Объектная модель документа

DOM – Document Object Model

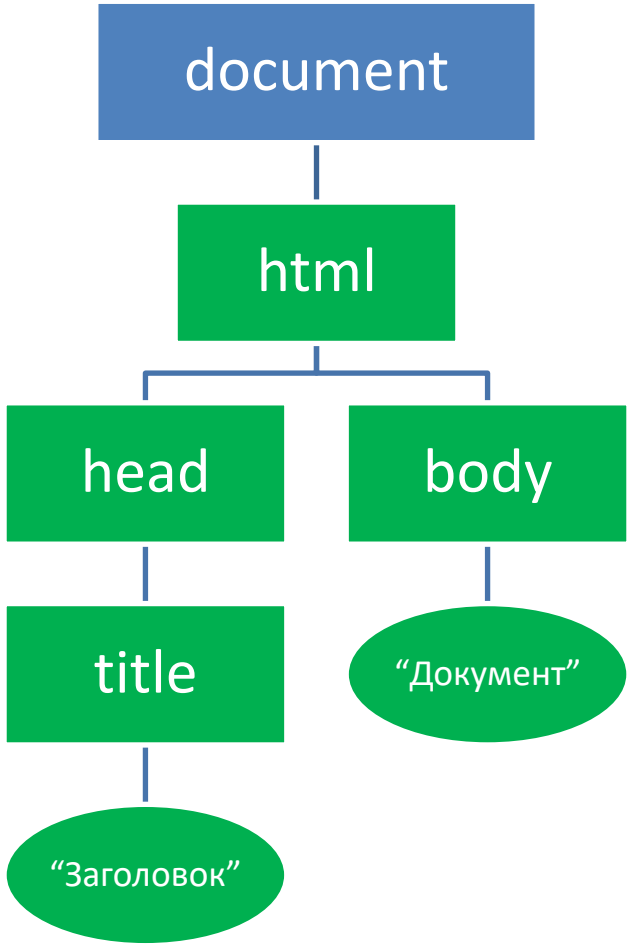
Модель DOM

JS

- Обеспечивает доступ ко всем элементам документа и их атрибутам посредством древовидной структуры объектов
- Позволяет создавать, удалять и изменять элементы документа и их содержимое
- DOM – представление документа, загруженного в окно браузера, в виде дерева тегов

DOM

```
<html>
  <head>
     <title>Заголовок</title>
  </head>
  <body>
      Документ
  </body>
</html>
```



Терминология DOM

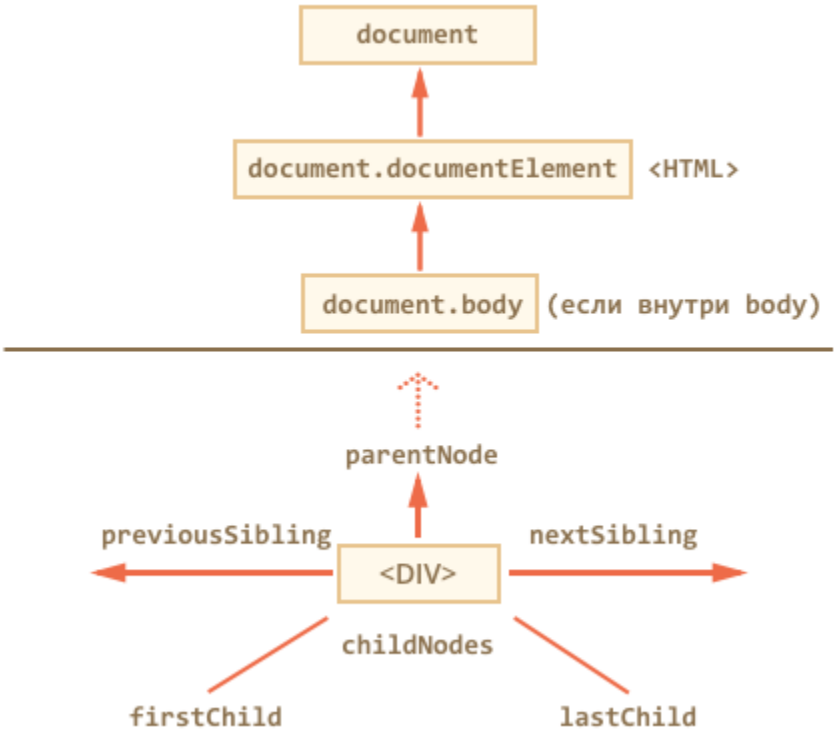
JS

- Каждый документ содержит элементы, представляемые узлами
- Всего существует 12 типов узлов, основных узлов два: узел-элемент и узел-текст
- Тег представляется узлом-элементом
- Текст представляется узлом-текстом
- Узел-элемент и узел-текст – равноправные узлы дерева

- **ELEMENT_NODE = 1**
- ATTRIBUTE_NODE = 2
- **TEXT_NODE = 3**
- CDATA_SECTION_NODE = 4
- ENTITY_REFERENCE_NODE = 5
- ENTITY_NODE = 6
- PROCESSING_INSTRUCTION_NODE = 7
- COMMENT_NODE = 8
- DOCUMENT_NODE = 9
- DOCUMENT_TYPE_NODE = 10
- DOCUMENT_FRAGMENT_NODE = 11
- NOTATION_NODE = 12

http://www.w3.org/TR/DOM-Level-3-Core/idl-definitions.html

DOM



- Доступ к DOM начинается с объекта document. Из него можно добраться до любых узлов.
- Основные ссылки, по которым можно переходить между узлами DOM

DOM

JS

Существует 6 основных методов получения DOM элементов:

- getElementById
- getElementsByTagName
- getElementsByName
- getElementsByClassName (except IE<9)
- querySelector (except IE<8 and IE8 in compat mode)
- querySelectorAll

