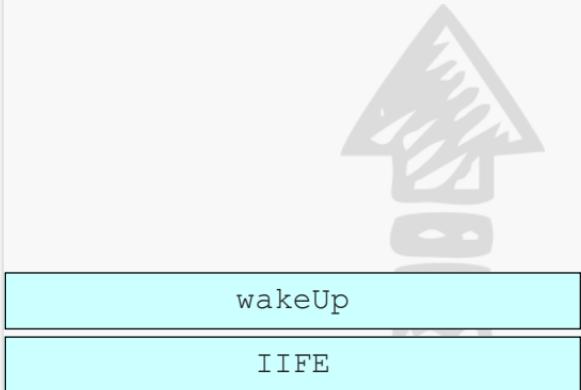
Стек вызовов



IIFE

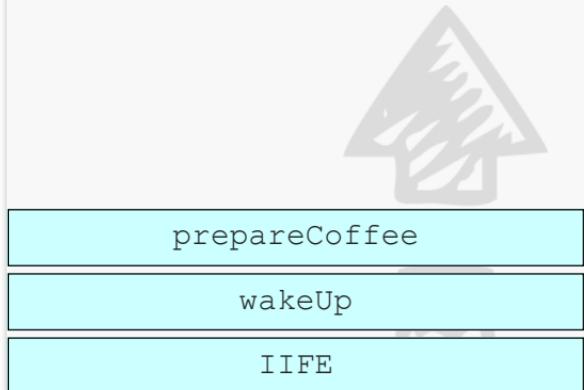
```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



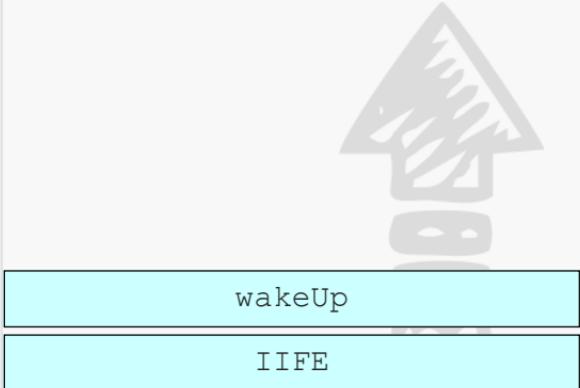
prepareCoffee

wakeUp

IIFE

```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



IIFE

```
function prepareCoffee() {  
    onStove();  
}  
  
function wakeUp() {  
    prepareCoffee();  
}  
  
wakeUp();
```

Стек вызовов



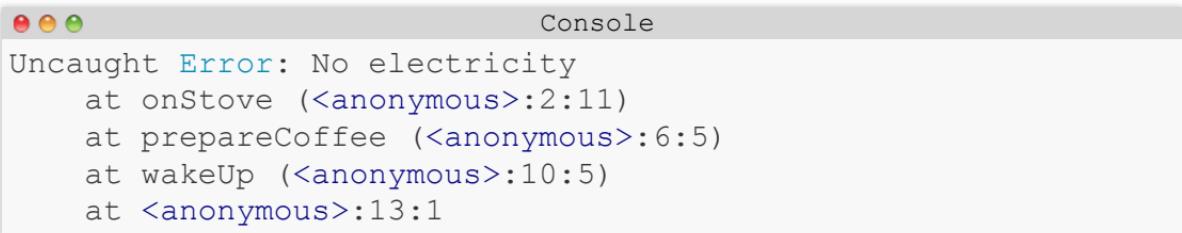
Если стек вызовов пуст, то программа успешно  
завершает свою работу

```
function onStove() {
    throw new Error('No electricity');
}

function prepareCoffee() {
    onStove();
}

function wakeUp() {
    prepareCoffee();
}

wakeUp();
```



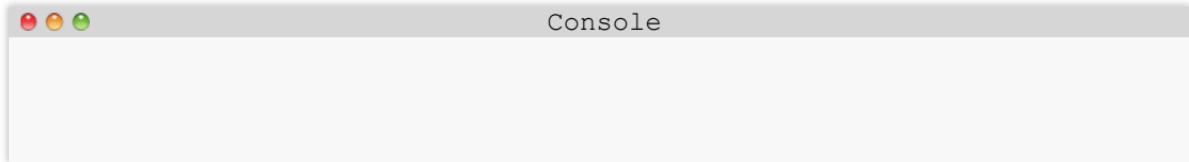
The screenshot shows a browser's developer tools console window. At the top, there are three colored circular icons (red, yellow, green). To the right of them, the word "Console" is displayed. Below this, the error message is shown in a monospaced font. The message indicates an "Uncaught Error: No electricity" and provides a stack trace:

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

# ОТКУДА АСИНХРОННОСТЬ?

```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



```
console.log(1);

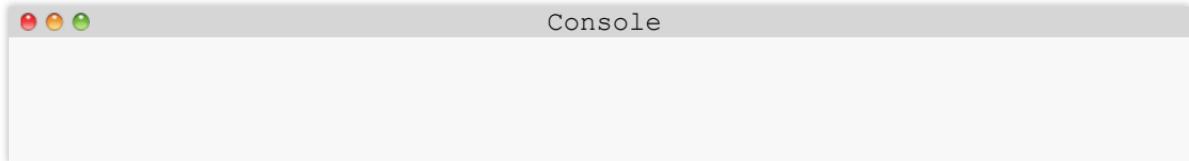
setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



IIFE



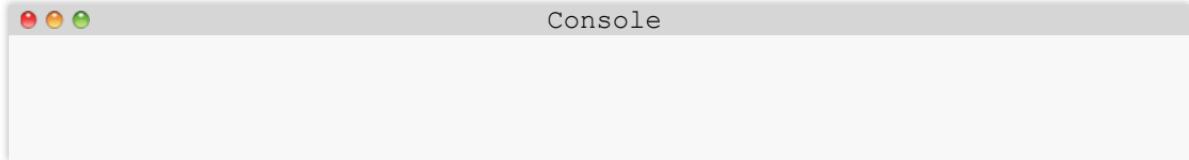
```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



console.log

IIFE



```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов

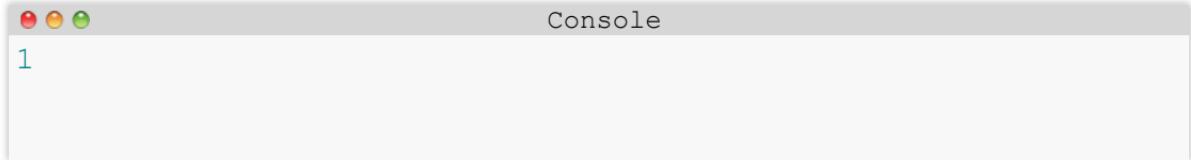
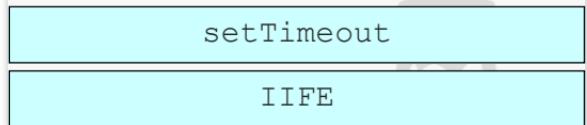


```
console.log(1);

setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

### Стек вызовов



```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



```
console.log(1);

setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



console.log

IIFE



Console

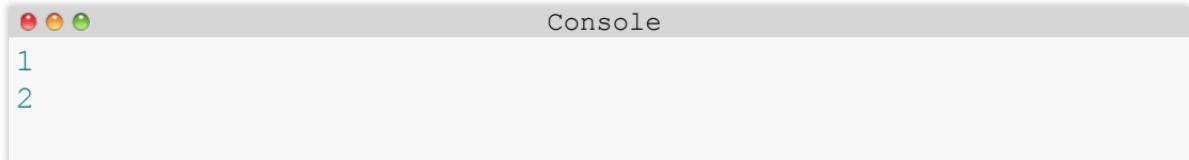
1

```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



IIFE



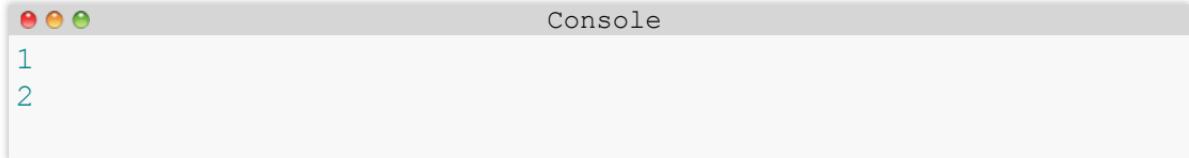
```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов

A screenshot of a browser's developer tools Console tab. The tab title is "Console". In the output area, the numbers "1", "2", and "3" are displayed sequentially, corresponding to the log statements in the code above. The browser interface includes standard window controls (red, yellow, green) in the top-left corner.

