Java Script

NEXT

GENIUS space

План

DOM

Навігація по DOM

Властивості

Робота з DOM- вузлами

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.

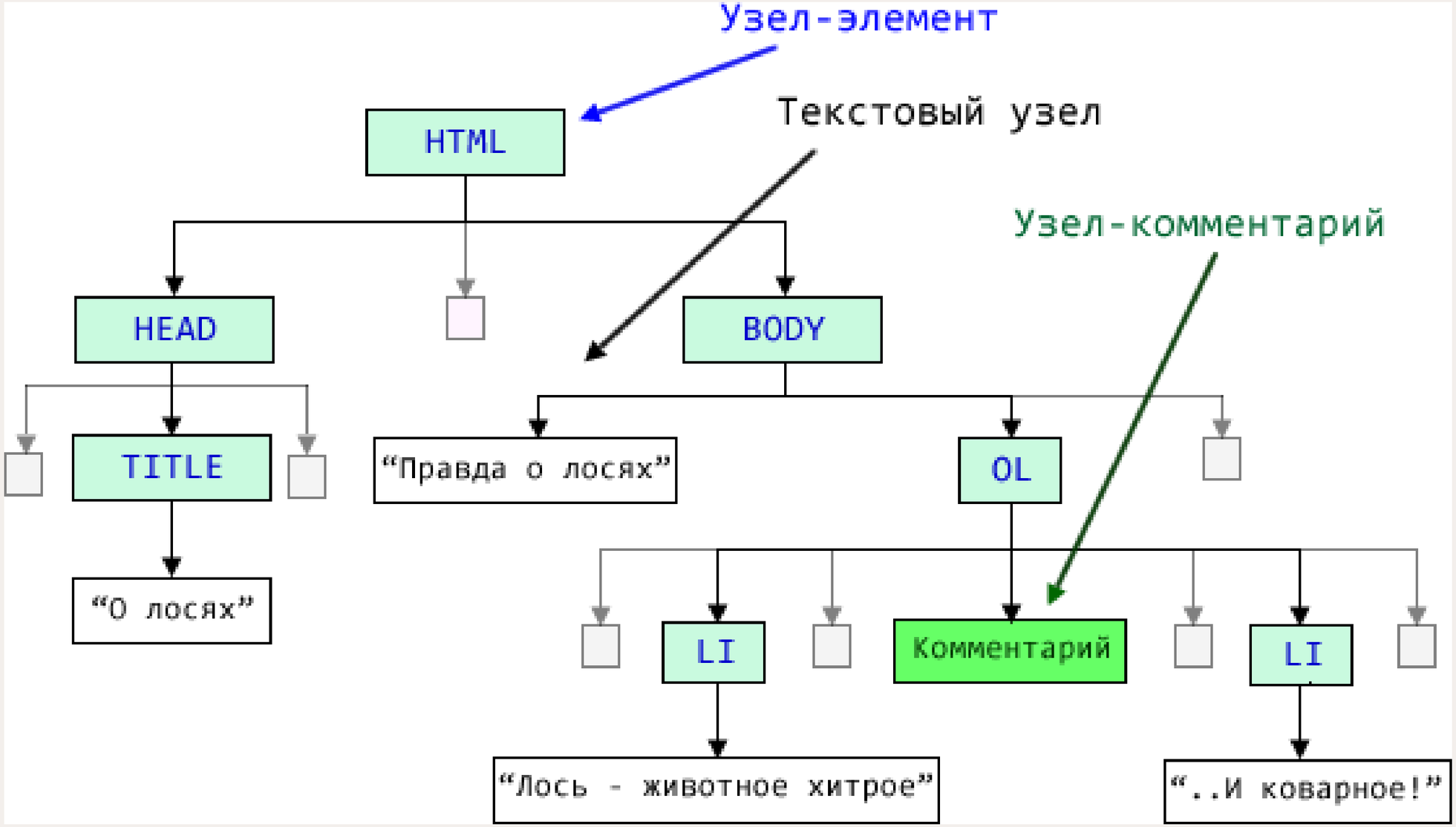
І уже з завантаженого з сервера HTML коду браузер формує - DOM.

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..

DOM (Document Object Model, Об'єктна модель документа) – міжплатформний, незалежний від мови інтерфейс для роботи з HTML-документом. Містить набір властивостей та методів, які дозволяють шукати, створювати та видаляти елементи, реагувати на дії користувача та інше.

BOM (Browser Object Model, Об'єктна модель браузера) – міжплатформний, незалежний від мови інтерфейс для роботи з вікном браузера. Містить набір властивостей та методів, що дозволяють отримати доступ безпосередньо до поточної вкладки та ряду функцій браузера. Включає об'єкт роботи з історією, місцем розташування та інше.

