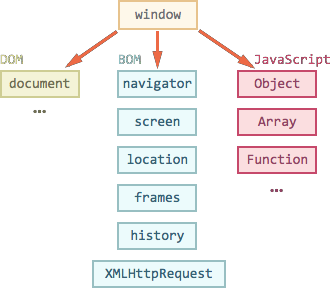
***Структура браузерних об’єктів*** DOM, BOM і JS

Структуру браузерних об'єктів можна зобразити такою схемою



**B**rowser **O**bject **M**odel (BOM)

У таблиці розглянуто деякі із складових

|  |  |
| --- | --- |
|  |  |
| **Navigator** | **window.navigator** містить інформацію про браузер відвідувача  деякі властивості:   * navigator.appName * navigator.appCodeName * navigator.platform |
| **Screen** | Об’єкт **window.screen** містить інформацію про екран користувача.  Властивості   * screen.width * screen.height * screen.availWidth * screen.availHeight * screen.colorDepth   screen.pixelDepth |
| **Location** | Об’єкт **location** може бути використатий для одержання поточної адреси і переадресації на нову сторінку.  Деякі властивості:   * window.location.href returns the href (URL) of the current page * window.location.hostname returns the domain name of the web host * window.location.pathname returns the path and filename of the current page * window.location.protocol returns the web protocol used (http: or https:) * window.location.assign loads a new document |
| **History** | **history** містить історію браузера  Деякі властивості:   * history.back() - same as clicking back in the browser * history.forward() - same as clicking forward in the browser |

**Об’єкт window**

Як видно з рисунку основним об’єктом є об’єкт window. Даний об’єкт представляє вікно браузера і містить властивості методи для управління вікном браузера.

Методи

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| [alert()](https://puzzleweb.ru/javascript/window_alert.php) | Выводит модальное диалоговое окно с сообщением и кнопкой OK. |
| atob() | Декодирует строку данных, которая была закодирована с использованием кодировки base-64. |
| btoa() | Создает строку ASCII в кодировке base-64 из строки бинарных данных. |
| [clearInterval()](https://puzzleweb.ru/javascript/window_clearinterval.php) | Отменяет выполнение действий, заданных с помощью метода setInterval(). |
| [clearTimeout()](https://puzzleweb.ru/javascript/window_cleartimeout.php) | Отменяет выполнение действий, заданных с помощью метода setTimeout(). |
| close() | Закрывает окно, которое было открыто с помощью метода window.open(). |
| confirm() | Отображает модальное диалоговое окно, содержащее сообщение и кнопки ОК и Отмена. |
| focus() | Устанавливает фокус на текущее окно. |
| moveBy() | Перемещает текущее окно на заданную величину. |
| moveTo() | Перемещает окно в соответствии с указанными координатами. |
| open() | Создает и открывает новое окно.  // Відкрити нове вікно / вкладку з URL [http://ya.ru](http://ya.ru/) window.open ('http://ya.ru'); |
| print() | Печатает содержимое текущего окна. |
| prompt() | Отображает диалоговое окно с сообщением и полем ввода для пользователя. Возвращает строку, содержащую введенные данные. |
| resizeBy() | Изменяет текущее окно на определенную величину. |
| resizeTo() | Изменяет размер окна до указанной ширины и высоты. |
| scrollBy() | Прокрутка документа в окне на указанное количество пикселей. |
| scrollTo() | Прокрутка документа до указанных координат. |
| [setInterval()](https://puzzleweb.ru/javascript/window_setinterval.php) | Вызывает функцию или вычисляет выражение в определенные промежутки времени (в миллисекундах). |
| [setTimeout()](https://puzzleweb.ru/javascript/window_settimeout.php) | Вызывает функцию или вычисляет выражение после указанного числа миллисекунд. |
| stop() | Останавливает загрузку окна. |