```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



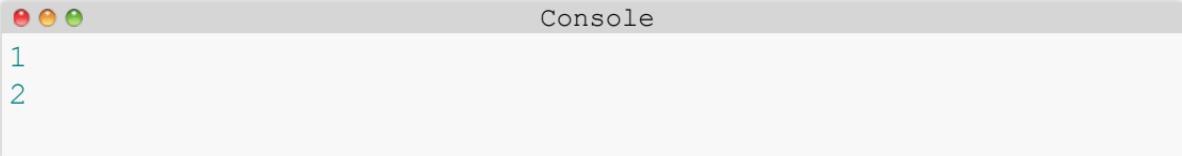
```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

### Стек вызовов



console.log

cb



```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

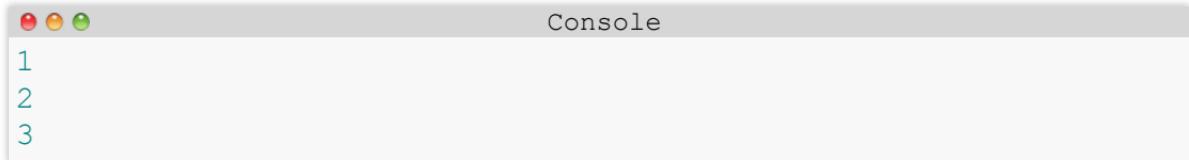
Стек вызовов



A screenshot of a browser window showing the developer tools Console tab. The tab title is 'Console'. The output area displays three lines of text: '1', '2', and '3', which correspond to the log statements in the code above.

```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



# ОЧЕРЕДЬ СОБЫТИЙ

```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



Очередь событий



Console

```
console.log(1);
```

```
setTimeout(function cb() {  
    console.log(3);  
, 5000);
```

```
console.log(2);
```

Стек вызовов



IIFE



Очередь событий



Console

```
console.log(1);
```

```
setTimeout(function cb() {  
    console.log(3);  
, 5000);
```

```
console.log(2);
```

Стек вызовов



console.log

IIFE



Очередь событий



Console

```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



IIFE



Очередь событий



Console

1



```
console.log(1);

setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



setTimeout

IIFE



Очередь событий



Console

1



```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



IIFE



Очередь событий



Console

1



```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



console.log

IIFE



Очередь событий



Console

1



```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



IIFE



Очередь событий



Console

1

2



```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



Очередь событий



Console

1

2



```
console.log(1);  
  
setTimeout(function cb() {  
    console.log(3);  
}, 5000);  
  
console.log(2);
```

Стек вызовов



cb



Очередь событий



Console

1

2

```
console.log(1);

setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



cb



Очередь событий



Console

1

2

```
console.log(1);

setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



console.log

cb



Очередь событий



Console

1

2

```
console.log(1);

setTimeout(function cb() {
    console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



cb



Очередь событий



Console

1

2

3

```
console.log(1);

setTimeout(function cb() {
  console.log(3);
}, 5000);

console.log(2);
```

Стек вызовов



Очередь событий



Console

1

2

3

Если стек вызовов пуст, то можем выполнить функцию из очереди событий

# setTimeout

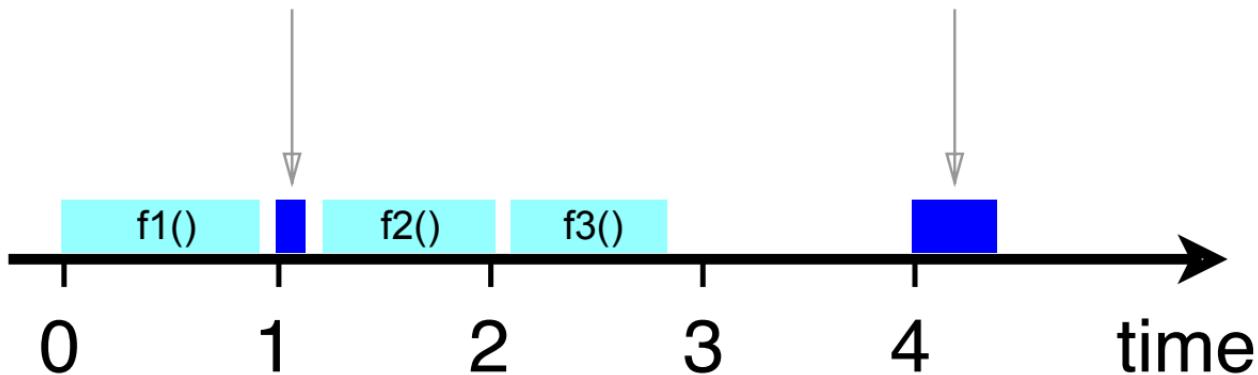
```
setTimeout(() => {  
    console.log('Я выполнюсь через 5 секунд.')  
, 5000);
```

# **setTimeout**

```
setTimeout(console.log, 5000, 'Я выполнюсь через 5 секунд.');
```

`setTimeout(cb, 3000)`

`cb()`



```
function onStove() {
    console.log('Поставить на плиту');
}

function fromStove() {
    console.log('Снять с плиты');
}

onStove();
setTimeout(fromStove, 5000);
```

```
function onStove() {
    console.log('Поставить на плиту');
}

function fromStove() {
    console.log('Снять с плиты');
}

onStove();
setTimeout(fromStove, 5000);
```



Очередь событий



Console

```
function onStove() {
    console.log('Поставить на плиту');
}

function fromStove() {
    console.log('Снять с плиты');
}

onStove();
setTimeout(fromStove, 5000);
```



Очередь событий



Console

'Поставить на плиту'



```
function onStove() {
    console.log('Поставить на плиту');
}

function fromStove() {
    console.log('Снять с плиты');
}

onStove();
setTimeout(fromStove, 5000);
```



Очередь событий



Console

'Поставить на плиту'



5 sec

```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function fromStove() {  
    console.log('Снять с плиты');  
}  
  
onStove();  
setTimeout(fromStove, 5000);
```

fromStove



Очередь событий



Console

'Поставить на плиту'

```
function onStove() {
    console.log('Поставить на плиту');
}

function fromStove() {
    console.log('Снять с плиты');
}

onStove();
setTimeout(fromStove, 5000);
```



Очередь событий



Console

'Поставить на плиту'  
'Снять с плиты'

ДЕМО

```
const timerId = setTimeout(fromStove, 5000);  
clearTimeout(timerId);
```

# ПРОБЛЕМА

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



Очередь событий



Console

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}
```

```
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



IIFE



Очередь событий



Console



0 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



setTimeout

IIFE



Очередь событий



Console



0 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



IIFE



Очередь событий



Console



0 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



hardWork

IIFE



Очередь событий



Console



0 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



Date.now

hardWork

IIFE



Очередь событий



Console



0 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



hardWork

IIFE



Очередь событий



Console



0 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



Date.now

hardWork

IIFE



Очередь событий



Console



