Window

BOM

DOM

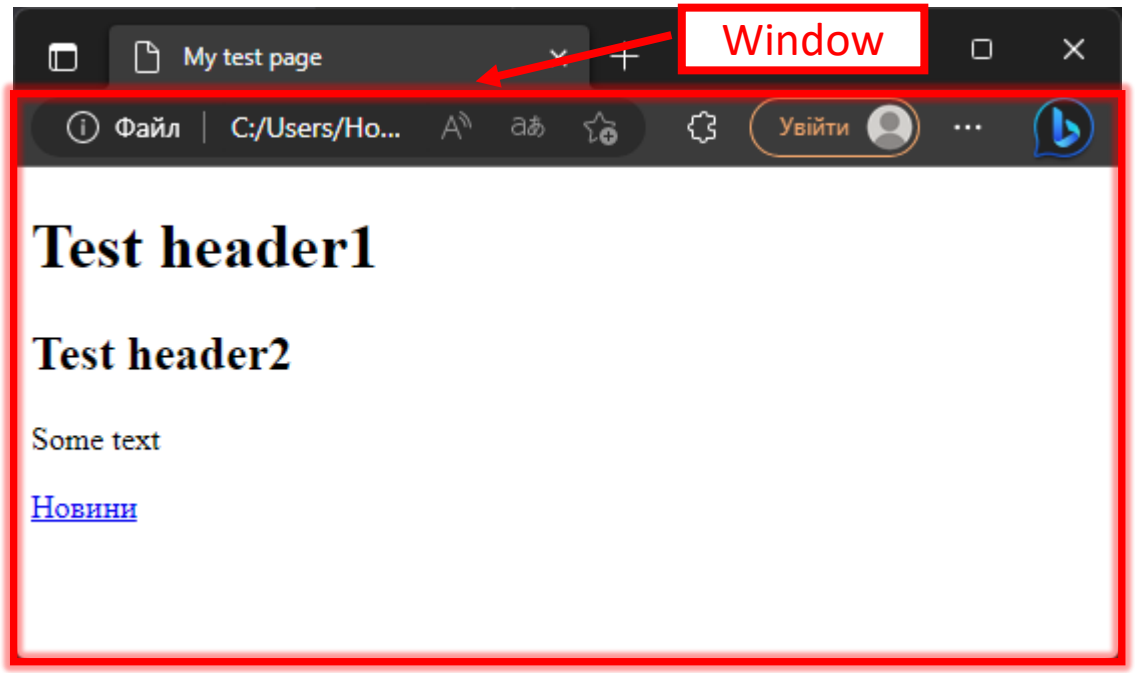
JavaScript

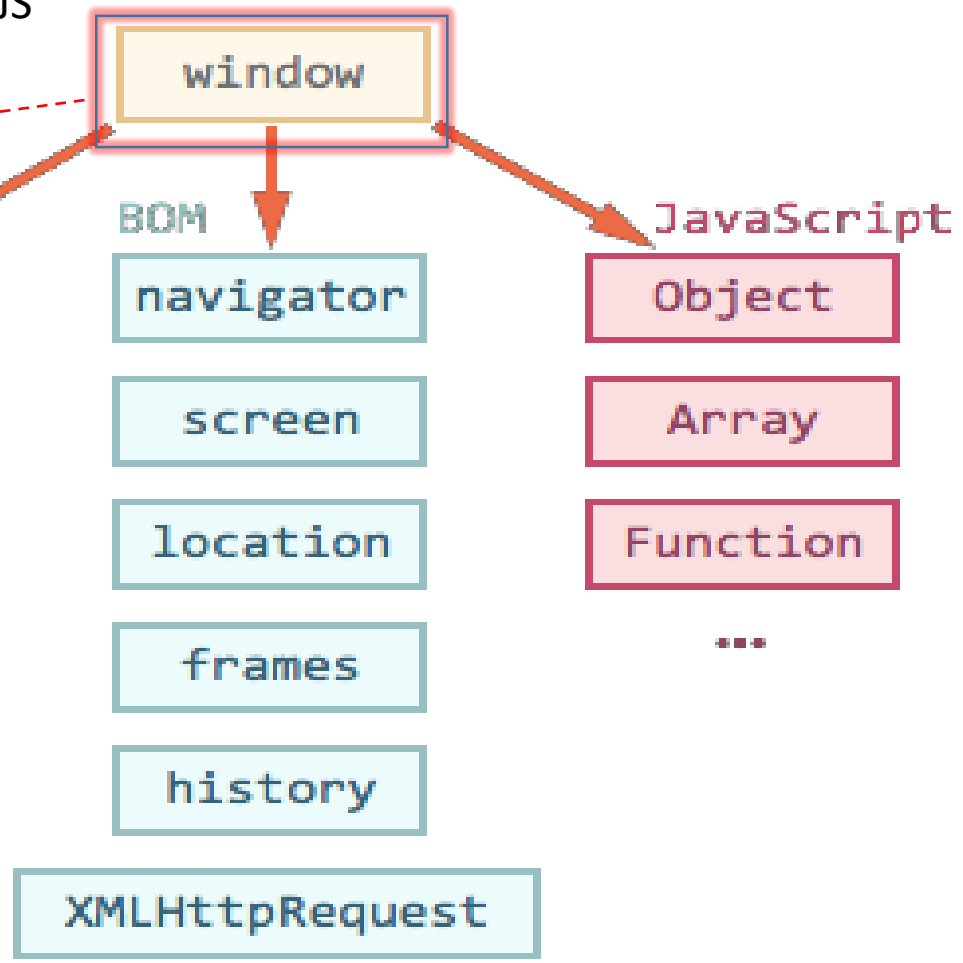
Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