### Властивості

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| closed | Возвращает логическое значение, указывающее закрыто окно или открыто. |
| [document](https://puzzleweb.ru/javascript/dom_document.php) | Возвращает объект Document текущего окна. |
| frames | Возвращает массив со всеми элементами [<iframe>](https://puzzleweb.ru/html/tag_iframe.php) текущего окна. |
| [history](https://puzzleweb.ru/javascript/bom_history.php) | Возвращает ссылку на объект History. |
| innerHeight | Возвращает высоту области просмотра окна. |
| innerWidth | Возвращает ширину области просмотра окна. |
| lenght | Возвращает количество [<iframe>](https://puzzleweb.ru/html/tag_iframe.php) элементов в окне. |
| [location](https://puzzleweb.ru/javascript/bom_location.php) | Возвращает ссылку на объект Location. |
| name | Задает или получает значение, указывающее имя окна. |
| [navigator](https://puzzleweb.ru/javascript/bom_navigator.php) | Возвращает ссылку на объект Navigator. |
| opener | Задает или получает ссылку на окно, которое было создано в текущем окне. |
| outerHeight | Возвращает внешнюю высоту окна, включая панели инструментов и полосы прокрутки. |
| outerWidth | Возвращает внешнюю ширину окна, включая панели инструментов и полосы прокрутки. |
| pageXOffset | Возвращает количество пикселей, на которое текущий документ был прокручен (по горизонтали) от верхнего левого угла окна. |
| pageYOffset | Возвращает количество пикселей, на которое текущий документ был прокручен (по вертикали) от верхнего левого угла окна. |
| parent | Возвращает родительское окно текущего окна. |
| screen | Возвращает ссылку на объект Screen, связанный с окном. |
| screenLeft | Получает x-координату верхнего левого угла окна относительно верхнего левого угла экрана. |
| screenTop | Получает y-координату верхнего угла окна, по отношению к верхней части экрана. |
| scrollX | Эквивалент свойства pageXOffset. |
| scrollY | Эквивалент свойства pageYOffset. |
| self | Извлекает ссылку на текущее окно или фрейм. |
| status | Получает/устанавливает текст в строке состояния в нижней части браузера. |

**Об’єкт Document**

Кожна сторінка у браузері має свій власний об’єкт Document. Цей об’єкт використовується для отримання інформації з сторінки, зміни інформації та додавання нової інформації.

### Методы

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| [addEventListener()](https://puzzleweb.ru/javascript/element_addeventlistener.php) | Позволяет установить обработчик события для документа. |
| [createElement()](https://puzzleweb.ru/javascript/document_createelement.php) | Создает новый элемент. |
| createTextNode() | Создает и возвращает текстовый узел с указанным текстом. Принимает один аргумент (строку с текстом). |
| [getElementById()](https://puzzleweb.ru/javascript/document_getelementbyid.php) | Возвращает ссылку на элемент, с указанным идентификатором. |
| [getElementsByClassName()](https://puzzleweb.ru/javascript/document_getelementsbyclassname.php) | Возвращает коллекцию элементов, соответствующих указанному классу. |
| getElementsByName() | Возвращает коллекцию всех элементов в документе, которые имеют атрибут name с указанным значением. Элементы хранятся в коллекции в том порядке, в котором они встречаются в исходном коде документа.  Метод document.getElementsByName() относится к объекту HTMLDocument, а не к объекту Document, поэтому он доступен только в HTML-документах и не доступен в XML-документах. |
| [getElementsByTagName()](https://puzzleweb.ru/javascript/document_getelementsbytagname.php) | Возвращает массив элементов, указанного типа. |
| [querySelector()](https://puzzleweb.ru/javascript/document_queryselector.php) | Возвращает первый элемент, совпавший с одним из указанных селекторов. |
| [querySelectorAll()](https://puzzleweb.ru/javascript/document_queryselectorall.php) | Возвращает массив элементов, соответствующих указанным селекторам. |
| [removeEventListener()](https://puzzleweb.ru/javascript/element_removeeventlistener.php) | Удаляет обработчик события, который был установлен с помощью метода addEventListener(). |
| [write()](https://puzzleweb.ru/javascript/3_out.php#a3) | Записывает переданные ему аргументы в документ. |
| writeln() | Записывает переданные ему аргументы в документ, вставляя после вывода данных пробел. |

### Свойства

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| activeElement | Возвращает элемент, который в данный момент находится в фокусе. |
| documentURI | Возвращает строку, содержащую URL-адрес документа. |
| images | Возвращает массив, содержащий все элементы [<img>](https://puzzleweb.ru/html/tag_img.php) документа. |
| links | Возвращает массив, содержащий все элементы [<a>](https://puzzleweb.ru/html/tag_a.php) и [<area>](https://puzzleweb.ru/html/tag_area.php), которые имеют атрибут href. |