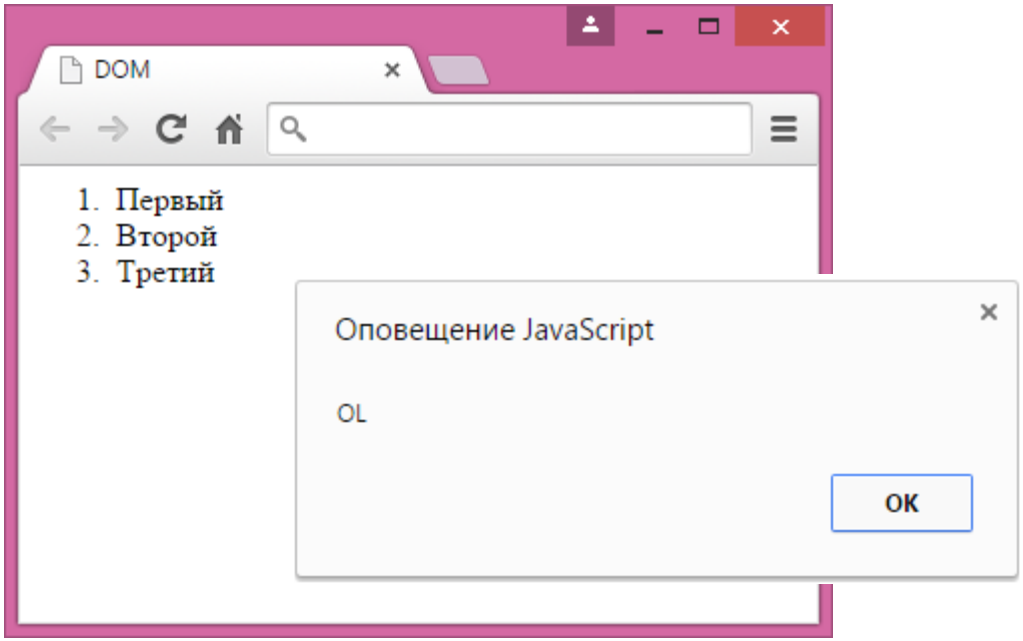
Навигация по дереву

JS

- Начать навигацию по дереву можно с любого узла, для которого известен идентификатор, задаваемый параметром тега id
- Метод **getElementById**(идентификатор_узла)

Навигация по дереву

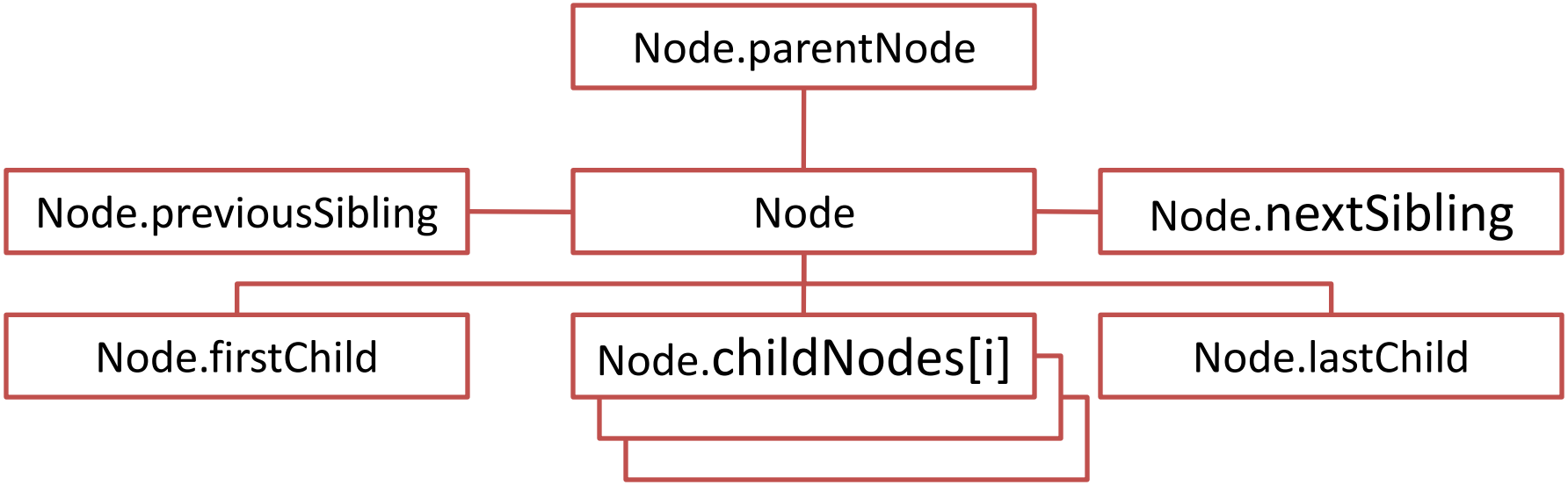
```
<body>
    <ol id=ordlist>
    <li>Первый</li>
    <li>Второй</li>
    <li>Третий</li>
    </ol>
    <script>
    alert(document.getElementById('ordlist').tagName)
    </script>
</body>
```



Навигация по дереву

Свойства для чтения (Node – текущий узел)

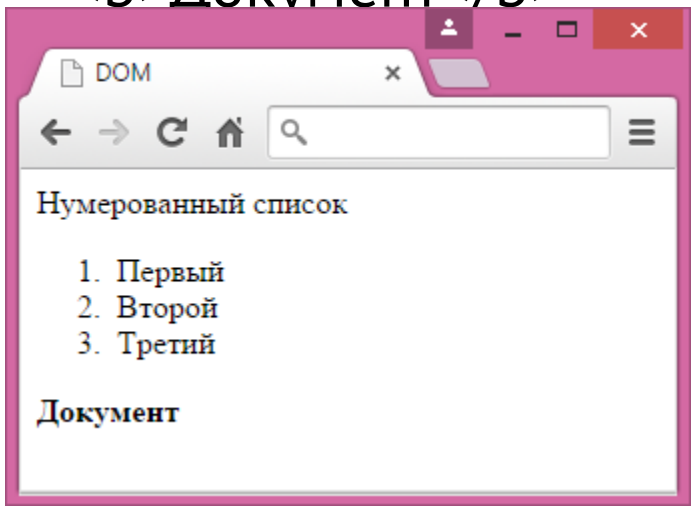
- parentNode
- childNodes[i]
- firstChild
- lastChild
- previousSibling
- nextSibling

