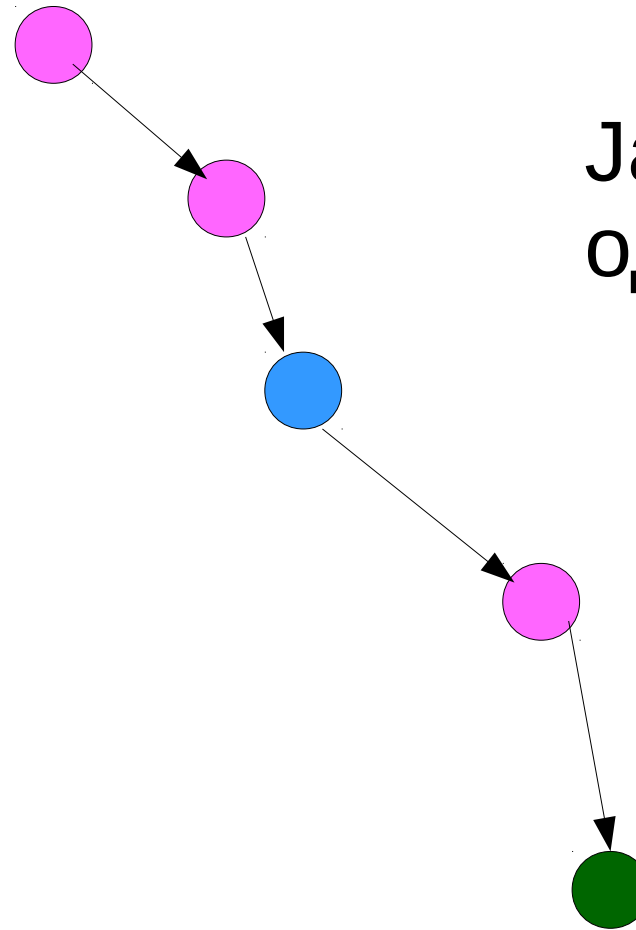


Асинхронное программирование в JavaScript



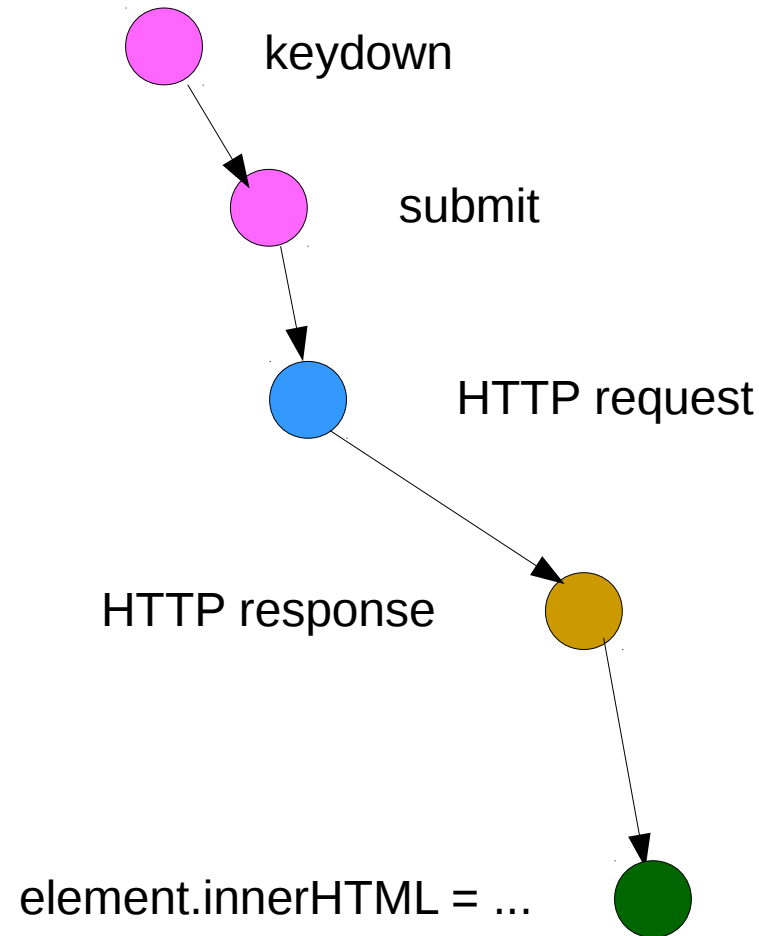
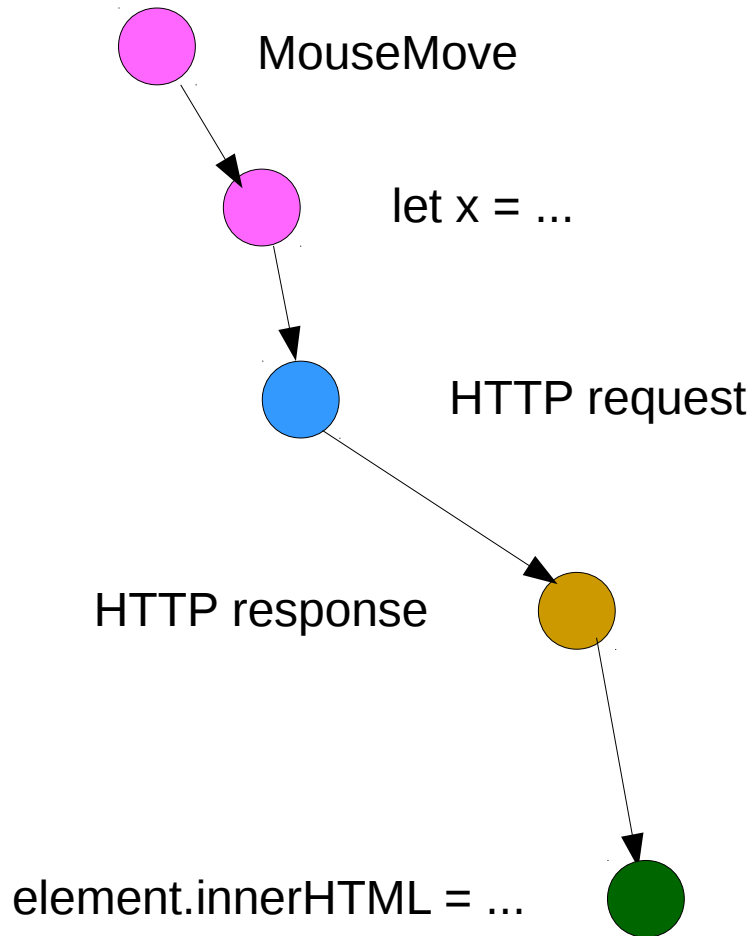
- Асинхронное программирование
- Преимущества асинхронного программирования
- Функция обратного вызова - основа асинхронного программирования
- XMLHttpRequest
- Использование XML для AJAX
- Использование JSON для AJAX
- Использование jQuery для AJAX вызовов
- Отложенные (deferred) объекты.

