Лабораторна робота № 1

Тема: Структура DOM та методи доступу до вузлів дерева

Мета: Ознайомитись з ієрархічною структурою об'єктів JavaScript та об'єктами документа і браузера.

Модель DOM

Web-сторінка може мати вигляд дерева, вузли якого є об'єктами, до властивостей яких доступаються операторами мови програмування Javascript.

Модель DOM (Document Object Model, об'єктна модель документа) містить низку стандартних глобальних об'єктів. Зокрема, це `window`, `navigator`, `document`, `screen`, `history`, `location`. Ієрархію основних об'єктів у моделі DOM подано на рис. 1.2.

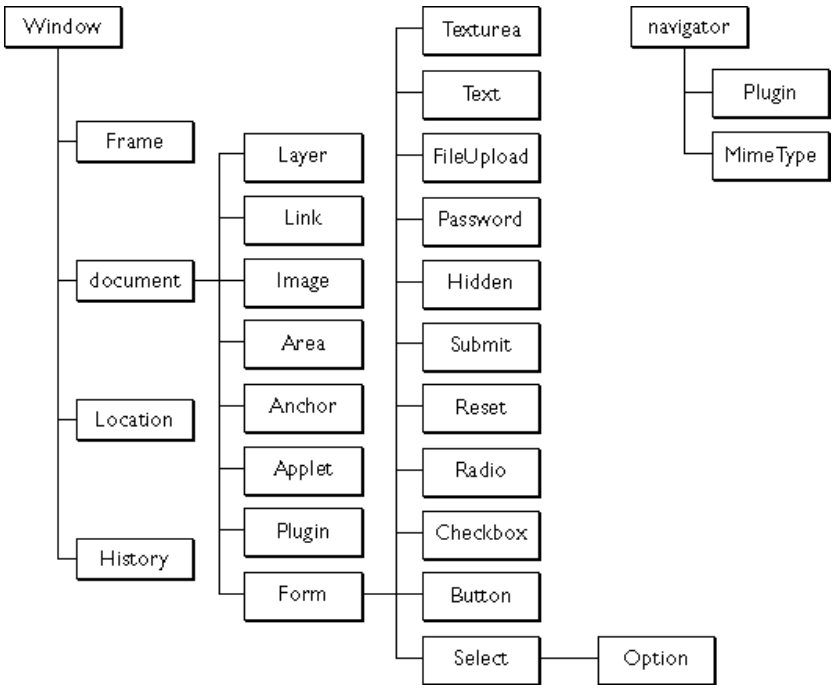

Рис. 1.2. Дерево моделі DOM об'єкта `window` браузера

Перерахуємо глобальні об'єкти та їх призначення, а також розглянемо методи та властивості глобальних об'єктів DOM:

- `window` – вікно браузера, найвищий об'єкт ієрархії DOM. Усі глобальні змінні стають частинами об'єкта window. Його методи такі:

`alert`, `blur`, `clearInterval`, `clearTimeout`, `close`, `confirm`, `focus`, `moveBy`, `moveTo`, `open`, `print`, `prompt`, `resizeBy`, `resizeTo`, `scrollBy`, `scrollTo`, `setInterval`, `setTimeout`

Властивості: `document`, `history`, `location`, `name`

- `navigator` – інформація про браузер.

Властивості: `appName`, `appVersion`, `browserLanguage`, `cookieEnabled`, `platform`, `userAgent`

- `screen` – інформація про екран, який використовує браузер.

Властивості: `availHeight`, `availWidth`, `colorDepth`, `height`, `pixelDepth`, `width`

• history – список викликаних сторінок.

Властивості: length

Методи: back, forward, go

• location – URL поточної web-сторінки.

Властивості: host, hostname, href, pathname, port, protocol, search

Методи: assign, reload, replace

• document – поточна Web-сторінка.

Властивості: anchors, body, cookie, domain, forms, images, links, referrer, title, URL

Методи: close, getElementById, getElementsByName, open, write, writeln, getElementsByTagName

Приклад ієрархічного дерева DOM для об'єкта document наведено на рис. 1.3.