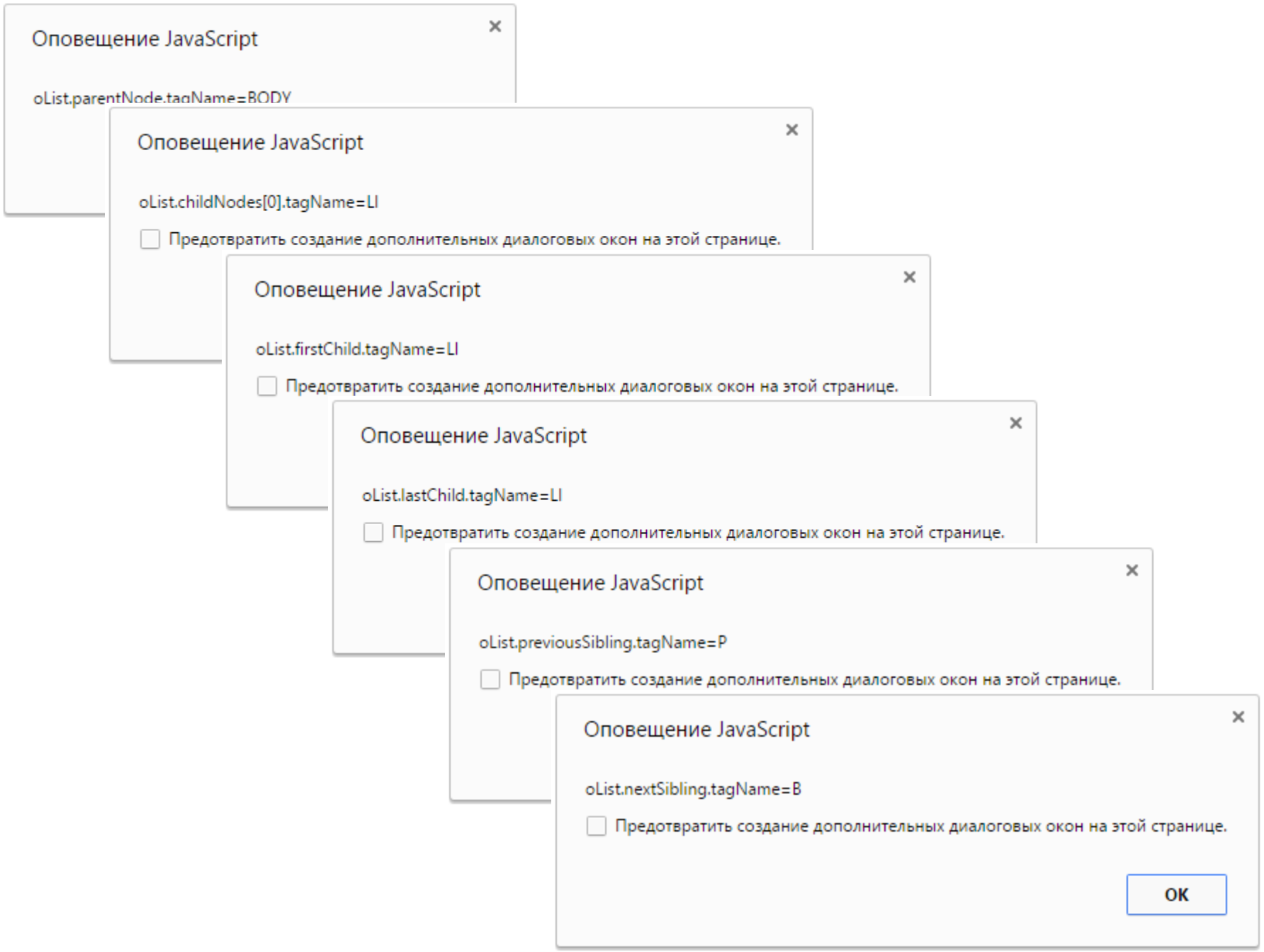


Навигация по дереву

```
<p>Нумерованный
список</p>
<ol id=ordlist>
<li>Первый</li>
<li>Второй</li>
<li>Третий</li>
</ol>
<b>Документ</b>
```

```
<script>
oList=document.getElementById('ordlist'
)
alert(oList.parentNode.tagName)
alert(oList.childNodes[0].tagName)
alert(oList.firstChild.tagName)
alert(oList.lastChild.tagName)
alert(oList.previousSibling.tagName)
alert(oList.nextSibling.tagName)
</script>
```




Оповещение JavaScript
oList.parentNode.tagName=BODY
Оповещение JavaScript
oList.childNodes[0].tagName=LI
Предотвратить создание дополнительных диалоговых окон на этой странице.
Оповещение JavaScript
oList.firstChild.tagName=LI
Предотвратить создание дополнительных диалоговых окон на этой странице.
Оповещение JavaScript
oList.lastChild.tagName=LI
Предотвратить создание дополнительных диалоговых окон на этой странице.
Оповещение JavaScript
oList.previousSibling.tagName=P
Предотвратить создание дополнительных диалоговых окон на этой странице.
Оповещение JavaScript
oList.nextSibling.tagName=B
Предотвратить создание дополнительных диалоговых окон на этой странице.
OK

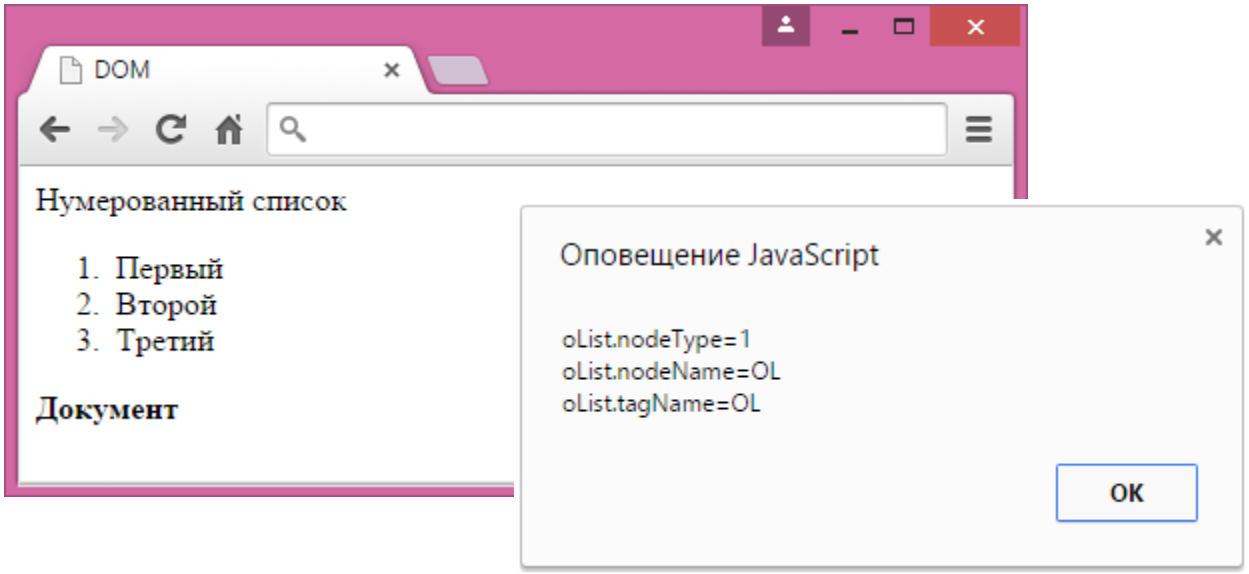
Проблемы навигации

JS

- Некоторые браузеры трактуют символы пробелов и переходы на новую строку как текстовые узлы
- Это вызывает проблемы при использовании свойств: firstChild, lastChild, nextSibling, previousSibling

Свойства узлов

- nodeType – тип узла
- nodeName и tagName – имя тега
- nodeName определено для многих типов узлов
- tagName определено только для элементов-узлов