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


Узел-элемент
Текстовый узел
Узел-комментарий
HTML
HEAD
BODY
TITLE
“Правда о лосях”
OL
“О лосях”
LI
Комментарий
LI
“Лось - животное хитрое”
“..И коварное!”

Інші типи вузлів

Окрім елементів та текстових вузлів є деякі інші типи вузлів.

Наприклад, коментарі:

```
<!DOCTYPE HTML>
<html>
<body>
  Правда про оленів.
  <ol>
    <li>Олень -- це розумний</li>
    <!-- comment -->
    <li>...та хитрий звір!</li>
  </ol>
</body>
</html>
```

Тут ми бачимо новий тип вузла дерева – вузол-коментар, позначений як #comment, між двома текстовими вузлами.

Ми могли подумати – чому коментар додається до DOM? Це не впливає на візуальне уявлення. Але є правило – якщо щось є в HTML, то воно також повинно бути в DOM дереві.

Все в HTML, навіть коментарі, стає частиною DOM.

Навіть директива <!DOCTYPE...> на самому початку HTML також є вузлом DOM. Вона є DOM дереві прямо перед <html>. Мало хто знає про це. Ми не збираємося звертатися до цього вузла, ми навіть не малюємо його на діаграмах, але він там є.

Об’єкт document, який представляє весь документ, формально, також є вузлом DOM.

Існує 12 типів вузлів. На практиці ми зазвичай працюємо з 4-ма з них:

document – “пункт входу” в DOM.

вузли-елементи – HTML-теги, будівельні блоки дерев.

текстові вузли – містять текст.

коментарі – іноді ми можемо записати туди інформацію, вона не буде показана, але JS може читати її з DOM.

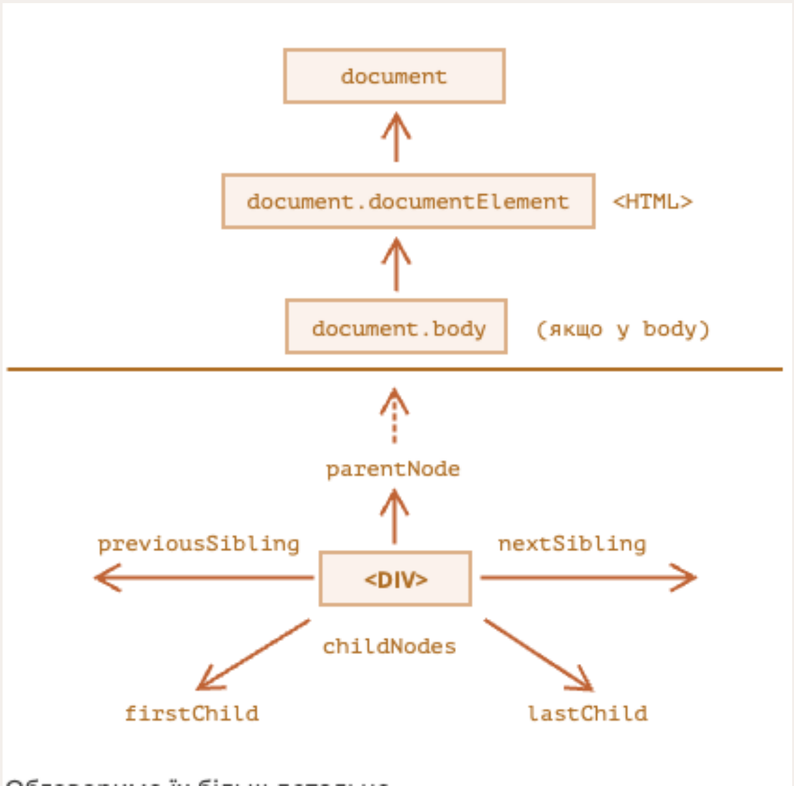
Коли ми працюємо з DOM, ми також можемо застосувати до нього JavaScript. Наприклад, отримайти вузол і запустити якийсь код, щоб змінити його так, щоб побачити результат.

Навігація по DOM

DOM дозволяє нам робити будь-що з елементами та їх вмістом, але спочатку нам потрібно знайти відповідний DOM об’єкт.

Усі операції з DOM починаються з об’єкта document. Це головна “точка входу” в DOM. З нього ми можемо отримати доступ до будь-якого вузла.

Ось зображення структури посилань, які дозволяють переміщатися між вузлами DOM:



Зверху: documentElement і body

Найвищі вузли дерева доступні безпосередньо як властивості document:

<html> = document.documentElement
Найвищий вузол документа – document.documentElement. Це вузол DOM тегу <html>.
<body> = document.body
Для доступу до DOM вузлу <body> часто використовують елемент – document.body.
<head> = document.head
Тег <head> доступний як document.head.

