Об'єктна модель документа

Об'єктна модель документа (англ. **D**ocument **O**bject **M**odel, **DOM**) — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами (як правило, документами XML). Визначається ця специфікація консорціумом W3C.

З точки зору об'єктно-орієнтованого програмування, **DOM** визначає класи, методи та атрибути цих методів для аналізу структури документів та роботи із представленям документів у вигляді дерева. Все це призначено для того, аби надати можливість комп'ютерній програмі доступу та динамічної модифікації структури, змісту та оформлення документа.

Разом із поширенням та розвитком вебтехнологій і вебпереглядачів почали з'являтись різні, часто несумісні інтерфейси роботи із HTML документами в інтерпретаторах JavaScript, вбудованих у вебпереглядачі. Це спонукало World Wide Web Consortium (W3C) узгодити та визначити низку стандартів, які отримали назву W3C Document Object Model (W3C DOM). Специфікації W3C не залежать від платформи або мови програмування.

Через те, що структура документа представляється у вигляді дерева, повний зміст документа аналізується та зберігається в пам'яті комп'ютера. Тому, DOM підходить для застосувань в програмах, які вимагають багаторазовий доступ до елементів документа в довільному порядку. В разі, якщо треба лише послідовний або одноразовий доступ до елементів документа, рекомендується, для пришвидшення переробки та зменшення обсягів необхідної пам'яті комп'ютера, використовувати послідовну модель роботи зі структурованими документами (SAX).

Стандарти DOM

Починаючи з 1998 року DOM визнається стандартом W3C. Відтоді, його було багаторазово розширено та вдосконалено. Існують кілька версій DOM, які отримали назву рівнів (англ. Level). Кожен рівень складається із декількох обов'язкових та необов'язкових модулів. Для того, щоб стверджувати про підтримку DOM певного рівня, програма має задовольняти всім вимогам стандарту DOM заявленого рівня, та всім вимогам нижчих рівнів. Також, реалізація інтерфейсу може підтримувати певні розширення, якщо вони не суперечать вимогам стандарту. У 2005 році, рівні 1 та 2 (Level 1, Level 2) та деякі модулі 3-го рівня (Level 3) було визнано як W3C Recommendation, що означає, що вони набули кінцевої форми.

Level 0
Не було стандартизовано, став основою для появи DOM Level 1. Як приклад можна навести DHTML Object Model, або реалізацію DOM в вебпереглядачах Netscape ранніх версій. **Level 1**
Обхід структури (дерева) документа (HTML та XML), та модифікація змісту (включаючи додавання елементів). Також включаються специфічні елементи HTML. **Level 2**
Підтримка просторів імен XML, фільтрованих представлень та подій. **Level 3**
Складається із 6 різних специфікацій:

1. DOM Level 3 Core;
2. DOM Level 3 Load and Save;
3. DOM Level 3 XPath;
4. DOM Level 3 Views and Formatting;

5. DOM Level 3 Requirements;
6. DOM Level 3 Validation.

DOM

DOM - об'єктна модель документа (Document Object Model).

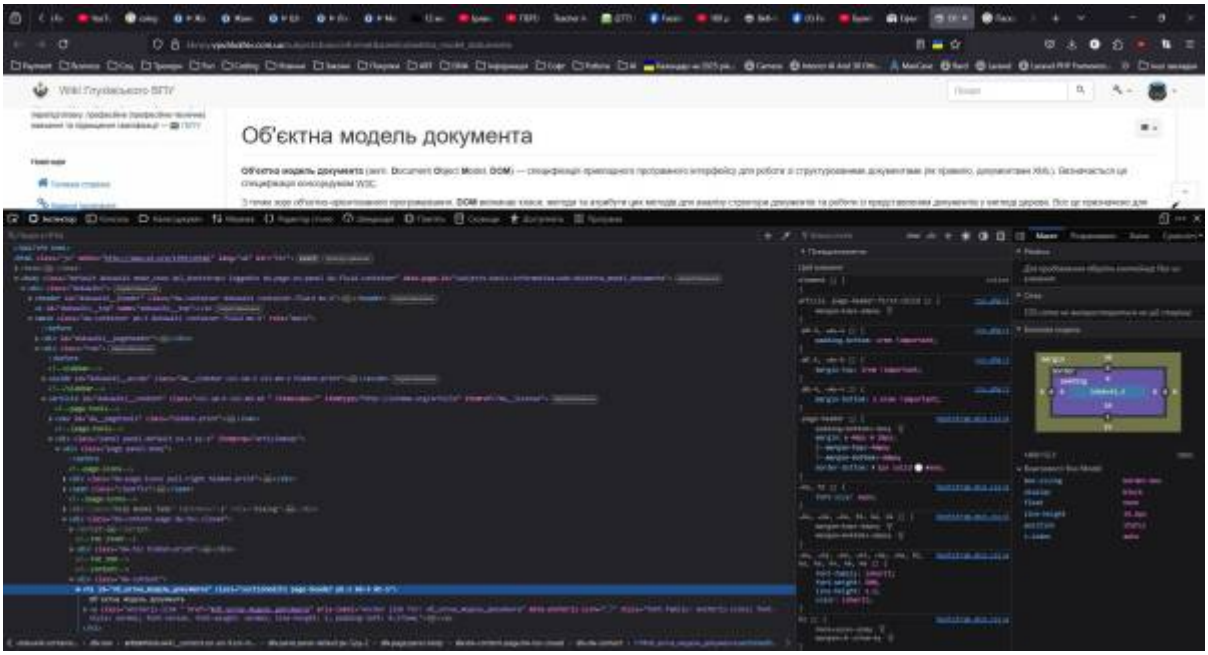
DOM це зовсім інше представлення веб-сторінки ніж HTML код.

