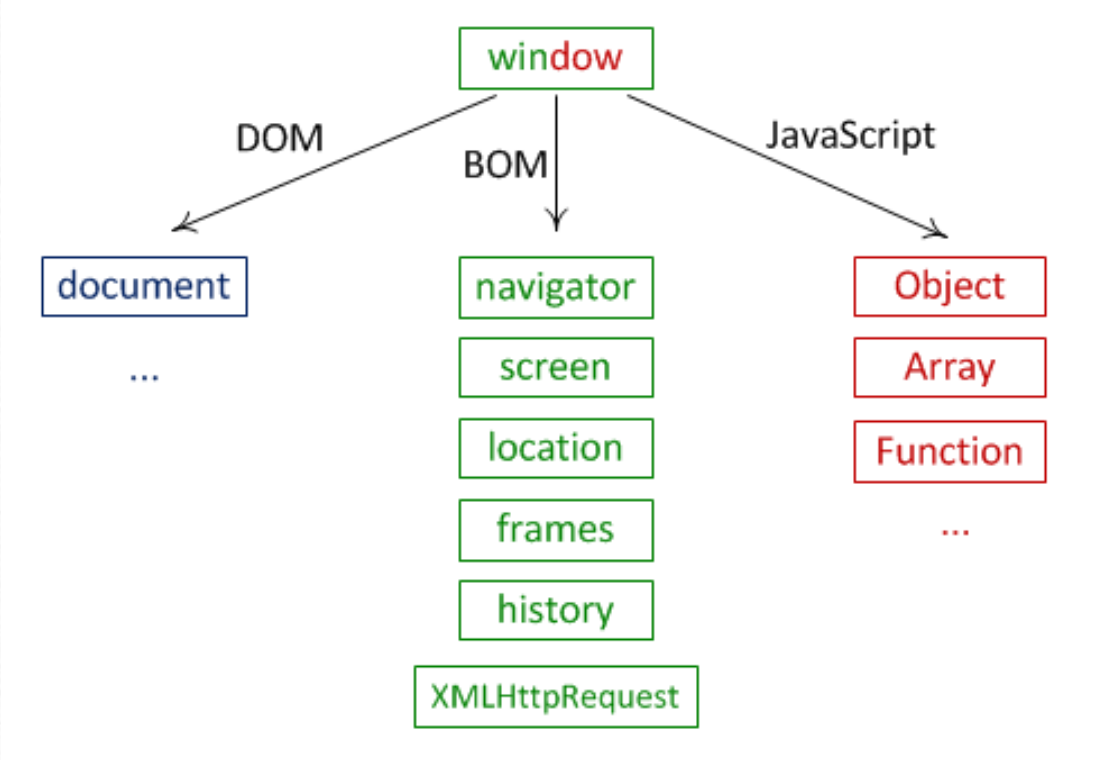
Інтернет програмування

лек 7

Лекція 7. DOM – Document Object Model

Зав.каф.КН, PhD Граф М.С.

Оточення: DOM, BOM та JS

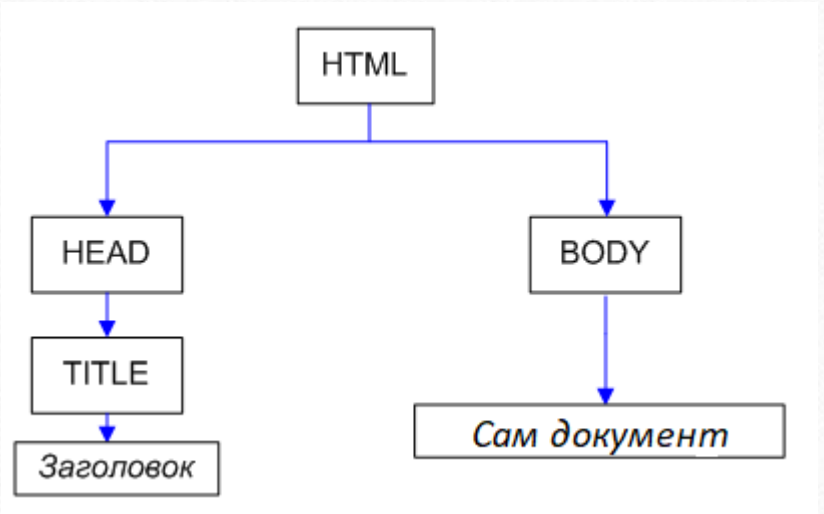


DOM у прикладах

- Основним інструментом роботи та динамічних змін на сторінці є **DOM (Document Object Model)** – об'єктна модель, що використовується для XML/HTML-документів.
- У DOM сторінки подаються як ієрархічного дерева елементів. Кожен лист цього дерева називається **вузлом** і безпосередньо пов'язаний з якимось елементом сторінки. Вузли, розташовані знизу, називаються **дочірніми** по відношенню до вищих.