Дочірні елементи: childNodes, firstChild, lastChild
Відтепер ми будемо використовувати два терміни:

- Дочірні вузли (або діти) – елементи, які є безпосередніми дітьми. Іншими словами, вони вкладені саме в цей вузол. Наприклад, <head> і <body> є дочірніми елементами <html>.
- Нащадки – всі елементи, які вкладені в даний, включаючи дітей, їхніх дітей тощо.

Вузли HTML-дерева мають ієрархічне відношення один до одного. Терміни ancestor (предок), descendant (нащадок), parent (батько), child (дитина) і sibling (сусід) використовуються для опису відносин.

У дереві вузлів верхній вузол називається кореневим (root node).

Кожен вузол, окрім root node, має лише одного з батьків.

У вузла може бути скільки завгодно дітей.

Сусіди - це вузли із спільним батьком.

Дочірні елементи (діти) – елементи, які лежать безпосередньо всередині цього.

Нащадки – всі елементи, які лежать усередині цього, разом із їхніми дітьми, дітьми їхніх дітей тощо. Тобто все піддерево.

Ви можете використовувати такі властивості для навігації між вузлами.
elem.parentNode - вибере батька elem.
elem.childNodes - псевдомасив зберігає всі дочірні елементи, включаючи текстові.
elem.children - псевдо-масив зберігає лише дочірні вузли-елементи, тобто відповідні тегам.
elem.firstChild – вибере перший дочірній елемент усередині elem, включаючи текстові вузли.
elem.firstElementChild – вибере перший дочірній вузол-елемент усередині elem.
elem.lastChild – вибере останній дочірній елемент усередині elem, включаючи текстові вузли.
elem.lastElementChild - вибере останній дочірній вузол-елемент усередині elem.
elem.previousSibling - вибере елемент "зліва" від elem (його попереднього сусіда)
elem.previousElementSibling - вибере вузол-елемент "зліва" від elem (його попереднього сусіда)
elem.nextSibling - вибере елемент "праворуч" від elem (його наступного сусіда)
elem.nextElementSibling - вибере вузол-елемент "праворуч" від elem (його попереднього сусіда)

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.

Пошук: getElement*, querySelector*
Властивості навігації по DOM чудові, коли елементи розташовані близько один до одного. А якщо ні? Як отримати довільний елемент сторінки?

Для цього існують додаткові методи пошуку.

document.getElementById або просто id
Якщо елемент має атрибут id, ми можемо отримати його за допомогою методу document.getElementById(id), незалежно від того, де він знаходиться.

```
<div id="elem">
<div id="elem-content">Елемент</div>
</div>
<script>
// отримати елемент let elem = document.getElementById('elem');
// зробити його фон червоним elem.style.background = 'red';
</script>
```

querySelectorAll

До сьогодні найуніверсальніший метод – це elem.querySelectorAll(css), він повертає всі елементи всередині elem, що відповідають заданому CSS-селектору.

Тут ми шукаємо всі елементи <li>, які є останніми дочірніми:

```
<ul>
  <li>Цей</li>
  <li>тест</li>
</ul>
<ul>
  <li>повністю</li>
  <li>пройдено</li>
</ul>
<script>
  let elements = document.querySelectorAll('ul > li:last-child');
  for (let elem of elements) {
    alert(elem.innerHTML); // "тест", "пройдено"
  }
</script>
```

querySelector
Виклик elem.querySelector(css) повертає перший елемент, що відповідає даному CSS-селектору.

Іншими словами, результат такий самий, як і elem.querySelectorAll(css)[0], але останній шукає всі елементи та вибирає один, а elem.querySelector просто шукає один. Тому його писати швидше і коротше.

Також можна використовувати псевдокласи
Псевдокласи в CSS-селекторі, такі як :hover і :active, також підтримуються. Наприклад, document.querySelectorAll(':hover') поверне колекцію елементів, що знаходяться під курсором миші (у порядку вкладення: від крайнього <html> до найбільш вкладеного).

getElementsBy*
Існують також інші методи пошуку елементів за тегом, класом тощо.

Сьогодні вони здебільшого історичні, оскільки querySelector є потужнішим і коротшим для написання.

Тому тут ми розглянемо їх переважно для повноти, тоді як ви все ще можете знайти їх у старому коді.

elem.getElementsByTagName(tag) шукає елементи з заданим тегом і повертає їх колекцію. Параметр tag також може бути зірочкою "*" для “будь-яких тегів”.
elem.getElementsByClassName(className) повертає елементи, які мають заданий CSS-клас.
document.getElementsByName(name) повертає елементи з заданим атрибутом name для всього документа. Використовується дуже рідко.