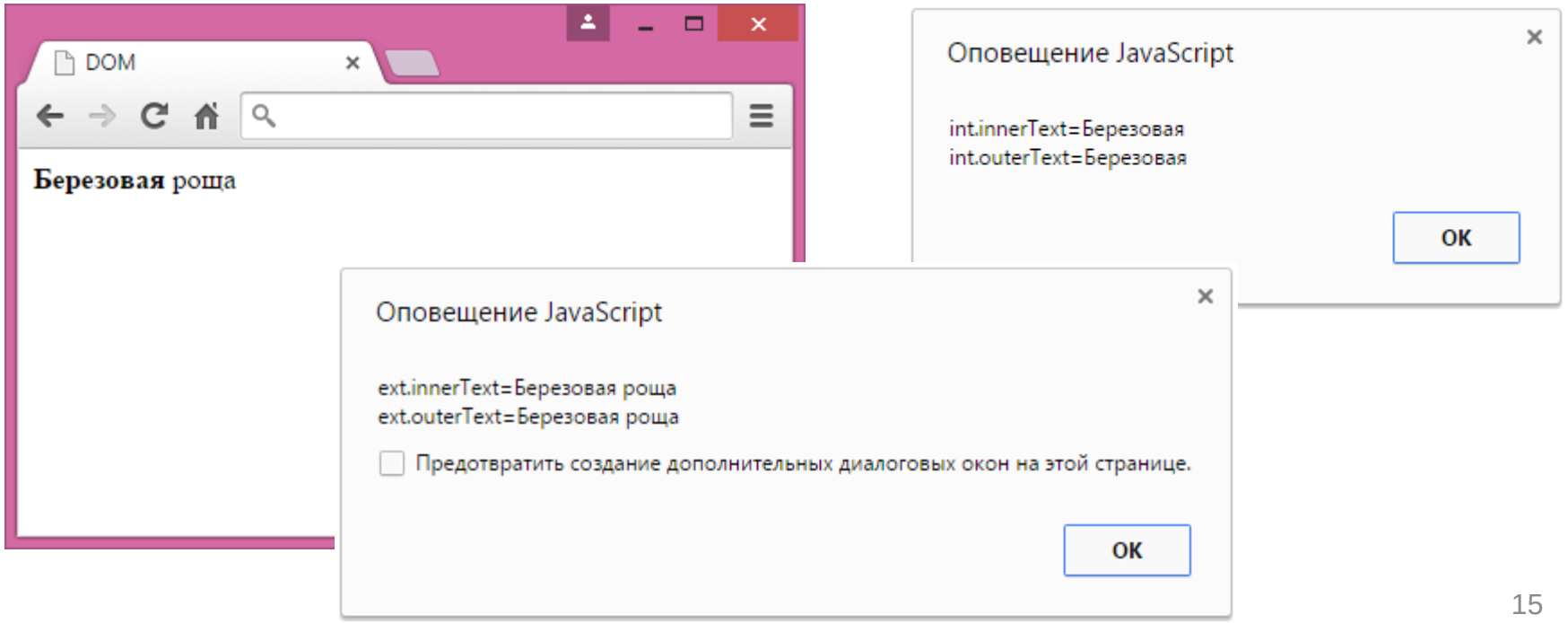
Свойства узлов

JS

- innerTEXT – текст элемента без тегов HTML
- outerTEXT – текст элемента без тегов HTML
- innerHTML – содержит текст и HTML-элементы для данного тега (html-содержимое узла в виде строки)
- outerHTML – содержит текст и HTML-элементы для данного и вложенных тегов (html-узел целиком)

innerTEXT и outerTEXT

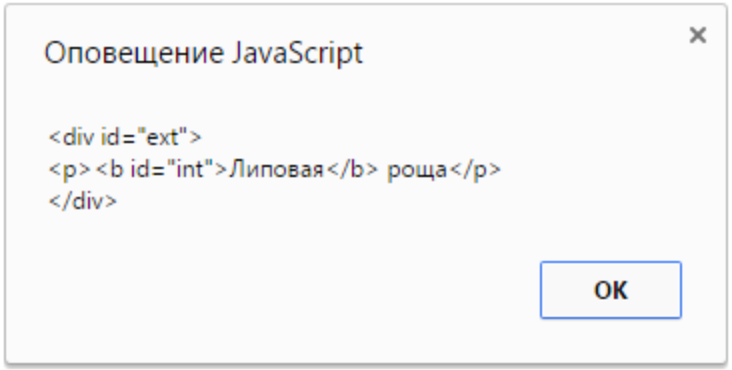
- Чтение
-
  <P><B id=int>Березовая</B> роща</P>

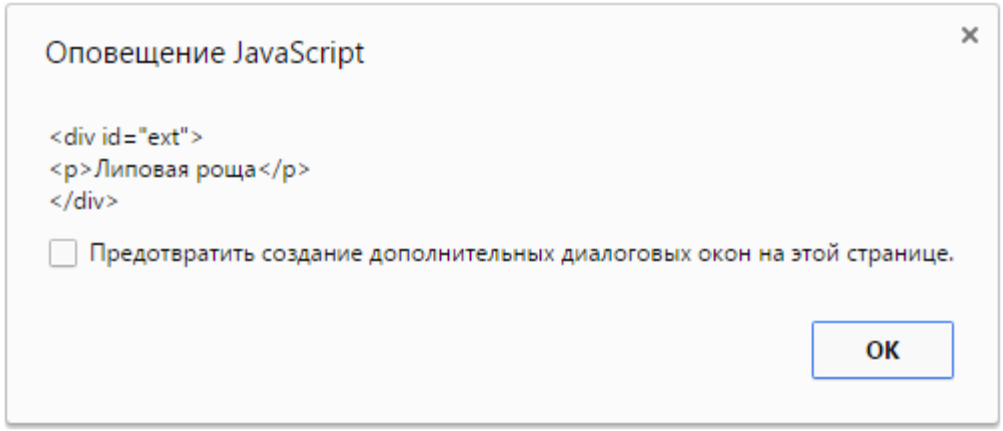


innerTEXT и outerTEXT

- Запись
-
  <P><B id=int>Березовая</B> роща</P>

- int.innerText='Липовая'
  int.outerText='Липовая'

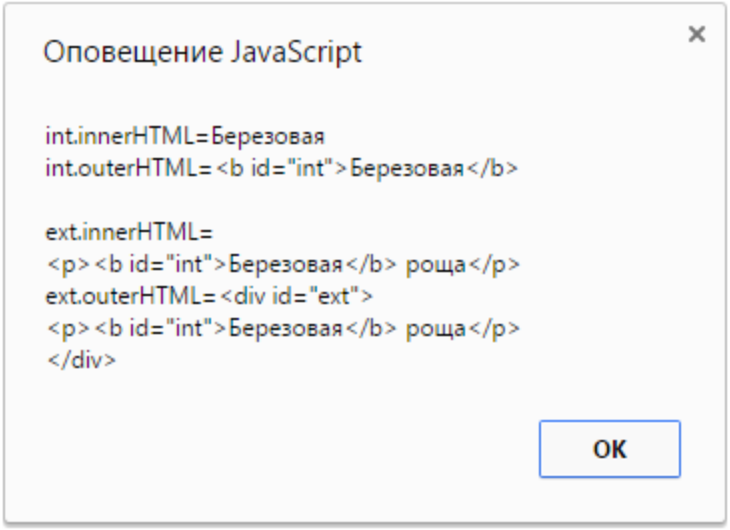




innerHTML и outerHTML

JS

-
  <P><B id=int>Березовая</B> роща</P>



innerHTML и outerHTML

JS

```
<style>
.new
{border: 3px solid #999999; border-radius: 5px; color:green}
</style>
<div id=ext><P><B id=int>Березовая</B> роща</P></div>

int.innerHTML='Липовая'
int.outerHTML='<i id=int>Вербная</i>'
ext.innerHTML='<h1><i id=int>Ясеневая</i> роща</h1>'
ext.outerHTML='<div id=ext class=new><h1>
<i id=int>Буковая</i> роща</h1></div>'
```