Об'єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів





https://uk.javascript.info/browser-environment

Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

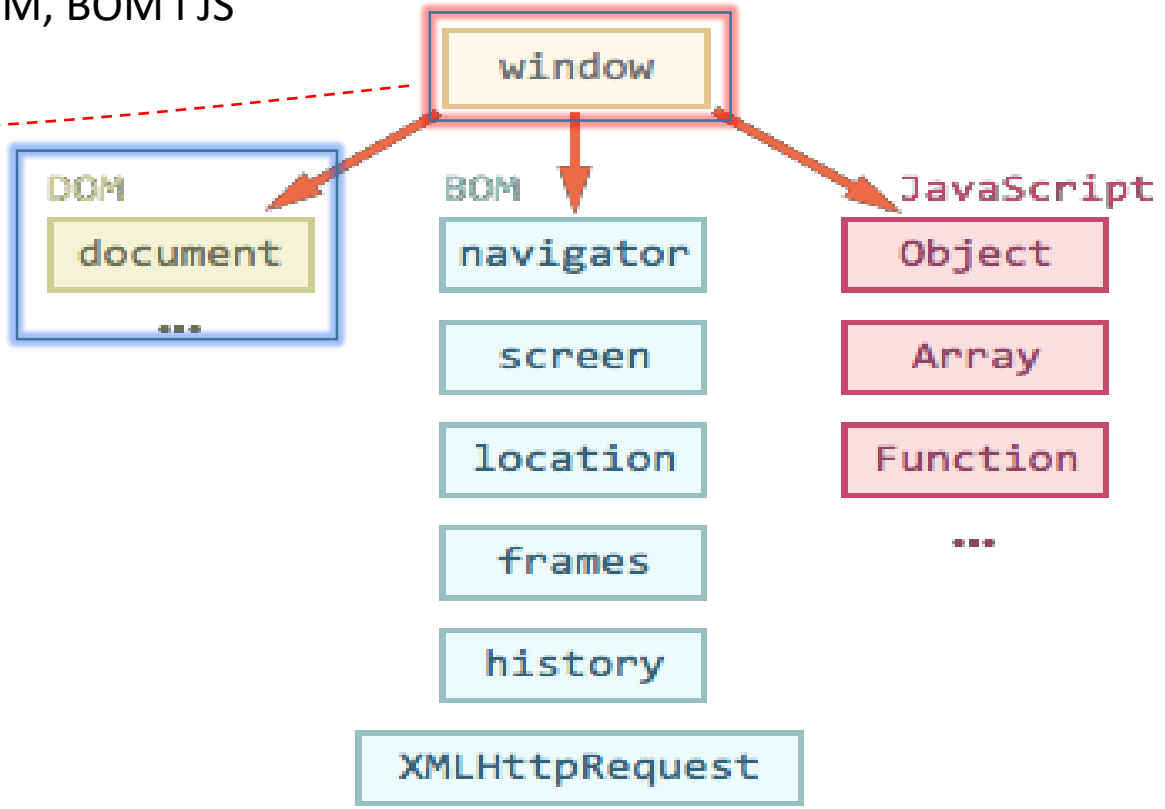
Об'єкт ***window***

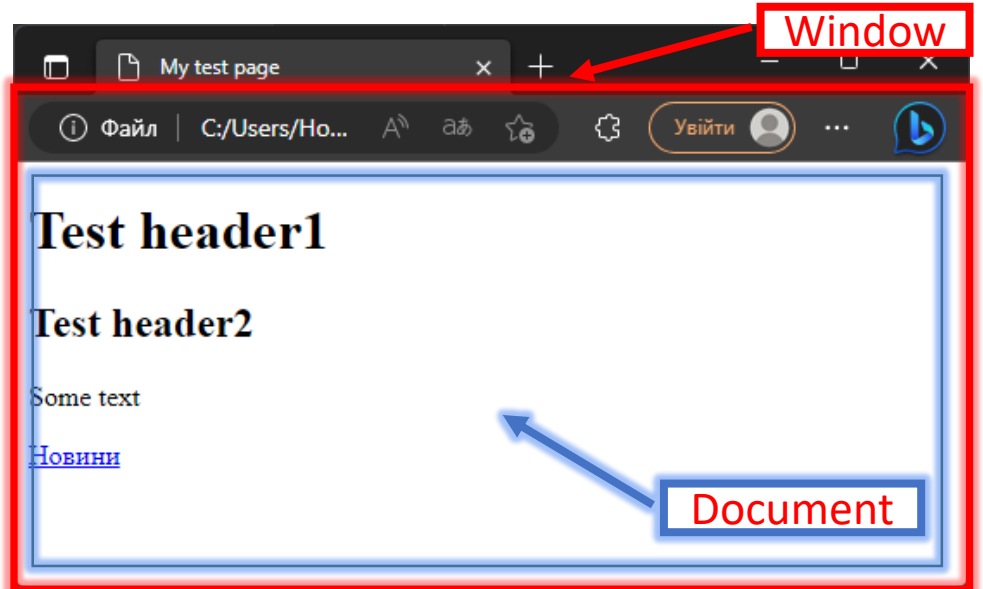
- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів

Об'єкт ***document***

- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об'єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами





Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

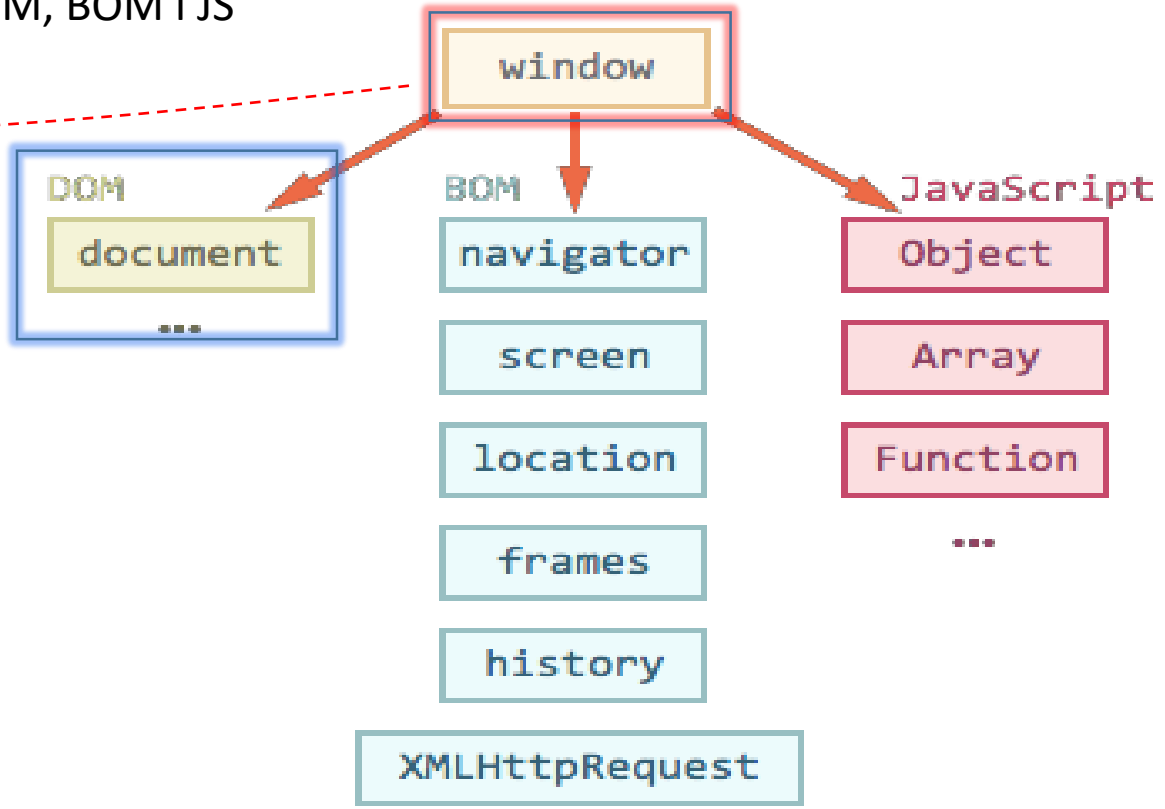
Об'єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів

Об'єкт ***document***

- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об'єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами



1. Браузер отримав HTML

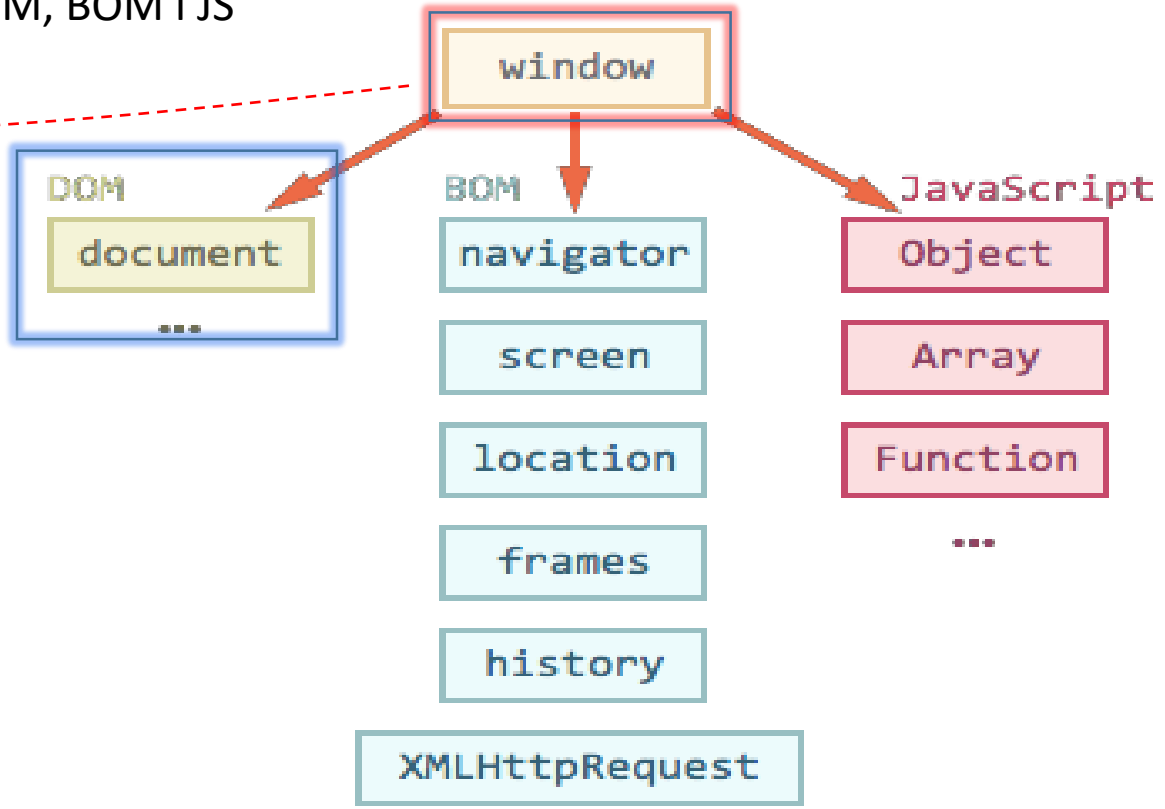
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>My test page</title>
  </head>
  <body>
    <h1>Test header1</h1>
    <div class="container">
      <h2>Test header2</h2>
      <p>Some text</p>
      <a href="https://www.ukr.net/">Новини</a>
    </div>
  </body>
</html>
```

Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

Об'єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів



Об'єкт ***document***

- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об'єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами

1. Браузер отримав HTML

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>My test page</title>
  </head>
  <body>
    <h1>Test header1</h1>
    <div class="container">
      <h2>Test header2</h2>
      <p>Some text</p>
      <a href="https://www.ukr.net/"> Новини </a>
    </div>
  </body>
</html>
```