Браузер по вказаній URL адресі відправляє запит і отримує (завантажує) з сервера веб-сторінку у вигляді HTML коду, який часто називається вихідний код сторінки. І якщо у коді вказані інші файли такі як стилі `css`, `js` - то завантажує і їх.

І уже з завантаженого з сервера HTML коду браузер формує - DOM.

Браузер створює **DOM** для того щоб за допомогою JavaScript можна було швидко маніпулювати веб-документом: шукати потрібний елемент, додавати нові елементи, отримати наступний дочірній елемент і т.п..

Вигляд DOM документа можна глянути у панелі розробника в браузері.



DOM подібний на вихідний код HTML але не є ним, а лише формується з нього.

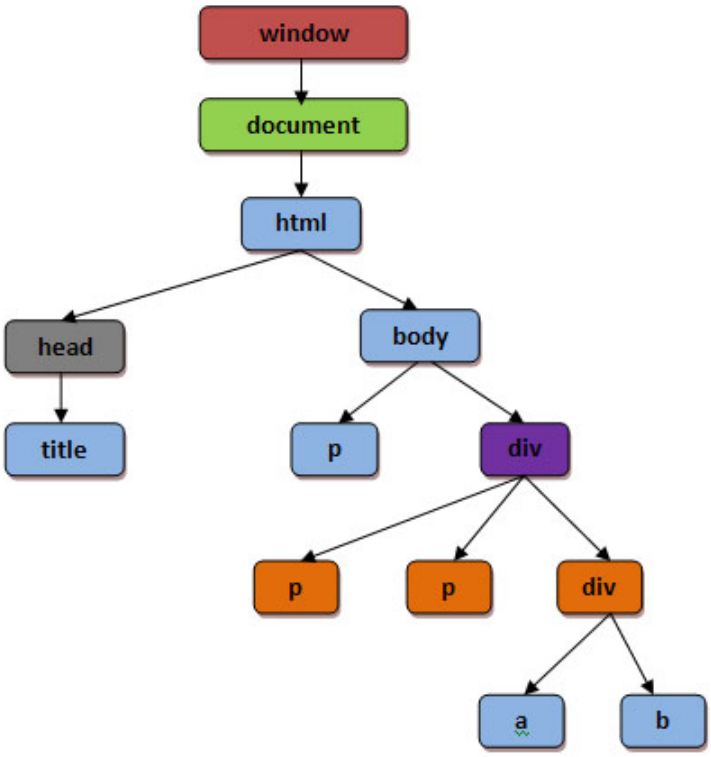
Браузер автоматично виправляє помилки якщо вони є у HTML коді. Тобто закриває не закриті теги HTML, вставляє обов'язкові теги якщо вони опущені.

```
<a href="/url1.html">силка 1
<a href="/url2.html">силка 2</a>

<ul>Список:
<li>1 пункт
<li>2 пункт</li>
<li>3 пункт</li>
</ul>

<table>
<tr><th>приклад</th><td>таблиця</td></tr>
</table>
```

DOM має деревоподібну ієархію. Документ DOM складається з вузлів Node. Кожен вузол може містити у собі вбудований вузол, елемент, текст чи коментар.

HTML коментар <!– коментар `html` –> для браузера це також вузол.

```
<html>
<head>
  <title>Заголовок веб-сторінки</title>
</head>
<body>
  <p>Абзац 1</p>
  <div>
    <p>Абзац 2</p>
    <p>Абзац 3</p>
    <div>
      <b>Жирний текст</b>
      <a href="http://яваскрипт.укр/">JavaScript</a>
    </div>
  </div>
</body>
</html>
```



Кожен вузол **DOM** формується з HTML тегу і отримує властивості, події, стилі які вказані у самих атрибутах тегу, CSS стилях і в JavaScript коді.

DOM підтримує об'єктно орієнтоване представлення веб-сторінки і дозволяє змінювати документ веб-сторінки за допомогою JavaScript.