Синхронное программирование

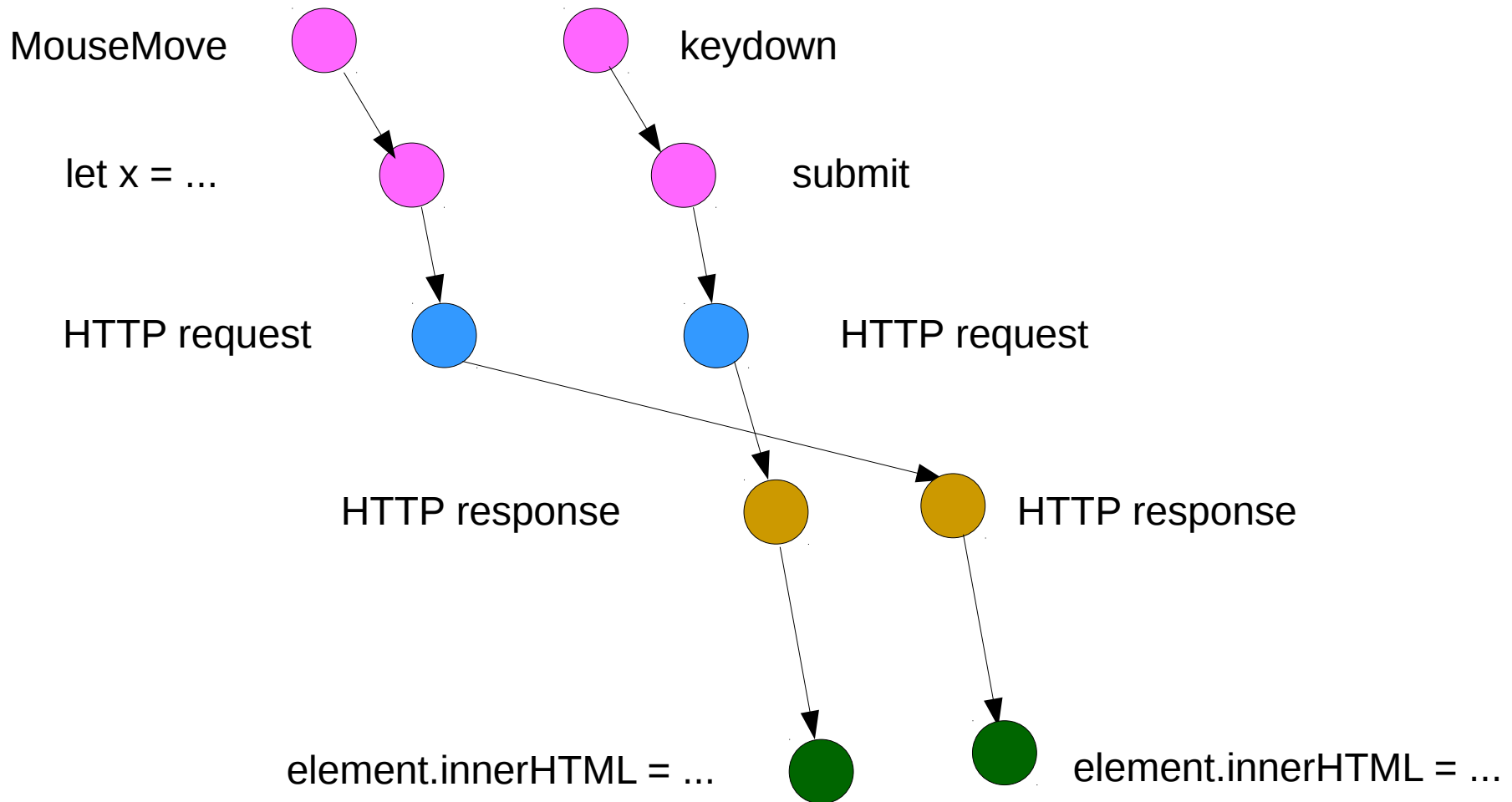


JavaScript -
однопоточный

Асинхронное программирование



Асинхронное программирование



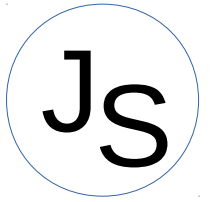
Асинхронное программирование



Проблемы

- Управление асинхронными действиями
- Обработка ошибок

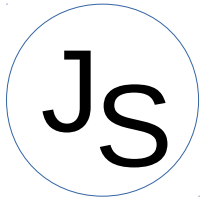
Функция обратного вызова



Callback

- ~~Управление асинхронными действиями~~
- События - обработчики

Функция обратного вызова



Callback

- События - обработчики
- Функция передаётся заранее

Функция обратного вызова

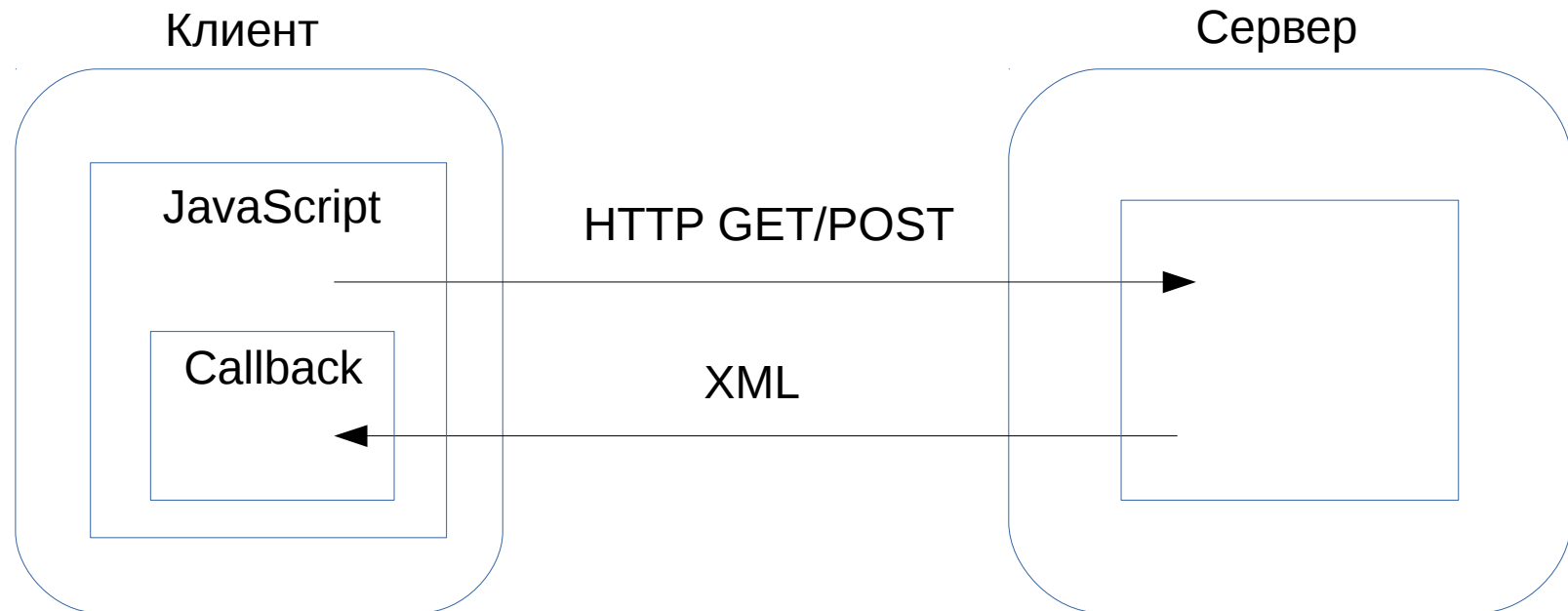


```
async_function(  
  'parameter 1',  
  'parameter 2',  
  ...,  
  function (err, data) {  
    ...  
  }  
);
```

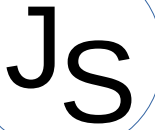

AJAX



Asynchronous JavaScript and XML

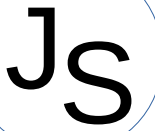


XMLHttpRequest



```
var xhr = new XMLHttpRequest();  
  
xhr.open('GET', '/test.html', false);  
xhr.send(null);  
if(xhr.status == 200) {  
    alert(xhr.responseText);  
}
```

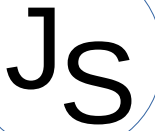
XMLHttpRequest



```
var xhr = new XMLHttpRequest();

xhr.open('GET', '/xhr/test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
        if(xhr.status == 200) {
            alert(xhr.responseText);
        }
    }
};
xhr.send(null);
```

XMLHttpRequest

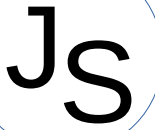


```
var xhr = new XMLHttpRequest();

xhr.open('POST', '/xhr/test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
        if(xhr.status == 200) {
            alert(xhr.responseText);
        }
    }
};

http.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");
xhr.send('name=John&age=23');
```

jQuery AJAX



```
$.ajax({  
  url: '/test.html',  
  success: (data,status,jqXHR)=>{...},  
  error: (jqXHR,status,message)=>{...}  
});
```

```
$.ajax({  
  type: "POST",  
  url: '/test.html',  
  data: {name:value, ...},  
  success: (data,status,jqXHR)=>{...},  
  error: (jqXHR,status,message)=>{...}  
});
```

Ад callback'ов

JS

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' +
height)
            this.resize(width, height).write(dest + 'w' + width + '_' +
filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }).bind(this))
        }
      })
    })
  }
})
```

Отложенные объекты



jQuery.Deferred
Promise

```
Something.save()  
  .then(function() {  
    console.log('успех');  
  })  
  .error(function() {  
    // обработка ошибок  
  })
```