2. Побудував модель документа у пам'яті

DOM

document

Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

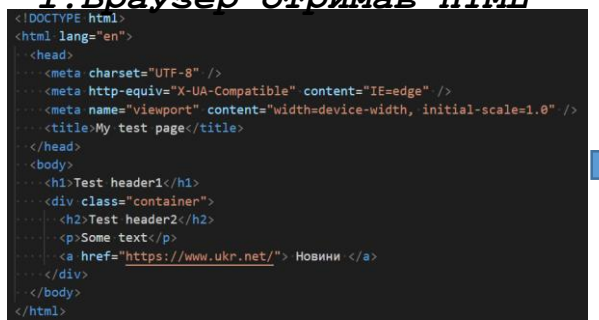
Об'єкт ***window***

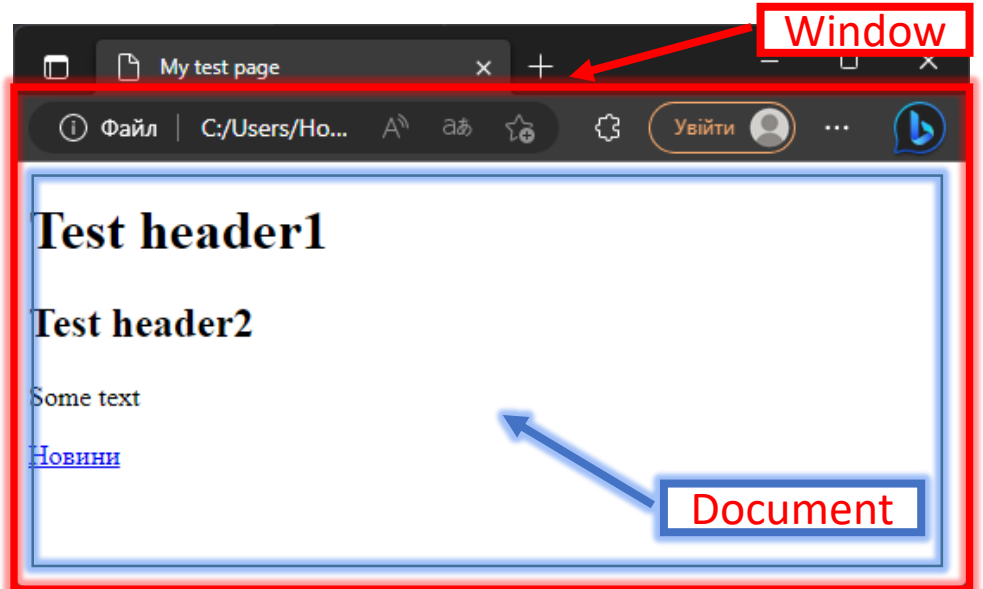
- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів

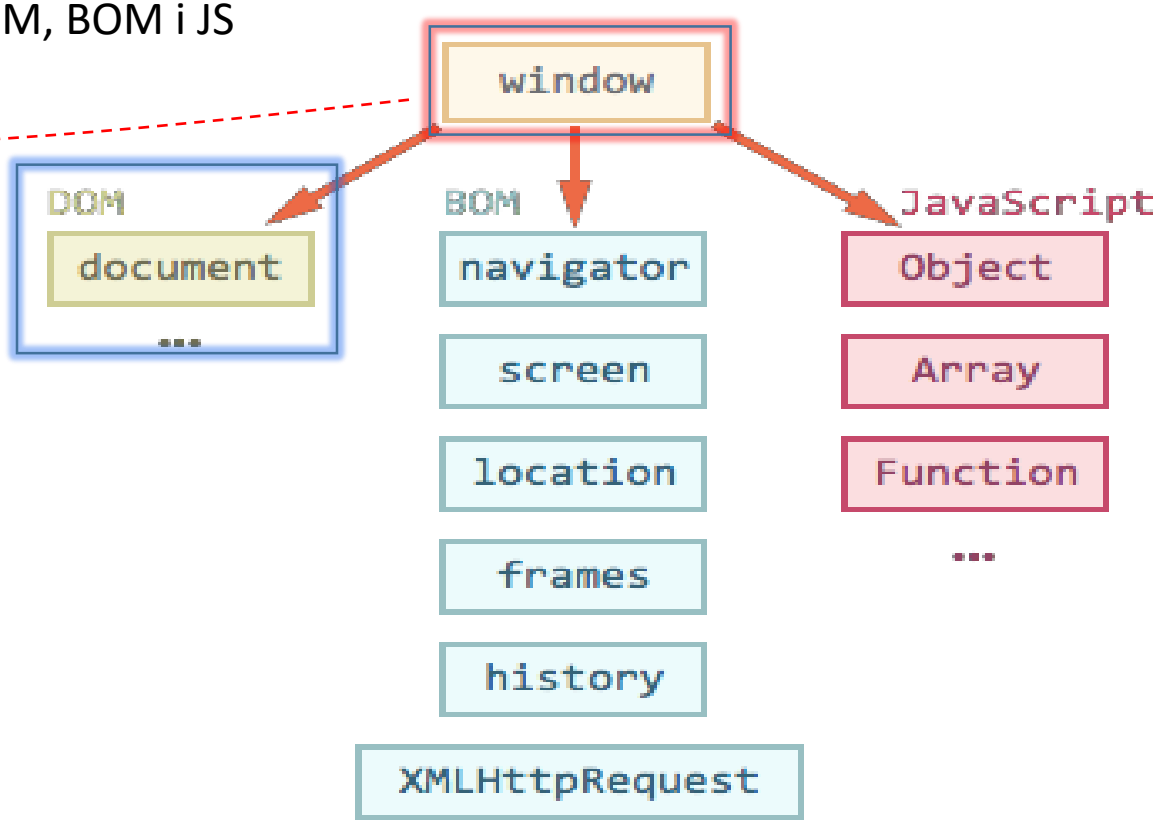
Об'єкт ***document***

- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об'єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами







Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

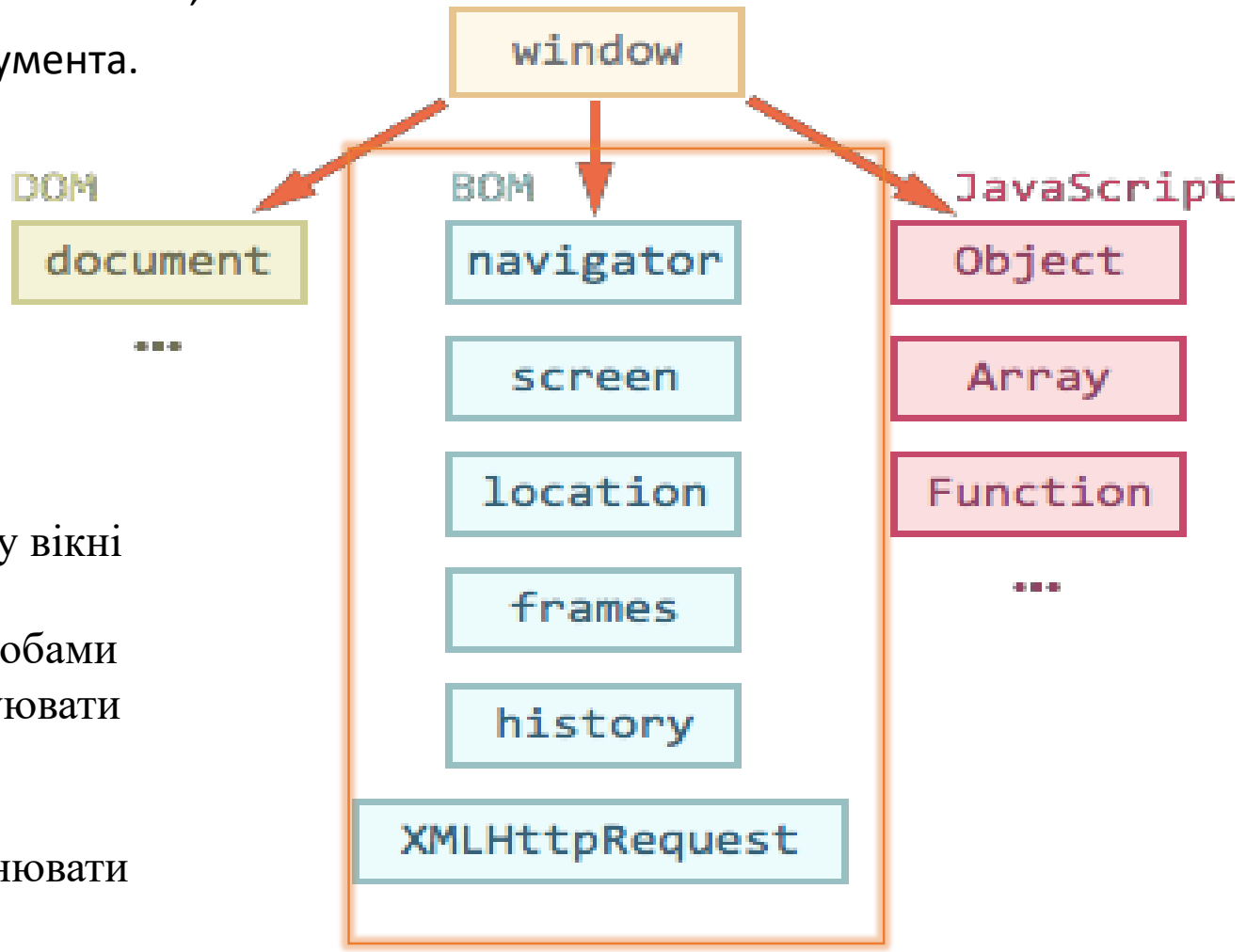
Об'єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів

Об'єкт ***document***

- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об'єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

