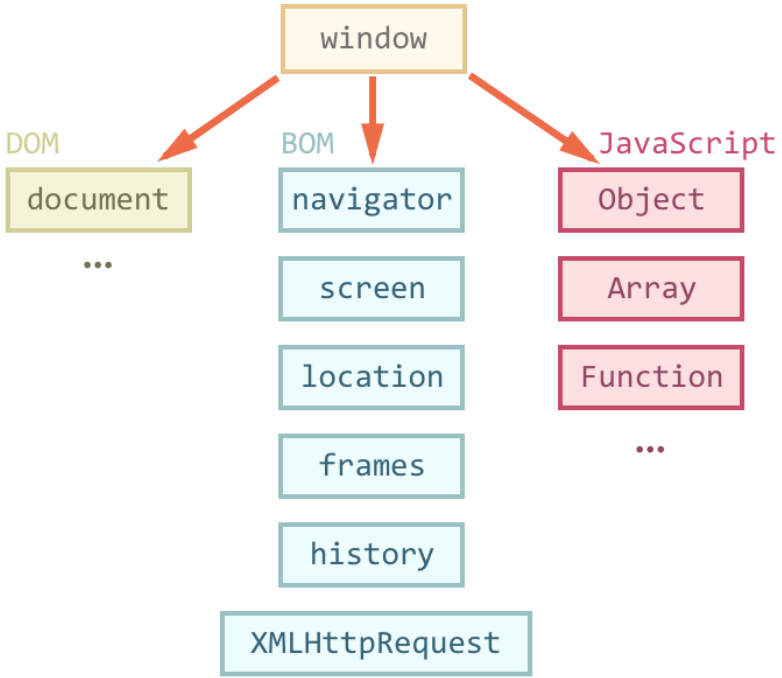
skillUp

DOM

План

- навігація по DOM
- пошук елементів
- додавання і видалення вузлів

Об’єкт Window



Browser Object Model (BOM)

це додаткові об'єкти, що надаються браузером, щоб працювати з усім, крім документа.

Об'єкт **navigator** дає інформацію про самому браузері і операційній системі.
navigator.userAgent - інформація про поточний браузері,
navigator.platform - інформація про ОС.

Об'єкт **location** дозволяє отримати поточний URL і перенаправити браузер за новою адресою.

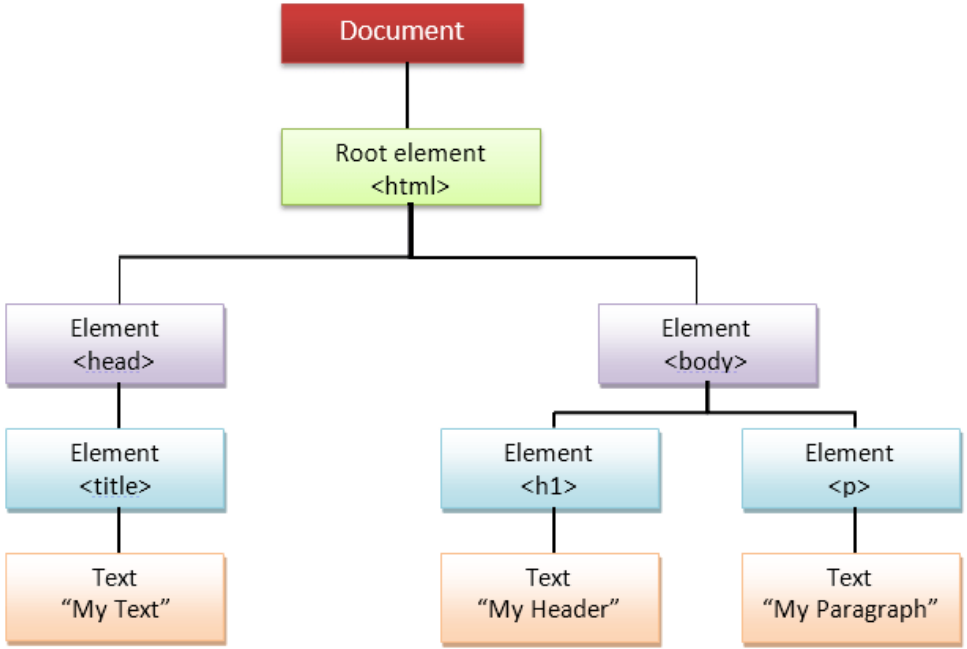
location.href - поточний url

Document Object Model (DOM)

