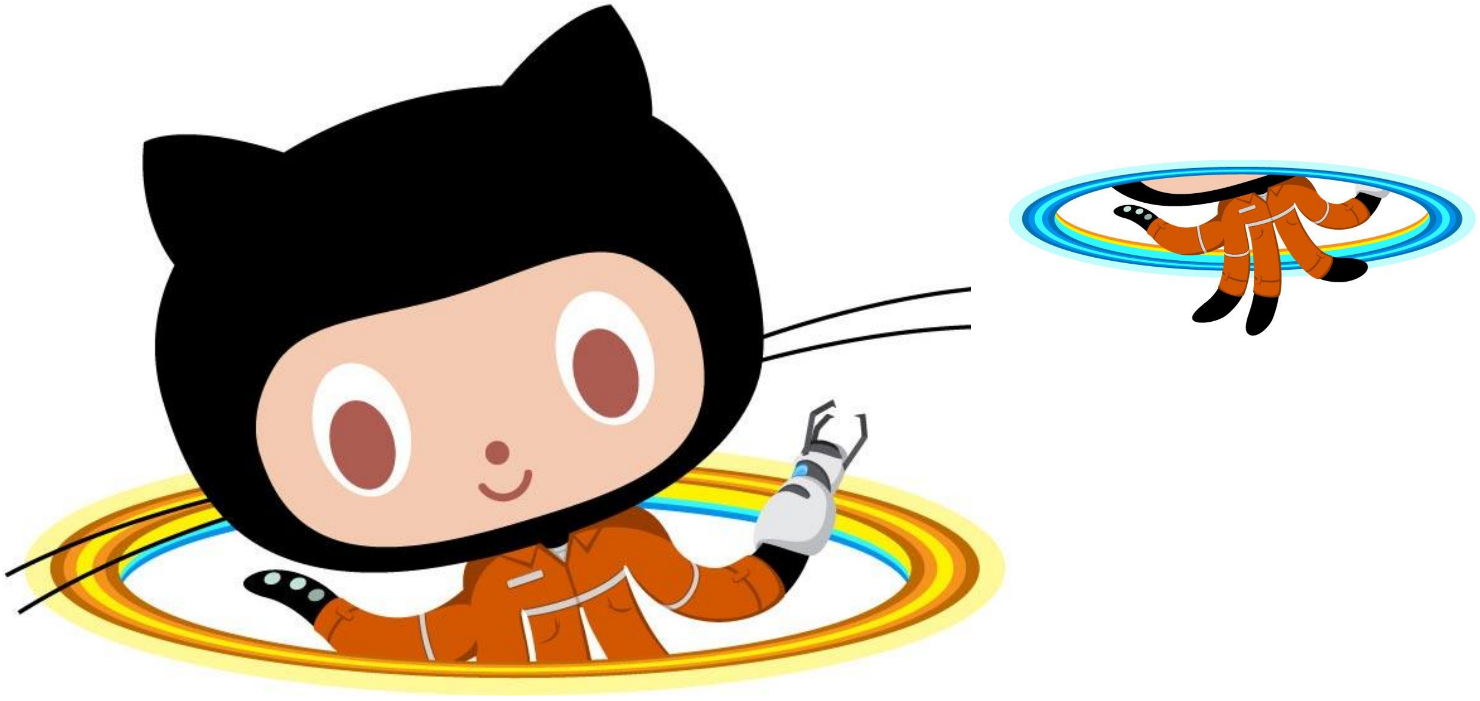


Лекция 3.5
асинхронность

Tinkoff.ru

План занятия

– обработка событий

– event loop

– пример

– callback

– promise

Работа кода

```
const btn = document.getElementById('button');

btn.onclick = function () {
  console.log('Привет!');
};
```

БРАУЗЕР ОЖИДАЕТ СОБЫТИЯ

События

События бывают:

– пользовательские (клики, скролл, ввод)

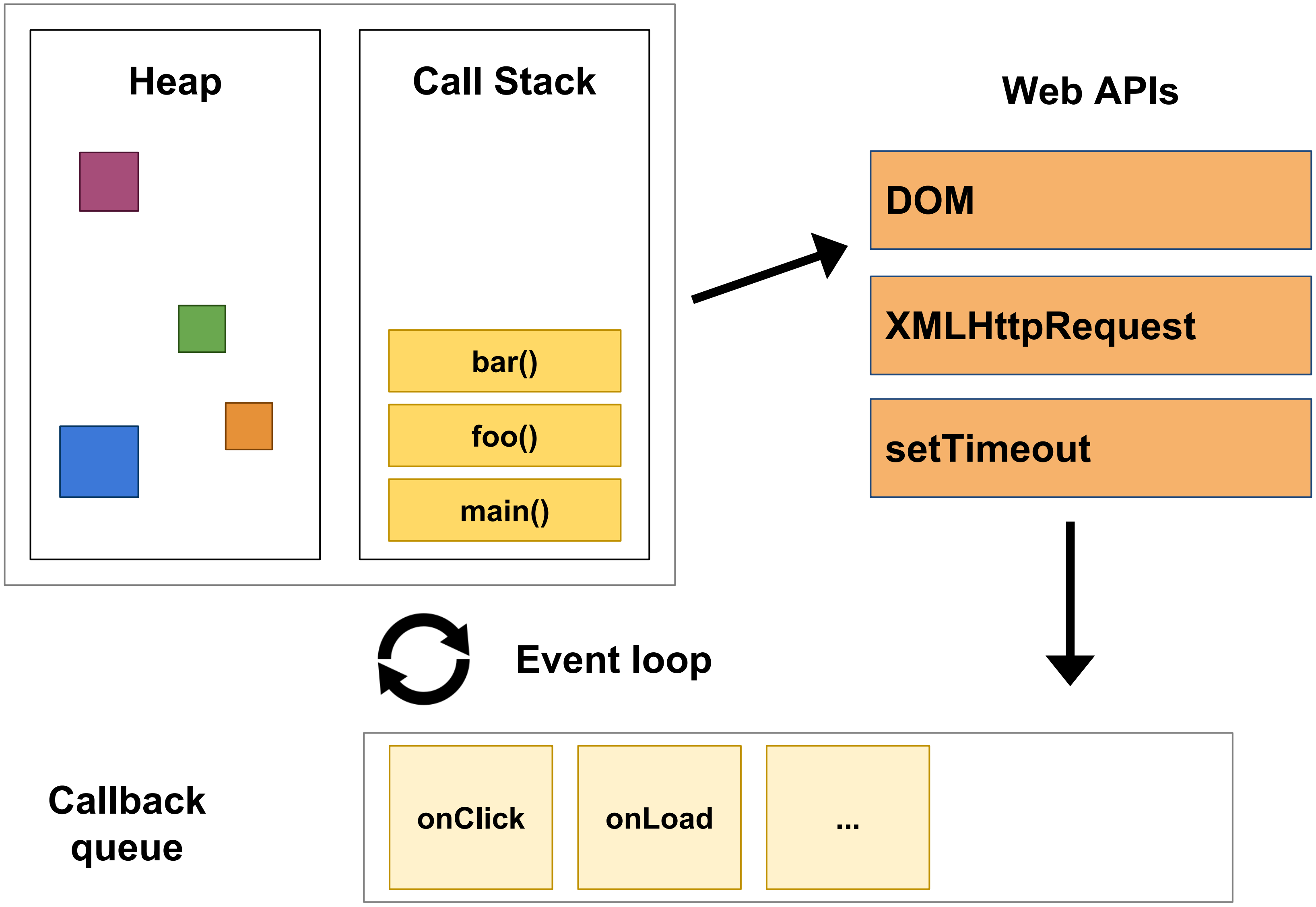
– браузерные (загрузка страницы, запросы к серверу)

– кастомные (изменение в модели данных)

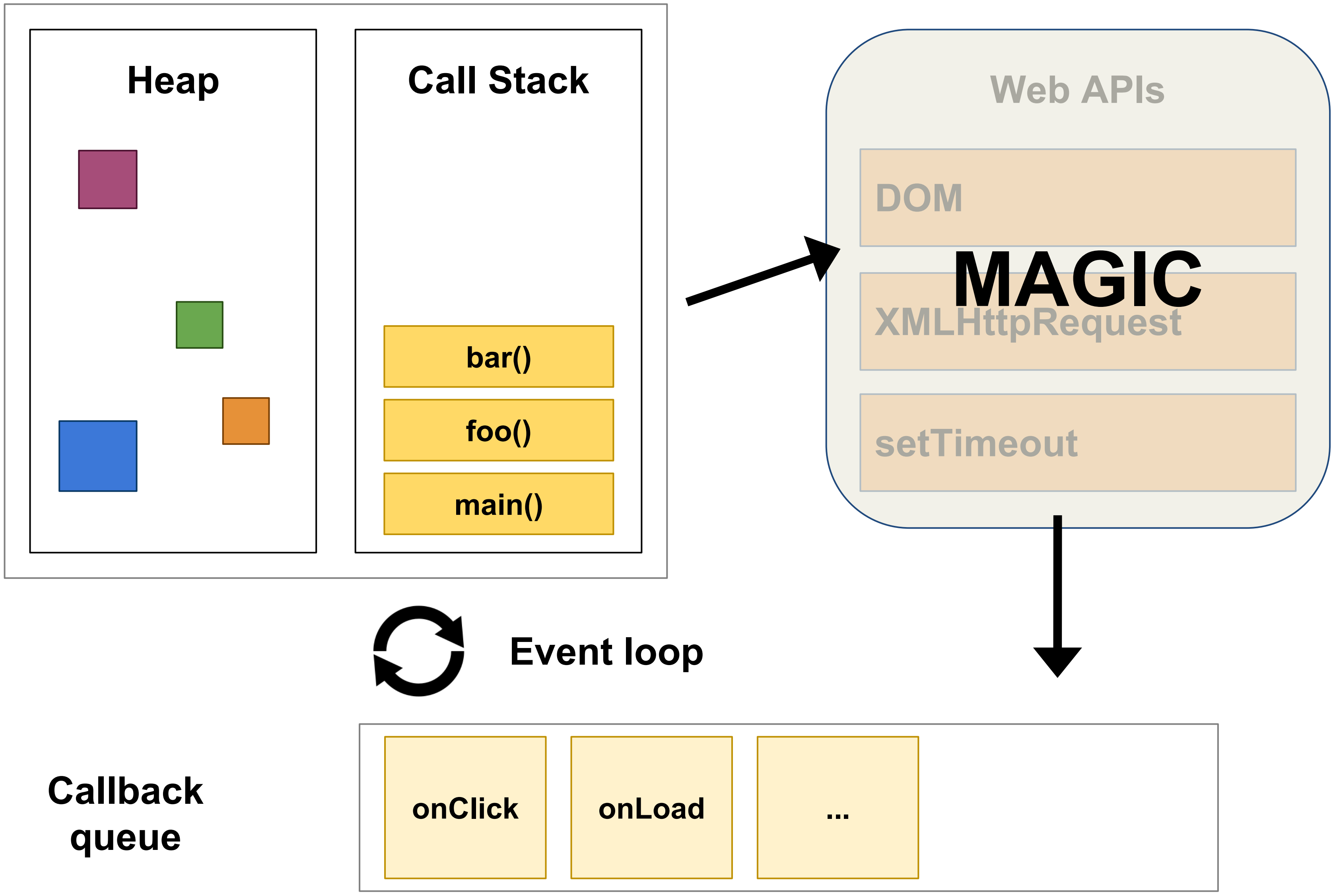
EVENT LOOP
(цикл событий)

https://www.youtube.com/watch?v=8cV4ZvHXQL4

Как работает JS

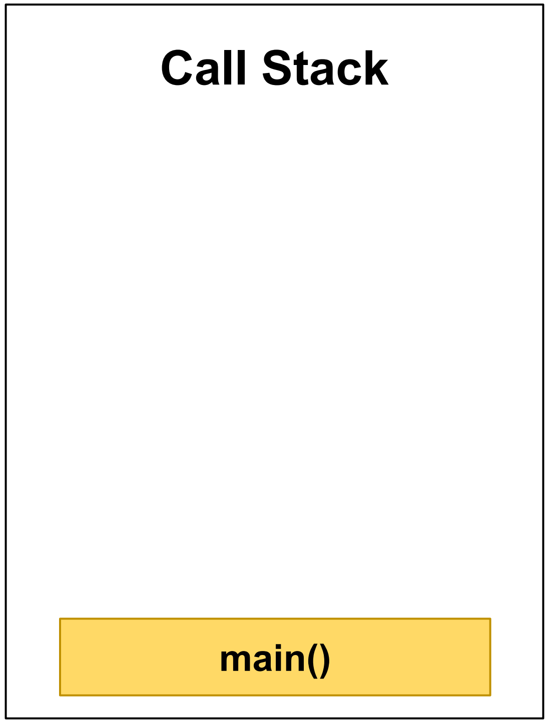


Как работает JS



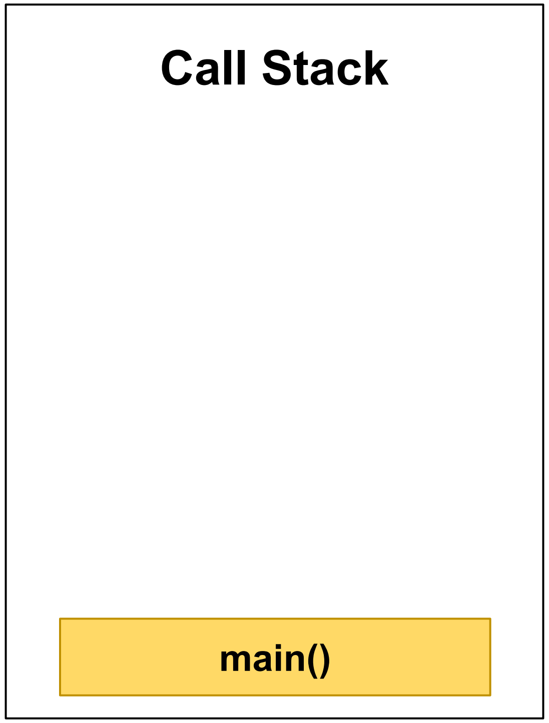
Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   var squared = square(x);
   console.log(squared);
}