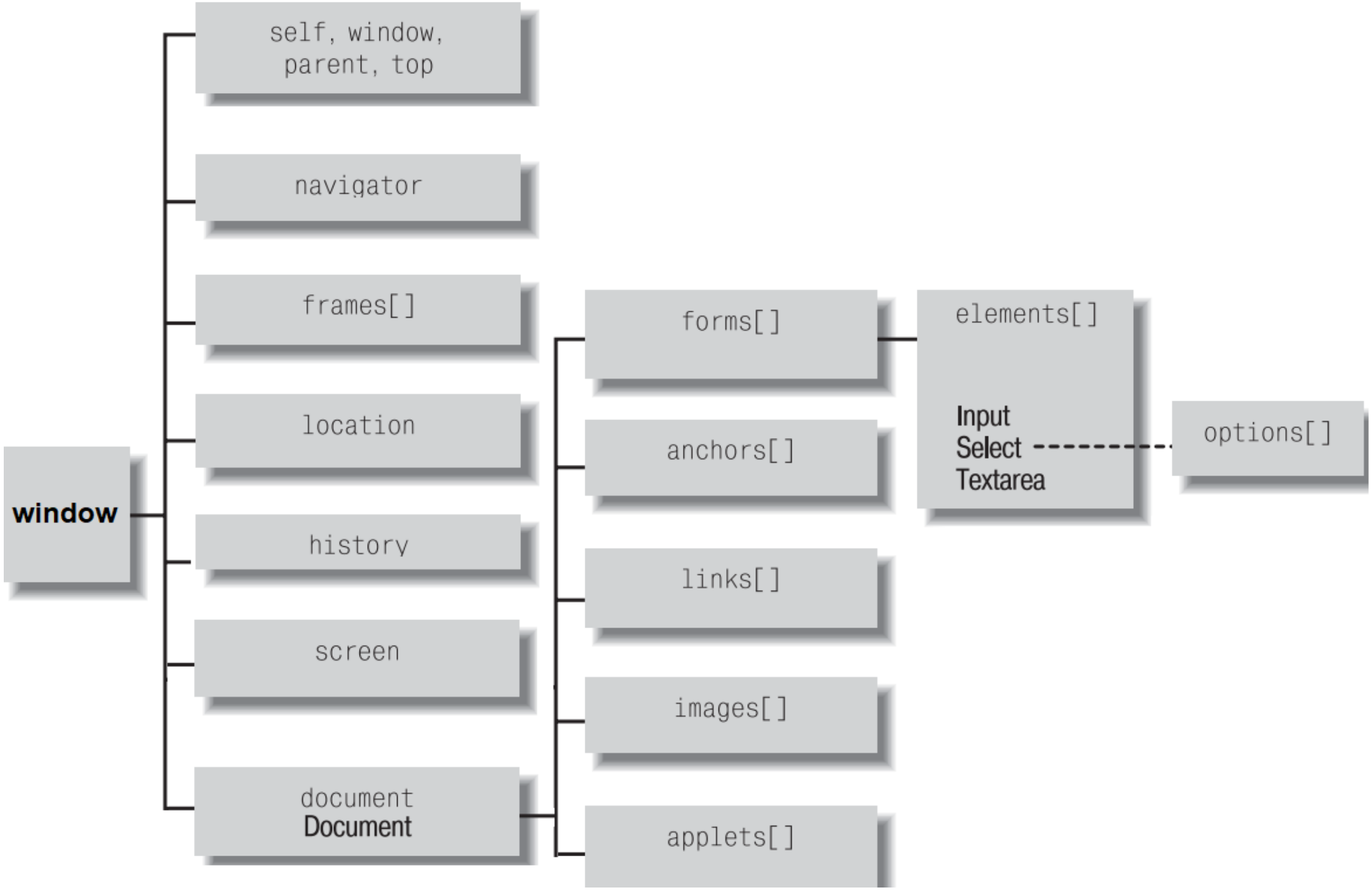
DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами



The Browser Object Model (***BOM***) is a browser-specific convention referring to all the objects exposed by the web browser.

Тобто ***BOM*** представляє собою об'єктну модель браузера, що дозволяє отримувати його властивості та маніпулювати ним

https://uk.javascript.info/browser-environment


window
self, window,
parent, top
navigator
frames[]
location
history
screen
document
Document
forms[]
anchors[]
links[]
images[]
applets[]
elements[]
Input
Select
Textarea
options[]

https://developer.mozilla.org/ru/docs/Web/API/Window

Об’єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об’єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об’єктів

Властивості

document	Повертає об’єкт Document поточного вікна.
frames	Повертає масив елементів <iframe> поточного вікна
history	Повертає посилання на об’єкт History.
location	Повертає посилання на об’єкт Location.
navigator	Повертає посилання на об’єкт Navigator.
screen	Повертає посилання на об’єкт Screen, що зв’язаний з вікном
opener	Задає або отримує посилання на вікно, через яке було відкрите поточне вікно
parent	Повертає батьківське вікно поточного вікна
self	Повертає посилання на поточне вікно або фрейм
status	Повертає/встановлює текст у рядку стану у нижній частині браузера

https://developer.mozilla.org/ru/docs/Web/API/Window

Об’єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об’єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об’єктів

Методи. Діалогові вікна

Метод	Опис
alert()	Виводить модальне вікно з повідомленням
confirm()	Відображає модальне вікно з повідомленням і кнопками ОК и Cancel
prompt()	Відображає діалогове вікно з повідомленням і полем вводу для користувача. Повертає введений користувачем рядок

Методи. Інтервали

Метод	Опис
setInterval()	Викликає функцію або обчислює вираз через вказані інтервали часу (у мілісекундах).
setTimeout()	Викликає функцію або обчислює вираз після вказаного інтервалу часу (у мілісекундах).
clearInterval()	Відміняє дії, що задані за допомогою методу setInterval().
clearTimeout()	Відміняє дії, що задані за допомогою методу setTimeout().