```
saveSomething()  
  .then(doOtherThing)  
  .then(deleteStuff)  
  .then(logResults);
```

Отложенные объекты



ES7: async functions

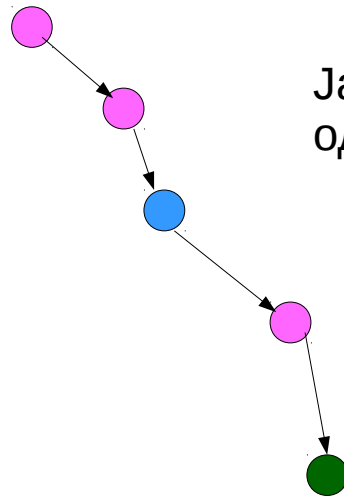
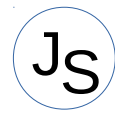
```
async function save(Something) {  
  try {  
    await Something.save()  
  } catch (ex) {  
    // обработка ошибок  
  }  
  console.log('успех');  
}
```


Асинхронное программирование в JavaScript



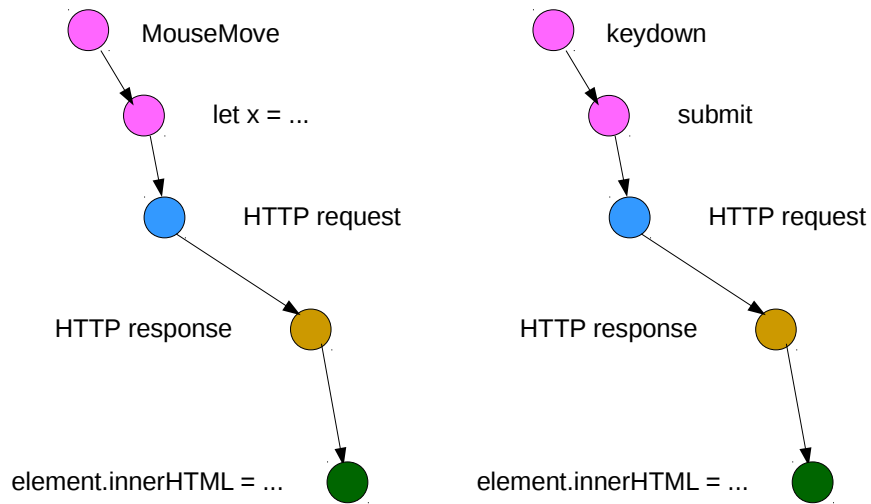
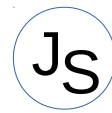
- Асинхронное программирование
- Преимущества асинхронного программирования
- Функция обратного вызова - основа асинхронного программирования
- XMLHttpRequest
- Использование XML для AJAX
- Использование JSON для AJAX
- Использование jQuery для AJAX вызовов
- Отложенные (deferred) объекты.

Синхронное программирование

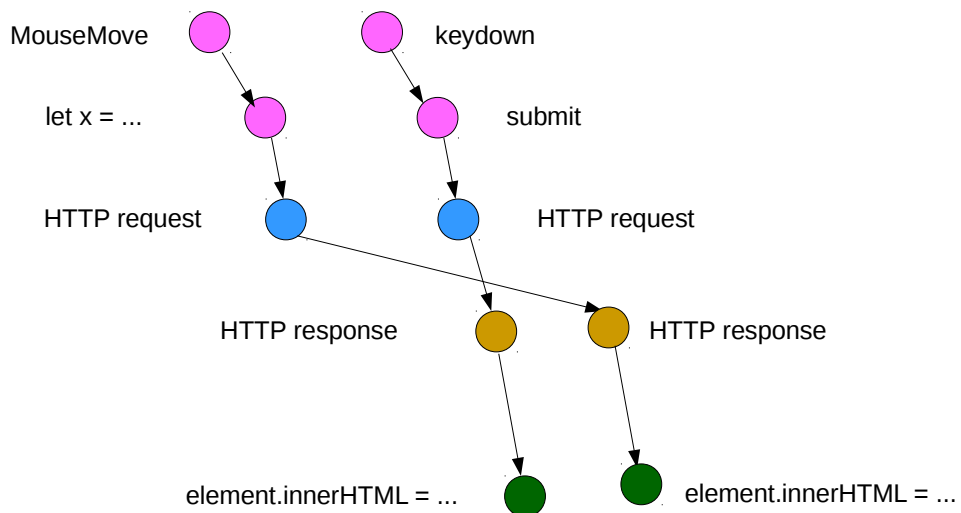
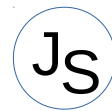


JavaScript -
однопоточный

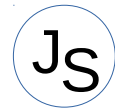
Асинхронное программирование



Асинхронное программирование



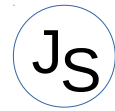
Асинхронное программирование



Проблемы

- Управление асинхронными действиями
- Обработка ошибок

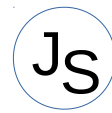
Функция обратного вызова



Callback

- Управление асинхронными действиями
- События - обработчики

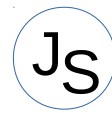
Функция обратного вызова



Callback

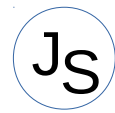
- События - обработчики
- Функция передаётся заранее

Функция обратного вызова

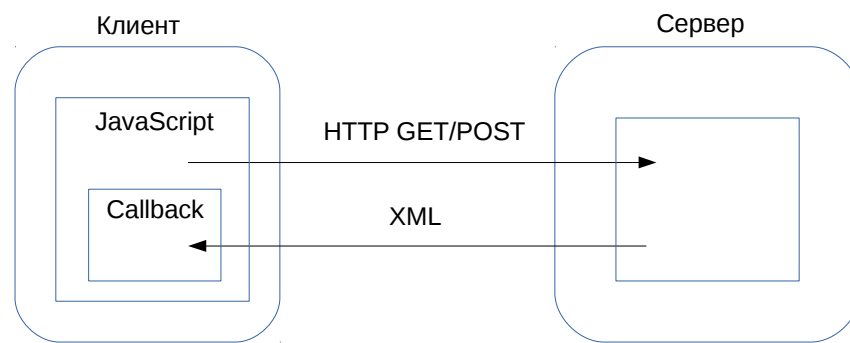