printSuared(4);
```

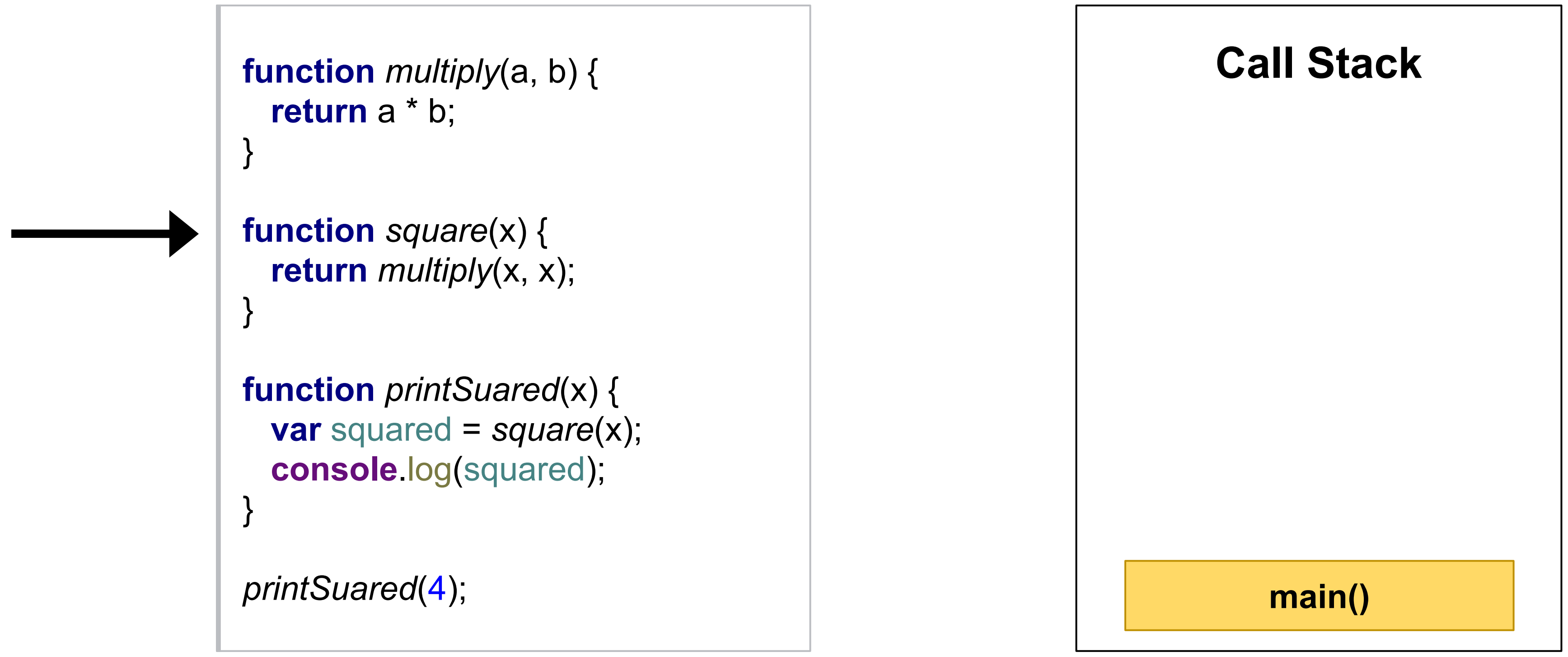


Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   var squared = square(x);
   console.log(squared);
}

printSuared(4);
```



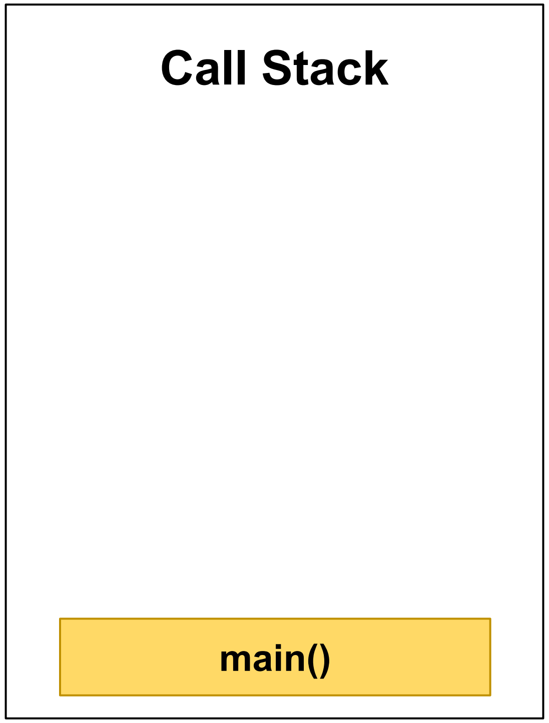
Call Stack



```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   var squared = square(x);
   console.log(squared);
}

printSuared(4);
```

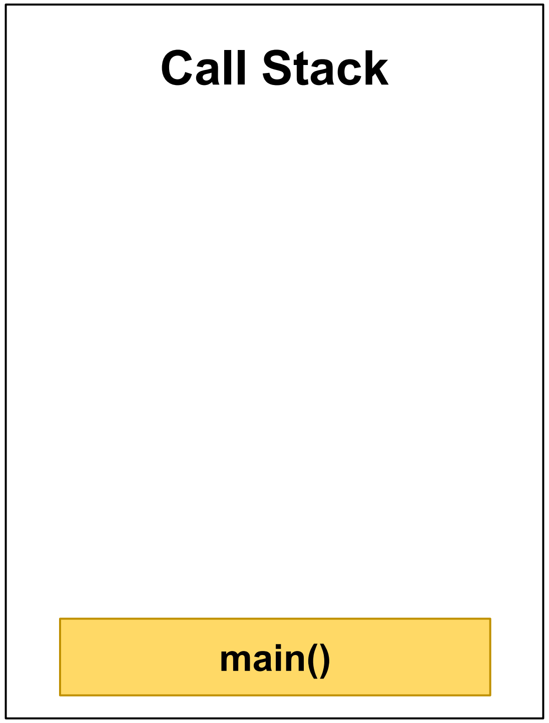
Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   const squared = square(x);
   console.log(squared);
}

printSuared(4);
```



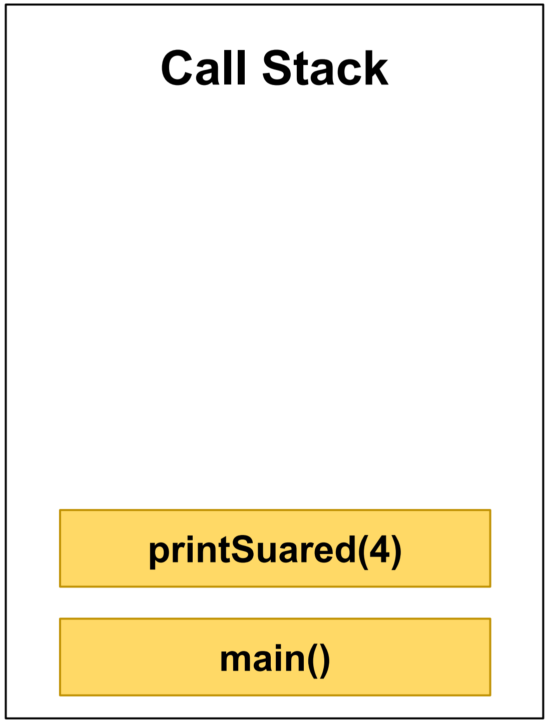
Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   const squared = square(x);
   console.log(squared);
}

printSuared(4);
```



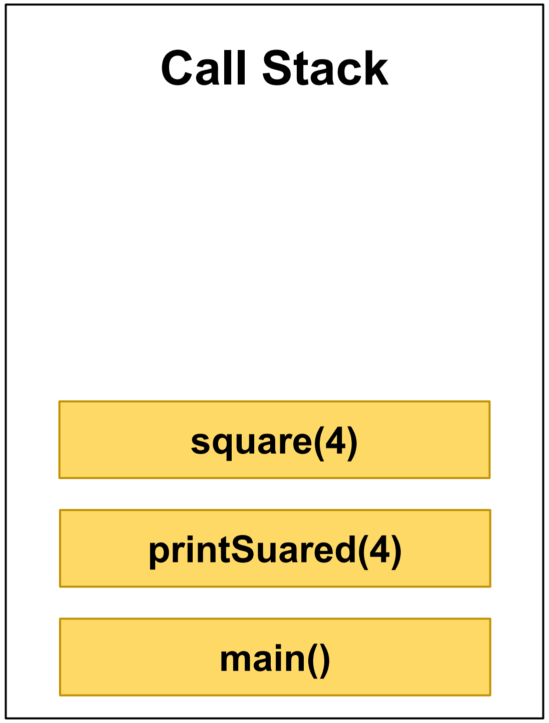
Call Stack

```
function multiply(a, b) {
  return a * b;
}

function square(x) {
  return multiply(x, x);
}

function printSuared(x) {
  const squared = square(x);
  console.log(squared);
}

printSuared(4);
```



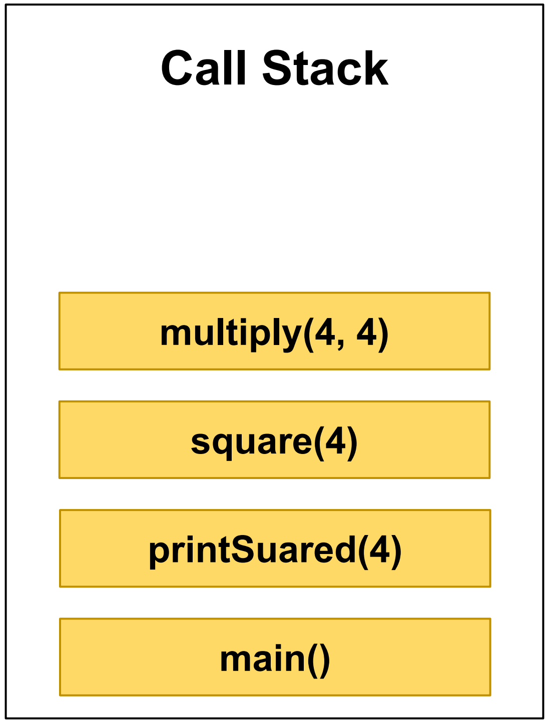
Call Stack

```
function multiply(a, b) {
  return a * b;
}

function square(x) {
  return multiply(x, x);
}

function printSuared(x) {
  const squared = square(x);
  console.log(squared);
}

printSuared(4);
```



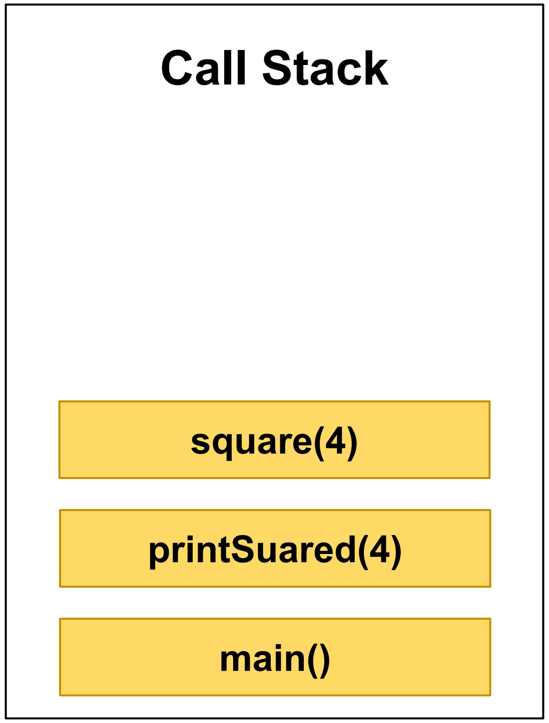
Call Stack

```
function multiply(a, b) {
  return a * b;
}

function square(x) {
  return multiply(x, x);
}

function printSuared(x) {
  const squared = square(x);
  console.log(squared);
}

printSuared(4);
```