```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



Date.now

hardWork

IIFE

cb



Очередь событий



Console



```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



Date.now

hardWork

IIFE

cb



Очередь событий



Console



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



hardWork

IIFE

cb



Очередь событий



Console



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



console.log

hardWork

IIFE

cb



Очередь событий



Console



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



hardWork

IIFE

cb



Очередь событий



Console

'Done'



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



IIFE

cb



Очередь событий



Console

'Done'



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



cb



Очередь событий



Console

'Done'



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



cb



Очередь событий



Console

'Done'



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



console.log

cb



Очередь событий



Console

'Done'



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



cb



Очередь событий



Console

'Done'  
'Callback'



3 sec

```
function hardWork() {  
    const start = Date.now();  
    while (Date.now() - start < 3000) {}  
    console.log('Done');  
}  
  
setTimeout(function cb() {  
    console.log('Callback');  
}, 1000);  
  
hardWork();
```

Стек вызовов



Очередь событий

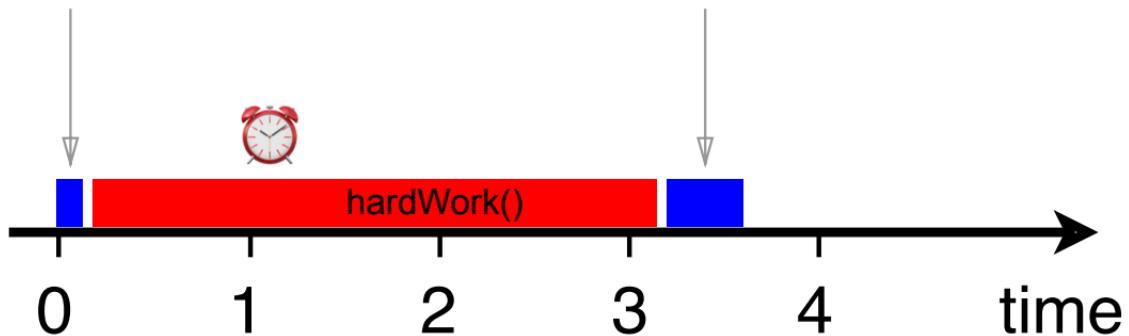


Console

```
'Done'  
'Callback'
```

`setTimeout(cb, 1000)`

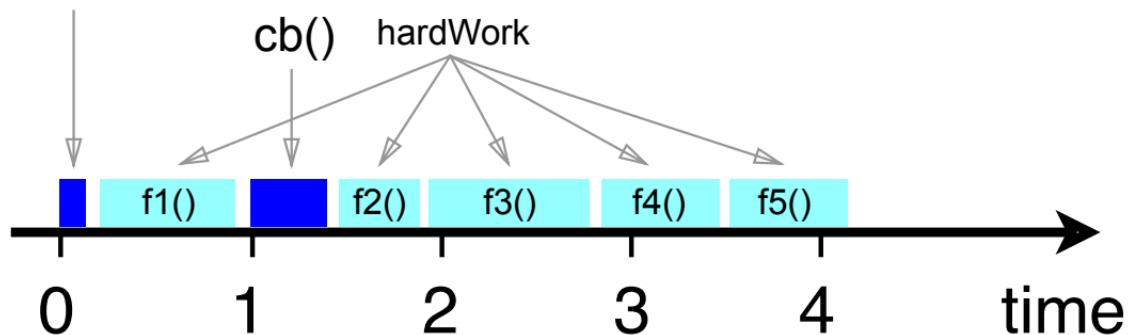
`cb()`



## ВЫВОДЫ

- setTimeout гарантирует, что событие произойдёт не раньше указанного времени
- Разбивать большие функции на маленькие

`setTimeout(cb, 1000)`



Код

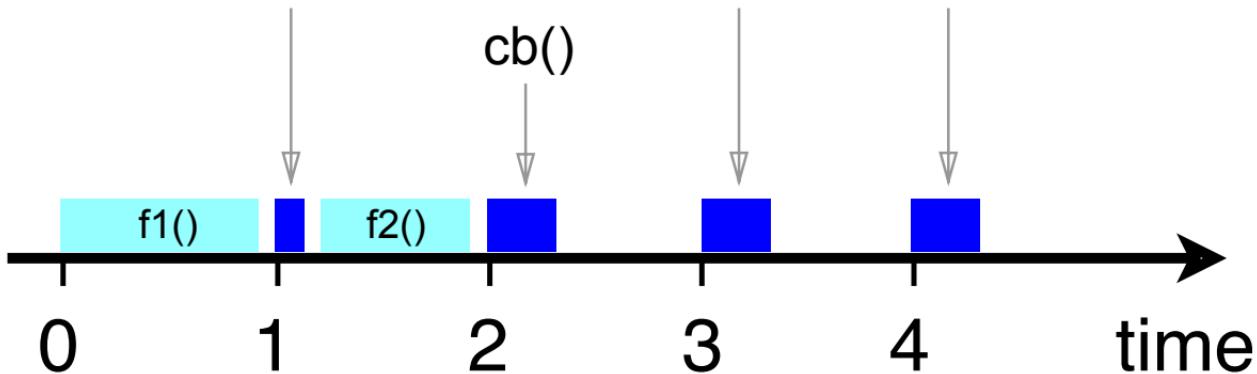
# **setInterval**

```
setInterval(() => {  
    console.log('Я буду выполняться каждые 5 секунд.')  
, 5000);
```

# setInterval

```
setInterval(console.log, 5000, 'Я буду выполняться каждые 5 секунд.');
```

`setInterval(cb, 1000)`



```
function onStove() {
    console.log('Поставить на плиту');
}

function stir() {
    console.log('Помешивать');
}

onStove();
setInterval(stir, 1000);
```

```
function onStove() {
    console.log('Поставить на плиту');
}

function stir() {
    console.log('Помешивать');
}

onStove();
setInterval(stir, 1000);
```



Очередь событий



Console

```
function onStove() {
    console.log('Поставить на плиту');
}

function stir() {
    console.log('Помешивать');
}

onStove();
setInterval(stir, 1000);
```



Очередь событий



Console

'Поставить на плиту'



0 sec

```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```



Очередь событий



Console

'Поставить на плиту'



```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```

stir



Очередь событий



Console

'Поставить на плиту'



```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```



Очередь событий



Console

'Поставить на плиту'  
'Помешивать'



```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```

stir



Очередь событий



Console

'Поставить на плиту'  
'Помешивать'



```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```



Очередь событий



Console

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



```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```

stir



Очередь событий



Console

'Поставить на плиту'  
'Помешивать'  
'Помешивать'



```
function onStove() {  
    console.log('Поставить на плиту');  
}  
  
function stir() {  
    console.log('Помешивать');  
}  
  
onStove();  
setInterval(stir, 1000);
```



Очередь событий



Console

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

```
const timerId = setInterval(stir, 1000);  
clearInterval(timerId);
```

# ПРОБЛЕМА

```
function hardWork() {  
    // этот код выполняется 50 мс  
};  
  
console.log('Start');  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
, 100);
```

Стек вызовов



Очередь событий



Console

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



IIFE



Очередь событий



Console

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

### Стек вызовов



console.log

IIFE



Очередь событий



Console

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

### Стек вызовов



IIFE



Очередь событий



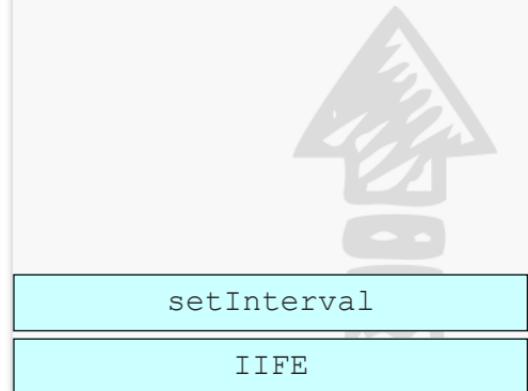
Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
, 100);
```