Для роботи з DOM у JavaScript є об'єкт document, який дозволяє:

- видалити HTML-елементи і атрибути
- змінити всі HTML-елементи на сторінці
- змінити всі атрибути HTML на сторінці

- змінювати всі стилі CSS на сторінці
- додавати нові елементи HTML і атрибути
- створювати нові події на сторінці
- реагувати на існуючі подій на сторінці

У **JavaScript** для роботи з DOM є об'єкт document, який містить методи і властивості для роботи з документом.

Методи і властивості для роботи з DOM

JavaScript дозволяє на етапі форматування документу додавати до нього дані за допомогою методів `document.write()` і `document.writeln()`.

Методи для отримання елементу (ів) з документу:

- `document.getElementById()` - повертає елемент за вказаним id.
- `document.getElementsByName()` - повертає список елементів з вказаним name.
- `document.getElementsByTagName` - повертає список елементів за вказаною назвою тегу.
- `document.getElementsByClassName()` - повертає список елементів за вказаним ім'ям класу.
- `document.querySelector()` - повертає перший елемент в документі який співпадає з вказаним CSS селектором.
- `document.querySelectorAll()` - повертає список всіх елементів в документі, які відповідають зазначеним CSS селекторам.

Властивості для переходу по дереву DOM:

- `document.documentElement` - повертає елемент який є батьком документу.
- `Element.parentElement` - батьківський елемент поточного елемента.
- `Element.children` - список дочірніх елементів.
- `Element.firstElementChild` - перший дочірній елемент.
- `Element.lastElementChild` - останній дочірній елемент.
- `Element.nextElementSibling` - наступний елемент у батьківському списку.
- `Element.previousElementSibling` - попередній елемент у батьківському списку.

Реалізація DOM у веббраузерах

Враховуючи існуючі суттєві відмінності у реалізації DOM у веббраузерах, серед програмістів розповсюджена звичка перевіряти дієздатність тих чи інших можливостей DOM для кожного з браузерів, і тільки потім використовувати їх. Код нижче ілюструє можливість перевірки стандартів W3CDOM перед тим як запускати код, що залежить від результату перевірки.

```
if (document.getElementById && document.getElementsByTagName) {
    // якщо методи getElementById та getElementsByTagName
    // існують, то можна з майже впевнено сподіватись на підтримку W3CDOM.
```

```
    obj = document.getElementById("navigation")
    // далі йде інший код з використанням можливостей  W3CDOM.
    // .....
}
```

Ще один фрагмент коду JavaScript, що дозволяє перевірити заявлену підтримку різних доповнень DOM у відповідному браузері.

```
<html>
<head>
<title>Test DOM Implementation</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<script type="text/javascript">
function domImplementationTest() {
  var featureArray = ['HTML', 'XML', 'Core', 'Views',
            'StyleSheets', 'CSS', 'CSS2', 'Events',
            'UIEvents', 'MouseEvents', 'HTMLEvents',
            'MutationEvents', 'Range', 'Traversal'];
  var versionArray = ['1.0', '2.0', '3.0'];
  var i;
  var j;
  if (document.implementation && document.implementation.hasFeature){
    document.write('<table border="1" cellpadding="2" style="border-
collapse:collapse;">');

    // header of table
    document.write('<tr>');
    document.write('<td>' + 'Підтримка доповнення ' + '</td>')
    for (j = 0; j < versionArray.length; j++) {
      document.write('<td>'+ 'версія ' + versionArray[j] + '</td>');
    }
    document.write('</tr>');

    // content of table
    for (i = 0; i < featureArray.length; i++){
      document.write('<tr>');
      document.write('<td>' + featureArray[i] + '</td>');

      for (j = 0; j < versionArray.length; j++) {
        var res = document.implementation.hasFeature(featureArray[i],
versionArray[j]);
        document.write('<td style="background-color:' + (res ? 'blue' :
'red') + '; color:white;">'+ res + '</td>');
      }

      document.write('</tr>');
    }

    document.write('</table>');
  }
}
```

```
</script>
</head>
<body>
<h1>Перевірка доповнень DOM</h1>

<script type="text/javascript">
  domImplementationTest();
</script>

</body>
</html>
```

Модель документу

Після аналізу структурованого документа, будується його представлення у вигляді дерева. Дерево, в моделі DOM, складається із множини зв'язних вузлів (Node) різних типів. Як правило, розрізняють вузли наступних типів:

- Документ (`Document`) — корінь дерева, представляє цілий документ.
- Фрагмент документа (`DocumentFragment`) — вузол, який є коренем піддерева основного документа.
- Елемент (`Element`) — представляє окремий елемент HTML або XML документа.
- Атрибут (`Attr`) — представляє атрибут елемента.
- Текст (`Text`) — представляє текстові дані, які містяться в елементі або атрибуті.

Стандартом визначаються і деякі інші типи вузлів у моделі документа.

Вузли деяких типів можуть мати гілки, інші ж можуть бути лише листами дерева. Спеціальні методи об'єктів вузлів дають можливість обходу дерева.

Джерела

- Об'єктна модель документа
- DOM - об'єктна модель документа - JavaScript довідка