За змістом вузли дерева поділяють на три типи:

• елементів (HTML теги) – породжені вузли або атрибути;

• тексту (текст або блоки) – не мають породжених вузлів чи атрибутів;

• атрибутів (пари атрибут/значення) – не мають породжених вузлів чи атрибутів.

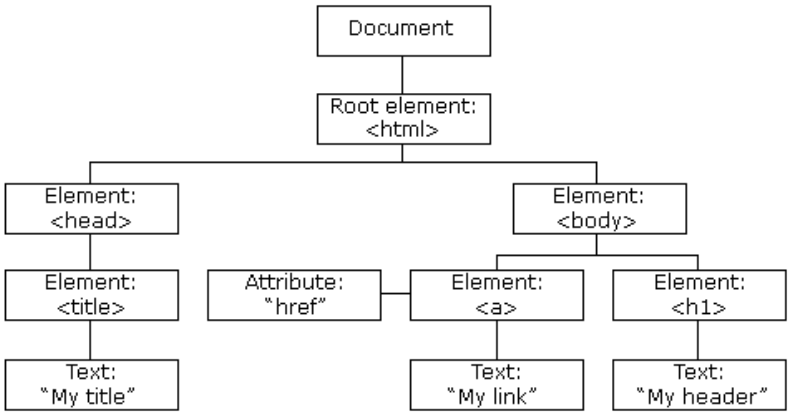

Рис. 1.3. Дерево об'єкта document у моделі DOM

Наведемо фрагмент сторінки і відповідне йому дерево (рис. 1.4)

```
<p>This is a paragraph of text with a
<a href="/path/to/another/page.html">link inside </a>.
```

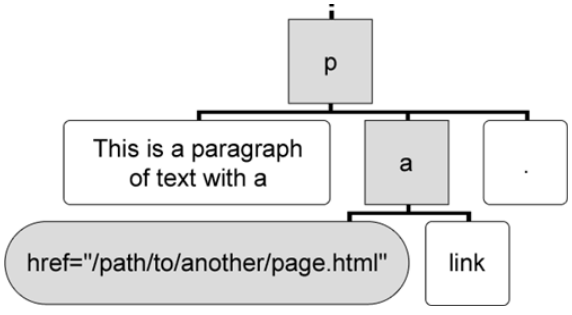

Рис. 1.4. Фрагмент дерева з вузлом параграфа

Типи вузлів та методи доступу до них

За місцем розташування на дереві виділено такі типи вузлів: `parent` – батьківський, `child` – породжений, `sibling` – сусідній, `firstchild` – перший породжений, `lastchild` – останній породжений.

Назви вузлів є складовими назвами властивостей вузлів та параметрів методів, які ними оперують (ліквідують, модифікують, додають тощо). Вузли DOM-дерева мають такі властивості:

- `firstChild, lastChild` – початковий та кінцевий породжені вузли;
- `childNodes` – масив породжених вузлів;
- `nextSibling, previousSibling` – сусідні вузли (по одному), що мають одного батька;
- `parentNode` – елемент, що породив цей вузол.

Мова програмування дає змогу змінювати склад вузлів, вигляд та зміст web-сторінки. На моделі DOM програміст має можливість виконувати операції двох типів: на рівні вузлів для зміни структури дерева додаванням або відніманням вузла та заміни вузла; на рівні вузла для зміни змісту елемента. Зокрема, для першого типу операцій застосовуються методи:

- `appendChild(node)`: розміщає вузол у кінці списку породжених вузлів до заданого;
- `insertBefore(newChild, oldChild)`: розміщає новий вузол перед вузлом `oldChild` у списку породжених вузлів до заданого;
- `removeChild(node)`: знищує наведений вузол зі списку породжених вузлів до заданого;
- `replaceChild(newChild, oldChild)`: заміняє породжений на новий вузол.

Властивості набувають значення

- `nodeType`: 1 – елемент, 2 – атрибут, 3 – текст, 4 – коментарі, 9 – документ.
- `nodeName`: повертає версію тега, наприклад, `"div"` чи `"article"`. Текстові вузли мають назву `"#text"`. Назва вузла `document` є `"#document"`,
- `nodeValue`: текст вузла або значення атрибута.