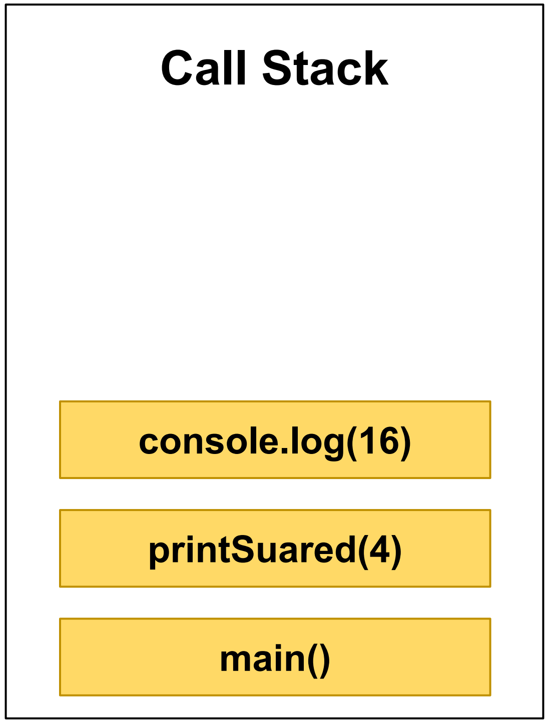
Call Stack

```
function multiply(a, b) {
  return a * b;
}

function square(x) {
  return multiply(x, x);
}

function printSuared(x) {
  const squared = square(x);
  console.log(squared);
}

printSuared(4);
```



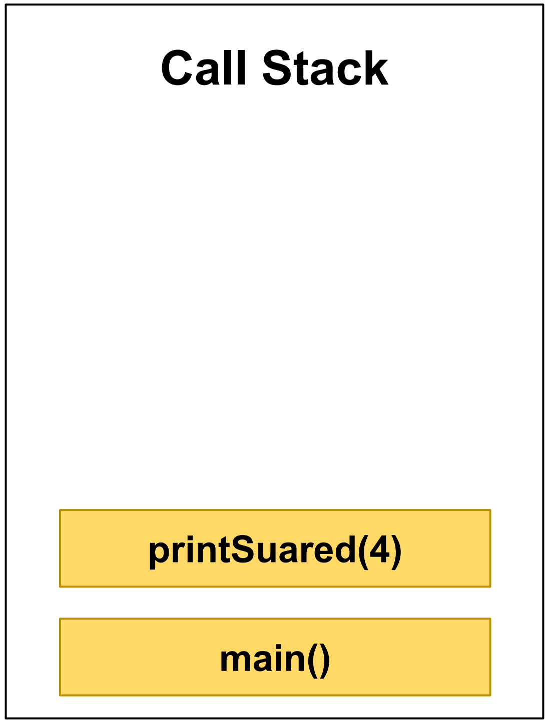
Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   const squared = square(x);
   console.log(squared);
}

printSuared(4);
```



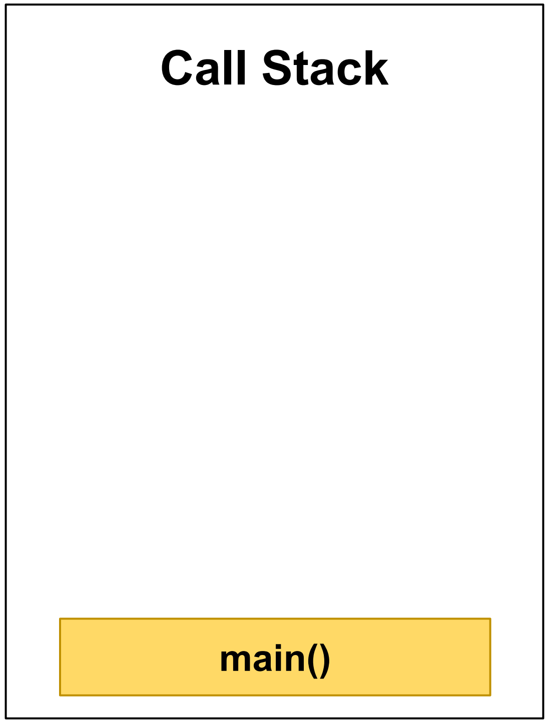
Call Stack

```
function multiply(a, b) {
  return a * b;
}

function square(x) {
  return multiply(x, x);
}

function printSuared(x) {
  const squared = square(x);
  console.log(squared);
}

printSuared(4);
```



Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   const squared = square(x);
   console.log(squared);
}

printSuared(4);
```

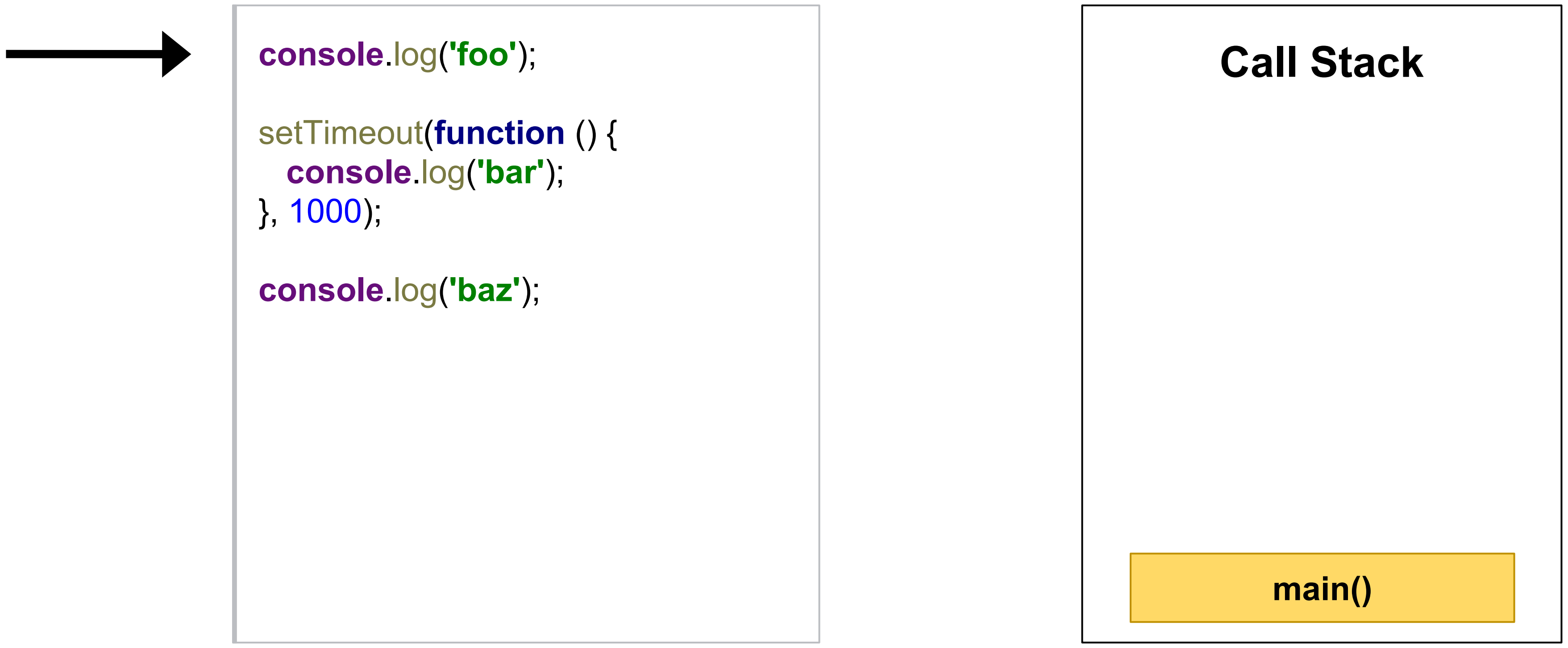


Call Stack

```
function multiply(a, b) {
   return a * b;
}

function square(x) {
   return multiply(x, x);
}

function printSuared(x) {
   const squared = square(x);
   console.log(squared);
}

printSuared(4);
```

Call Stack

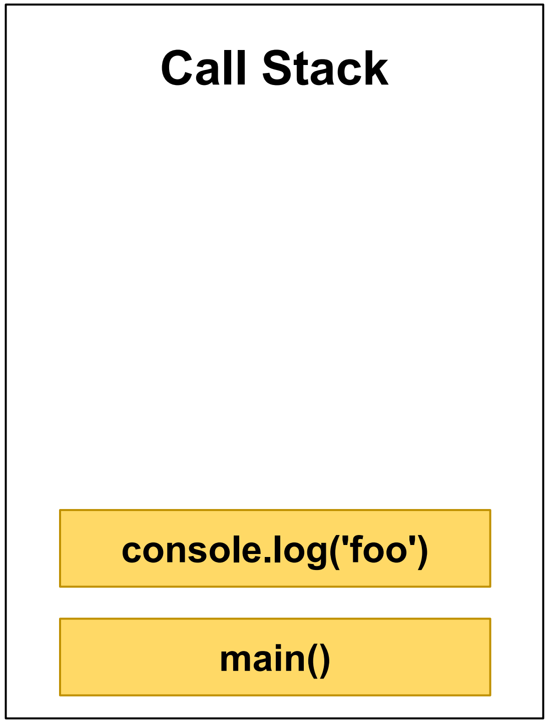
Асинхронный Call Stack



```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```

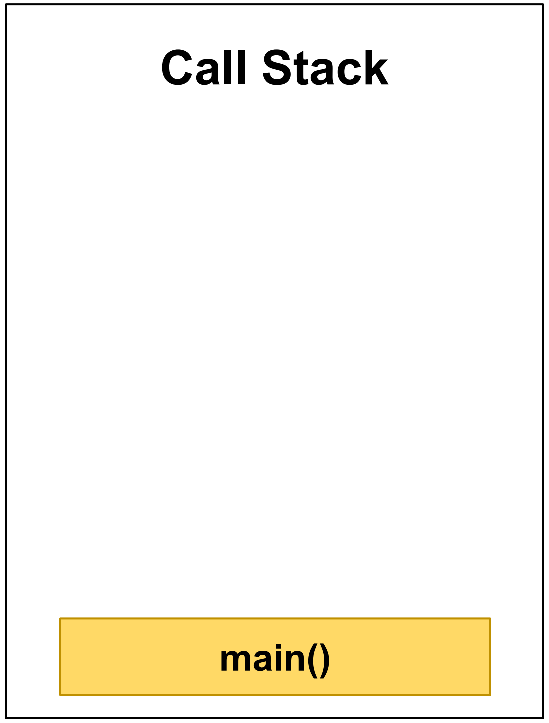
Асинхронный Call Stack

```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```



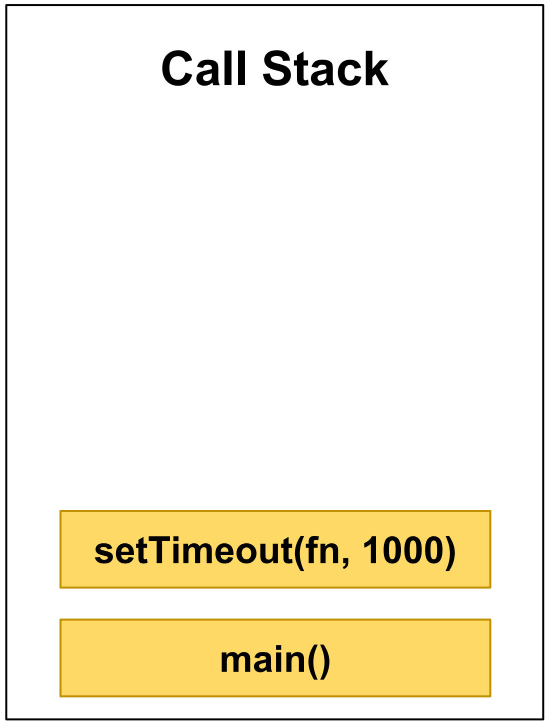
Асинхронный Call Stack

```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```



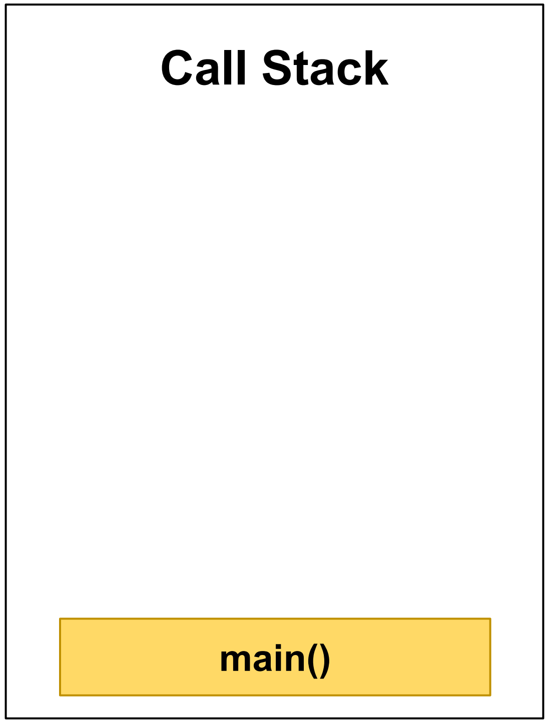
Асинхронный Call Stack