https://developer.mozilla.org/ru/docs/Web/API/Window

Маніпуляція з вікнами

Метод	Опис

open()	Створює і викдриває нове дочірнє вікно
	// Відкрити нове вікно / вкладку з URL https://www.ukr.net/
	window.open ('https://www.ukr.net/')
	`let w = open( 'https://www.ukr.net/', '_blank', `location=yes, height=${parseInt(screen.height) / 2},width=${ parseInt(screen.width) / 2 },scrollbars=yes,status=yes` )`

close()	Закриває вікно, яке було створено з використанням метода window.open().
focus()	Встановлює фокус на поточне окно.
moveBy()	Переміщає поточне вікно на заданную величину.
moveTo()	Переміщує окно у відповідності з вказаними координатами.
print()	Друкує вміст поточного вікна (на принтері)
resizeBy()	Змінює розмір вікна на задану величину.
resizeTo()	Змінює розмір вікна до вказаної ширини і висоти
scrollBy()	Прокрутка документа на вказану кількість пікселів
scrollTo()	Прокрутка документа до вказаних координат.
stop()	Зупиняє завантаження вікна

https://developer.mozilla.org/ru/docs/Web/API/Window

Розміри вікна

Об’єкт **`window`**

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об’єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об’єктів

innerHeight	Повертає висоту області перегляду вікна.
innerWidth	Повертає ширину області перегляду вікна.
outerHeight	Повертає зовнішню висоту вікна, включаючи панель инструментів і полоси прокрутки.
outerWidth	Повертає зовнішню ширину вікна, включаючи панель инструментів і полоси прокрутки.
pageXOffset	Повертає кількість пікселів, на яку поточинй документ був прокручений (по горизонталі) от від верхнього лівого кута вікна
pageYOffset	Повертає кількість пікселів, на на яку поточний документ був прокручений (по вертикалі) від вернього лівого кута вікна
screenLeft	Отримує x-координату верхнього лівого кута вікна відносно верхнього лівого кута екрану
screenTop	Отримує y-координату верхнього кута вікна, стосовно верхньогої чистини екрану
scrollX	Эквивалент властивості pageXOffset.
scrollY	Эквивалент властивості pageYOffset.

https://uk.javascript.info/size-and-scroll-window

Задача. Створити дочірнє вікно (url='https://www.ukr.net/') з розмірами, що дорівнюють половині розмірів поточного вікна і закрити його через 2 секнуди

Задача. Створити дочірнє вікно (url='https://www.ukr.net/') з розмірами, що дорівнюють половині розмірів поточного вікна і закрити його через 2 секнуди

```
let w = open(
  'https://www.ukr.net/',
  '_blank',
  `location=yes, height=${parseInt(screen.height) / 2},
   width=${
    parseInt(screen.width) / 2
  },scrollbars=yes,status=yes`
)
setTimeout(() => {
  w.close()
}, 2000)
```

https://developer.mozilla.org/ru/docs/Web/API/Window

Прокрутка вікна

Об’єкт ***window***

- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об’єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об’єктів

Властивість	Опис
innerHeight	Повертає висоту області перегляду вікна.
innerWidth	Повертає ширину області перегляду вікна.
outerHeight	Повертає зовнішню висоту вікна, включаючи панель инструментів і полоси прокрутки.
outerWidth	Повертає зовнішню ширину вікна, включаючи панель инструментів і полоси прокрутки.
pageXOffset	Повертає кількість пікселів, на яку поточний документ був прокручений (по горизонталі) от від верхнього лівого кута вікна
pageYOffset	Повертає кількість пікселів, на на яку поточний документ був прокручений (по вертикалі) від вернього лівого кута вікна
screenLeft	Отримує x-координату верхнього лівого кута вікна відносно верхнього лівого кута екрану
screenTop	Отримує y-координату верхнього кута вікна, стосовно верхньої чистини екрану
scrollX	Еквівалент властивості pageXOffset.
scrollY	Еквівалент властивості pageYOffset.

https://uk.javascript.info/size-and-scroll-window

The Browser Object Model (***BOM***) is a browser-specific convention referring to all the objects exposed by the web browser.

Тобто ***BOM*** представляє собою об'єктну модель браузера, що дозволяє отримувати його властивості та маніпулювати ним

Navigator	**window.navigator** містить інформацію про браузер відвідувача деякі властивості: • navigator.appName • navigator.appCodeName • navigator.platform	https://developer.mozilla.org/en-US/docs/Web/API/Navigator
Screen	Об'єкт **window.screen** містить інформацію про екран користувача. Властивості • screen.width • screen.height • screen.availWidth • screen.availHeight • screen.colorDepth • screen.pixelDepth	https://developer.mozilla.org/en-US/docs/Web/API/Screen

The Browser Object Model (***BOM***) is a browser-specific convention referring to all the objects exposed by the web browser.

https://developer.mozilla.org/ru/docs/Web/API/Location

Тобто ***BOM*** представляє собою об'єктну модель браузера, що дозволяє отримувати його властивості та маніпулювати ним

Location	Об'єкт **location** може бути використатий для одержання поточної адреси і переадресації на нову сторінку. Деякі властивості: • window.location.href returns the href (URL) of the current page • window.location.hostname returns the domain name of the web host • window.location.pathname returns the path and filename of the current page • window.location.protocol returns the web protocol used (http: or https:) • window.location.assign loads a new document

https://developer.mozilla.org/en-US/docs/Web/API/Location

History	**history** містить історію браузера Деякі властивості: • history.back() - same as clicking back in the browser • history.forward() - same as clicking forward in the browser

https://developer.mozilla.org/en-US/docs/Web/API/History_API

Об’єкт Document

Кожна сторінка у браузері має свій власний об’єкт Document.