Стек вызовов



IIFE



Очередь событий



Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
, 100);
```

Стек вызовов



Очередь событий



Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



cb



Очередь событий



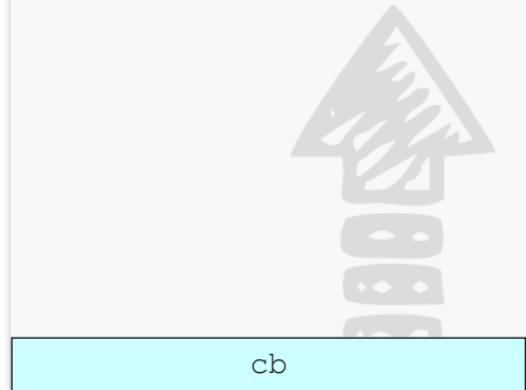
Console

'Start'

 100 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
, 100);
```

Стек вызовов



Очередь событий



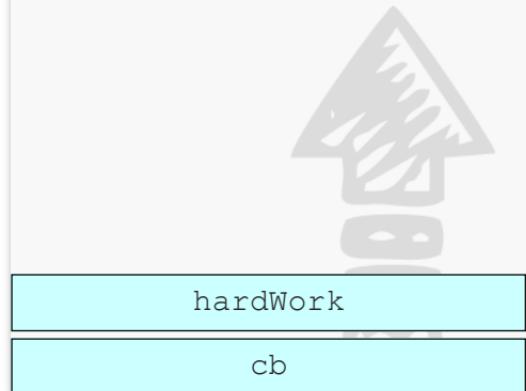
Console

'Start'

 100 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



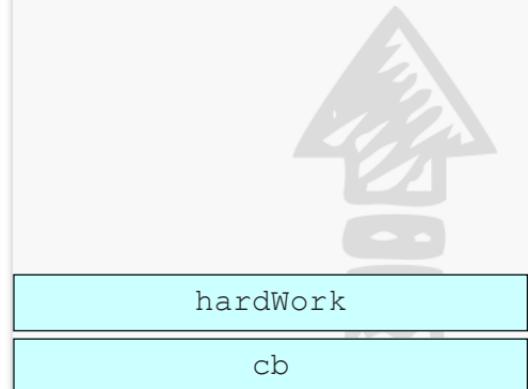
Console

'Start'

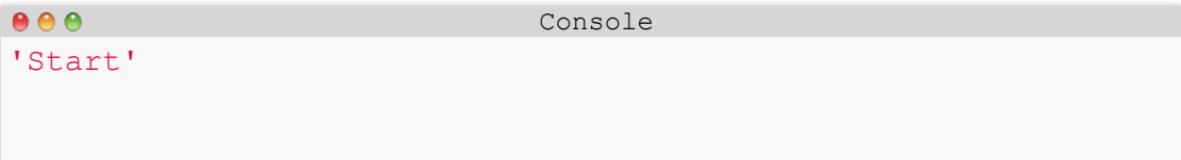
 100 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



cb



Очередь событий



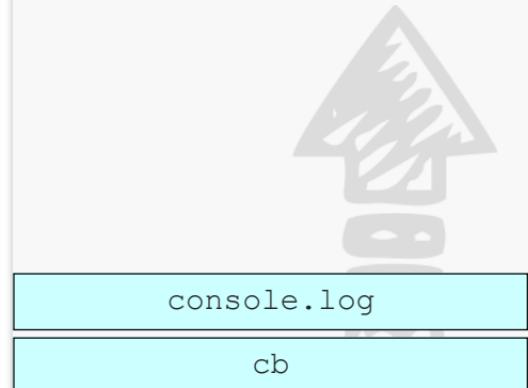
Console

'Start'

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



Console

'Start'

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



cb



Очередь событий



Console

```
'Start'  
'Done'
```

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



Console

```
'Start'  
'Done'
```

 200 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



cb



Очередь событий



Console

```
'Start'  
'Done'
```

 200 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



cb



Очередь событий



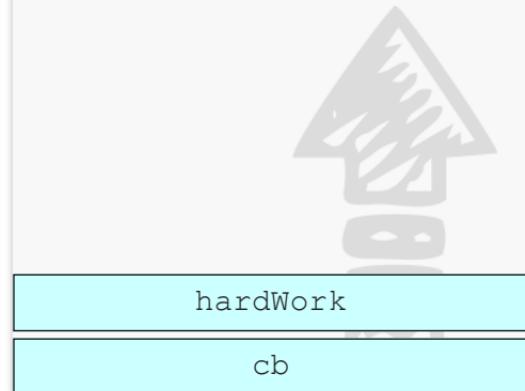
Console

```
'Start'  
'Done'
```

 200 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



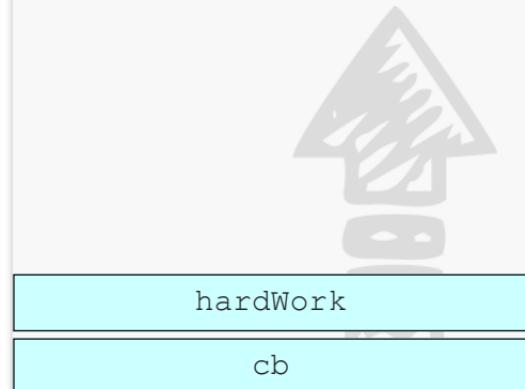
Console

```
'Start'  
'Done'
```

 200 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



Очередь событий



Console

```
'Start'  
'Done'
```

 250 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов



cb



Очередь событий



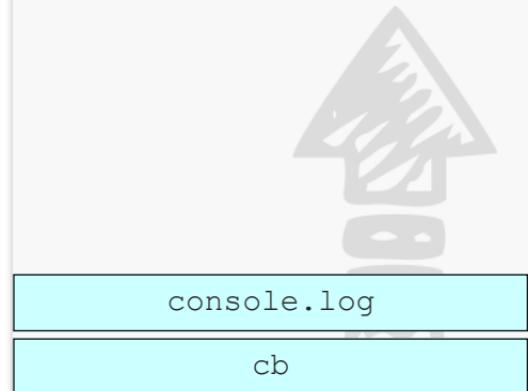
Console

```
'Start'  
'Done'
```



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

### Стек вызовов



### Очередь событий



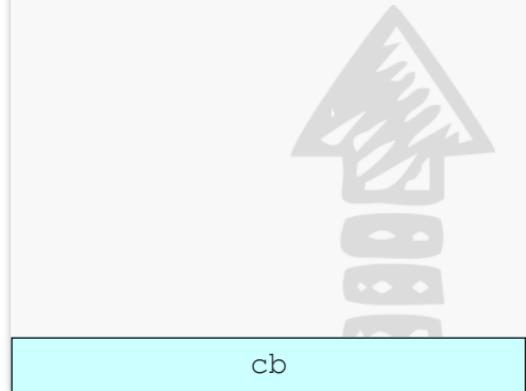
### Console

```
'Start'  
'Done'
```

 250 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
, 100);
```

Стек вызовов



cb



Очередь событий



Console

```
'Start'  
'Done'  
'Done'
```



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setInterval(function cb() {  
    hardWork();  
    console.log('Done');  
}, 100);
```