```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```



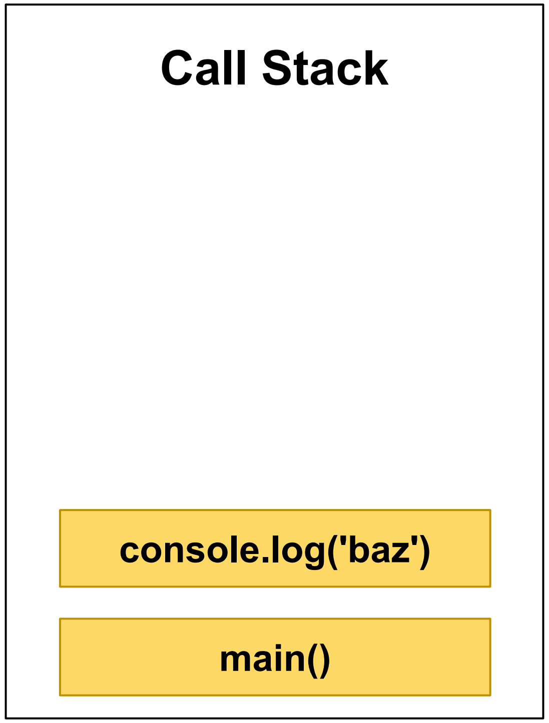
Асинхронный Call Stack

```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```

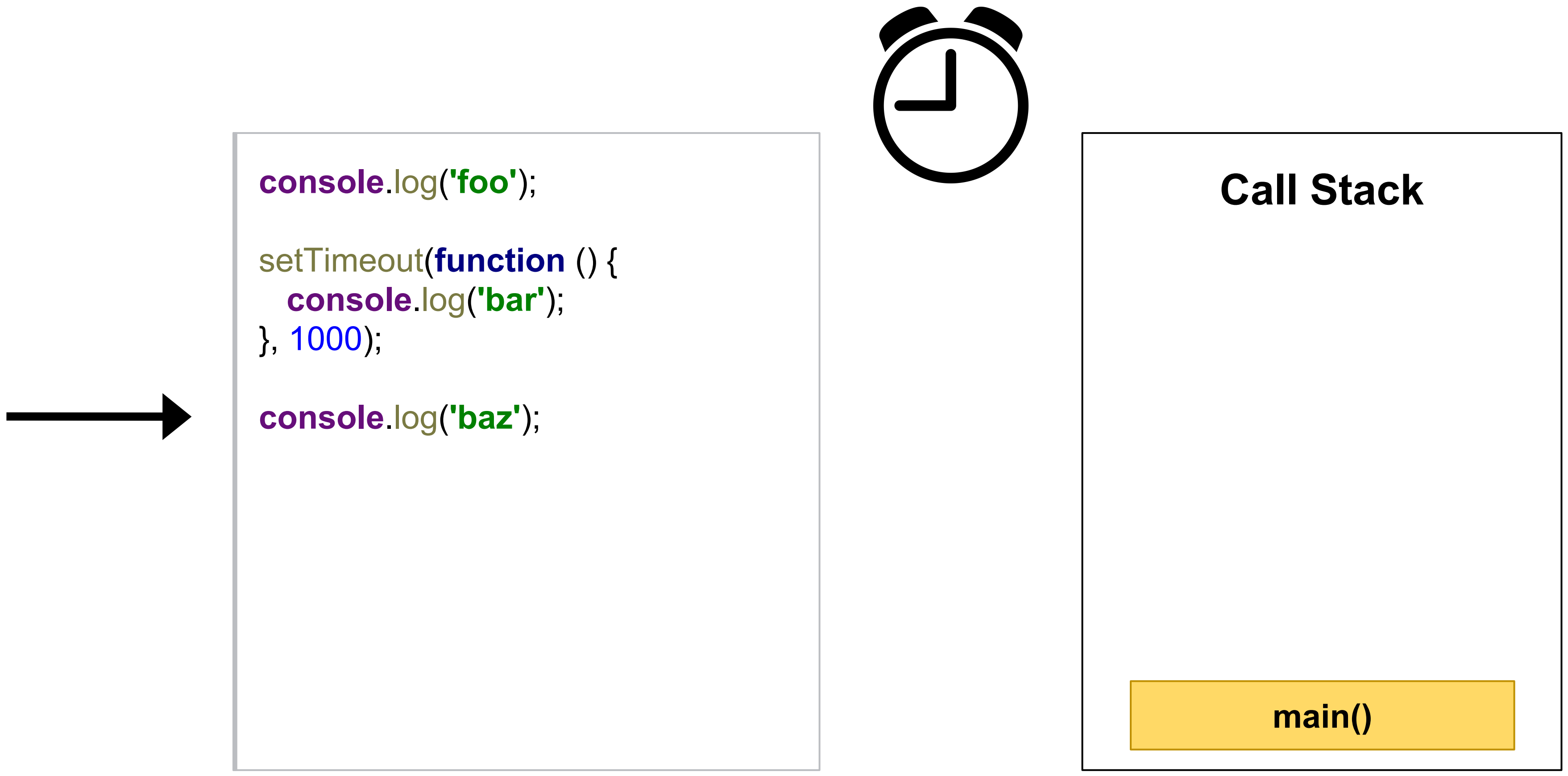


Асинхронный Call Stack

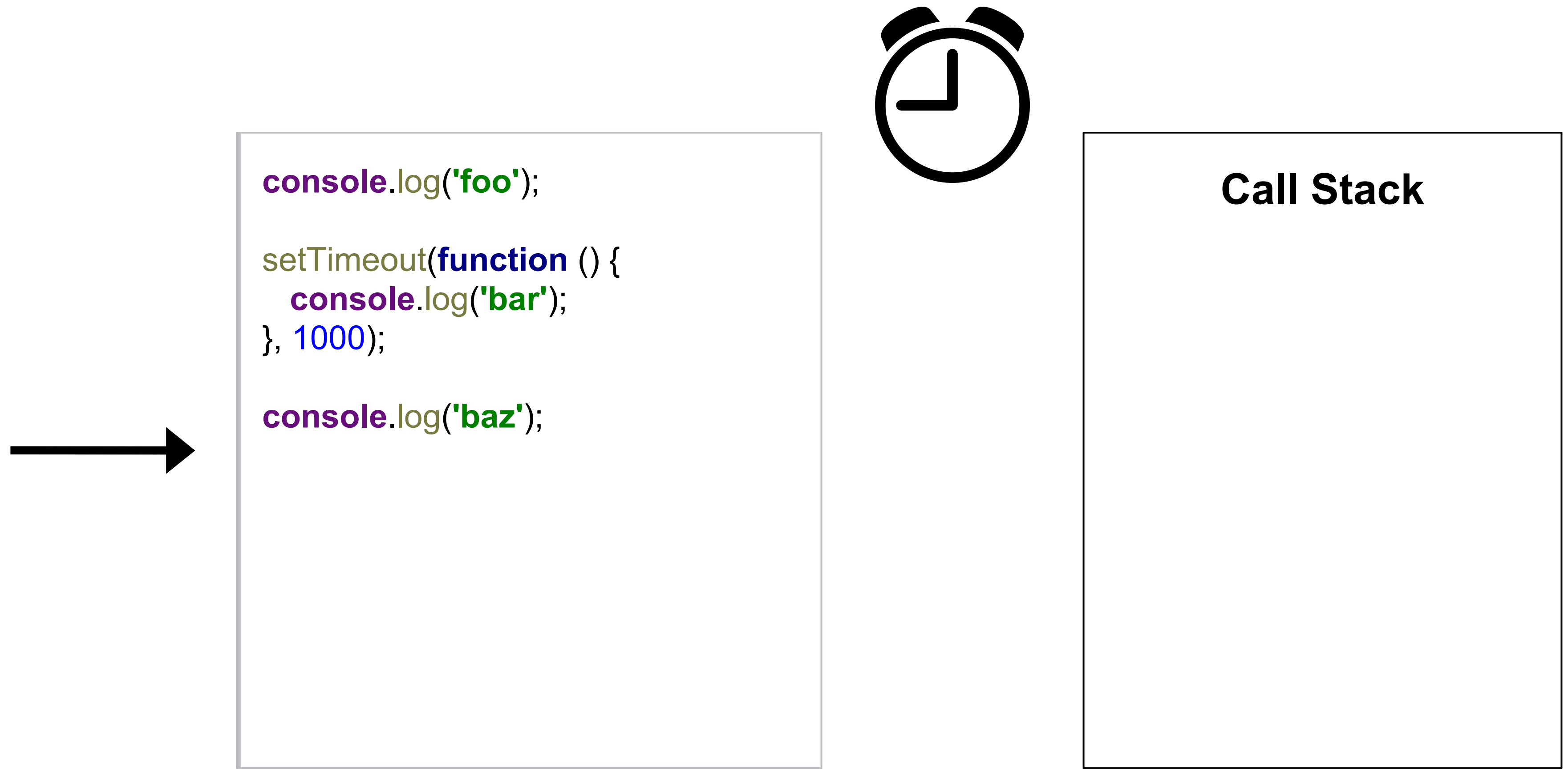
```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```



Асинхронный Call Stack


```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```

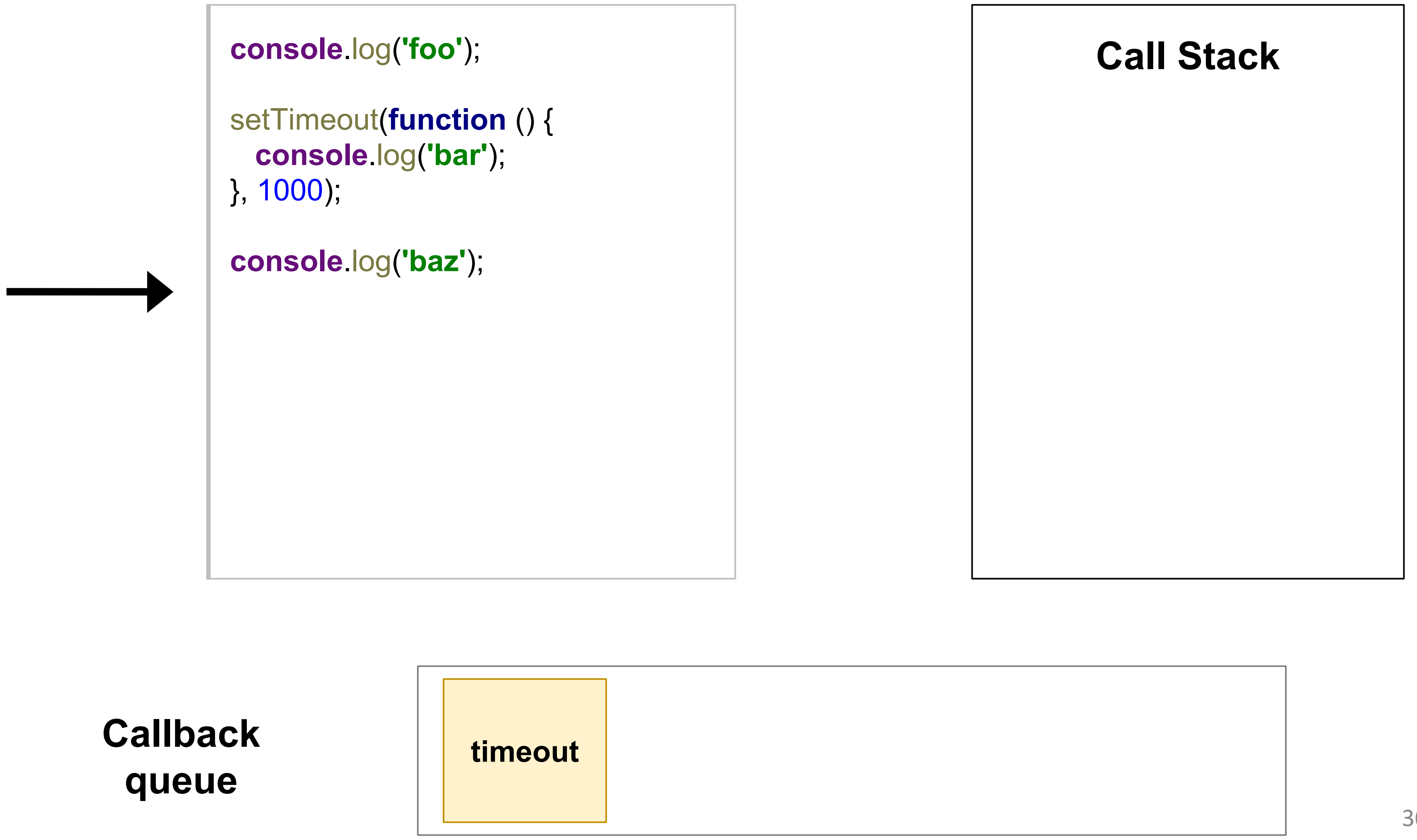
Асинхронный Call Stack