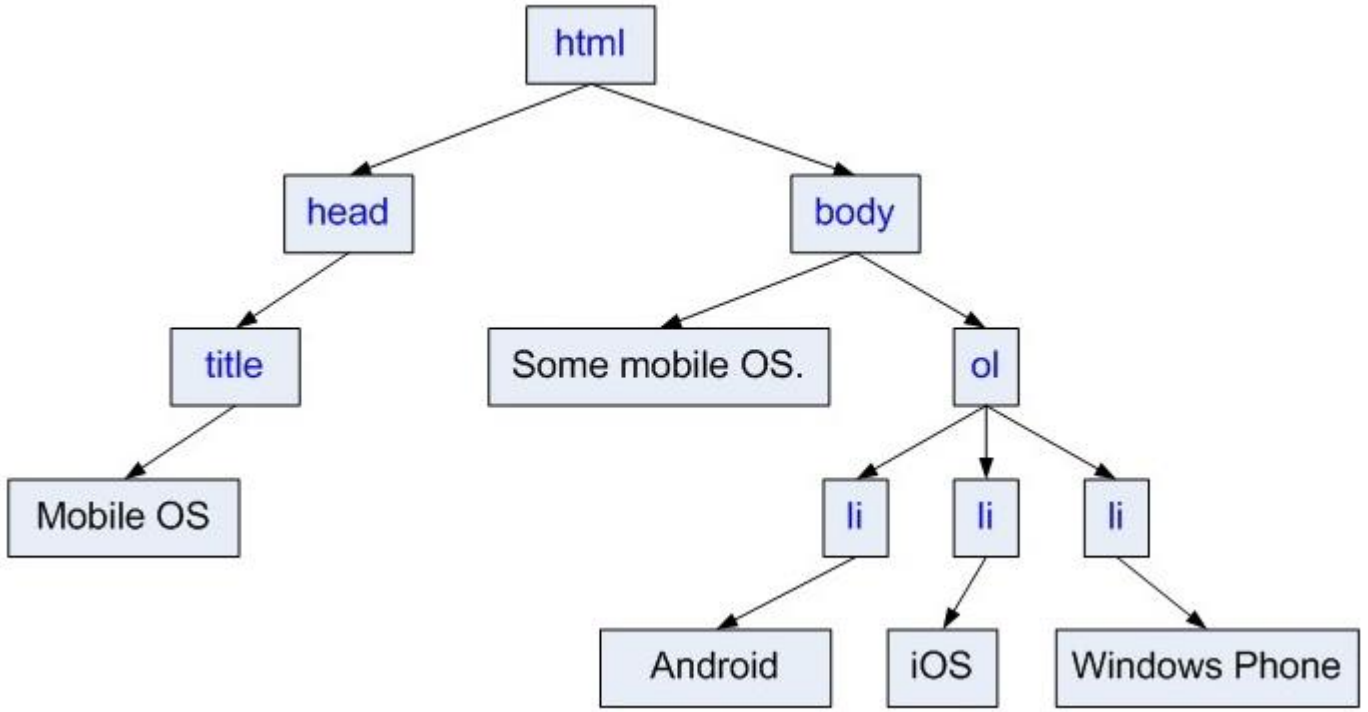
Найпростіший DOM

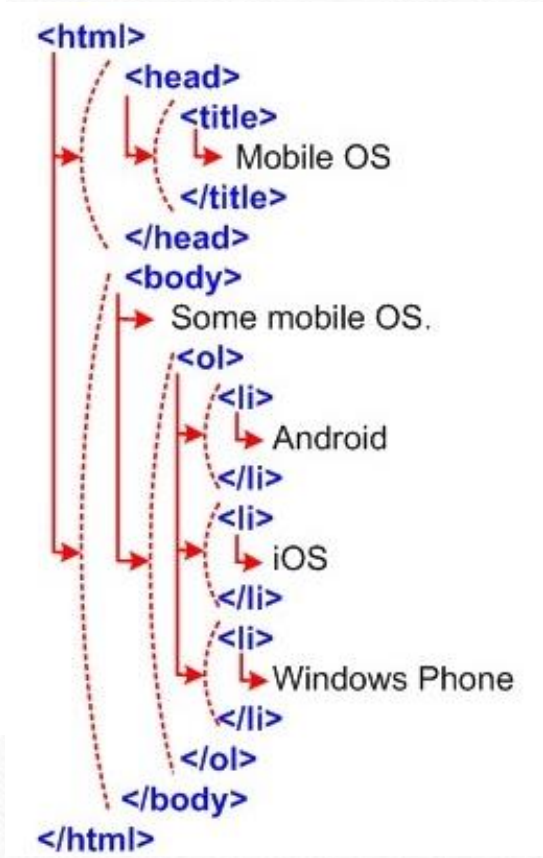
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Title</title>
</head>
<body>
   Сам документ
</body>
</html>
```



Теги утворюють вузли-елементи (**element node**). Текст представлений текстовими вузлами (**text node**). І те й інше – рівноправні вузли дерева **DOM** .

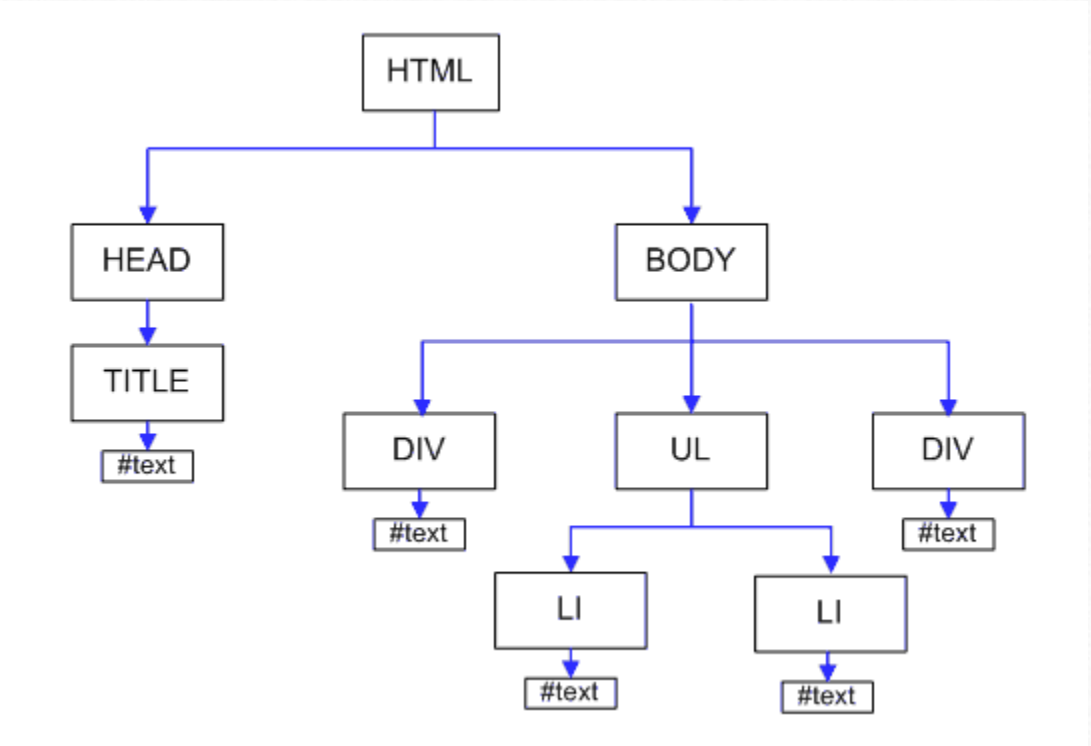
```
<html>
<head>
   <title>
      Mobile OS
   </title>
</head>
<body>
Some mobile OS.
<ol>
   <li>
      Android
   </li>
   <li>
      iOS
   </li>
   <li>
      Windows Phone
   </li>
</ol>
</body>
</html>
```





Приклад із атрибутами

```
<html>
<head>
   <title>Документ</title>
</head>
<body>
<div id="dataKeeper">Data</div>
<ul>
   <li style="background-color:red">Обережно</li>
   <li class="info">Інформація</li>
</ul>
<div id="footer">Made in Ukraine &copy;</div>
</body>
</html>
```



Можливості, які дає DOM

- Для маніпуляцій з DOM використовується об'єкт **document**.
- Наприклад, наступний код отримує перший елемент із тегом **ol** , послідовно видаляє два елементи списку і потім додає їх у зворотному порядку:

```
var ol = document.getElementsByTagName('ol')[0]
var android = ol.removeChild(ol.firstChild)
var ios = ol.removeChild(ol.firstChild)
var windows = ol.removeChild(ol.firstChild)
ol.appendChild(windows)
ol.appendChild(ios)
ol.appendChild(android)
```

Доступ до елементів

- **document.documentElement** - верхній тег.

Приклад при натисканні на кнопку видасть текстове представлення об'єктів document.documentElement та document.body.

```
<html>
<body>
<script>
   function go() {
      alert(document.documentElement)
      alert(document.body)
   }
</script>
<input type="button" onclick="go()" value="Go"/>
</body>
</html>
```

Типи DOM -елементів

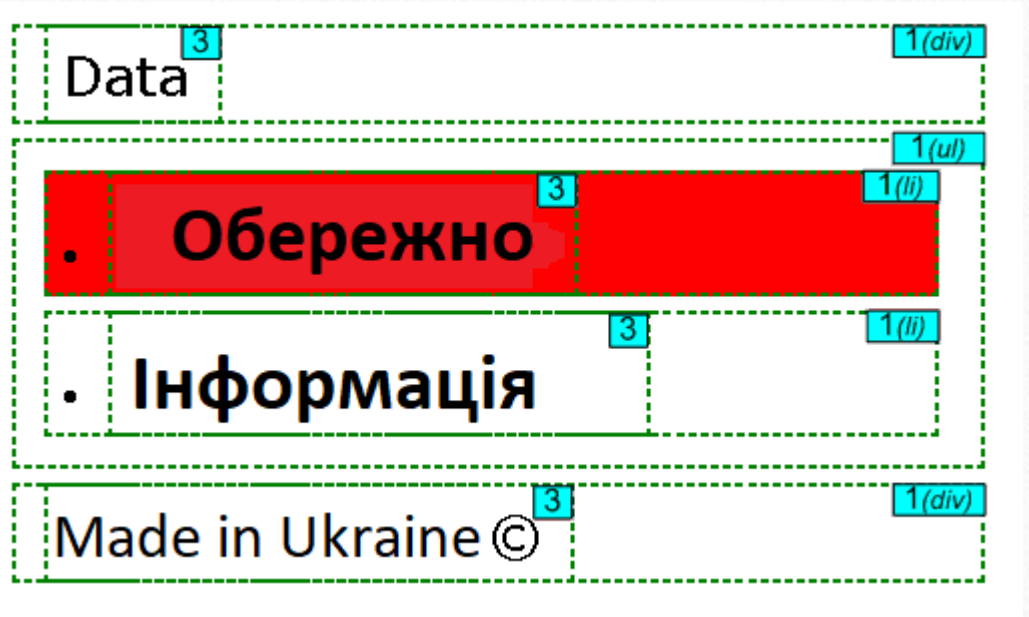
- Кожен елемент у DOM-моделі має тип. Його номер зберігається в атрибуті **elem.nodeType**
- тип **Node.TEXT _ NODE**

Наступний приклад при натисканні на кнопку виведе типи **document.documentElement**, а потім тип останнього нащадка вузла **document.body** .