- Глобальний об'єкт **document** дає можливість взаємодіяти з вмістом сторінки
- Кожен HTML-тег утворює вузол дерева з типом «елемент»
- Вкладені в нього теги являються дочірніми вузлами

```
<!DOCTYPE html>
<html>
<head>
  <title>My text</title>
<body>
            <h1>My header</h1>
      <p>My paragraph</p>
</body>
</html>
```

https://learn.javascript.ru/dom-nodes



Можливості javascript в браузері

- JavaScript може змінювати всі HTML-елементи на сторінці
- JavaScript може змінювати всі атрибути HTML на сторінці
- JavaScript може змінювати всі стилі CSS на сторінці
- JavaScript може видаляти існуючі елементи HTML та атрибути
- JavaScript може додати нові елементи HTML та атрибути
- JavaScript може реагувати на всі існуючі HTML-події на сторінці
- JavaScript може створити нові HTML-події на сторінці

Навігація по DOM

- Доступ до DOM починається з об'єкта document

<HTML> = **document.documentElement**

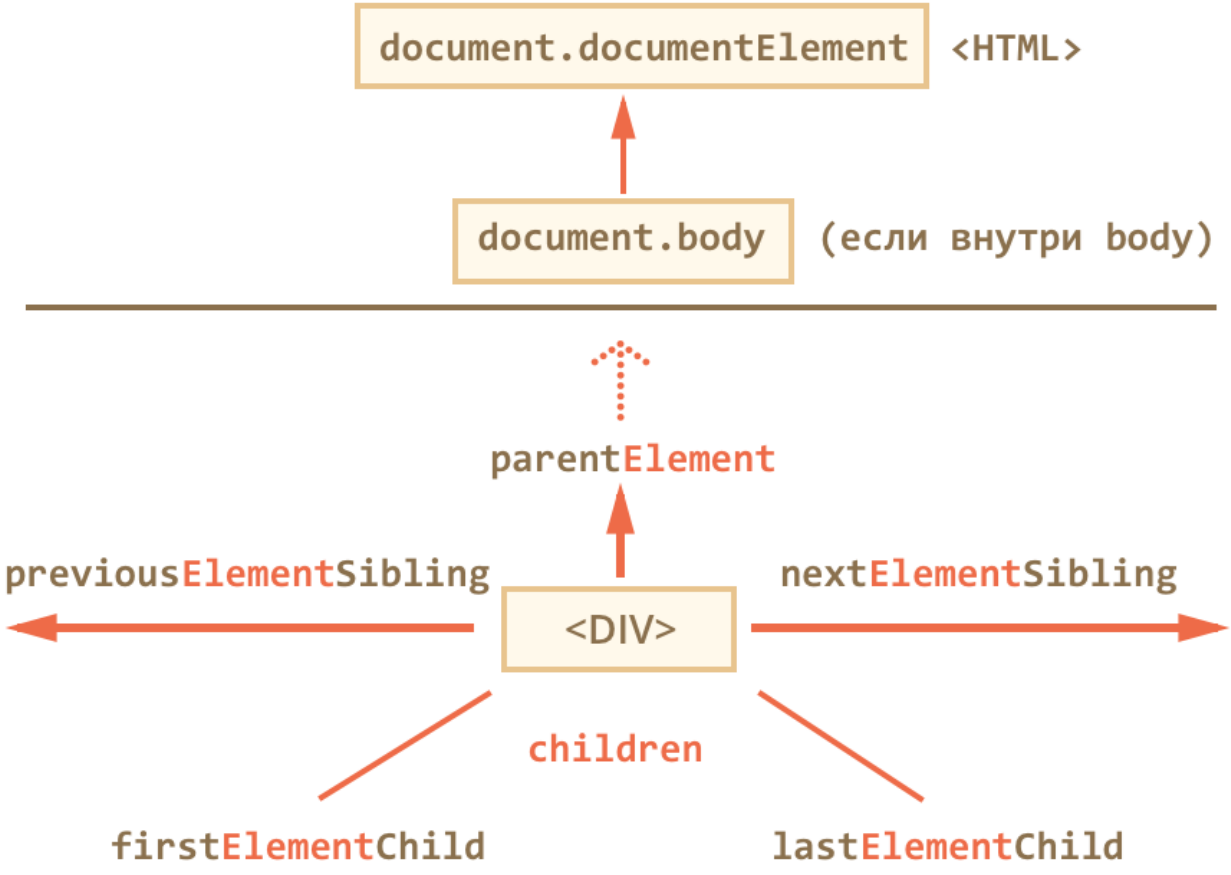
<BODY> = **document.body**

Дочірні елементи (або діти) - елементи, які лежать безпосередньо всередині даного. Наприклад, всередині <HTML> зазвичай лежать <HEAD> і <BODY>.

Нащадки - все елементи, які лежать всередині даного, разом з їхніми дітьми, дітьми їхніх дітей і так далі. Тобто, все піддерево DOM.

Батьківський елемент - по відношенню до елемента лежить безпосередньо вище в дереві DOM

https://learn.javascript.ru/traversing-dom


document.documentElement
<HTML>
document.body
(если внутри body)
parentElement
previousElementSibling
<DIV>
nextElementSibling
children
firstElementChild
lastElementChild

Властивості об’єкта DOM

- **children** - тільки дочірні вузли-елементи, повертає колекцію
- **firstElementChild**, **lastElementChild** - відповідно, перший і останній діти-елементи