```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```

Асинхронный Call Stack

```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```



Асинхронный Call Stack

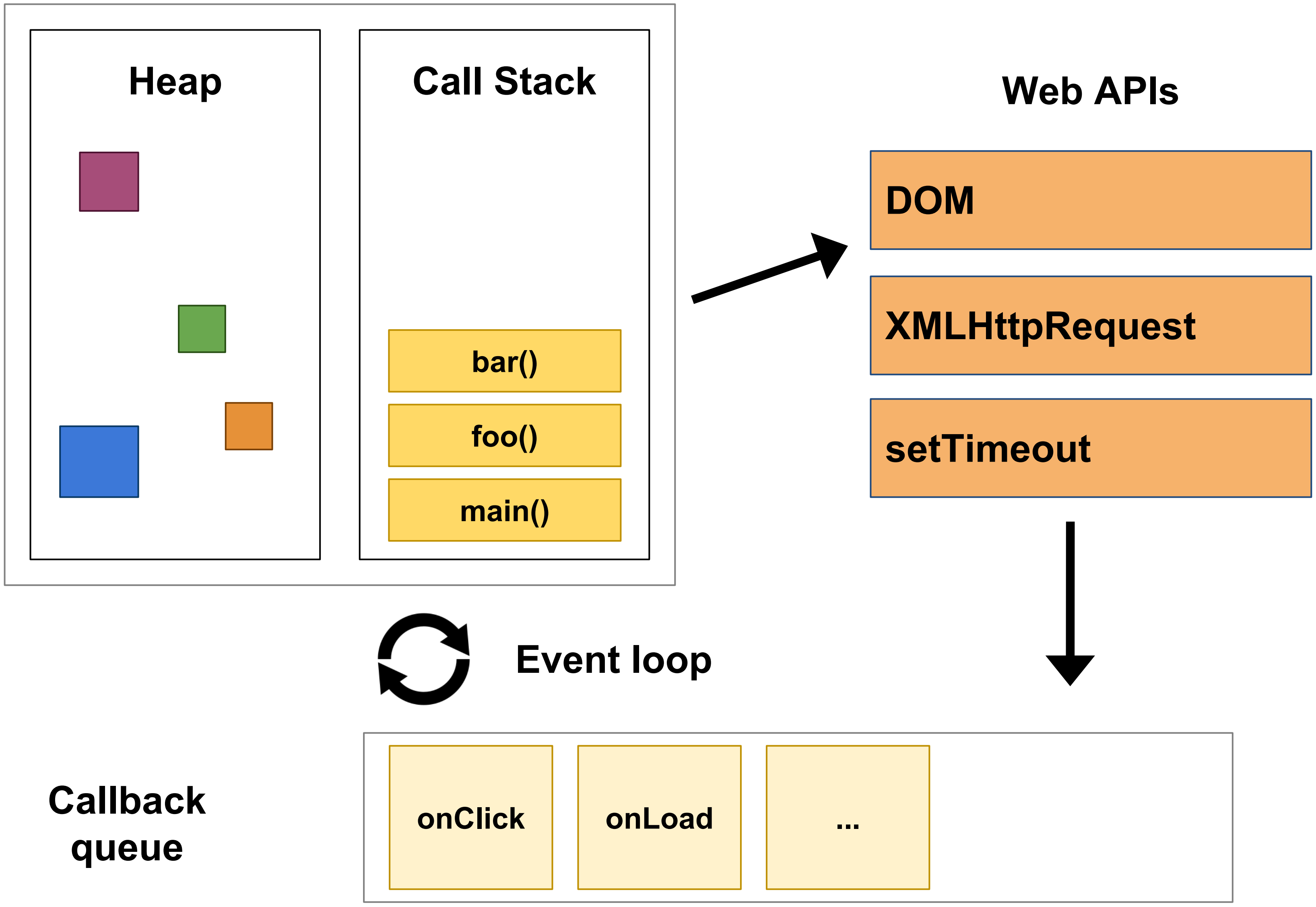
```
console.log('foo');

setTimeout(function () {
   console.log('bar');
}, 1000);

console.log('baz');
```

Call Stack

console.log('bar')

Callback queue

Как работает JS



JAVASCRIPT АСИНХРОННЫЙ

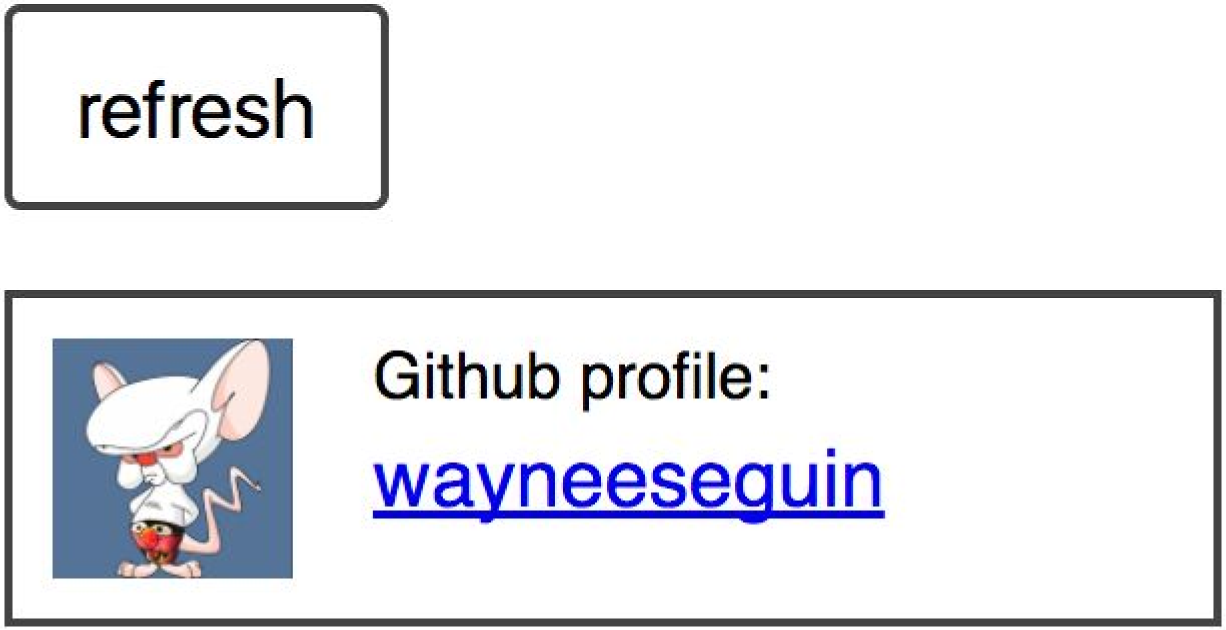
JAVASCRIPT АСИНХРОННЫЙ, ОДНОПОТОЧНЫЙ

JAVASCRIPT АСИНХРОННЫЙ, ОДНОПОТОЧНЫЙ

но это неточно. Воркеры

ПРИМЕР

Пример: случайный пользователь



Github profile:
wayneeseguin

XHR
(*древнегреческ.* XMLHttpRequest)

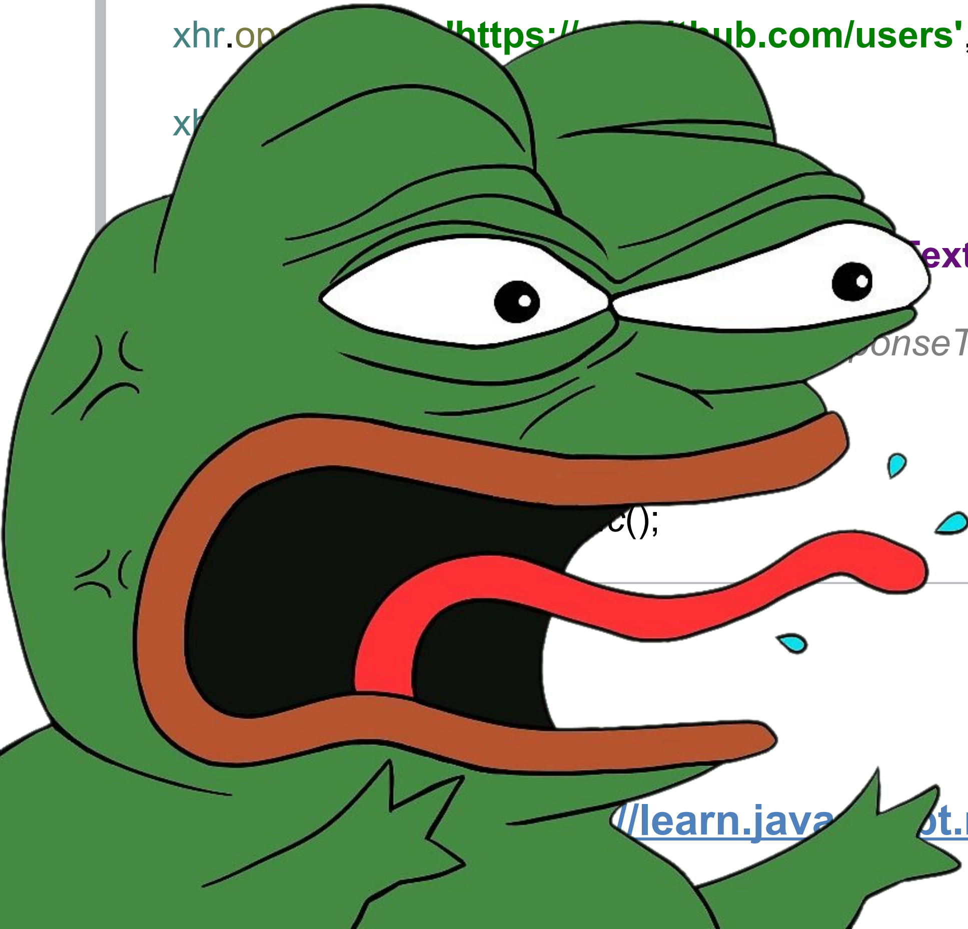
Синхронный XHR

```
function getRandomGithubUsersSync() {
  const xhr = new XMLHttpRequest();

  xhr.open('GET', 'https://api.github.com/users', false);

  xhr.send();

  if (xhr.status != 200) {
     console.log(xhr.status + ': ' + xhr.statusText); // 404: Not Found
  } else {
     console.log(xhr.responseText); // responseText -- текст ответа.
  }
}

getRandomGithubUsersSync();
```

https://learn.javascript.ru/ajax-xmlhttprequest

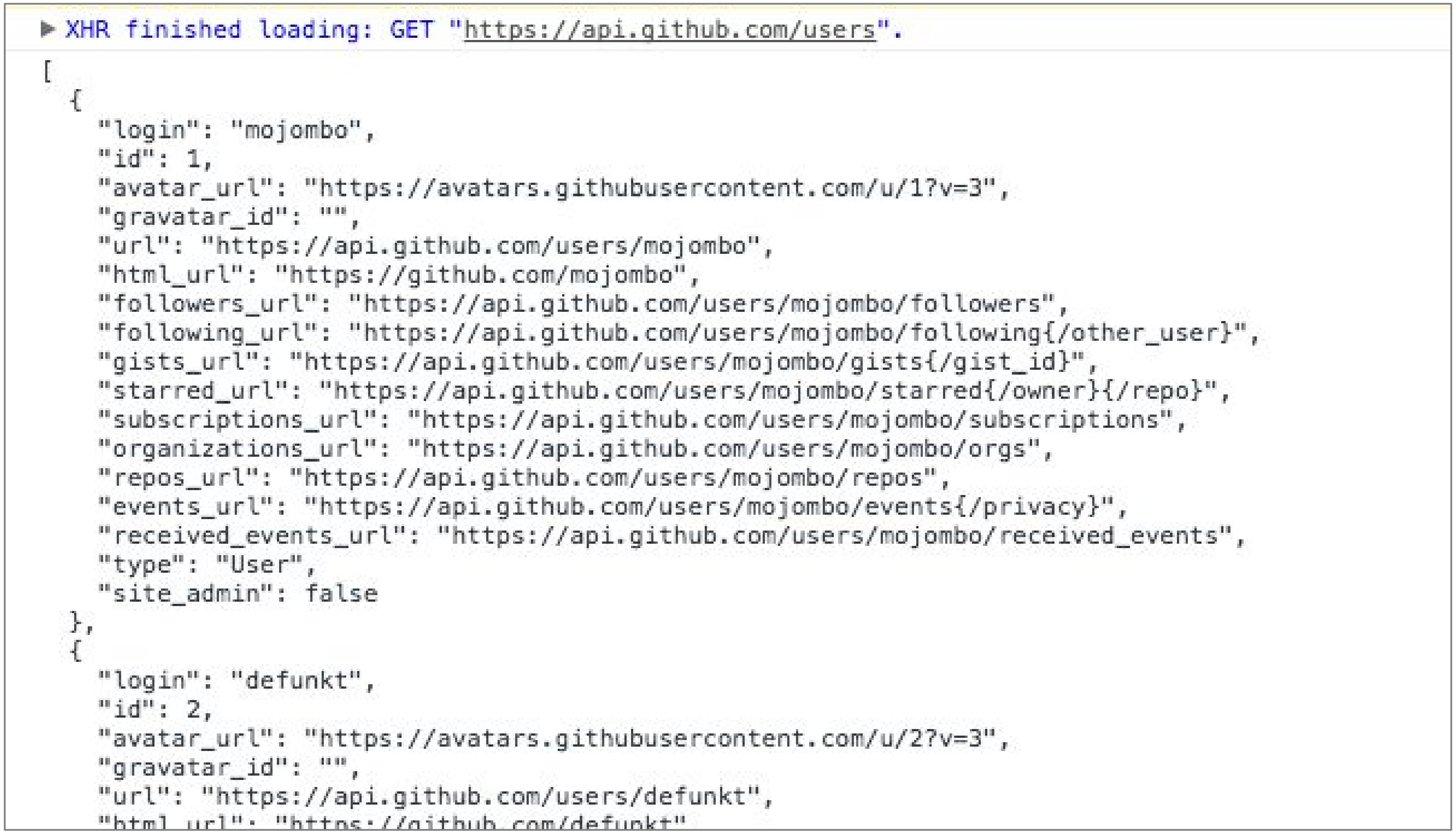
Синхронный XHR