Згідно DOM-моделі (Document Object Model), документ є ієрархією, деревом. Кожен HTML-тег утворює вузол дерева з типом «елемент». Вкладені в нього теги стають дочірніми вузлами. Для представлення тексту створюються вузли з типом«текст».

|  |  |
| --- | --- |
| Документ | Схематичне представлення |
| <! DOCTYPE HTML>  <Html>  <Head>  <Title>О лосях</Title>  </ Head>  <Body>Правда о лосях</Body>  </Html> |  |

Теги утворюють **вузли-елементи (element node).** Текст всередині елементів утворює **текстові вузли (text node),** позначені як #text. Текстовий вузол містить виключно рядок тексту і не може мати нащадків, тобто він завжди на самому нижньому рівні.Зверніть увагу на спеціальні символи в текстових вузлах:

Перевід рядка: ↵

пробіл: ␣

Пробіли і перевід рядка – це теж текст, повноправні символи, які враховуються в DOM.

# Усього розрізняють 12 типів вузлів, але на практиці ми працюємо з чотирма з них:

Документ - точка входу в DOM.

Елементи - основні будівельні блоки. Текстові вузли - містять, власне, текст.

Коментарі - іноді в них можна включити інформацію, яка не буде показана, але доступна з JS.

Вузол HTML можна отримати як **document.documentElement**, а BODY - як document.body.

# document.body.style.backgroundColor = 'red';

IE8- не генерує текстові вузли, якщо вони складаються тільки з пропусків.

**Об’єкт Element**

Об’єкт Element є інтерфейсом, який надає доступ до властивостей і методів елементав

### Методы

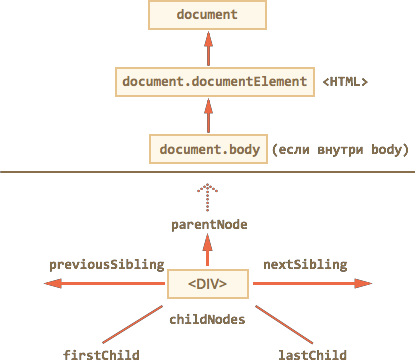
|  |  |
| --- | --- |
| **Метод** | **Описание** |
| [addEventListener()](https://puzzleweb.ru/javascript/element_addeventlistener.php) | Позволяет установить обработчик события для элемента. |
| [appendChild()](https://puzzleweb.ru/javascript/element_appendchild.php) | Добавляет дочерний узел в конец родительского элемента. |
| getAttribute() | Извлекает значение указанного атрибута. |
| [getElementsByClassName()](https://puzzleweb.ru/javascript/document_getelementsbyclassname.php) | Возвращает коллекцию элементов, соответствующих указанному классу. |
| [getElementsByTagName()](https://puzzleweb.ru/javascript/document_getelementsbytagname.php) | Возвращает массив элементов, указанного типа. |
| hasAttribute() | Возвращает значение true, если элемент имеет указанный атрибут, или false в обратном случае. |
| [insertBefore()](https://puzzleweb.ru/javascript/element_insertbefore.php) | Вставляет элемент перед указанным элементом. |
| removeAttribute() | Удаляет указанный атрибут из текущего элемента. |
| [removeChild()](https://puzzleweb.ru/javascript/element_removechild.php) | Удаляет указанный дочерний узел из текущего родительского элемента. |
| [removeEventListener()](https://puzzleweb.ru/javascript/element_removeeventlistener.php) | Удаляет обработчик события, который был установлен с помощью метода addEventListener(). |
| [setAttribute()](https://puzzleweb.ru/javascript/element_setattribute.php) | Добавляет к элементу атрибут и устанавливает для него значение. |