Доступ до вузлів DOM за тегами чи ідентифікаторами здійснюється методами:

- `document.getElementById("id")` – елемент;
- `element.getElementsByTagName("tag")` – усі породжені вузли;

`document.createElement("tag")` - команда створює новий порожній вузол для подання елемента певного типу. Рис.1. Загальна схема ієрархічної структури об’єктів.

Опрацювання подій

Події та обробники подій є дуже важливою частиною у програмування на JavaScript. ***Події*** (*Events*), головним чином, ініціюються тими або іншими діями користувача. Події – це дії, які відбуваються, внаслідок того, що робить користувач. Наприклад, якщо користувач клацає по деякій кнопці, відбувається подія `Click`. Якщо миша перетинає яке-небудь посилання - відбувається подія `MouseOver`. Існує певний набір подій, які розпізнає той чи інший броузер.

Ми можемо примусити нашу JavaScript-програму реагувати на деякі з них. І це може бути виконано за допомогою спеціальних програм ***обробки подій***. Так, в результаті клацання по кнопці може створюватися випадаюче вікно. Це означає, що створення вікна повинно бути реакцією на подію `Click`. Програма - обробник подій, яку ми повинні

використати в даному випадку, називається `onClick`. І вона повідомляє комп'ютер, що потрібно робити, якщо відбудеться дана подія.

Обробник подій записується в документ як атрибут тега HTML, до якого ви приписуєте код JavaScript. Наприклад, ви створили ***функцію*** JavaScript, і назвали її `compute`. Ви можете примусити броузер виконувати цю функцію, коли користувач натискає на кнопку, в якій до `onClick` приписаний обробник результату кнопки:

```
INPUT TYPE="button" VALUE="Calculate" onClick="compute(this.form)"
```

Ви можете поміщати будь-які твердження JavaScript усередині кавичок `onClick`. Ці твердження будуть виконані, коли користувач натискатиме на кнопку. Якщо Ви хочете включити більш ніж одне твердження, то окремі твердження записуються через крапку з комою (;).

Взагалі, це – непогана ідея визначати ***функцію*** для обробників подій тому що:

- це робить ваш код мобільним, оскільки ви можете використовувати ту ж саму функцію в багатьох різних місцях.
- це робить ваші твердження більш легкими для читання.

В даному прикладі використовується `this.form`, щоб звернутися до поточної форми. Ключове слово звертається до об'єкту (об'єкту кнопки у вищезазначеному прикладі). Потім конструкція `this.form` звертається до форми, що містить кнопку. Далі – обробник події `onClick` робить запит до функції `compute()`, з поточною формою `this.form`, як параметр функції.

Події звертаються до тегів HTML таким чином:

- події **Focus, Blur, Change**: text fields, textareas, і selections;
- подія **Click**: buttons, radio buttons, checkboxes, submit buttons, reset buttons, links;
- подія **Select**: text fields, textareas;
- подія **MouseOver**: links.

Ви можете використовувати в скрипті безліч різних типів функцій обробки подій. Щоб дізнатись про усі існуючі обробники подій, звертайтеся до відповідного довідника. Наведемо лише деякі з них:

- **onLoad** - виконання скрипта або функції при завантаженні;
- **onChange** - породжується при зміні значення елемента форми;
- **onClick** - породжується при виборі об'єкту (button, checkbox і т.п.);
- **onSelect** - породжується при виборі текстового об'єкту (text, textarea);
- **onSubmit** - при натисненні на кнопку Submit;
- **onUnload** - при переході до іншої сторінки.

Завдання до лабораторної роботи № 1:

Розробити web-сторінку згідно макета (wireframe)
https://cacoo.com/diagrams/ZvVhYS3UpG5PdbBy/EDE3A

Обираємо для розробки шаблон Students
Використовуємо Gitlab (https://gitlab.com)
Сторінка має містити:
таблицю HTML.
анімацію засобами CSS3 (анімувати індикатор нових повідомлень).
посилання на web-сторінці.
зображення на web-сторінці.
додавання нового вузла (додавання студента по натисканню на +).
видалення вузла (видалення студента кожного окремо).
відображення блоків при наведені на елементи (нотифікації та студент) у шапці.