```
function getRandomGithubUsersSync() {
  const xhr = new XMLHttpRequest();

  xhr.op          'https://      ub.com/users', false);

  xh



                                   ext); // 404: Not Found

                                   ponseText -- текст ответа.




                   c();
```



//learn.java   t.ru/ajax-xmlhttprequest

Синхронный XHR



```
▶ XHR finished loading: GET "https://api.github.com/users".
[
  {
    "login": "mojombo",
    "id": 1,
    "avatar_url": "https://avatars.githubusercontent.com/u/1?v=3",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mojombo",
    "html_url": "https://github.com/mojombo",
    "followers_url": "https://api.github.com/users/mojombo/followers",
    "following_url": "https://api.github.com/users/mojombo/following{/other_user}",
    "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mojombo/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
    "organizations_url": "https://api.github.com/users/mojombo/orgs",
    "repos_url": "https://api.github.com/users/mojombo/repos",
    "events_url": "https://api.github.com/users/mojombo/events{/privacy}",
    "received_events_url": "https://api.github.com/users/mojombo/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "defunkt",
    "id": 2,
    "avatar_url": "https://avatars.githubusercontent.com/u/2?v=3",
    "gravatar_id": "",
    "url": "https://api.github.com/users/defunkt",
    "html_url": "https://github.com/defunkt"
```

Асинхронный XHR

```
function getRandomGithubUsersAsync() {
   const xhr = new XMLHttpRequest();
   xhr.open('GET', 'https://api.github.com/users', true);

   xhr.onreadystatechange = () => {
      if (xhr.readyState != 4) return;

      if (xhr.status != 200) {
         console.log(xhr.status + ': ' + xhr.statusText);
      } else {
         console.log(xhr.responseText);
      }
   };

   xhr.send();
}

getRandomGithubUsersAsync();
```

Асинхронный XHR

```
function getRandomGithubUsersAsync() {
  const xhr = new XMLHttpRequest();
  xhr.open('GET', 'https://api.github.com/users', true);

  xhr.onreadystatechange = () => {
    if (xhr.readyState != 4) return;

    if (xhr.status != 200) {
      console.log(xhr.status + ': ' + xhr.statusText);
    } else {
      console.log(xhr.responseText);
    }
  };

  xhr.send();
}

getRandomGithubUsersAsync();
```

readystatechange

UNSENT = 0	*начальное состояние*
OPENED = 1	*вызван* ***open***
HEADERS_RECEIVED = 2	*получены заголовки*
LOADING = 3	*загружается тело (получен очередной пакет данных)*
DONE = 4	*запрос завершён*

Асинхронный XHR

```
function getRandomGithubUsersAsync() {
   const xhr = new XMLHttpRequest();
   xhr.open('GET', 'https://api.github.com/users', true);

   xhr.onreadystatechange = () => {
       if (xhr.readyState != 4) return;

       if (xhr.status != 200) {
           console.log(xhr.status + ': ' + xhr.statusText);
       } else {
           console.log(xhr.responseText);
       }
   };

   xhr.send();
}

getRandomGithubUsersAsync();
```

Универсальный XHR

```
function makeGetRequest(url, successCallback, errorCallback) {
  const xhr = new XMLHttpRequest();
  xhr.open('GET', url, true);
  xhr.onreadystatechange = () => {
    if (xhr.readyState != 4) return;

    if (xhr.status != 200) {
      const error = new Error('Ошибка ' + xhr.status);
      error.code = xhr.statusText;
      errorCallback(error);
    } else {
      successCallback(xhr.responseText);
    }
  };

  xhr.send();
}
```

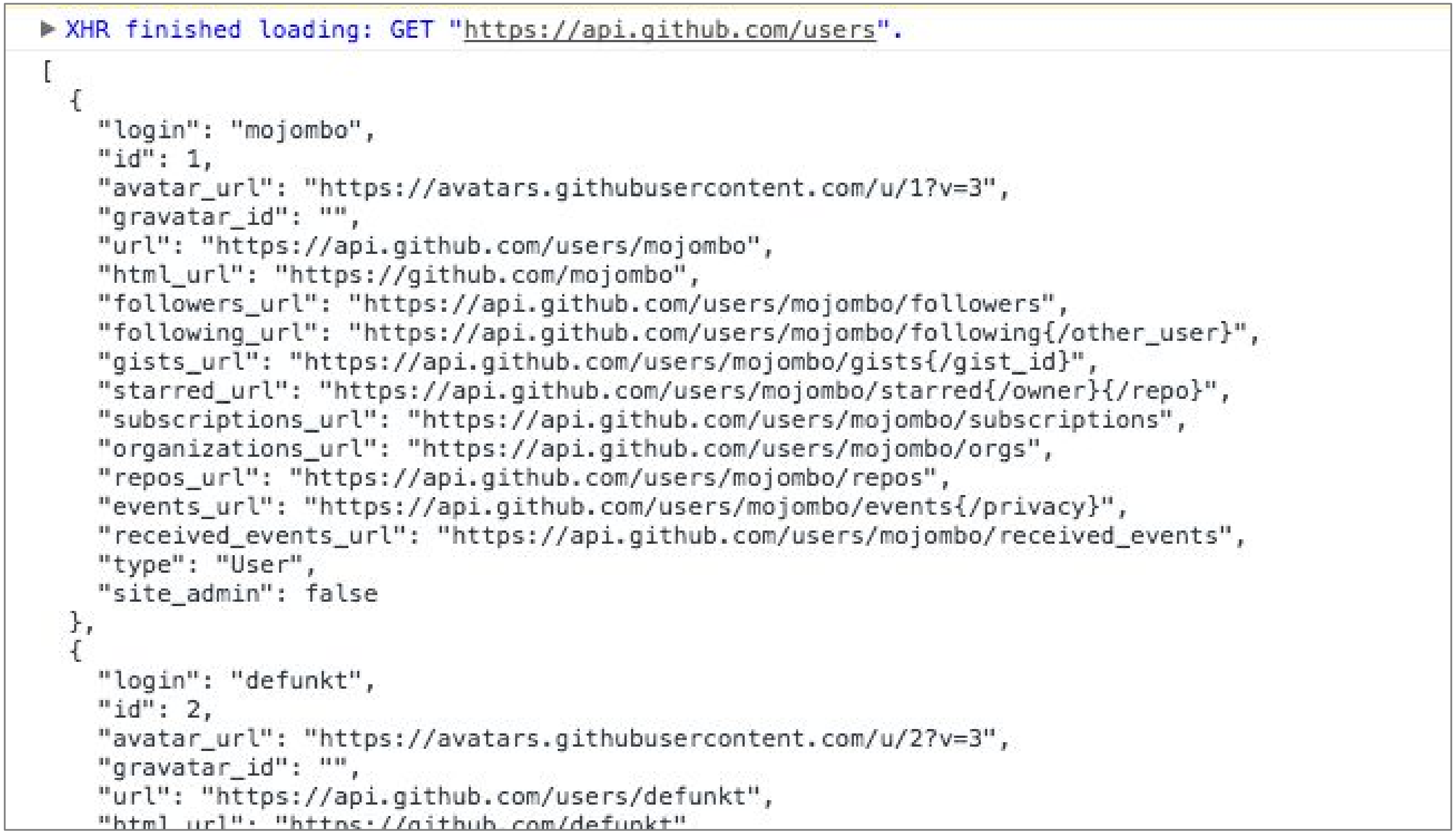
Вызовем makeGetRequest

```
makeRequest('https://api.github.com/users', (response) => {
  console.log(response);
}, (error) => {
  console.error(error);
});
```

Добавим клик

```
const randomButtonElement = document.getElementById('randomize');
randomButtonElement.onclick = () => {
   makeRequest('https://api.github.com/users', (data) => {
       console.log(data);
   }, (error) => {
       console.error(error);
   });
};
```

Асинхронный XHR



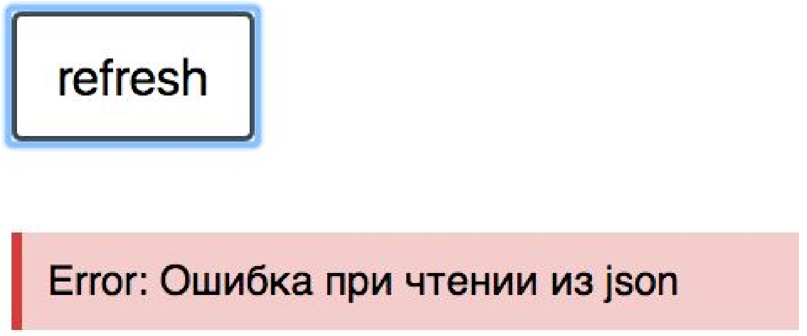
```
▶ XHR finished loading: GET "https://api.github.com/users".
[
  {
    "login": "mojombo",
    "id": 1,
    "avatar_url": "https://avatars.githubusercontent.com/u/1?v=3",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mojombo",
    "html_url": "https://github.com/mojombo",
    "followers_url": "https://api.github.com/users/mojombo/followers",
    "following_url": "https://api.github.com/users/mojombo/following{/other_user}",
    "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mojombo/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
    "organizations_url": "https://api.github.com/users/mojombo/orgs",
    "repos_url": "https://api.github.com/users/mojombo/repos",
    "events_url": "https://api.github.com/users/mojombo/events{/privacy}",
    "received_events_url": "https://api.github.com/users/mojombo/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "defunkt",
    "id": 2,
    "avatar_url": "https://avatars.githubusercontent.com/u/2?v=3",
    "gravatar_id": "",
    "url": "https://api.github.com/users/defunkt",
    "html_url": "https://github.com/defunkt"
```

JSON.parse

```
const randomButtonElement = document.getElementById('randomize');

randomButtonElement.onclick = () => {
  makeGetRequest('https://api.github.com/users',
      (request) => {
          let data;
          try {
              data = JSON.parse(request)
          } catch (err) {
              console.error(new Error('Ошибка при чтении из json'));
          }
          if (data) {
              console.log(data);
          }
      },
      (error) => {
          console.error(error);
      })
};
```

Добавим единый обработчик ошибок

```
const randomButtonElement = document.getElementById('randomize');
const errorElement = document.getElementById('error');
const randomUserElement = document.getElementById('user');

const showError = (err) => {
  errorElement.textContent = err;
  errorElement.classList.remove('hidden');
  randomUserElement.classList.add('hidden');
}

const hideError = () => {
  errorElement.classList.add('hidden');
  randomUserElement.classList.remove('hidden');
}
```



Error: Ошибка при чтении из json

Добавим единый обработчик ошибок

```
randomButtonElement.onclick = () => {
  makeGetRequest('https://api.github.com/users',
     (request) => {
        let data;
        try {
           data = JSON.parse(request)
        } catch (err) {
           showError(new Error('Ошибка при чтении из json'));
        }
        if (data) {
           console.log(data);
        }
     }, showError);
};
```

Выбор случайного пользователя

```
randomButtonElement.onclick = () => {
  makeGetRequest('https://api.github.com/users',
     (request) => {
        let data;
        try {
           data = JSON.parse(request)
        } catch (err) {
           showError(new Error('Ошибка при чтении из json'));
        }
        if (data) {
           const user = data[Math.floor(Math.random() * data.length)];
        }
     }, showError);
};
```

Отрисовка шаблона

```javascript
function drawUser(data) {
   const img =
         randomUserElement.querySelector('img');
   const link =
         randomUserElement.querySelector('a');
   img.src = data.avatar_url;
   img.alt = data.login;
   link.href = data.html_url;
   link.textContent = data.login;
}
```

```html
<div class="github-user" id="user">
   <img src="" alt="">
   Github profile:
   <a href=""></a>
</div>
```

Отрисовка шаблона

```
randomButtonElement.onclick = () => {
  makeGetRequest('https://api.github.com/users',
     (request) => {
        let data;
        try {
           data = JSON.parse(request)
        } catch (err) {
           showError(new Error('Ошибка при чтении из json'));
        }
        if (data) {
           const user = data[Math.floor(Math.random() * data.length)];
           drawUser(user);
        }
     }, showError);
};
```

Предзагрузка изображений

