DOM та Devtools: Вступ до об'єктної моделі документа

Коли ми працюємо з браузером, доступний функціонал складається з декількох модулів, оскільки JavaScript не має інструментів для роботи з браузером. DOM означає об'єктну модель документа (Document Object Model). Це інтерфейс, який дозволяє нам створювати, змінювати або видаляти елементи з документа.

Ми також можемо додавати події до цих елементів, щоб зробити нашу сторінку більш динамічною. DOM розглядає документ HTML як дерево вузлів (нод). Вузол (нода) представляє елемент HTML. Це фундаментальна концепція, яка лежить в основі всієї веб-розробки та дозволяє JavaScript взаємодіяти зі структурою веб-сторінки.

Ієрархічна структура

DOM організує HTML як дерево вузлів з батьківськими та дочірніми елементами

Динамічна взаємодія

Можливість змінювати вміст та структуру документа в реальному часі

JavaScript інтерфейс

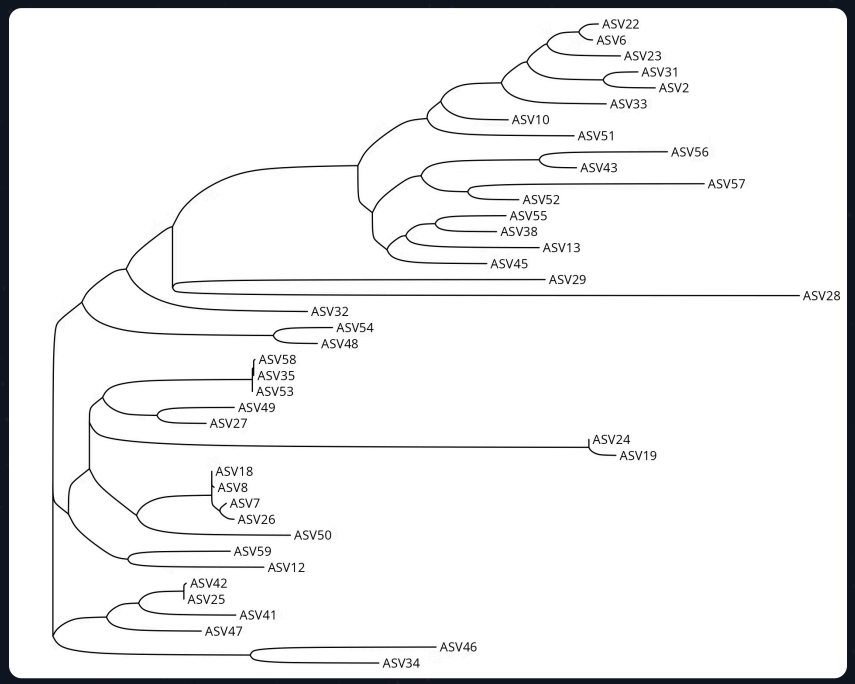
Надає методи та властивості для програмної роботи з елементами

Структура дерева DOM

Основні концепції

Наш документ називається кореневим вузлом (root node) і містить один дочірній вузол, який є елементом **<html>**. Елемент **<html>** містить два дочірні елементи, а саме **<head>** і **<body>**.

Обидва елементи **<head>** і **<body>** мають власних нащадків. Ця ієрархічна структура нагадує генеалогічне дерево, де кожен елемент може мати батьків, нащадків та сусідів.



Document

Кореневий об'єкт

1

2

HTML

Головний елемент

Head & Body

Основні розділи

3

4

Дочірні елементи

Вміст сторінки

Важливо розуміти: Кожен елемент у DOM є об'єктом з власними методами та властивостями, що дозволяє програмно керувати будь-якою частиною документа.

Візуалізація DOM-структури

Розглянемо конкретний приклад HTML-розмітки та її представлення в DOM. Коли браузер обробляє HTML-документ, він створює дерево об'єктів, де кожен елемент, атрибут і текстовий вузол стає частиною цієї структури.

HTML код

```
<p>
  <img src="dinosaur.png">
  Here we will add a link to
  the <a href="https://mozilla.org/">
  Mozilla homepage</a>
</p>
```

Трошки інший спосіб

Візуалізація нод представлених у DOM для HTML документа наданого вище показує, як елементи організовані ієрархічно. Кожен тег стає вузлом, а текстовий вміст - текстовим вузлом.

1

Елемент параграфа

Батьківський вузол, що містить всі інші елементи

2

Зображення

Дочірній вузол типу IMG з атрибутом src

3

Текстовий вузол

Простий текст між елементами

4

Посилання

Елемент anchor з атрибутом href та текстовим вмістом

> **Ключова ідея:** DOM-дерево - це не просто статична структура, а динамічне представлення документа, яке можна змінювати в реальному часі за допомогою JavaScript.