- **previousElementSibling**, **nextElementSibling** - сусіди-елементи
- **parentElement** - батько-елемент

Практика

```
<div>Технології:</div>
  <ul>
    <li>HTML</li>
    <li>CSS</li>
  </ul>
```

- знайти ul елемент
- знайти другий li

Пошук html елементів

- **document.getElementById(id)** - знаходить елемент за ідентифікатором елемента
- **document.getElementsByTagName(name)** - знаходить елементи за іменем тегів
- **document.getElementsByClassName(name)** - знаходить елементи за назвою класу
- **document.querySelectorAll(selector)** - знаходить елементи відповідному css селектору
- **document.querySelector(selector)** - знаходить **перший** елемент відповідному css селектору

- Метод **elem.matches(css)**, який перевіряє, чи задовольняє елемент CSS-селектору
- Метод **elem.closest(css)** шукає найближчий елемент вище за ієрархією DOM, що підходить під CSS-селектор css. Сам елемент теж включається в пошук

Зміна властивостей елементів HTML

- **element.innerHTML** = новий вміст html
  Змінює внутрішній HTML елемент
- **element.style.property** = new style
  Змінює стиль елемента HTML
- **getComputedStyle(element[, pseudo])** - повертає стилі елемента

```
document.getElementById('main').innerHTML = 'New text';
document.body.style.backgroundColor = 'red';
```

Практика

```
<div>Технології:</div>
  <ul class=”list”>
    <li>HTML</li>
    <li>CSS</li>
  </ul>
```

- знайти елемент з класом list
- знайти другий li і змінити в ньому текст
- зробити текст в div червоним

Атрибути

- **elem.hasAttribute(name)** - перевіряє наявність атрибута
- **elem.getAttribute(name)** - отримує значення атрибута
- **elem.setAttribute(ім'я, значення)** - встановлює атрибут
- **elem.removeAttribute(name)** - видаляє атрибут

```
document.getElementByTagName(a).getAttribute('href');
```