```
function loadImage(imageUrl, successCallback, errorCallback) {
   const img = new Image();

   img.onload = () => {
       successCallback(img);
   };

   img.onerror = () => {
       errorCallback(new Error('Что-то пошло не так'));
   };
   img.src = imageUrl;
}
```

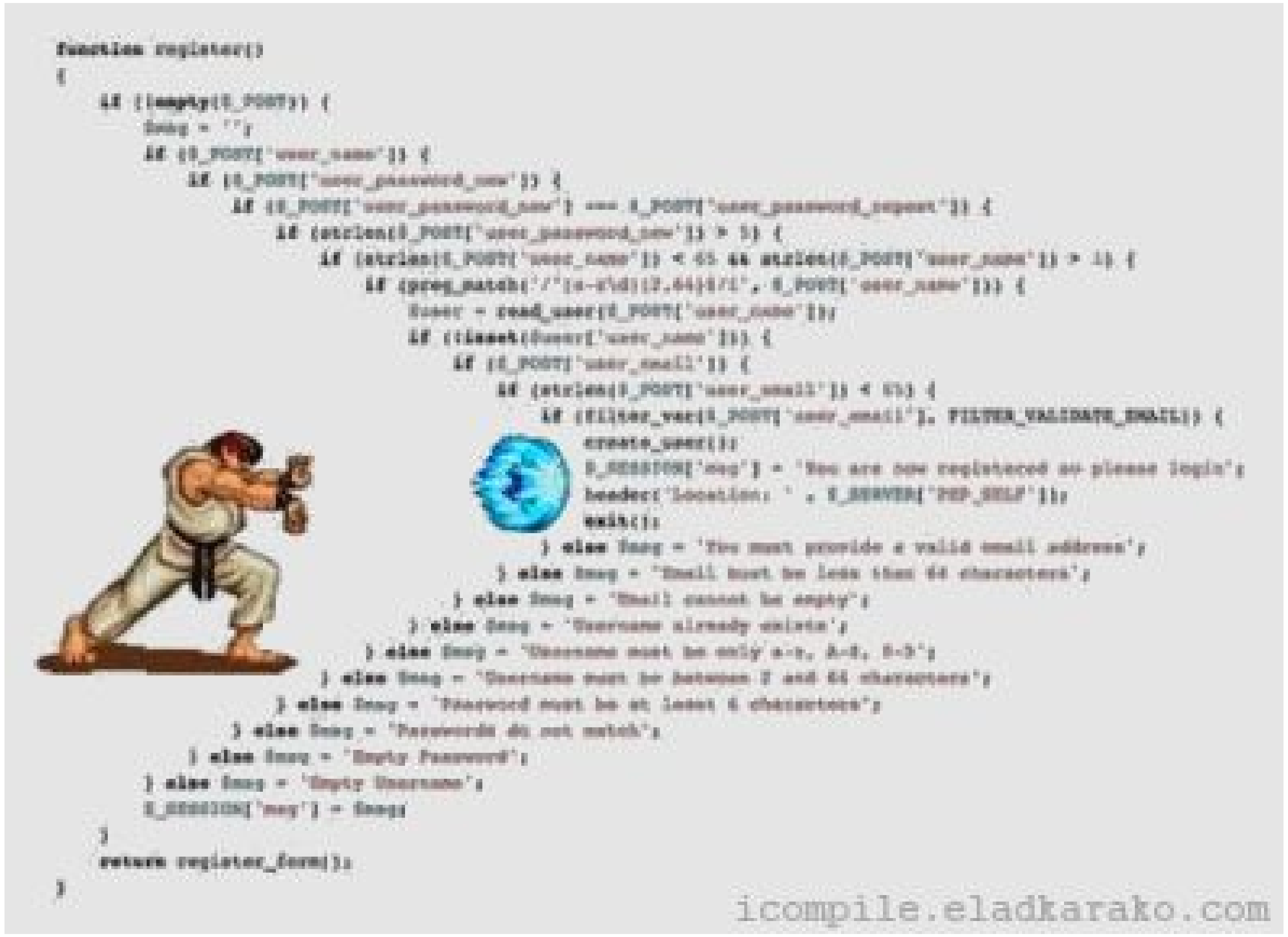
Вывод после загрузки изображения

```
randomButtonElement.onclick = () => {
  makeGetRequest('https://api.github.com/users',
     (request) => {
        let data;
        try {
           data = JSON.parse(request)
        } catch (err) {
           showError(new Error('Ошибка при чтении из json'));
        }
        if (data) {
           const user = data[Math.floor(Math.random() * data.length)];
           loadImage(user.avatar_url,() => {
              hideError();
              drawUser(user);
           }, showError);
        }
     }, showError);
};
```

https://jsbin.com/maragay/edit?html,js,output

CALLBACK
HELL

Callback hell



```
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

http://callbackhell.com/
http://callbackhell.ru/

PROMISE

Promise

Promise – «обещание» того, что асинхронная операция завершится.

```
const p = new Promise(...);

p.then(function onFulfilled(value) {
  // если все хорошо
}, function onRejected(error) {
  // что-то пошло не так
});
```

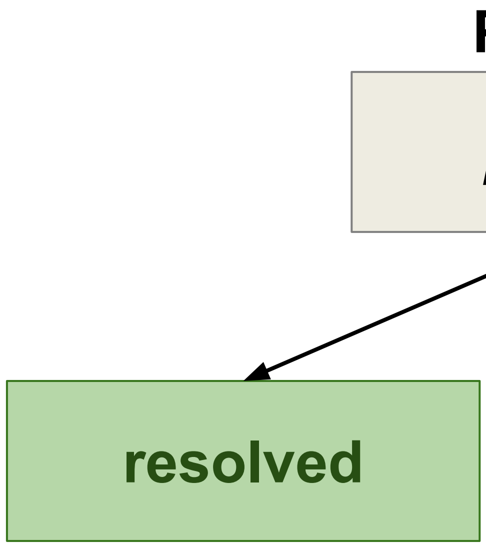
Promise

Promise

pending

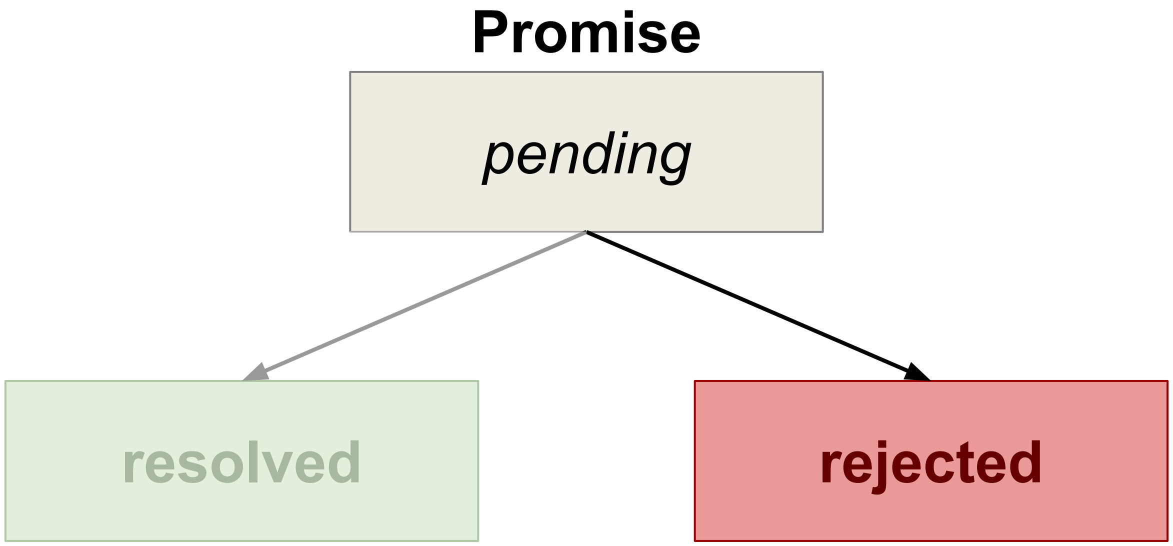
```
const p = new Promise(function(resolve, reject) {

});
```

Promise



```
const p = new Promise(function(resolve, reject) {
  resolve('foo'); // при успешном выполнении

});
```

Promise



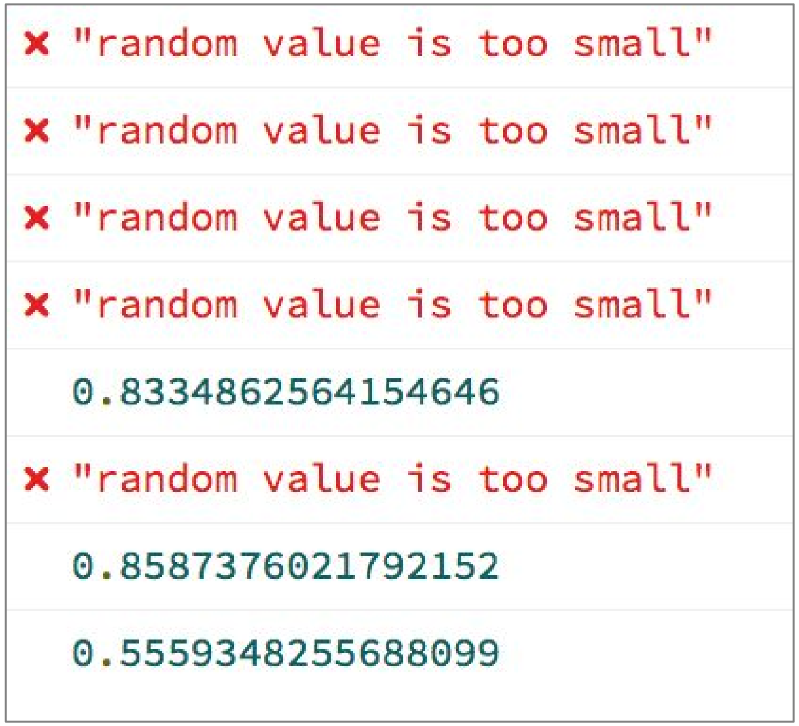
```
const p = new Promise(function(resolve, reject) {
  //resolve('foo'); // при успешном выполнении
  reject('bar'); // при ошибке
});
```

Promise

```
const p = new Promise(function(resolve, reject) {
   setTimeout(function () {
       const random = Math.random();
       if (random > 0.5) {
           resolve(random);
       } else {
           reject(new RangeError(
                   'random value is too small'));
       }
   }, 1000);
});

p.then(function(value) {
   console.log(value);
}, function(error) {
   console.error(error.message);
});
```

Promise

```javascript
const p = new Promise(function(resolve, reject) {
  setTimeout(function () {
     var random = Math.random();
     if (random > 0.5) {
        resolve(random);
     } else {
        reject(new RangeError(
              'random value is too small'));
     }
  }, 1000);
});

p.then(function(value) {
  console.log(value);
}, function(error) {
  console.error(error.message);
});
```



```
✖ "random value is too small"
✖ "random value is too small"
✖ "random value is too small"
✖ "random value is too small"
  0.8334862564154646
✖ "random value is too small"
  0.8587376021792152
  0.5559348255688099
```

http://jsbin.com/rabunap/1/edit?js,console

Цепочки

```
p
  .then(function (value) {
      console.log(value);
  }, function (error) {
      console.error(error.message);
  });
```

Цепочки

```
p
  .then(function (value) {
     console.log(value);
     return value;
  });
```

Цепочки

```
p
  .then((value) => {
     console.log(value);
     return value;
  })
```

Цепочки

```
p
  .then((value) => {
      console.log(value);
      return value;
  })
  .then(v => v * 2)
```

Цепочки

```
p
  .then((value) => {
      console.log(value);
      return value;
  })
  .then(v => v * 2)
  .then(v => v - 1)
```

Цепочки

```
p
  .then((value) => {
    console.log(value);
    return value;
  })
  .then(v => v * 2)
  .then(v => v - 1)
  .then(v => v * 10)
```

Цепочки

```
p
  .then((value) => {
      console.log(value);
      return value;
  })
  .then(v => v * 2)
  .then(v => v - 1)
  .then(v => v * 10)
  .then(v => console.log(v))
```

Цепочки

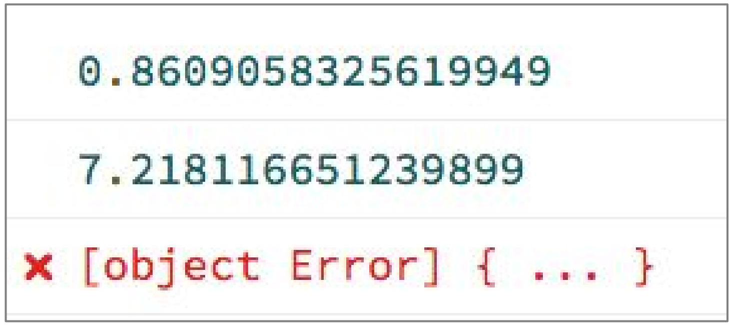
.catch(fn) – то же самое, что **.then(*null*, fn)**

```
p
  .then((value) => {
      console.log(value);
      return value;
  })
  .then(v => v * 2)
  .then(v => v - 1)
  .then(v => v * 10)
  .then(v => console.log(v))
  .catch(err => console.error(err.message));
```

Цепочки

.catch(fn) – то же самое, что **.then(*null*, fn)**

```
p
  .then((value) => {
    console.log(value);
    return value;
  })
  .then(v => v * 2)
  .then(v => v - 1)
  .then(v => v * 10)
  .then(v => console.log(v))
  .catch(err => console.error(err.message));
```