Методи пошуку елементів у DOM

Спочатку слід перейти до того, як ми можемо отримати доступ до елементів. Існує кілька основних методів для пошуку елементів у DOM, кожен з яких має свої особливості та сценарії використання.

01

getElementById

Найпростіший спосіб знайти елемент за унікальним ідентифікатором

02

getElementsByTagName

Знаходить всі елементи з певною назвою тегу

03

Вкладений пошук

Комбінація методів для пошуку елементів всередині інших елементів

Пошук за ідентифікатором

Найпростіший спосіб знайти елемент HTML у DOM - використовувати ідентифікатор елемента. Цей метод повертає один конкретний елемент:

```
const element =
document.getElementById("intro");
```

Швидкий та **ефективний** метод для унікальних елементів.

Пошук за тегом

У цьому прикладі знаходяться всі елементи **<p>** на сторінці:

```
const element =
document.getElementsByTagName("p");
```

Повертає живу колекцію всіх елементів з вказаним тегом.

Вкладений пошук елементів

У цьому прикладі знаходиться елемент із ідентифікатором "main", а потім знаходяться всі елементи **<p>** всередині "main":

```
const x = document.getElementById("main");
const y = x.getElementsByTagName("p");
```

Сучасні методи querySelector

Група методів **elem.querySelector*** - це сучасний стандарт для пошуку елементів. Вони дозволяють знаходити елемент або групу елементів за будь-яким складним CSS-селектором, що робить їх надзвичайно гнучкими та потужними інструментами.

element.querySelector(selector)

Цей метод використовується, коли потрібно знайти лише один, зазвичай унікальний елемент. Він повертає перший знайдений елемент всередині element, що відповідає CSS-селектору selector. Якщо нічого не знайдено, повертається null.

element.querySelectorAll(selector)

Цей метод використовується, коли потрібно знайти колекцію елементів, наприклад, всі елементи списку з класом menu-item. Він повертає псевдомасив всіх елементів всередині element, які відповідають CSS-селектору selector. Якщо нічого не знайдено, повертається порожній масив.

Переваги querySelector

- Підтримка будь-яких CSS-селекторів
- Інтуїтивний синтаксис для тих, хто знає CSS
- Можливість складних комбінованих запитів
- Сучасний стандарт веб-розробки

DOM елементи мають власний набір методів та властивостей які можна переглянути в документації.

> *Порада для розробників:* Використовуйте querySelector для простих запитів та querySelectorAll для колекцій елементів. Це робить код чистішим та легшим для підтримки.

Робота з атрибутами та операції з елементами

Атрибути елементів

HTML-теги відповідають DOM-елементам, що мають текстові атрибути. Доступ до цих атрибутів здійснюється за допомогою стандартних методів.

Операції з елементами

Якщо ми хочемо додати або видалити елементи, існують методи, які це дозволяють зробити ефективно та безпечно.

Методи роботи з атрибутами

1

hasAttribute(name)

Перевіряє, чи є в елемента атрибут з певною назвою, і повертає true або false

2

getAttribute(name)

Отримує значення конкретного атрибута елемента і повертає його

3

setAttribute(name, value)

Встановлює значення для певного атрибута елемента

4

removeAttribute(name)

Видаляє певний атрибут у елемента

Додавання елементів

- **append()** - додає після всіх дочірніх елементів
- **prepend()** - додає перед усіма дочірніми елементами
- **after()** - додає після елемента
- **before()** - додає перед елементом

У всіх цих методах el - це елементи або рядки у будь-якому поєднанні і кількості.

insertAdjacentHTML()

Це сучасний спосіб додавання рядка з HTML-тегами перед, після або всередину елемента.

Синтаксис:

```
elem.insertAdjacentHTML(
  position,
  string
);
```

Позиції для insertAdjacentHTML

- **"beforebegin"** - перед елементом elem
- **"afterbegin"** - всередині elem, перед усіма дочірніми
- **"beforeend"** - всередині elem, після всіх дочірніх
- **"afterend"** - після елемента elem

Видалення елементів: Для того, щоб видалити елемент, використовується метод **remove()**, який викликається на елементі elem, який потрібно видалити.

Цікавий факт: Якщо елемент, який потрібно додати, вже існує в DOM, він спочатку видаляється зі свого попереднього місця і додається у нове. Один елемент не може одночасно перебувати у двох місцях!