```
async_function(  
  'parameter 1',  
  'parameter 2',  
  .../  
  function (err, data) {  
    ...  
  }  
);
```


AJAX



Asynchronous JavaScript and XML



XMLHttpRequest



```
var xhr = new XMLHttpRequest();

xhr.open('GET', '/test.html', false);
xhr.send(null);
if(xhr.status == 200) {
    alert(xhr.responseText);
}
```

XMLHttpRequest



```
var xhr = new XMLHttpRequest();

xhr.open('GET', '/xhr/test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
        if(xhr.status == 200) {
            alert(xhr.responseText);
        }
    }
};
xhr.send(null);
```

XMLHttpRequest



```
var xhr = new XMLHttpRequest();

xhr.open('POST', '/xhr/test.html', true);
xhr.onreadystatechange = function() {
    if (xhr.readyState == 4) {
        if(xhr.status == 200) {
            alert(xhr.responseText);
        }
    }
};
http.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");
xhr.send('name=John&age=23');
```

jQuery AJAX



```
$.ajax({
  url: '/test.html',
  success: (data,status,jqXHR)=>{...},
  error: (jqXHR,status,message)=>{...}
});

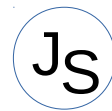
$.ajax({
  type: "POST",
  url: '/test.html',
  data: {name:value, ...},
  success: (data,status,jqXHR)=>{...},
  error: (jqXHR,status,message)=>{...}
});
```

Ад callback'ов



```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' +
height)
            this.resize(width, height).write(dest + 'w' + width + '_' +
filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }).bind(this))
        }
      })
    })
  }
})
})
```

Отложенные объекты



jQuery.Deferred

Promise

```
Something.save()  
  .then(function() {  
    console.log('успех');  
  })  
  .error(function() {  
    // обработка ошибок  
  })
```

```
saveSomething()  
  .then(doOtherThing)  
  .then(deleteStuff)  
  .then(logResults);
```

ES7: async functions

```
async function save(Something) {  
  try {  
    await Something.save()  
  } catch (ex) {  
    // обработка ошибок  
  }  
  console.log('успех');  
}
```