### Свойства

|  |  |
| --- | --- |
| **Свойство** | **Описание** |
| accessKey | Позволяет получить или задать клавишу доступа для элемента. |
| *атрибут*.value | Позволяет получить или задать значение для атрибута. |
| clientHeight | Возвращает высоту элемента в пикселях, включая внутренние отступы ([padding](https://puzzleweb.ru/css/pr_padding.php)). Внешние отступы ([margin](https://puzzleweb.ru/css/pr_margin.php)), рамки ([border](https://puzzleweb.ru/css/pr_border.php)) и горизонтальная полоса прокрутки (если она есть) не учитываются. |
| clientWidth | Возвращает ширину элемента в пикселях, включая внутренние отступы ([padding](https://puzzleweb.ru/css/pr_padding.php)). Внешние отступы ([margin](https://puzzleweb.ru/css/pr_margin.php)), рамки ([border](https://puzzleweb.ru/css/pr_border.php)) и вертикальная полоса прокрутки (если она есть) не учитываются. |
| [firstChild](https://puzzleweb.ru/javascript/element_firstchild.php) | Возвращает ссылку на первый дочерний узел внутри указанного узла. |
| firstElementChild | Возвращает первый дочерний элемент, если дочерних элементов нет, то возвращает null. |
| lastElementChild | Возвращает последний дочерний элемент, если дочерних элементов нет, то возвращает null. |
| [innerHTML](https://puzzleweb.ru/javascript/element_innerhtml.php) | Представляет содержимое, которое находится между открывающим и закрывающим тегами текущего элемента. |
| [parentNode](https://puzzleweb.ru/javascript/element_parentnode.php) | Возвращает родительский элемент указанного узла. |
| [style](https://puzzleweb.ru/javascript/element_style.php) | Объект, представляющий атрибут style элемента, он позволяет получить или установить встроенный стиль элемента. |
| textContent | Содержит текстовое содержимое узла и его потомков.   |  |  | | --- | --- | | 1  2  3  4  5  6  7 | <p>Внутри элемента &lt;p&gt;. <em>Внутри элемента &lt;em&gt;.</em></p>    <script>  var p = document.querySelector("p");    alert(p.textContent);   // Внутри элемента <p>. Внутри элемента <em>.  </script> | |

# Навігація по DOM-елементах

Основні посилання, за якими можна переходити між вузлами DOM:



# <HTML> = document.documentElement

**<BODY> = document.body**

У світі DOM як значення, що означає «немає такого елемента» або «вузол не знайдений», використовується не undefined, **а null.**

# Діти: childNodes, firstChild, lastChild

**Дочірні елементи (або діти)** - елементи, які лежать безпосередньо всередині даного. Наприклад, усередині <HTML> зазвичай лежать <HEAD> і

<BODY>.

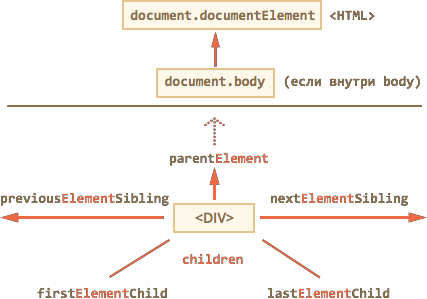
**Нащадки** - всі елементи, які лежать всередині даного, разом з їхніми дітьми, дітьми їхніх дітей і так далі. Тобто, все піддерево DOM.

Псевдо-масив **childNodes** зберігає всі дочірні елементи, включаючи текстові.

# Сусіди і батько

Доступ до елементів зліва і справа даного можна отримати по посиланнях **previousSibling** / **nextSibling.** Батько доступний через **parentNode**. Якщо довго йти від одного елемента до іншого, то рано чи пізно дійдеш до кореня DOM, тобто до document.documentElement, а потім і document.

# Навігація тільки за елементами



Ці посилання схожі на ті, що раніше, тільки в ряді місць стоїть слово Element:

**children** - тільки дочірні вузли-елементи, тобто відповідні тегами.

**firstElementChild**, **lastElementChild** - відповідно, перший і останній діти-елементи.

**previousElementSibling**, **nextElementSibling** - сусіди-елементи.

**parentElement** - батько-елемент.

Властивість **elem.parentNode** повертає батько елемента.Воно завжди одно

**parentElement**, крім одного винятку:

# alert (document.documentElement.parentNode); // Document alert (document.documentElement.parentElement); // Null Завжди вірні рівності:

**elem.firstElementChild === elem.children [0] elem.lastElementChild === elem.children [elem.children.length - 1]** У IE8- підтримується тільки children