Властивості

Під час побудови DOM-дерева багато стандартних HTML-атрибутів стають властивостями вузлів. Подивимося на кілька властивостей, що часто використовуються.

hidden – контролює видимість елемента.

value - містить поточний текстовий контент елементів input, select, textarea.

checked - зберігає стан чекбоксу чи радіокнопки.

name - зберігає значення, вказане в HTML-атрибуті name.

src – шлях до зображення тега <img>.

Створення елементу

Є два способи створення DOM вузлів:

document.createElement(tag)

Створює новий елемент з заданим тегом:

let div = document.createElement('div');
document.createTextNode(text)

Створює новий текстовий вузол з заданим текстом:

let textNode = document.createTextNode('От і я');
У більшості випадків нам потрібно створювати саме елементи, такі як div для повідомлень.

Створення повідомлень

Створення div елементу для повідомлення складається з трьох кроків:

```
// 1. Створіть елемент <div>
let div = document.createElement('div');

// 2. Задайте йому клас "alert"
div.className = "alert";

// 3. Наповніть <div> змістом
div.innerHTML = "<strong>Всім привіт!</strong> Ви прочитали важливе повідомлення.";
```

Ми створили елемент, але поки що він знаходиться лише у змінній під назвою div, а не на сторінці. Тому ми не можемо його побачити.

Методи вставки

Щоб div показався нам потрібно вставити його десь на сторінці в document. Наприклад, в елемент <body> який можна отримати звернувшись до document.body.

Для цього існує спеціальний метод append: document.body.append(div).

Нижче наведено більше методів для вставки, вони вказують куди саме буде вставлено вміст:

- node.append(...вузли або рядки) – додає вузли або рядки в кінець node,
- node.prepend(...вузли або рядки) – вставляє вузли вбо рядки на початку node,
- node.before(...вузли або рядки) – вставляє вузли або рядки попереду node,
- node.after(...вузли або рядки) – вставляє вузли або рядки після node,
- node.replaceWith(...вузли або рядки) – замінює node заданими вузлами або рядками.

Але що, як нам потрібно вставити рядок HTML «як html», з усіма тегами та іншим, в той самий спосіб як це робить elem.innerHTML?

Для цього ми можемо використовувати інший, досить універсальний метод: elem.insertAdjacentHTML(куди, html).

Перший параметр це кодове слово, яке вказує куди вставляти відносно elem. Його значеннння має бути одним з наступних:

"beforebegin" – вставити html безпосередньо перед elem,

"afterbegin" – вставити html в elem, на початку,

"beforeend" – вставити html в elem, в кінці,

"afterend" – вставити html безпосередньо після elem.

Об'єкт містить методи для роботи з класами елемента.
elem.classList.contains(cls) - повертає true або false, залежно від того, чи є елемент класу cls
elem.classList.add(cls) - додає клас cls до списку класів елемента
elem.classList.remove(cls) - видаляє клас cls зі списку класів елемента
elem.classList.replace(oldClass, newClass) - замінює існуючий клас на вказаний
elem.classList.toggle(cls) - якщо класу cls немає, додає його, якщо є, видаляє

DOM-елементам відповідають HTML-теги, які мають текстові атрибути. Доступ до атрибутів здійснюється за допомогою стандартних методів. Ці методи працюють із значенням, що знаходиться в HTML.
elem.hasAttribute(name) - перевіряє наявність аттрибута, повертає true чи false
elem.getAttribute(name) - отримує значення атрибуту та повертає його
elem.setAttribute(name, value) - встановлює атрибут
elem.removeAttribute(name) - видаляє атрибут
elem.attributes - властивість, що повертає колекцію всіх атрибутів елемента

Додаткові матеріали

- http://xn--80adth0aefm3i.xn--j1amh/DOMp
- https://uk.javascript.info/browser-environment
- https://uk.javascript.info/dom-nodes
- https://uk.javascript.info/dom-navigation
- https://uk.javascript.info/basic-dom-node-properties
- https://uk.javascript.info/searching-elements-dom
- https://uk.javascript.info/dom-attributes-and-properties
- https://uk.javascript.info/modifying-document
- https://uk.javascript.info/styles-and-classes
- https://uk.javascript.info/size-and-scroll
- https://uk.javascript.info/size-and-scroll-window
- https://uk.javashttps://www.sitepoint.com/how-why-use-html5-custom-data-attributes/cript.info/coordinates
- https://www.youtube.com/watch?v=dndeRnzkJDU
- https://medium.com/swlh/what-the-heck-is-repaint-and-reflow-in-the-browser-b2d0fb980c08
- https://web.dev/avoid-large-complex-layouts-and-layout-thrashing/

Домашнє завдання

-----1-----

Напишіть код, щоб вибрати елемент з атрибутом data-widget-name з документа та прочитати його значення.

```
<!DOCTYPE html>
<html>
<body>
<div data-widget-name="menu">Виберіть жанр</div>
<script>/* ваш код */</script>
</body>
</html>
```