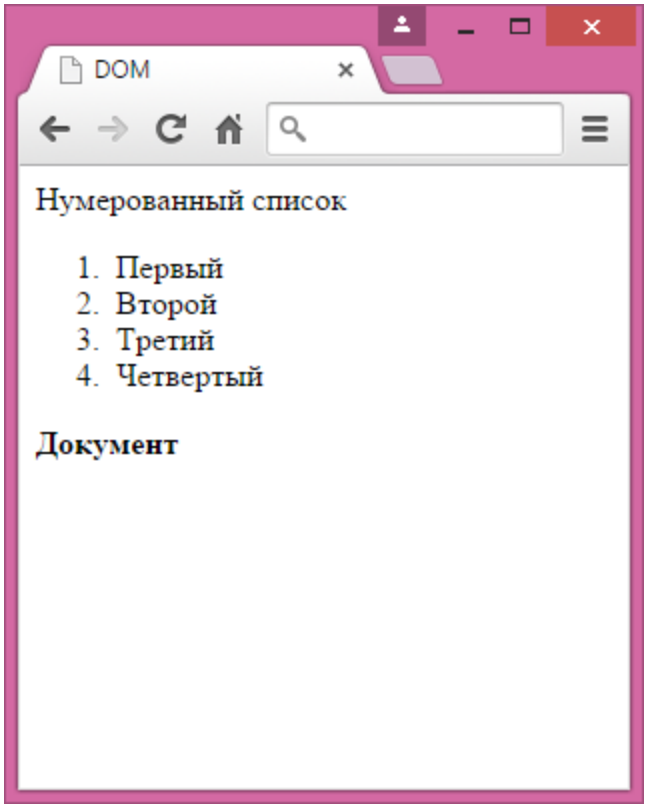
Редактирование дерева

JS

- Методы
  - createElement()
    создание нового узла-элемента
  - createTextNode()
    создание нового узла-текста
  - appendChild()
    добавление узла в конец коллекции childNodes узла, для которого метод был вызван
  - insertBefore()
    добавление узла с возможностью указания места, куда вставляется новый узел

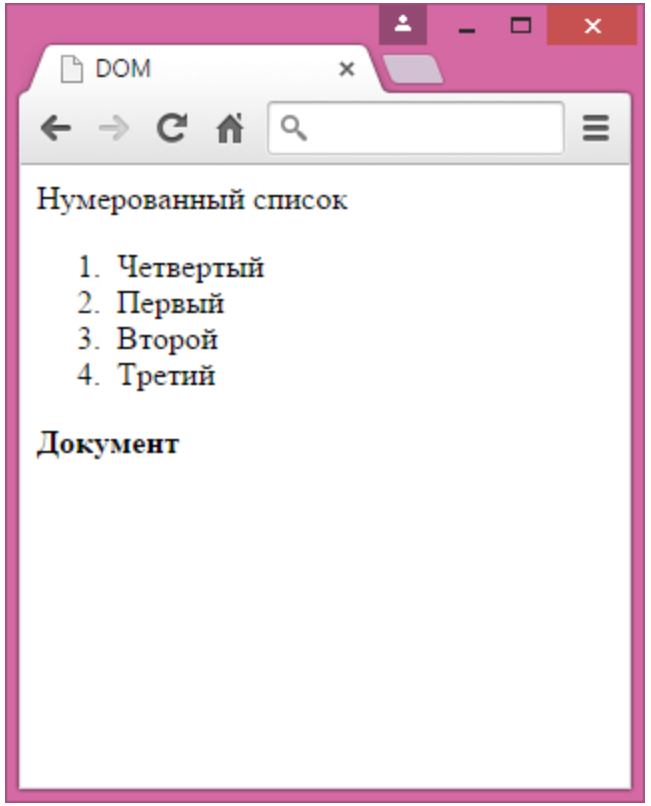
Редактирование дерева

```
<p>Нумерованный список</p>
<ol id=ordlist>
<li>Первый</li><li>Второй</li><li>Третий</li>
</ol>
<b>Документ</b>
<script>
oList=document.getElementById('ordlist')
newItem=document.createElement('LI')
newText=document.createTextNode('Четвертый')
newItem.appendChild(newText);
oList.appendChild(newItem);
</script>
```



Редактирование дерева

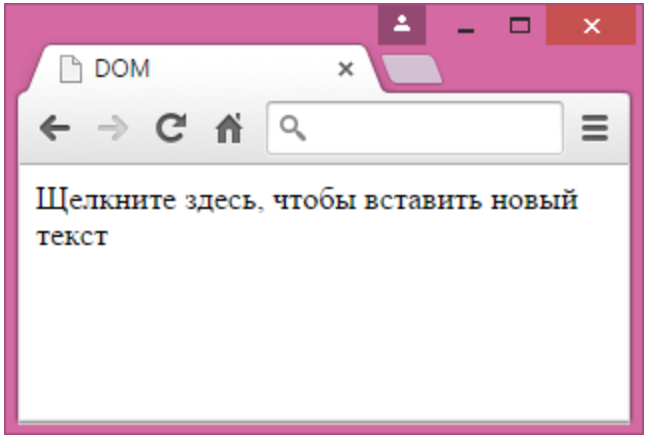
```
<p>Нумерованный список</p>
<ol id=ordlist>
<li>Первый</li><li>Второй</li><li>Третий</li>
</ol>
<b>Документ</b>
<script>
oList=document.getElementById('ordlist')
newItem=document.createElement('LI')
newText=document.createTextNode('Четвертый')
newItem.appendChild(newText);
curItem=oList.firstChild
oList.insertBefore(newItem,curItem)
</script>
```

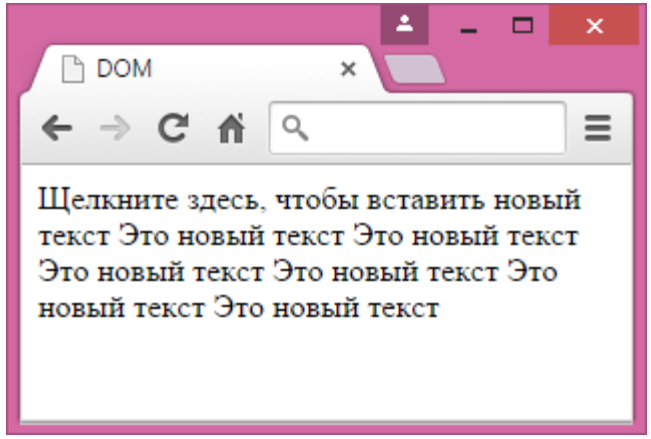


Редактирование дерева

JS

- <script>
  function addtext(text)
  {
   mytext=document.createTextNode(text)
   document.getElementById("example").appendChild(mytext)
  }
  </script>
-
  Щелкните здесь, чтобы вставить новый текст