====================================================

**Вибірка елементів сторінки**

|  |  |
| --- | --- |
|  |  |
| [getElementById()](https://puzzleweb.ru/javascript/document_getelementbyid.php) | Повертає елемент з вказаним ідентифікатором |
| [getElementsByClassName()](https://puzzleweb.ru/javascript/document_getelementsbyclassname.php) | Повертає колекцію елементів, які відповідають вказаному класу |
| getElementsByName() | Повертає колекцію елементів, які мають атрибут name з вказаним значенням.  Метод document.getElementsByName() относится к объекту HTMLDocument, а не к объекту Document, поэтому он доступен только в HTML-документах и не доступен в XML-документах. |
| [getElementsByTagName()](https://puzzleweb.ru/javascript/document_getelementsbytagname.php) | Повертає колекцію елементів вказаного типу |
| [querySelector()](https://puzzleweb.ru/javascript/document_queryselector.php) | Повертає перший елемент, який відповідає вказаному селектору |
| [querySelectorAll()](https://puzzleweb.ru/javascript/document_queryselectorall.php) | Повертає масив елементів, які відповідають вказаному селектору |

**document.getElementById ("ідентифікатор")**

Якщо елементу призначений спеціальний атрибут id, то можна отримати його викликом **document.getElementById ("ідентифікатор").**

# Приклад.

<div id="content"> Виділимо цей елемент </div>

<script>

var elem = document.getElementById('content'); elem.style.background = 'red';

alert( elem == content ); // true

content.style.background = ""; // Один і той же елемент

</script>

**elem.getElementsByTagName (tag)**

Метод **elem.getElementsByTagName (tag)** шукає всі елементи з заданим тегом tag всередині елемента elem і повертає їх у вигляді списку. Регістр тега не має значення.

# // Отримати всі div-елементи

**var elements = document.getElementsByTagName ('div');**

Звернемо увагу: на відміну від getElementById, який існує тільки в контексті document, метод getElementsByTagName може шукати всередині будь-якого елементу.

Наприклад, знайдемо всі елементи input всередині таблиці:

# Приклад 15.1.2

<table id="age-table">

<tr>

<td>Ваш возраст:</td>

<td>

<label>

<input type="radio" name="age" value="young" checked> младше 18

</label>

<label>

<input type="radio" name="age" value="mature"> от 18 до 50

</label>

<label>

<input type="radio" name="age" value="senior"> старше 60

</label>

</td>

</tr>

</table>

<script>

var tableElem = document.getElementById('age-table'); var elements = tableElem.getElementsByTagName('input'); for (var i = 0; i < elements.length; i++) {

var input = elements[i];

alert( input.value + ': ' + input.checked );

}

</script>

**Можна отримати всіх нащадків, передавши зірочку "\*" замість тега:**

// Отримати всі елементи документа

document.getElementsByTagName('\*');

// Отримати всіх нащадків елемента elem:

elem.getElementsByTagName('\*'); Повертається колекція, а не елемент. Колекцію потрібно або перебрати в циклі або отримати елемент за номером і вже йому привласнювати value,

наприклад так:

# document.getElementsByTagName ('input') [0] .value = 5;

**getElementsByName**

Виклик **document.getElementsByName (name)** дозволяє отримати всі елементи **з даними атрибутом name.**

Наприклад, всі елементи з ім'ям age:

# var elems = document.getElementsByName ('age');

До появи стандарту HTML5 цей метод повертав тільки ті елементи, в яких передбачена підтримка атрибуту name, зокрема: iframe, a, input та іншими. У сучасних браузерах (IE10 +) тег не має значення.

# getElementsByClassName

Виклик **elem.getElementsByClassName (className)** повертає колекцію елементів з класом className. Знаходить елемент і в тому випадку, якщо у нього кілька класів, а шуканий - один з них.

Підтримується всіма сучасними браузерами, крім IE8-.

# Приклад 15.1.3

<div class="article">Статья</div>

<div class="long article">Длинная статья</div>

<script>

var articles = document.getElementsByClassName('article'); alert( articles.length ); // 2, найдѐт оба элемента

</script>

Як і getElementsByTagName, цей метод може бути викликаний і в

контексті DOM-елемента і в контексті документа.

# querySelectorAll

Виклик **elem.querySelectorAll (css)** повертає всі елементи всередині elem, що задовольняють CSS-селектору css. Він є у всіх сучасних браузерах, включаючи IE8 + (в режимі відповідності стандарту).