Стек вызовов

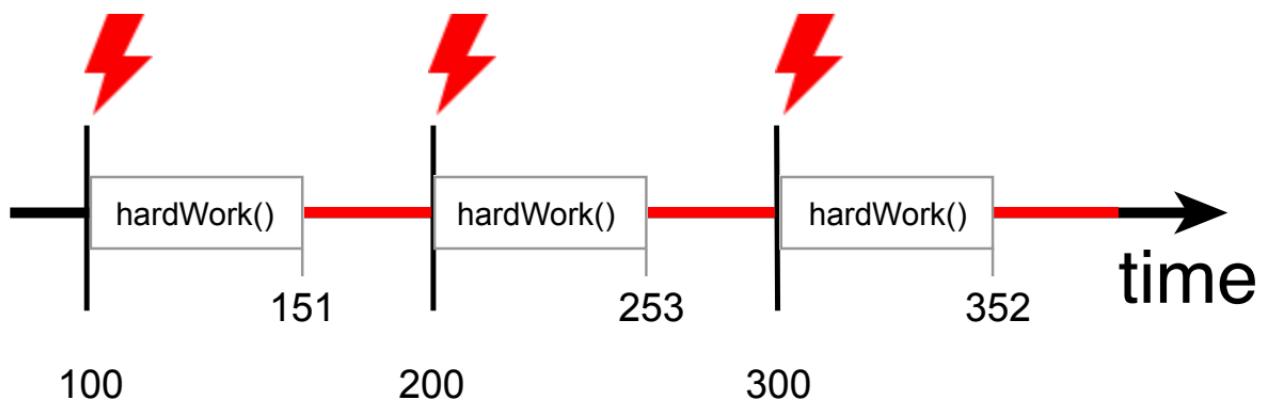


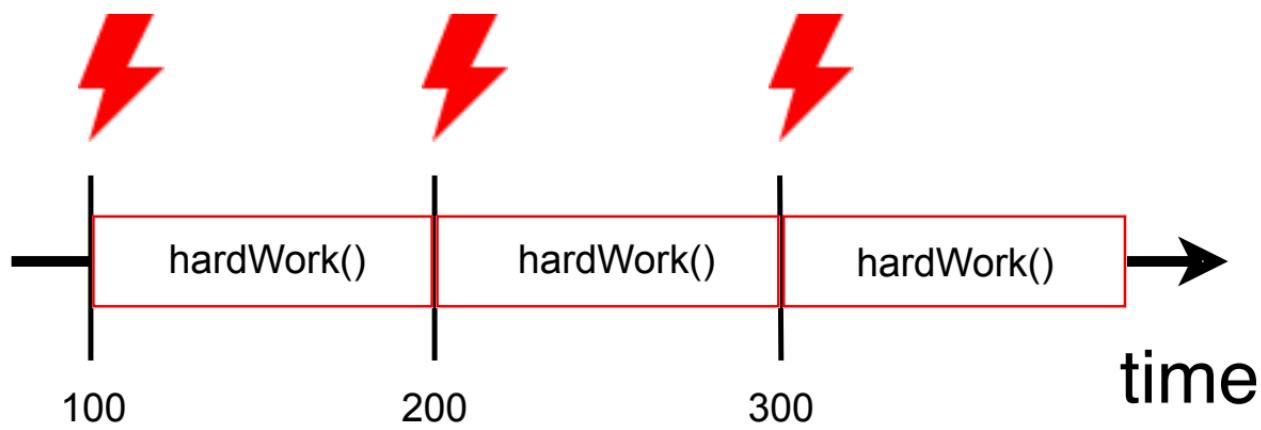
Очередь событий



Console

```
'Start'  
'Done'  
'Done'
```





## ВЫВОДЫ

- setInterval гарантирует, что событие произойдёт не раньше указанного времени
- setInterval vs. setTimeout

# **SETINTERVAL → SETTIMEOUT**

```
function hardWork() {  
    // этот код выполняется 50 мс  
};  
  
console.log('Start');  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



Console

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



IIFE



Очередь событий



Console

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

### Стек вызовов



console.log

IIFE



Очередь событий



Console

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

### Стек вызовов



IIFE



Очередь событий



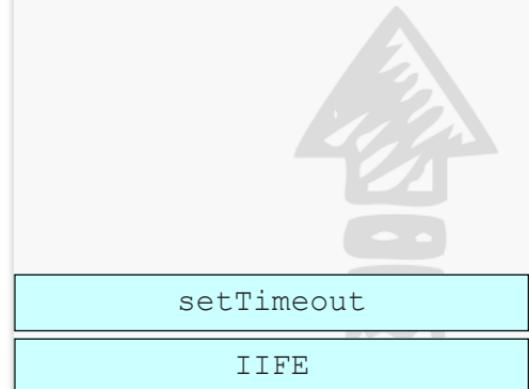
Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



IIFE



Очередь событий



Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



Console

'Start'



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



cb



Очередь событий



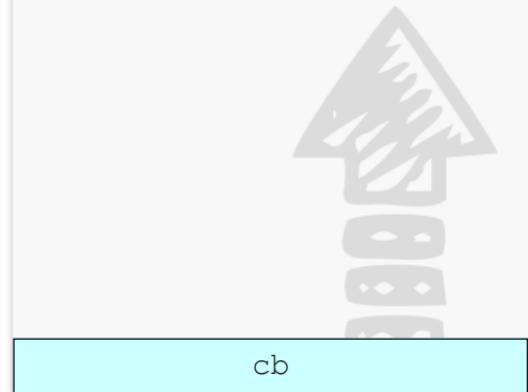
Console

'Start'

 100 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



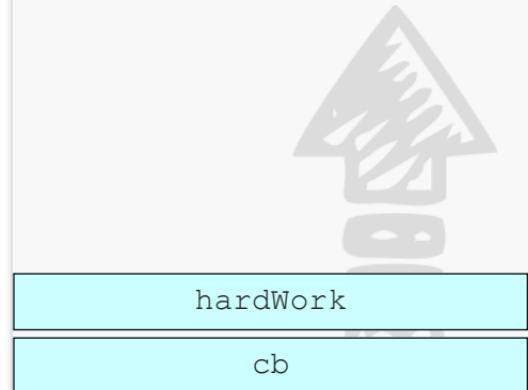
Console

'Start'

 100 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



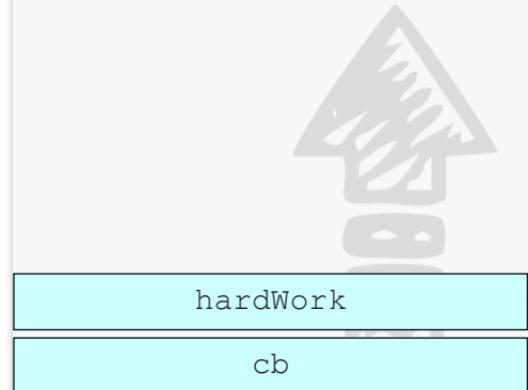
Console

'Start'

⌚ 100 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



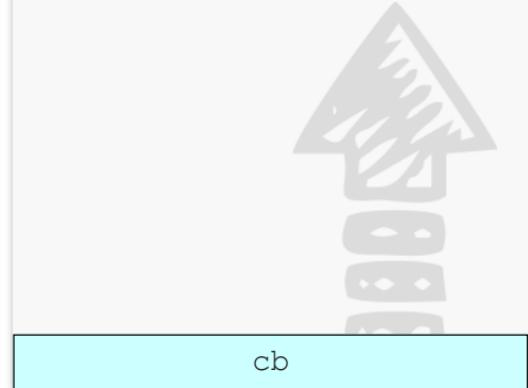
Console

'Start'

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



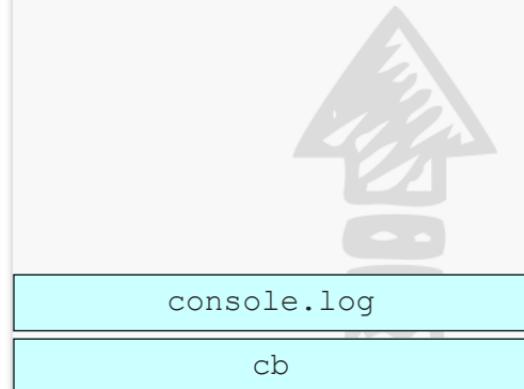
Console

'Start'

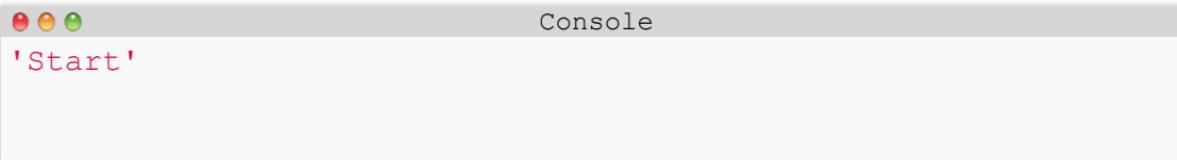
 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



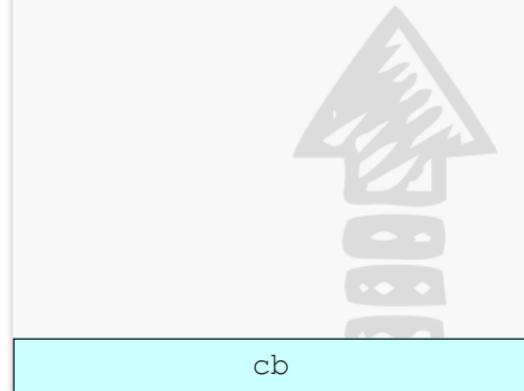
Очередь событий



 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



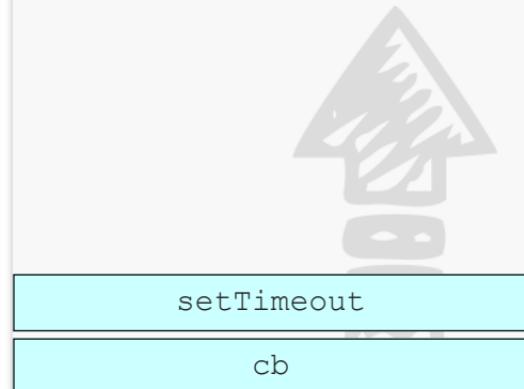
Console