За допомогою нестандартних атрибутів можна прив'язати до елементу дані, які будуть доступні в JavaScript. Це робиться за допомогою атрибутів з назвами, які починаються часткою на **data-**

Класи

- **elem.classList.contains("class")** - повертає true / false, залежно від того, чи є у елемента клас class.
- **elem.classList.add("class")** - додає клас class
- **elem.classList.remove("class")** - видаляє клас class
- **elem.classList.toggle("class")** - якщо класу class немає, додає його, якщо є - видаляє.

classList можна перебрати класи через for

Практика

- до чекбокса додати атрибут checked
- якщо параграф має клас “text” то видалити його і навпаки

Додавання і видалення елементів

- **document.createElement(елемент)** - створює елемент HTML
- **document.removeChild(елемент) -** видаляє елемент HTML
- **document.appendChild(елемент)** -додає елемент HTML
- **document.insertBefore(elem, nextSibling)** -додає елемент HTML після
- **document.replaceChild(елемент)** - замінює елемент HTML

https://learn.javascript.ru/modifying-document

Приклад додавання елемента

```
<ul id="list">
  <li>0</li>
  <li>1</li>
  <li>2</li>
</ul>
```

Додати елемент в кінець списку

```
var newLi = document.createElement('li');
  newLi.innerHTML = 'Привет, мир!';
  list.appendChild(newLi);
```

Додати елемент після 2 елементу списку

```
var newLi = document.createElement('li');
  newLi.innerHTML = 'Привет, мир!';
  list.insertBefore(newLi, list.children[1]);
```

Практика

```
<ul id="list"></ul>
```

- додати в список 5 елементів li з текстом від 1 до 5

Клонування елементів

- **elem.cloneNode(true)** створить «глибоку» копію елемента - разом з атрибутами, включаючи піделементи. Якщо ж викликати з аргументом false, то копія буде зроблена без дочірніх елементів

```
var div2 = div.cloneNode(true);
  // копію можна підправити
  div2.querySelector('strong').innerHTML = 'Супер!';
   // вставимо її після поточного повідомлення
  div.insertBefore(div2, div.nextSibling);
```

Видалення вузлів

- **parentElem.removeChild(elem)** - видаляє elem зі списку дітей parentElem.
- **parentElem.replaceChild(newElem, elem)** серед дітей parentElem видаляє elem і вставляє на його місце newElem.

Обидва ці методи повертають видалений вузол, тобто elem. Якщо потрібно, його можна вставити в інше місце DOM

Сучасні методи для вставки

- node.**append** (... nodes) - вставляє nodes в кінець node,
- node.**prepend** (... nodes) - вставляє nodes в початок node,
- node.**after** (... nodes) - вставляє nodes після вузла node,
- node.**before** (... nodes) - вставляє nodes перед вузлом node,
- node.**replaceWith** (... nodes) - вставляє nodes замість node.

```
var p = document.createElement('p');
    document.body.append(p);
var em = document.createElement('em');
    em.append('Мир!');

    p.append("Привет, ", em);
```

стиль getComputedStyle

- Для того, щоб отримати поточну css властивость, використовується метод window.getComputedStyle
- Для правильного отримання значення потрібно вказати точну властивість. Наприклад: paddingLeft, marginTop, borderLeftWidth.

```
getComputedStyle(element[, pseudo])
var computedStyle = getComputedStyle(document.body);
    alert( computedStyle.marginTop ); // виводить значення в пікселях
```

Посилання

https://learn.javascript.ru/traversing-dom
https://learn.javascript.ru/searching-elements-dom
https://learn.javascript.ru/attributes-and-custom-properties
https://learn.javascript.ru/modifying-document
https://learn.javascript.ru/styles-and-classes

Відео

https://www.youtube.com/watch?v=1FpuKVmotNc
https://www.youtube.com/watch?v=AZx45EHcA1w