Наступний запит отримує всі елементи LI, які є останніми нащадками в

UL:

# Приклад

<ul>

<li>Цейт</li>

<li>тест</li>

</ul>

<ul>

<li>повністю</li>

<li>пройдений</li>

</ul>

<script>

var elements = document.querySelectorAll('ul > li:last-child');

for (var i = 0; i < elements.length; i++) {

alert( elements[i].innerHTML ); // "тест", "пройден"

}

</script>

**querySelector**

Виклик **elem.querySelector (css)** повертає не всі, а тільки перший елемент, відповідний CSS-селектору css. Інакше кажучи, результат - такий же, як і при **elem.querySelectorAll (css) [0],** але в останньому виклику спочатку шукаються всі елементи, а потім береться перший, а в **elem.querySelector (css)** шукається тільки перший, тобто він ефективніше.

# matches

Метод **elem.matches (css)** нічого не шукає, а перевіряє, чи задовольняє elem селектору css. Він повертає true або false. не підтримується в IE8-. Цей метод буває корисним, коли ми перебираємо елементи (в масиві або за звичайними навігаційним посиланнях) і намагаємося відфільтрувати ті з них, які нам цікаві.

# closest

Метод **elem.closest (css)** шукає найближчий елемент вище по ієрархії DOM, що підходить під CSS-селектор css. Сам елемент теж включається в пошук.Інакше кажучи, метод closest біжить від поточного елемента вгору по ланцюжку батьків і перевіряє, чи підходить елемент під вказаний CSS- селектор. Якщо підходить - зупиняється і повертає його.

# nodeName і tagName

Існує цілих дві властивості: nodeName і tagName, які містять назву (тег) елемента вузла.

Назва HTML-тега завжди знаходиться у верхньому регістрі.Наприклад, для document.body:

# alert (document.body.nodeName); // BODY alert (document.body.tagName); // BODY

<body>

<!-- комментарий -->

<script>

// для комментария

alert( document.body.firstChild.nodeName ); // #comment

alert( document.body.firstChild.tagName ); // undefined (в IE8- воскл. знак "!")

// для документа

alert( document.nodeName ); // #document, т.к. корень DOM -- не элемент alert( document.tagName ); // undefined

</script>

</body>

При роботі з елементами, як це зазвичай буває, має сенс використовувати

властивість tagName - воно коротше.

# innerHTML: вміст елемента

Властивість innerHTML дозволяє отримати HTML-вміст елемента у вигляді рядка. У innerHTML можна і читати і писати.

Приклад виведе на екран весь вміст document.body, а потім замінить його на інше:

# Приклад 15.1.6

<body>

<p>Параграф</p>

<div>Div</div>

<script>

alert( document.body.innerHTML ); // читаем текущее содержимое document.body.innerHTML = 'Новий BODY!'; // заменяем содержимое

</script>

</body>

Для таблиць в IE9- - innerHTML тільки для читання

В Internet Explorer версії 9 і раніше, innerHTML доступно тільки для читання для елементів COL, COLGROUP, FRAMESET, HEAD, HTML, STYLE, TABLE, TBODY, TFOOT, THEAD, TITLE, TR.

Зокрема, в IE9- заборонений запис в innerHTML для будь-яких табличних елементів, крім комірок (TD / TH).

# Додавання innerHTML + = здійснює перезапис

**Якщо в innerHTML є тег script - він не буде виконаний.**

**outerHTML: HTML елемента цілком** Властивість **outerHTML** містить HTML елемента цілком. **Приклад 15.1.7.**

<div>Привет <b>Мир</b></div>

<script>

var div = document.body.children[0];

alert( div.outerHTML ); // <div>Привет <b>Мир</b></div>

</script>

Змінити outerHTML елемента неможливо.

# nodeValue / data: вміст текстового вузла

Властивість innerHTML є тільки у вузлів-елементів.

Вміст текстових вузлів або коментарів на читання і запис через властивість **data.**

# Приклад 15.1.8.

<body> Привет

<!-- Комментарий -->

<script>

for (var i = 0; i < document.body.childNodes.length; i++) { alert( document.body.childNodes[i].data );

}

</script> Пока

</body>

Якщо ви запустите цей приклад, то побачите, як виводяться послідовно:

Привіт - це вміст першого вузла (текстового). Коментар - це вміст другого вузла (коментаря).

Прогалини - це вміст невеликого пробільного вузла після коментарю до скрипта.

undefined - далі цикл дійшов до <script>, але це вузол-елемент, у нього немає data.