```
<html>
<body>
<script>
   function go() {
      alert(document.documentElement.nodeType)
      alert(document.body.lastChild.nodeType)
   }
</script>
<input type="button" onclick="go()" value="Go"/>
Текст
</body>
</html>
```

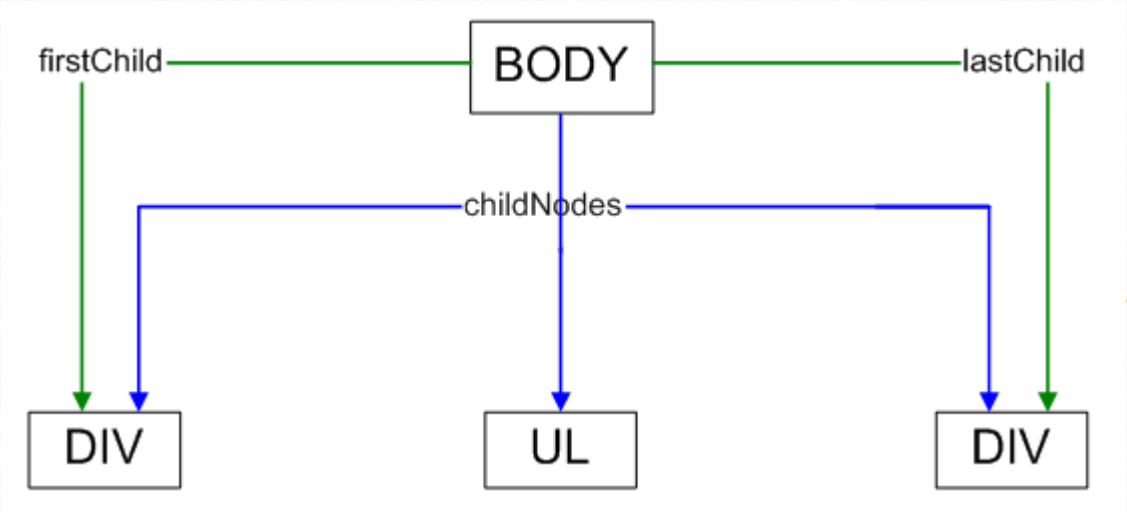
Якщо кожен видимий елемент обвести рамкою з цифрою **nodeType** у верхньому правому куті

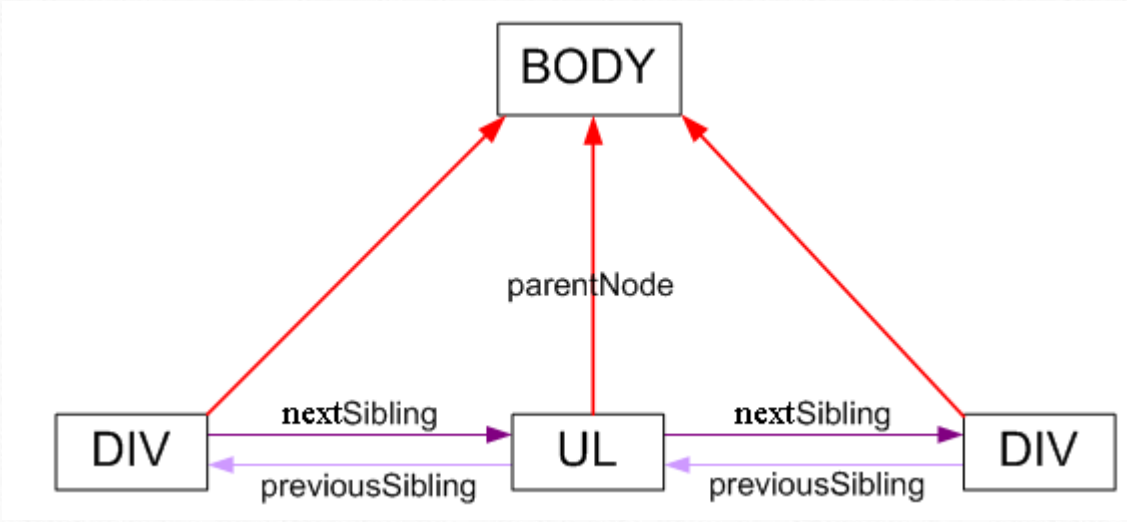
```
<html>
<head><title>...</title></head>
<body>
<div id="dataKeeper">Data</div>
<ul>
   <li style="background-color:red">Обережно</li>
   <li class="info">Інформація</li>
</ul>
<div id="footer">Made in Ukraine &copy;</div>
</body>
</html>
```



Дочірні елементи

- **Властивість 1:** Всі дочірні елементи, включаючи текстові, перебувають у масиві **childNodes** .
- **Властивість 2:** Властивості **firstChild** і **lastChild** показують на перший і останній дочірні елементи і дорівнюють null, якщо дітей немає.
- **Властивість 3:** Властивість **parentNode** вказує на батька. Наприклад, для <body> таким елементом є <html>.
- **Властивість 4:** Властивості **previousSibling** і **nextSibling** вказують на лівого і правого братів вузла.


firstChild
BODY
lastChild
childNodes
DIV
UL
DIV


BODY
parentNode
DIV
nextSibling
previousSibling
UL
nextSibling
previousSibling
DIV

Властивості елементів

- tagName

```
alert(document.body.tagName) // => BODY
```

- style

```
<input
      type="button"
      style="width: 300px"
      onclick="this.style.width = parseInt(this.style.width)-10+'px'"
      value="Укоротить на 10px"
/>
```

- className
- onclick, onkeypress, onfocus...

Властивості та атрибути

Властивості DOM-елементів

Вузол DOM — це об'єкт, тому, як і будь-який об'єкт у JavaScript, він може містити власні властивості та методи.