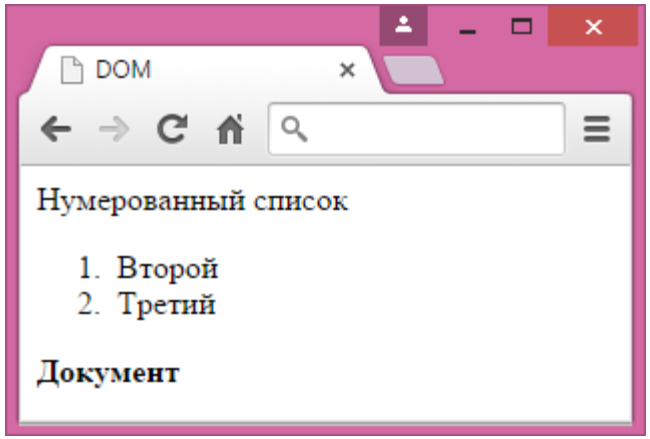
Редактирование дерева

JS

- Удаление узла
  - removeChild()
    удаление потомка узла, для которого метод был вызван
  - removeNode()
    удаление узла из документа

Редактирование дерева

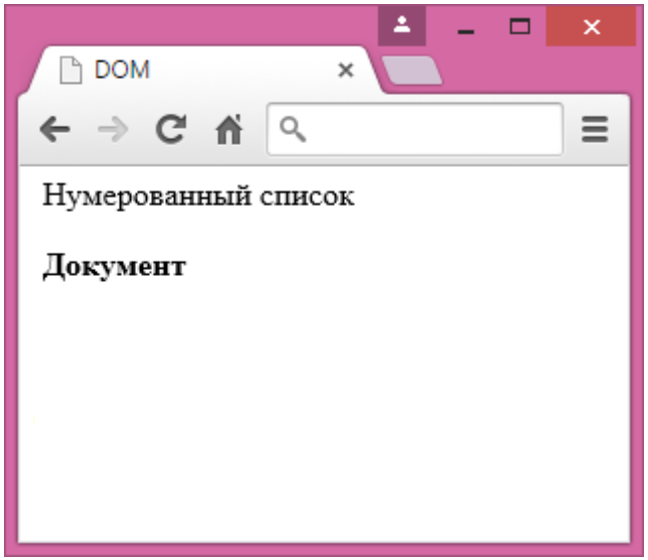
```
<p>Нумерованный список</p>
<ol id=ordlist>
<li>Первый</li><li>Второй</li><li>Третий</li>
</ol>
<b>Документ</b>
<script>
oList=document.getElementById('ordlist')
oList.removeChild(oList.firstChild)
</script>
```

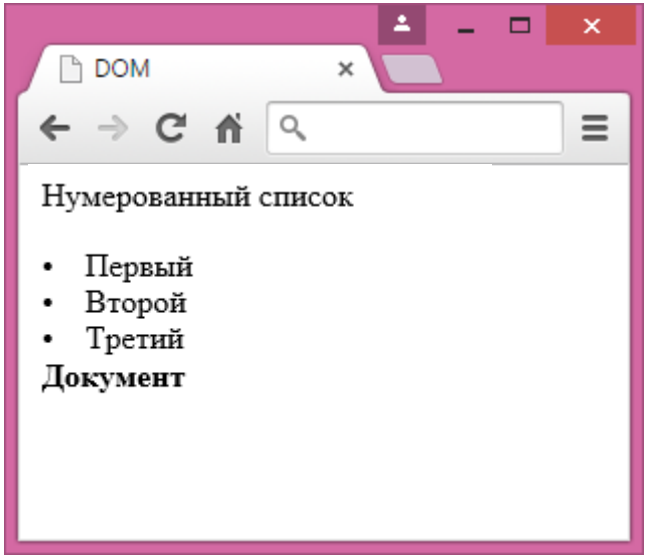


Редактирование дерева

```
<p>Нумерованный список</p>
<ol id=ordlist>
<li>Первый</li><li>Второй</li><li>Третий</li>
</ol>
<b>Документ</b>
<script>
oList=document.getElementById('ordlist')
oList.removeNode(true)   //false
</script>
```

true – удаляется узел-элемент
и его коллекция childNodes
false – удаляется узел-элемент без коллекции





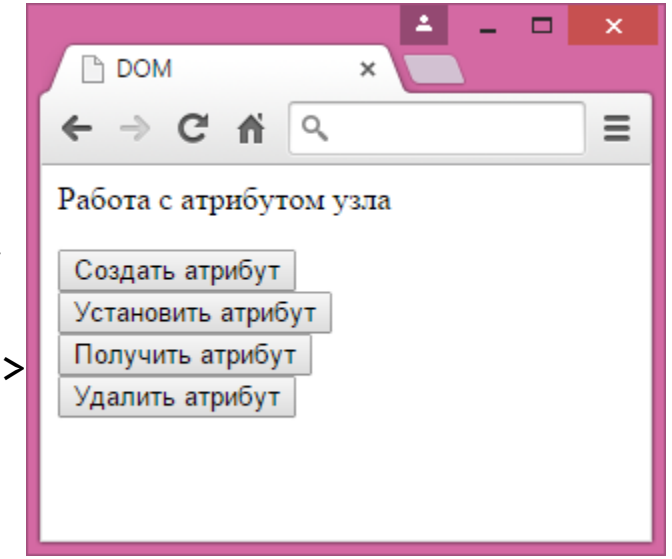
Редактирование дерева

JS

- Атрибуты узлов
  - createAttribute()
    создает новый атрибут узла
  - setAttribute()
    присваивает значение атрибуту
  - removeAttribute()
    удаляет атрибут
  - getAttribute()
    возвращает текущее значение атрибута

Работа с атрибутами

- <script>
  function createAtt()
  {id=document.getElementById('heading'); id.createAttribute('align')}
- function setAtt()
  {id=document.getElementById('heading'); id.setAttribute('align','center')}
- function getAtt()
  {id=document.getElementById('heading'); alert(id.getAttribute('align'))}
- function removeAtt()
  {id=document.getElementById('heading'); id.removeAttribute('align')}
  </script>
- <body>
  Работа с атрибутом узла
  <button onclick=createAtt()>Создать атрибут</button>
  <button onclick=setAtt()>Установить атрибут</button>
  <button onclick=getAtt()>Получить атрибут</button>
  <button onclick=removeAtt()>Удалить атрибут</button>
  </body>