Взагалі кажучи, після <script> ... </ script> і до закриття </ body> в документі є ще один текстовий вузол. Однак, на момент роботи скрипта браузер ще не знає про нього, тому не виведе.

# Текст: textContent

Властивість textContent містить тільки текст всередині елемента, за вирахуванням всіх <тегів>.

Воно підтримується скрізь, крім IE8-, повертається в точності весь текст, включаючи переведення рядків і прогалини, але без тегів.Іншими словами, elem.textContent повертає конкатенацію всіх текстових вузлів всередині elem.

У цьому прикладі ім'я відвідувача потрапить в перший div як innerHTML, а в другій - як текст:

# Приклад 15.1.9.

<div></div>

<div></div>

<script>

var name = prompt("Введите имя?", "<b>Винни-пух</b>");

document.body.children[0].innerHTML = name; document.body.children[1].textContent = name;

</script>

Вивід:

# Винни-пух

<b>Винни-пух</b>

# Створення та видалення вузлів

Для створення елементів використовуються такі методи:

|  |  |
| --- | --- |
| **document.createElement (tag)** | Створює новий елемент із зазначеним тегом:  var div = **document.createElement** ('div'); |
| **document.createTextNode (text)** | Створює новий текстовий вузол з даним текстом:  var textElem = **document.createTextNode** ('Тут був я "); |

Створення повідомлення

У нашому випадку ми хочемо зробити DOM-елемент div, дати йому класи і заповнити текстом:

var div = document.createElement('div');

div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

Після цього коду у нас є готовий DOM-елемент. Поки що він присвоєний

в змінну div, але не видно, так як ніяк не пов'язаний зі сторінкою.

# Додавання елемента: appendChild, insertBefore

|  |  |
| --- | --- |
| parentElem.**appendChild** (elem) | Додає elem в кінець дочірніх елементів parentElem |
| parentElem.insertBefore (elem, nextSibling) | Вставляє elem в колекцію дітей parentElem, перед елементом nextSibling |

Щоб DOM-вузол був показаний на сторінці, його необхідно вставити в document.Для цього насамперед потрібно вирішити, куди ми будемо його вставляти. Припустимо, що ми вирішили, що вставляти будемо в якийсь елемент parentElem, наприклад var parentElem = document.body.

Для вставки всередину parentElem є такі методи:

# parentElem.appendChild (elem)

Додає elem в кінець дочірніх елементів parentElem. Наступний приклад додає новий елемент в кінець <ol>:

# Приклад 15.1.11.

<ol id="list">

<li>0</li>

<li>1</li>

<li>2</li>

</ol>

<script>

var newLi = document.createElement('li'); newLi.innerHTML = 'Привіт, світ!'; list.appendChild(newLi);

</script>

1. 0

2. 1

3. 2

4. Привіт, світ!

# parentElem.insertBefore (elem, nextSibling)

Вставляє elem в колекцію дітей parentElem, перед елементом nextSibling. Наступний код вставляє новий елемент перед другим <li>:

# Приклад 15.1.12.

<ol id="list">

<li>0</li>

<li>1</li>

<li>2</li>

</ol>

<script>

var newLi = document.createElement('li'); newLi.innerHTML = 'Привіт, світ!';

list.insertBefore(newLi, list.children[1]);

</script>

Для вставки елемента в початок досить вказати, що вставляти будемо перед першим нащадком:

# list.insertBefore(newLi, list.firstChild);

У читача, який подивиться на цей код уважно, напевно виникне питання:

«А що, якщо list взагалі порожній, в цьому випадку адже **list.firstChild = null**, чи відбудеться вставка?»Відповідь - так, відбудеться.

Справа в тому, що якщо другим аргументом вказати null, то insertBefore спрацює як appendChild:

# parentElem.insertBefore (elem, null);

// Те ж, що і:

# parentElem.appendChild (elem)

Так що insertBefore універсальний.