```
'Start'  
'Done'
```

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



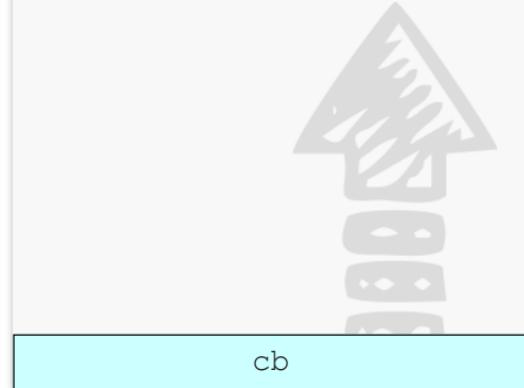
Console

```
'Start'  
'Done'
```

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



Console

```
'Start'  
'Done'
```

 150 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



Console

```
'Start'  
'Done'
```



```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



cb



Очередь событий



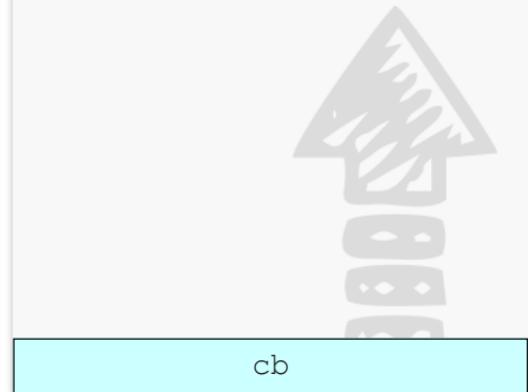
Console

'Start'  
'Done'

 250 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



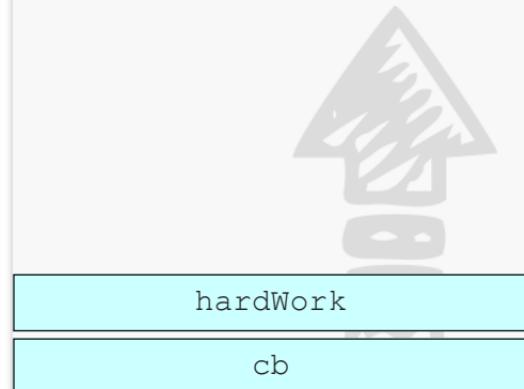
Console

```
'Start'  
'Done'
```

 250 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов



Очередь событий



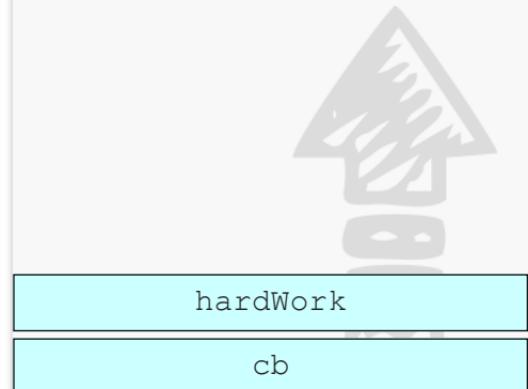
Console

```
'Start'  
'Done'
```

 250 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
};  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



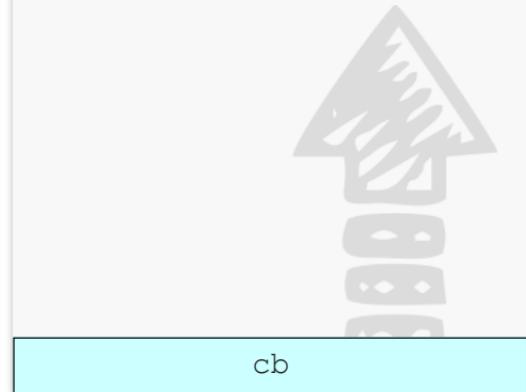
Console

```
'Start'  
'Done'
```

 300 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



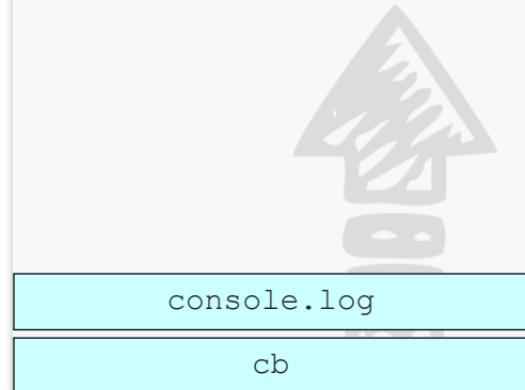
Console

```
'Start'  
'Done'
```

 300 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



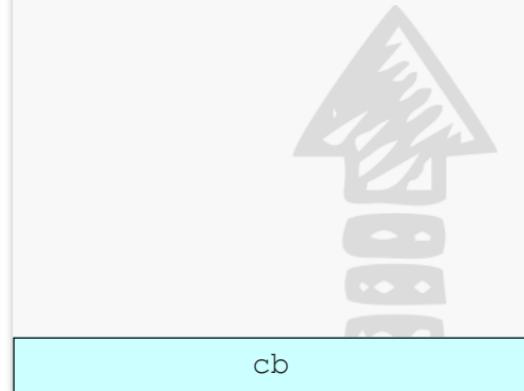
Console

```
'Start'  
'Done'
```

 300 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



cb



Очередь событий



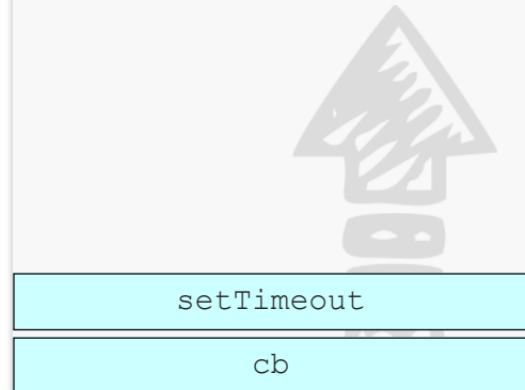
Console

```
'Start'  
'Done'  
'Done'
```