```
var elem = document.getElementById('MyElement')
elem.mySuperProperty = 5
```

Значенням якості може бути будь-який об'єкт.

```
elem.master = {
   name: petya
}
alert(elem.master.name)
```

Користувальницькі DOM-властивості:

- Можуть мати будь-яке значення.
- Працюють завдяки тому, що DOM-вузли є об'єктами JavaScript.
- Назви властивостей *чутливі* до регістру.

Властивості та атрибути

DOM-атрибути

У наступному прикладі елемент має атрибути id, class і нестандартний атрибут alpha.

```
<div id="MyElement" class="big" alpha="omega"></div>
```

setAttribute(name, value) – Встановлює значення атрибута.

Атрибути можна додавати, видаляти та змінювати. Для цього є спеціальні **методи** :
getAttribute(name) – Отримати значення атрибуту
hasAttribute(name) – Перевірити, чи є такий атрибут
removeAttribute(name) – Видалити атрибут

```
document.body.setAttribute('test', 123)
```

```
document.body.getAttribute('TEST') // 123
```

Синхронізація

Наприклад:

```
document.body.id = 5
alert(document.body.getAttribute('id'))
```

А тепер – навпаки

```
document.body.setAttribute('id', 'NewId')
alert(document.body.id)
```

Можливі значення

Атрибути з іменами "abc" і "ABC" - той самий атрибут.

```
document.body.setAttribute('abc', 1)
document.body.setAttribute('ABC', 5)
alert(document.body.getAttribute('abc')) // => стало 5
```

Але якості у різних регістрах - дві різні характеристики.

```
document.body.abc = 1
document.body.ABC = 5
alert(document.body.abc) // => все ще 1
```

Атрибут можна встановити будь-який, а властивість – ні. Але якості у різних регістрах - дві різні характеристики.

```
document.body.setAttribute('tagName',1)
document.body.getAttribute('tagName') // 1
document.body.tagName  // "BODY"
```

- Атрибути це те, що написано в HTML.
- Властивість це те, що знаходиться у властивості DOM-об'єкта.

Таблиця порівнянь для атрибутів та властивостей:

Властивості	Атрибути
Будь-яке значення	Рядок
Назви реєстрозалежні	Не чутливі до регістру
Не видно в innerHTML	Видно в innerHTML

Пошук елементів у DOM

Стандарт DOM передбачає кілька засобів пошуку елемента. Це методи **getElementById, getElementsByTagName та getElementsByName.**

Пошук по id

Наприклад, наступний код змінить колір тексту на блакитний у div'і с id="dataKeeper":

```
document.getElementById('dataKeeper').style.color = 'blue'
```

Пошук за тегом

Наприклад, можна отримати другий елемент (нумерація в масиві йде з нуля) з тегом li:

```
document.getElementsByTagName('LI')[1]
```

Наприклад, наступний виклик отримує список елементів LI, що знаходяться всередині першого div тега:

```
document.getElementsByTagName('DIV')[0].getElementsByTagName('LI')
```

Пошук елементів у DOM

Отримати всіх нащадків

```
<input type="submit" value="Нажми на меня" OnClick="ImageClick()">
<p>Проба</p>
<p>Проба</p>
<p>Проба</p>
<p>Проба</p>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
   function ImageClick() {
      arr=document.getElementsByTagName("P")
      for(i=0; i<=3; i++) arr[i].innerHTML="Абзац "+i;
   }
```

Пошук елементів у DOM

Наприклад, на такому DOM:

```
<div id="d1">
   <ol id="ol1">
      <li id="li1">1</li>
      <li id="li2">2</li>
   </ol>
</div>
```

Такий код:

```
var div = document.getElementById('d1')
var elems = div.getElementsByTagName('*')
for(var i=0; i<elems.length; i++) alert(elems[i].id)
```

Виведе послідовність: **ol1, li1, li2.**

Пошук елементів у DOM

Пошук по name:

Метод **document.getElementsByName(name)** повертає всі елементи, у яких ім'я (атрибут name) дорівнює даному.

Створення та додавання елементів

```
var newDiv = document.createElement('div')
```

Тут же можна і проставити властивості та зміст створеного елементу.