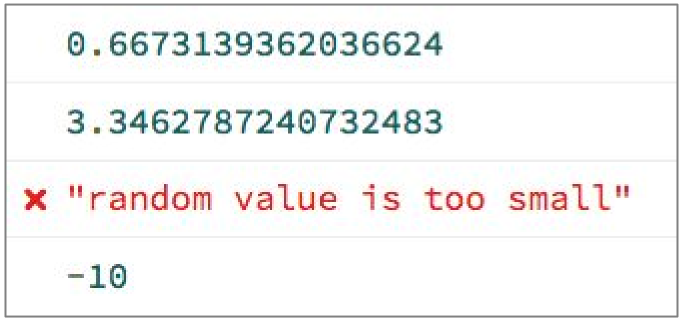


```
0.8609058325619949
7.218116651239899
✖ [object Error] { ... }
```

http://jsbin.com/jakida/edit?js,console

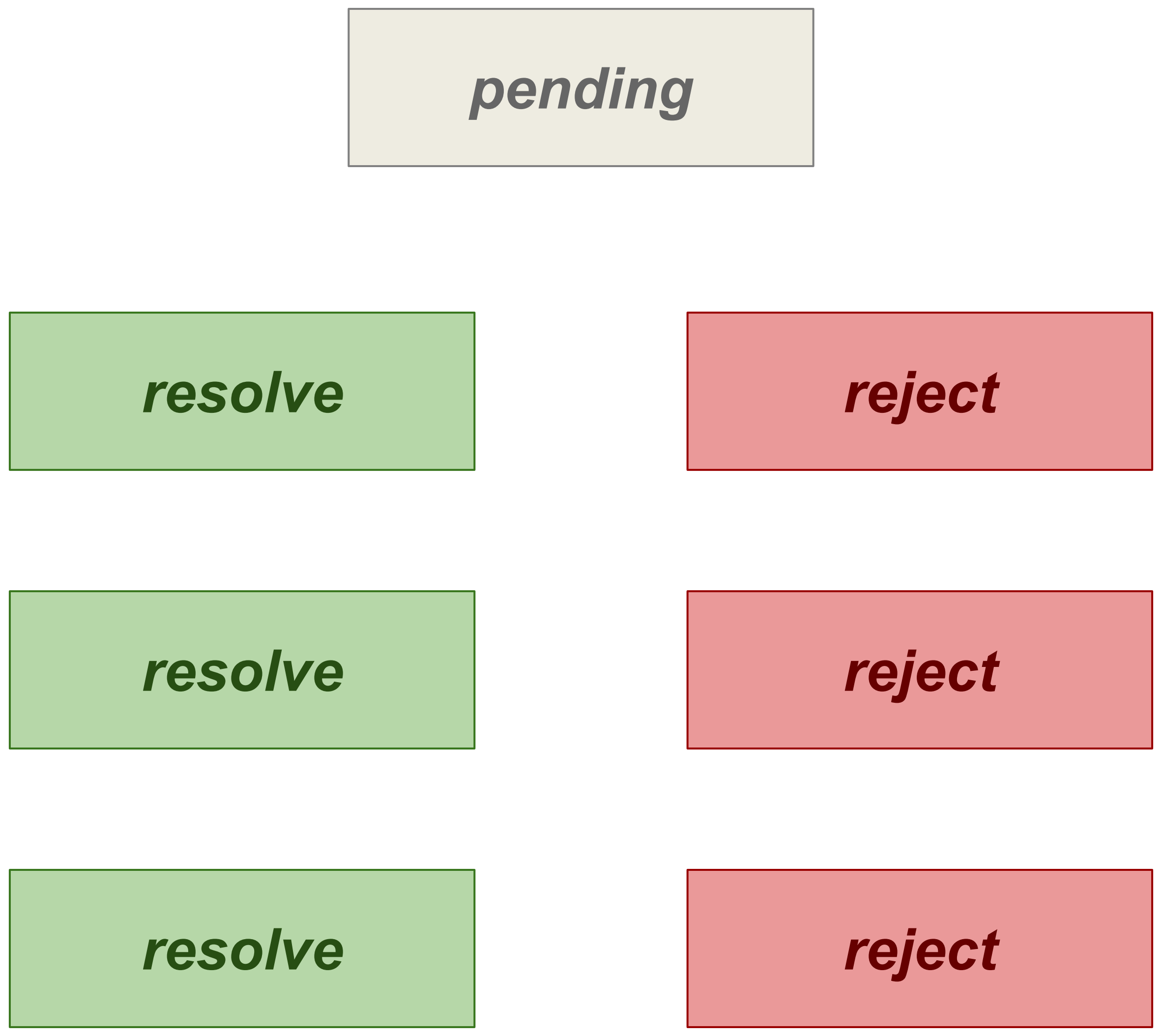
catch

```
p
  .then((value) => {
      console.log(value);
      return value;
  })
  .then(v => v * 2)
  .catch((err) => {
      console.error(err.message);
      return 0;
  })
  .then(v => v - 1)
  .then(v => v * 10)
  .then(v => console.log(v))
  .catch(err => console.error(err.message));
```

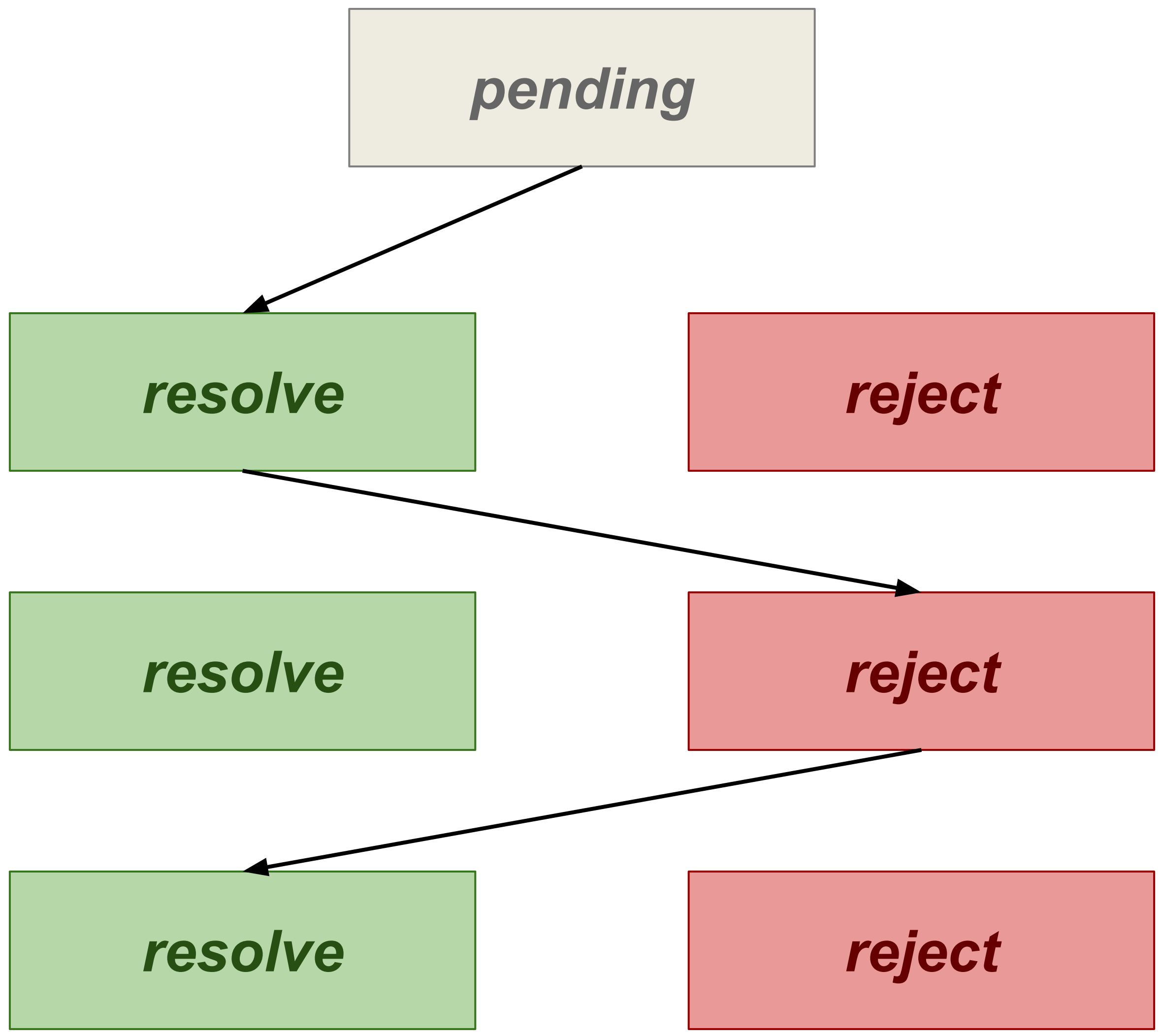


```
0.6673139362036624
3.346278724073248З
✖ "random value is too small"
-10
```

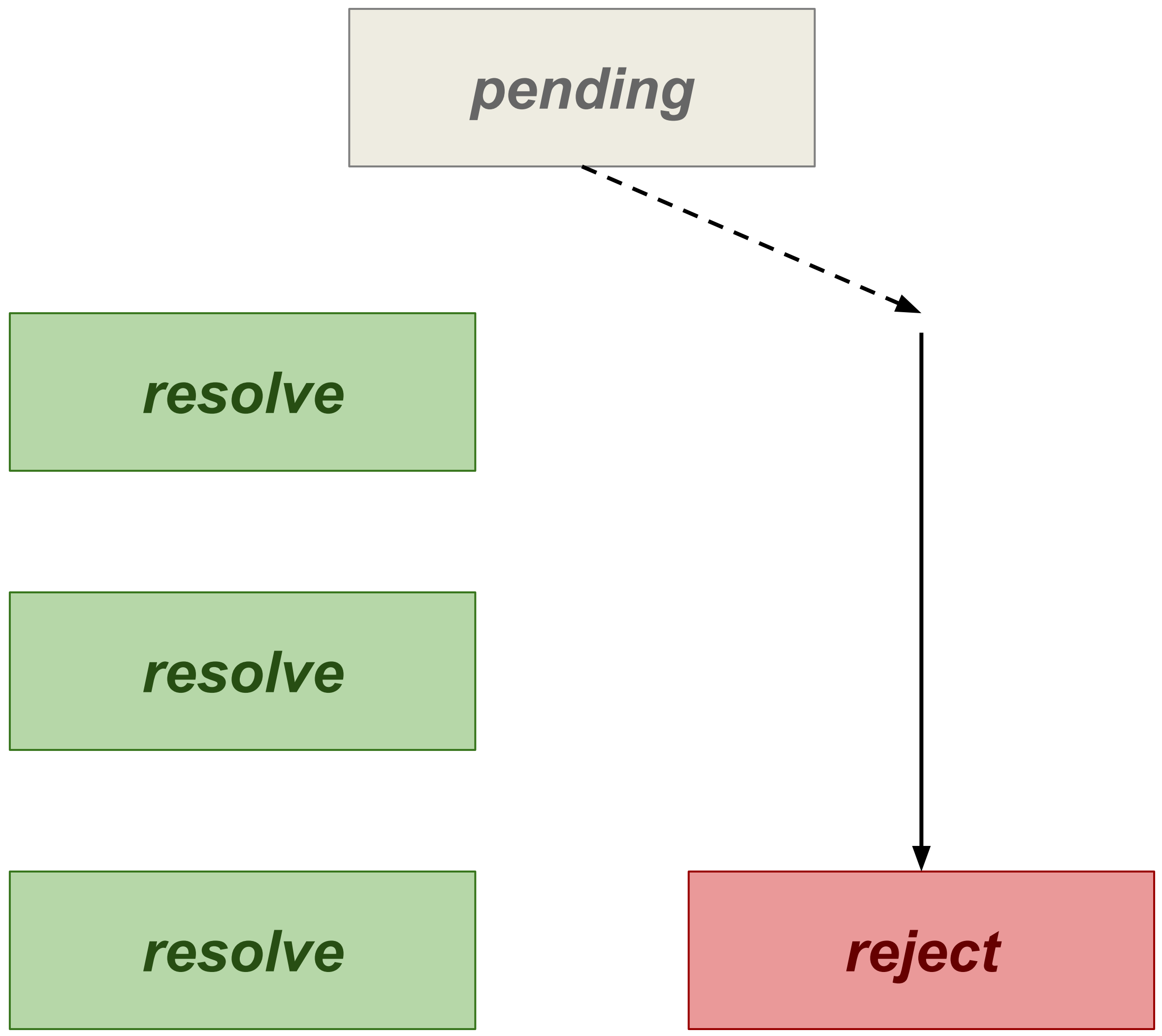
Цепочка выполнения



Цепочка выполнения



Цепочка выполнения



Сохранение значений

```
const promise = new Promise(function (resolve) {
   resolve(42);
});

promise
   .then(v => v * 2)
   .then(v => console.log(v)); // 84

setTimeout(() => {
   promise
       .then(v => v / 2)
       .then(v => console.log(v)); // 21
}, 1000);
```

Вопросы?