 300 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



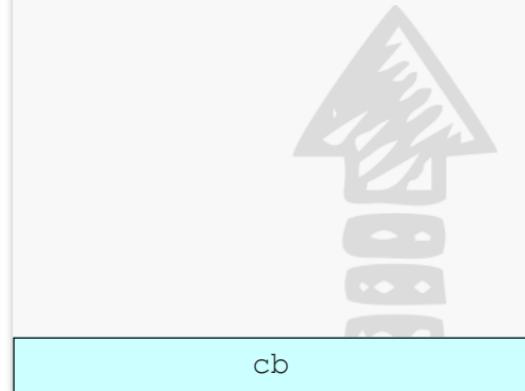
Console

```
'Start'  
'Done'  
'Done'
```

 300 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
}, 100);
```

Стек вызовов



Очередь событий



Console

```
'Start'  
'Done'  
'Done'
```

 300 ms

```
function hardWork() {  
    // этот код выполняется 50 мс  
}  
  
console.log('Start');  
  
setTimeout(function cb() {  
    hardWork();  
    console.log('Done');  
    setTimeout(cb, 100);  
, 100);
```

Стек вызовов

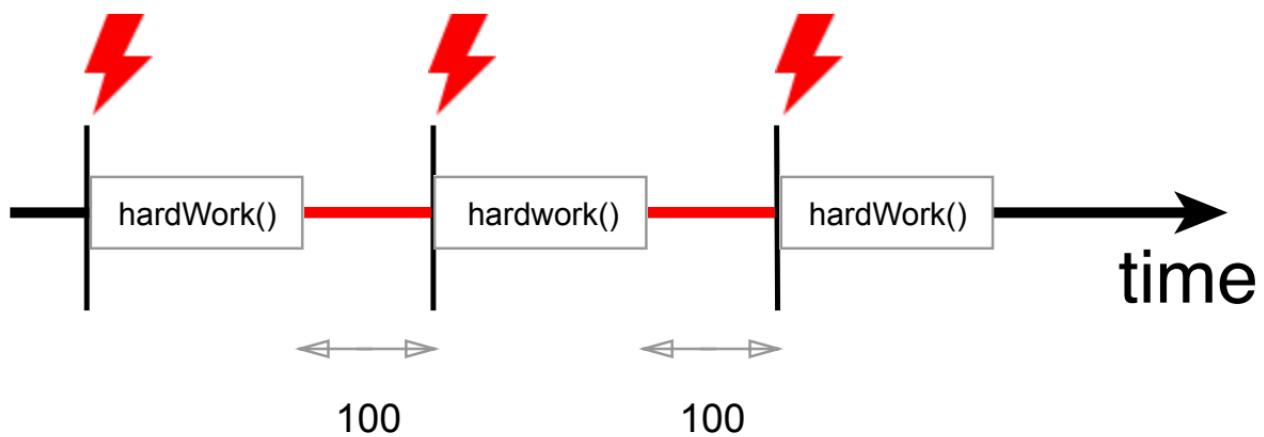


Очередь событий



Console

```
'Start'  
'Done'  
'Done'
```



# ПЕРЕРЫВ

# **НУЛЕВАЯ ЗАДЕРЖКА**

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

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

Стек вызовов



Очередь событий



Console

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

Стек вызовов



IIFE



Очередь событий



Console



0 sec

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

Стек вызовов



setTimeout

IIFE



Очередь событий



Console



0 sec

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

Стек вызовов



IIFE

beHappy



Очередь событий



Console

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

Стек вызовов



console.log

IIFE

beHappy



Очередь событий



Console

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

Стек вызовов



IIFE

beHappy



Очередь событий



Console

'Много-много дел'

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

Стек вызовов



beHappy



Очередь событий



Console

'Много-много дел'

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

Стек вызовов



beHappy



Очередь событий



Console

'Много-много дел'

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

Стек вызовов



Очередь событий



Console

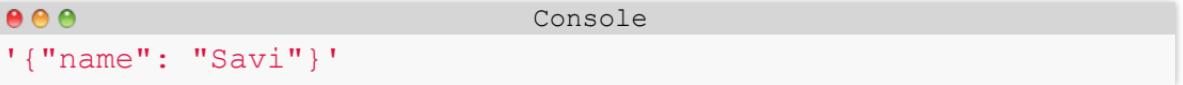
'Много-много дел'  
'Насладиться результатом'

# **РАБОТА С ФАЙЛАМИ**

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

const data = fs.readFileSync(fileName, 'utf-8');

console.log(data);
```



A screenshot of a terminal window titled "Console". The window has three small colored icons (red, yellow, green) in the top-left corner. The text area contains the command "console.log(data);" followed by its output: '{"name": "Savi"}'.

```
Console
'{"name": "Savi"}'
```

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

const start = Date.now();

const data = fs.readFileSync(fileName, 'utf-8');
console.log(`readFileSync: ${Date.now() - start}ms`);

console.log(data);
```



Console

```
readFileSync: 4ms
'{"name": "Savi"}'
```

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

const fileName = __dirname + '/data.json';

console.time('readFileSync');
const data = fs.readFileSync(fileName, 'utf-8');
console.timeEnd('readFileSync');

console.log(data);
```



Console

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

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

const fileName = __dirname + '/bigData.json';

console.time('readFileSync');
const data = fs.readFileSync(fileName, 'utf-8');
console.timeEnd('readFileSync');

console.log(data);
```



Console

```
readFileSync: 3567ms
<Много-много текста>
```

Во время синхронных операций не обрабатываются другие события: таймеры, пользовательские события и т.п.



```
const fs = require('fs');
const fileName = __dirname + '/data.json';

const data = fs.readFile(fileName, 'utf-8');

console.log(data);
```



Console

```
undefined
```

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



IIFE



Очередь событий



Console

# API

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
  console.log(data);
});
```

Стек вызовов



fs.readFile

IIFE



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



IIFE



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



cb



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



cb



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

## Стек вызовов



console.log

cb



Очередь событий



Console

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



cb



Очередь событий



Console

```
'{"name": "Savi"}'
```

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function cb(err, data) {
    console.log(data);
});
```

Стек вызовов



Очередь событий



Console

'{"name": "Savi"}'

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFileSync(fileName1, 'utf-8');
fs.readFileSync(fileName2, 'utf-8');
fs.readFileSync(fileName3, 'utf-8');
```

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

console.time('readFileSync')
fs.readFileSync(fileName1, 'utf-8');
fs.readFileSync(fileName2, 'utf-8');
fs.readFileSync(fileName3, 'utf-8');
console.timeEnd('readFileSync')
```



Console

```
readFileSync: 9217ms
```

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

### Стек вызовов



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

### Стек вызовов



IIFE



Очередь событий

# API

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



fs.readFile

IIFE



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



fs.readFile

IIFE



Очередь событий



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



IIFE



Очередь событий

# API



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



fs.readFile

IIFE



Очередь событий



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



Очередь событий

# API



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



cb2



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



cb2



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



cb3



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



cb3



Очередь событий

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

## Стек вызовов



Очередь событий

## API



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



cb1



Очередь событий

# API



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



cb1



Очередь событий

## API