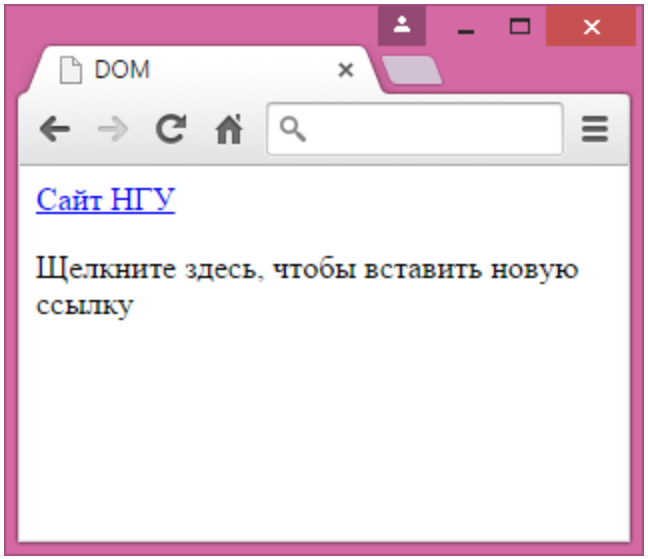
ПРИМЕРЫ РАБОТЫ С DOM

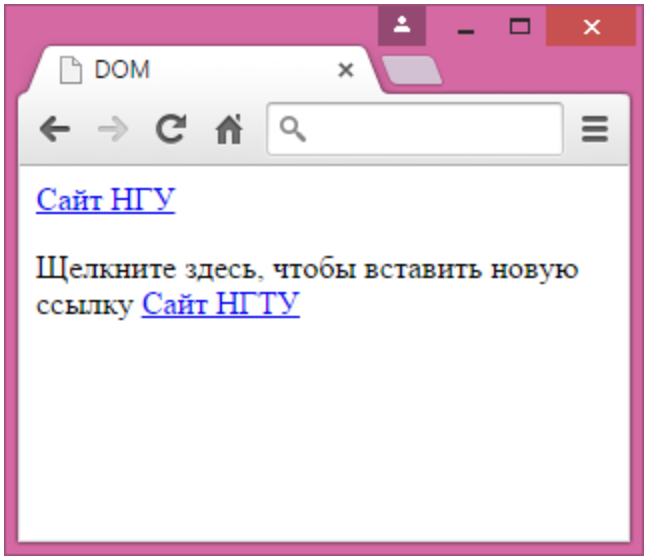
Вставка новой ссылки

JS

- <script>
  function addlink()
  {
   mylink=document.createElement('a')
   mylink.setAttribute('href','http://math.isu.ru')
   mylink.setAttribute('name','nstu')
   text=document.createTextNode('Сайт ИГУ')
   mylink.appendChild(text)
   document.getElementById('firstlink').appendChild(mylink)
   }
  </script>
- Сайт ИГУ
  Щелкните здесь, чтобы вставить новую ссылку

Вставка новой ссылки





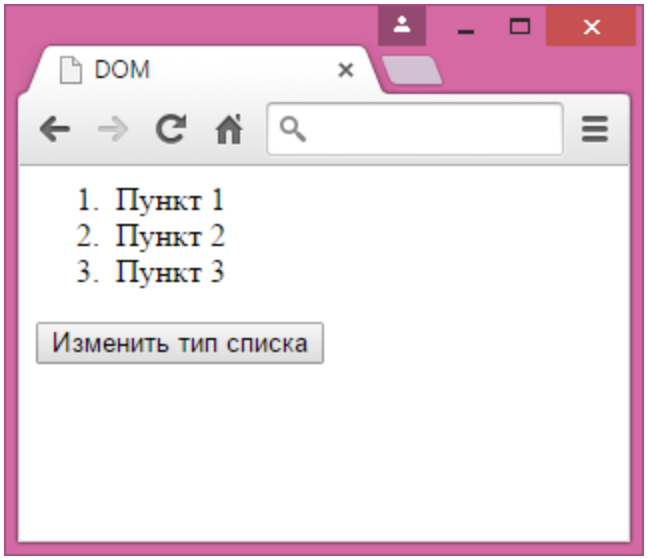
Изменение типа списка

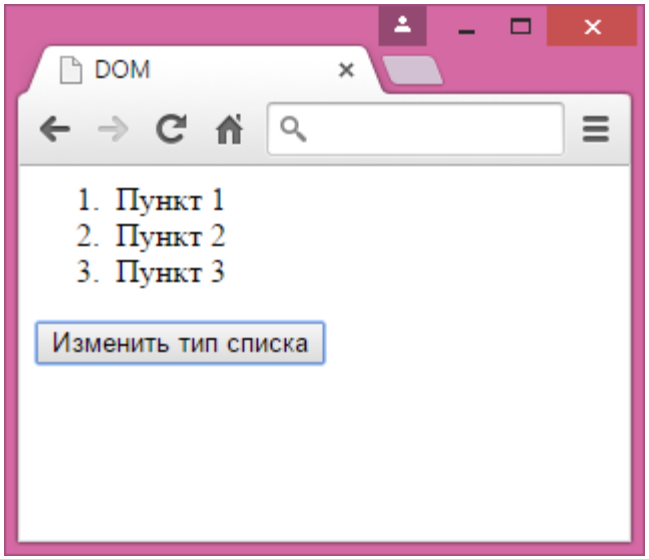
JS

- <script>
  function changelist()
  {
   oldlistitems=mylist.innerHTML
   newnode=document.createElement('ul')
   mylist.replaceNode(newnode)
   newnode.innerHTML=oldlistitems
  }
  </script>
- <ol id=mylist>
   Пункт 1
   Пункт 2
   Пункт 3