**Всі методи вставки повертають вставлений вузол.** Наприклад, parentElem.appendChild (elem) повертає elem. Додамо повідомлення в кінець <body>:

# Приклад 15.1.13.

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6; border-radius: 4px;

color: #3c763d; background-color: #dff0d8;

}

</style>

<body>

<h3>Моя страница</h3>

</body>

<script>

var div = document.createElement('div'); div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.appendChild(div);

</script>

... **А тепер - на початок <body>: Приклад 15.1.14.**

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6; border-radius: 4px;

color: #3c763d; background-color: #dff0d8;

}

</style>

<body>

<h3>Моя страница</h3>

</body>

<script>

var div = document.createElement('div'); div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.insertBefore(div, document.body.firstChild);

</script>

**==============================================================================**

**Клонування вузлів: cloneNode**

**Виклик elem.cloneNode (true**) створить «глибоку» копію елемента - разом з атрибутами, включаючи піделементи. Якщо ж викликати з аргументом false, то копія буде зроблена без дочірніх елементів. Це потрібно набагато рідше.

Приклад зі вставкою копії повідомлення:

# Приклад 15.1.15

<style>

.alert {

padding: 15px;

border: 1px solid #d6e9c6; border-radius: 4px;

color: #3c763d; background-color: #dff0d8;

}

</style>

<body>

<h3>Моя страница</h3>

</body>

<script>

var div = document.createElement('div'); div.className = "alert alert-success";

div.innerHTML = "<strong>Ура!</strong> Вы прочитали это важное сообщение.";

document.body.insertBefore(div, document.body.firstChild);

// создать копию узла

var div2 = div.cloneNode(true);

// копию можно подправить div2.querySelector('strong').innerHTML = 'Супер!';

// вставим еѐ после текущего сообщения div.parentNode.insertBefore(div2, div.nextSibling);

</script>

Зверніть увагу на останній рядок, яка вставляє div2 після div: div.parentNode.insertBefore (div2, div.nextSibling);

Для вставки нам потрібен майбутній батько. Ми, можливо, не знаємо, де точно знаходиться div (або не хочемо залежати від того, де він), але якщо потрібно вставити поруч з div, то батьком виразно буде div.parentNode. Ми хотіли б вставити після div, але методу insertAfter немає, є тільки insertBefore, тому вставляємо перед його правим сусідом div.nextSibling.

**=======================================================**

**Видалення вузлів**

Для видалення вузла є два методи:

|  |  |
| --- | --- |
| **parentElem.removeChild (elem)** | Видаляє elem зі списку дітей parentElem. |
| **parentElem.replaceChild (newElem, elem)** | Серед дітей parentElem видаляє elem і вставляє на його місце newElem. |

Обидва ці методи повертають віддалений вузол, тобто elem. Якщо потрібно, його можна вставити в інше місце DOM тут же або в майбутньому.Якщо ви хочете перемістити елемент на нове місце - не потрібно його видаляти зі старого.Всі методи вставки автоматично видаляють вставляється елемент зі старого місця.

# Приклад 15.1.16

<div>Первый</div>

<div>Второй</div>

<script>

var first = document.body.children[0]; var last = document.body.children[1];

// нет необходимости в предварительном removeChild(last) document.body.insertBefore(last, first); // поменять местами

</script>

**Метод document.write**

Метод document.write (str) працює тільки поки HTML-сторінка знаходиться в процесі завантаження. Він дописує текст в поточне місце HTML ще до того, як браузер побудує з нього DOM.

HTML-документ нижче буде містити 1 2 3.

# <body> 1

**<Script> document.write (2);**

**</Script> 3**

**</body>**

Також існує метод **document.writeln (str)** - не менш древній, який додає після str символ перекладу рядка "\ n". Методи document.write і document.writeln пишуть безпосередньо в текст документа, до того як браузер побудує з нього DOM, тому вони можуть записати в документ все, що завгодно, будь-які стилі і незакриті теги.Браузер врахує їх при побудові DOM, точно так само, як враховує чергову порцію HTML-тексту.Технічно, викликати document.write можна в будь-який час, однак, коли HTML завантажився, і браузер повністю побудував DOM, документ стає «закритим». Спроба дописати щось в закритий документ відкриває його заново. При цьому всі поточний зміст видаляється.

Наприклад, document.write використовують для вставки рекламних скриптів і різних лічильників, коли URL скрипта необхідно генерувати динамічно, додаючи в нього параметри з JavaScript, наприклад:

<script>

// в url указано текущее разрешение экрана посетителя

var url = ['http://a](http://ads.com/buyme?screen=%27)d[s.com/buyme?screen='](http://ads.com/buyme?screen=%27) + screen.width + "x" + screen.height;

// загрузить такой скрипт прямо сейчас document.write('<script src="' + url + '"></scr' + 'ipt>');

</script>