```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

Стек вызовов



Очередь событий

# ПРОБЛЕМА

```
const fs = require('fs');
const fileName1 = __dirname + '/data_1.json';
const fileName2 = __dirname + '/data_2.json';
const fileName3 = __dirname + '/data_3.json';

fs.readFile(fileName1, 'utf-8', cb1); // читается 5 секунд
fs.readFile(fileName2, 'utf-8', cb2); // читается 1 секунду
fs.readFile(fileName3, 'utf-8', cb3); // читается 3 секунды
```

cb2

cb3

cb1

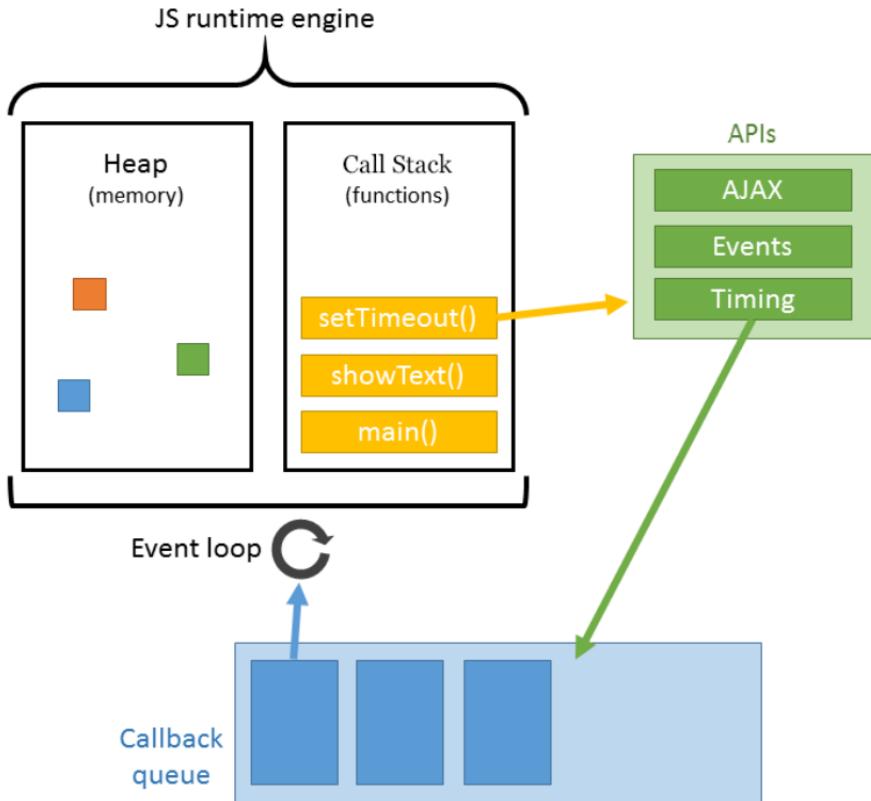


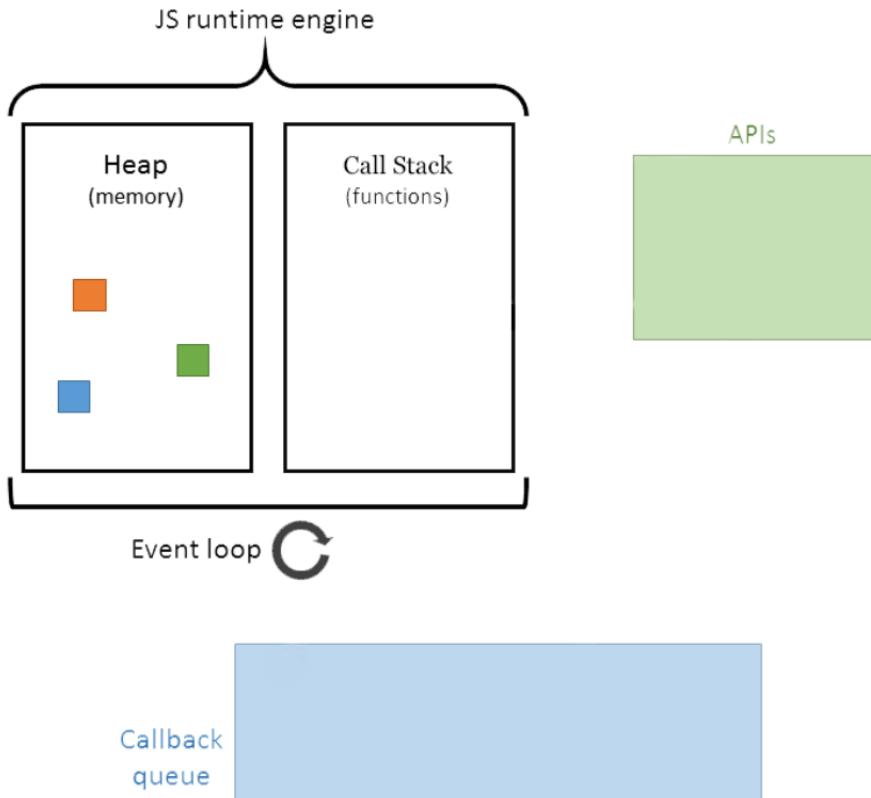
Очередь событий

# **EVENT LOOP**

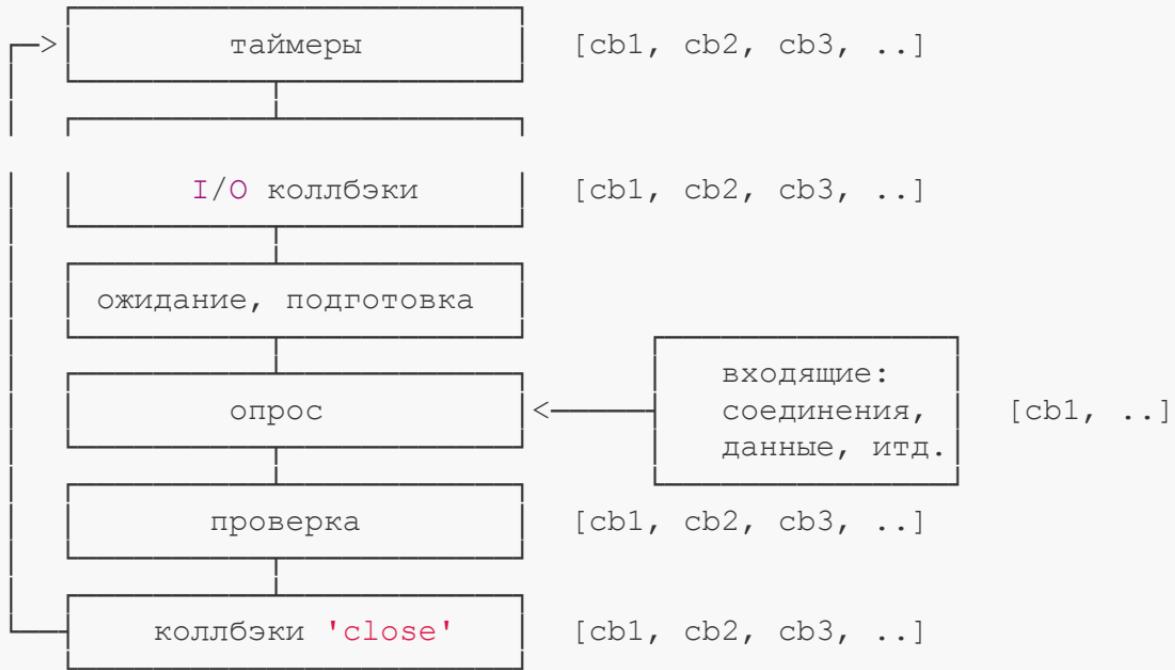
Цикл событий (Event Loop) — это то, что позволяет Node.js выполнять блокирующие операции ввода/вывода путем выгрузки операций в ядро системы, не блокируя основной поток.

```
while (queue.waitForMessage()) {  
    queue.processNextMessage();  
}
```





Если стек вызовов и очередь событий пусты, и при этом не планировалось никаких асинхронных операций, то программа успешно завершается



## **ФАЗА "ТАЙМЕРЫ"**

setTimeout

setInterval

## ФАЗА "I/O КОЛЛБЭКИ"

```
const fs = require('fs');
const fileName = __dirname + '/data.json';

fs.readFile(fileName, 'utf-8', function (err, data) {
  console.log(data);
});
```



Console

```
'{"engine": "V8"}'
```

# ПОЧИТАТЬ

- Параллельная модель и цикл событий
- setTimeout и setInterval
- Асинхронность в JavaScript: Пособие для тех, кто хочет разобраться
- Знай свой инструмент: Event Loop в libuv
- The Node.js Event Loop, Timers, and process.nextTick()

# **ПОСМОТРЕТЬ**

- Про цикл событий в JavaScript или "как на самом деле работает асинхронность"?

# ВОПРОСЫ?

Сальвадор Дали «Постоянство времени»