```
var newDiv = document.createElement('div')
newDiv.className = 'my-class'
newDiv.id = 'my-id'
newDiv.style.backgroundColor = 'red'
newDiv.innerHTML = 'Привет, мир!'
```

```
var textElem = document.createTextNode('Текстовий елемент');
```

Додавання до DOM

appendChild

Код із наступного прикладу додає нові елементи до списку:

```
<ul id="list">
   <li>Перший елемент</li>
</ul>
```

Перелік:

- Перший елемент

Елемент-список UL

```
var list = document.getElementById('list')
```

Новий елемент

```
var li = document.createElement('LI')
li.innerHTML = 'Новий елемент списку'
```

Додавання в кінець

```
list.appendChild(li)
```

Додавання до конкретного місця

Для цього використовується метод **insertBefore** батьківського елемента.

Наприклад, у тому списку додамо елементи перед першим li.

```
<ul id="list2">
   <li>Первый элемент</li>
</ul>
```

Перший елемент

Батьківський елемент UL

```
var list = document.getElementById('list2')
```

Елемент для вставки перед ним (перший LI)

```
var firstLi = list.getElementsByTagName('LI')[0]
```

Новий елемент

```
var newListElem = document.createElement('LI')
newListElem.innerHTML = 'Новий елемент списку'
```

Додавання

```
list.insertBefore(newListElem, firstLi)
```

Видалення вузла DOM

Щоб видалити вузол з документа - достатньо викликати метод **removeChild** з його батька.

parentElem.removeChild(elem)

Метод document.write

Методи document.writei document.writeln пишуть безпосередньо в текст документа, перш ніж браузер побудує з нього DOM, тому вони можуть записати в документ все, що завгодно будь-які стилі та незакриті теги.

```
<input type="button" onClick='document.write("Пуста сторінка!");'
    value=«Завантажити document.write('Пустая страница!')">
```

BOM-об'єкти: navigator, screen, location, frames

Об'єкт **navigator** містить загальну інформацію про браузер та операційну систему.

Має дві властивості:

- navigator.userAgent— містить інформацію про браузер.
- navigator.platform— містить інформацію про платформу, дозволяє розрізняти Windows/Linux/Mac тощо.

Для вашого браузера значення:

```
alert(navigator.userAgent);
alert(navigator.platform);
```

Поточна роздільна здатність екрана відвідувача по горизонталі/вертикалі знаходиться в alert(screen.width+"x"+screen.height);

BOM-об'єкти: navigator, screen, location, frames

Методи та властивості Location

Найголовніший метод - **toString**. Він повертає повну URL-адресу.

Код, якому потрібно провести рядкову операцію над Location , повинен *спочатку привести об'єкт до рядка* . Ось так буде помилка:

```
// буде помилка, тому що location - не рядок
alert( window.location.indexOf('://') );
```

… А так – правильно:

```
// привели до рядка, що передує indexOf
alert( (window.location + '').indexOf('://') );
```

Усі наступні властивості є рядками.
Колонка «Приклад» містить значення для тестового URL:
http://www.google.com:80/search?q=javascript#test
(У коді набирати так: window.location.href)

Властивість	Опис	приклад
hash	частина URL, яка йде після символу решітки '#', включаючи символ '#'	#test
host	хост та порт	www.google.com:80
href	весь URL	http://www.google.com:80/search?q=javascript#test
hostname	хост (без порту)	www.google.com
pathname	рядок колії (щодо хоста)	/search
port	номер порту (якщо порт не вказано, то порожній рядок)	80
protocol	протокол	http: (двокрапка на кінці)
search	частина адреси після символу "?", включаючи символ "?"	?q=javascript

Методи об'єкту Location

assign (url)

reload([forceget])

replace(url)

toString()

frames

У *frames* містяться window-об'єкти дочірніх кадрів.

Наступний код перекладає кадр на новий URL:

```
<iframe name="example" src="http://example.com" width="200" height="100"></iframe>
<script>
  window.frames.example.location = 'http://example.com';
</script>
```

Можна перенаправити і явним присвоєнням location, наприклад:

```
// браузер ззавантажить сторінку http://javascript.ru
window.location = "http://javascript.ru";
```

Він перезавантажить сторінку з новими параметрами після "?":

```
function refreshSearch(search) {
  window.location.search = search;
}
```

Дякую за увагу!