- <button onclick=changelist()>Изменить тип списка</button>

Изменение типа списка

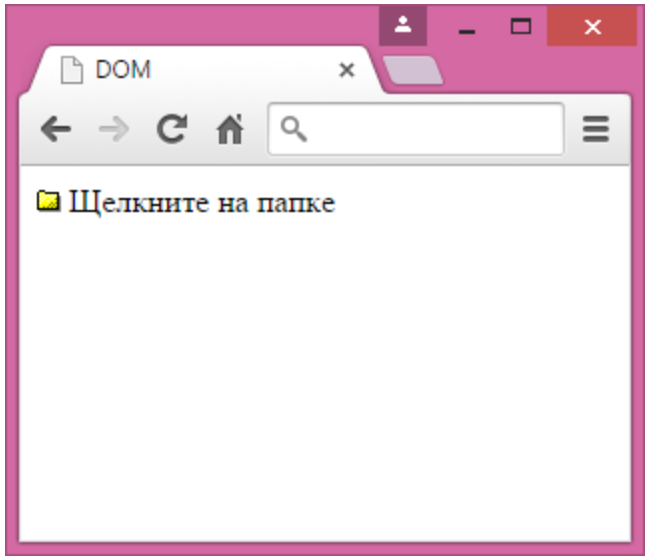


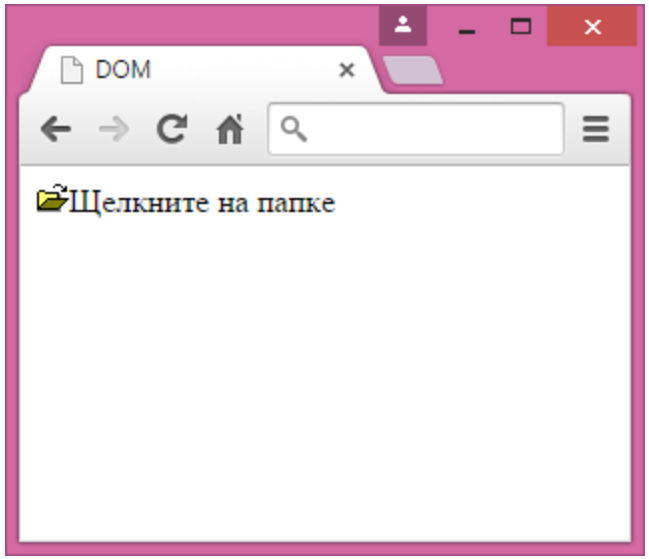


Замена изображения

- <img id="img" src="open1.gif" onClick="changeimg()">
  Щелкните на папке
- <script>
  tr=1;
  function changeimg()
  {
  elem=document.all('img')
  if (tr==1)
  {elem.outerHTML='<img id="img" src="open2.gif"
  onClick="changeimg()">'
  tr=2}
  else
  {elem.outerHTML="<img id='img' src='open1.gif'
  onClick='changeimg()'>"
  tr=1}}
  </script>

Замена изображения



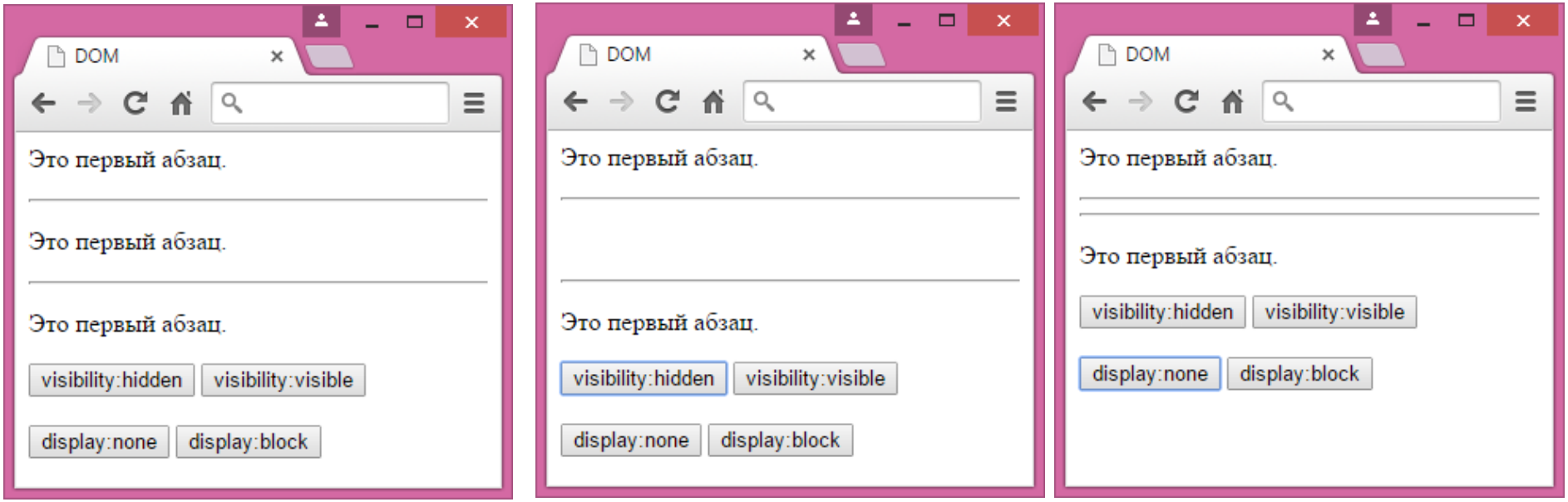


Отображение и скрытие

```
<p id="p1">Это первый абзац.</p><hr>
<p id="p2">Это первый абзац.</p><hr>
<p id="p3">Это первый абзац.</p>


<input type="button"
onclick="document.getElementById('p2').style.visibility='hidden'"
>
<input type="button"
onclick="document.getElementById('p2').style.visibility='visible'">
<input type="button"
onclick="document.getElementById('p2').style.display='none'">
<input type="button"
onclick="document.getElementById('p2').style.display='block'">
```

Отображение и скрытие



ОБРАБОТКА СОБЫТИЙ

События

JS

- Для реакции на действия посетителя и внутреннего взаимодействия скриптов существуют *события*.
- *Событие* – это сигнал от браузера о том, что что-то произошло.

События

- Мышь:
  - click – происходит, когда кликнули на элемент левой кнопкой мыши
  - contextmenu – происходит, когда кликнули на элемент правой кнопкой мыши
  - mouseover – возникает, когда на элемент наводится мышь
  - mousedown и mouseup – когда кнопку мыши нажали или отжали
  - mousemove – при движении мыши
- Элементах управления:
  - submit – посетитель отправил форму <form>
  - focus – посетитель фокусируется на элементе, например нажимает на <input>
- Клавиатура:
  - keydown – когда посетитель нажимает клавишу
  - keyup – когда посетитель отпускает клавишу
- Документ:
  - DOMContentLoaded – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

Обработка событий

JS

- Событию можно назначить *обработчик*, то есть функцию, которая сработает, как только событие произошло.
- Именно благодаря обработчикам JavaScript-код может реагировать на действия посетителя.

1. Через атрибут HTML
2. Через свойства DOM-объекта
3. Через метод addEventListener

Через атрибут HTML

JS

```
<input
  value="Нажми меня“
  onclick="alert('Клик!')"
  type="button">
<input
  value="Нажми меня"
  onclick="someFunction()"
  type="button">
```

Через свойства DOM-объекта

```
<input
  id="elem"
  type="button"
  value="Нажми меня" />

<script>
  elem.onclick = function() {
     alert( 'Спасибо' );
  };
</script>
```

Через addEventListener

JS

```
elem.addEventListener(
  "click" ,
  function() {
    alert('Спасибо!')
  }
)
```

Через addEventListener

JS

```
<input id="elem" type="button" value="Нажми меня"/>
<script>
  function handler1() { alert('Спасибо!'); };
  function handler2() { alert('Спасибо ещё раз!'); }
  elem.onclick = function() { alert("Привет"); };
  elem.addEventListener("click", handler1);
  elem.addEventListener("click", handler2);
</script>

// Привет, Спасибо!, Спасибо ещё раз!
```

Через addEventListener

JS

```
function addHandler(obj, msg) {
  function say() {
    alert(msg)
  }
  obj.addEventListener('click',say);
}
```