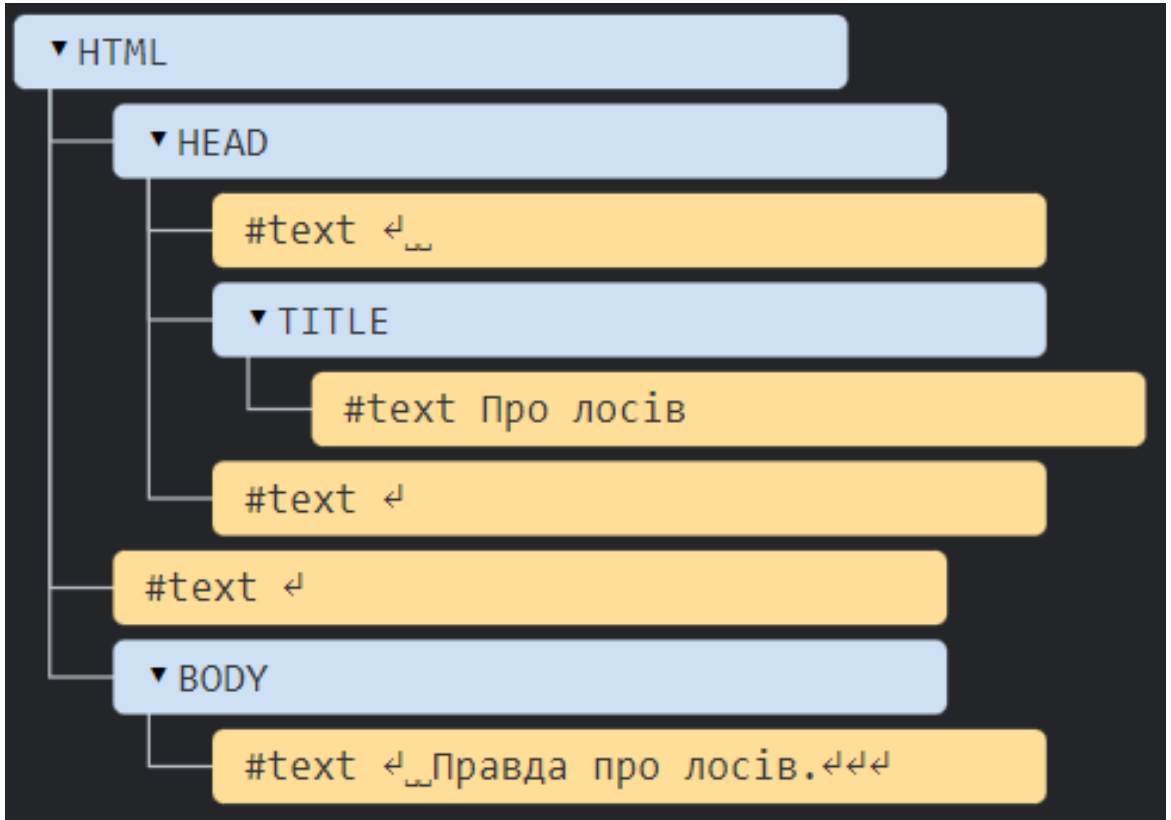
Об’єкт ***document***

- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об’єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами

Згідно DOM-моделі (Document Object Model), документ є ієрархією, деревом. Кожен HTML-тег утворює вузол дерева з типом «елемент». Вкладені в нього теги стають дочірніми вузлами. Для представлення тексту створюються вузли з типом«текст».

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Про лосів</title>
</head>
<body>
  Правда про лосів.
</body>
</html>
```



https://uk.javascript.info/document

Усього розрізняють 12 типів вузлів, але на практиці ми працюємо з чотирма з них:

Документ - точка входу в DOM .

Елементи - основні будівельні блоки.

Текстові вузли - містять, власне, текст.

Коментарі - іноді в них можна включити інформацію, яка не буде показана, але доступна з JS.

Деякі важливі вузли:

- вузол HTML можна отримати як ***document.documentElement***
- BODY - як ***document.body***

https://uk.javascript.info/document

Вибірка елементів сторінки

getElementById()	Повертає елемент з вказаним ідентифікатором
getElementsByClassName()	Повертає колекцію елементів, які відповідають вказаному класу
getElementsByName()	Повертає колекцію елементів, які мають атрибут name з вказаним значенням
getElementsByTagName()	Повертає колекцію елементів з вказаним тегом
querySelector()	Повертає перший елемент, який відповідає вказаному селектору
querySelectorAll()	Повертає масив елементів, які відповідають вказаному селектору

Відформатувати текст

```
<div>
    <h1 > Title </h1>
    <p id="Frirst" class="edge">
        Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello.
        Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello. Hello.
    </p>
    <h1 > Title </h1>
    <section>

        <p id="MyTitle2" class="middle"> From From From From From From From From From From
        From From From From From From From From From From From From From From From From
        From From From From From From
        </p>
    </section>
    <h1 > Title </h1>
    <p>
        From From From From From From From From From From From From From From From From
        From From From From From From From From From From From From From From From From
    </p>
    <section>
        <p>
        <b> Text Text Text Text Text Text </b>
        <b> Text Text Text Text Text Text </b>
        <b> Text Text Text Text Text Text </b>
        </p>
    </section>
</div>
```

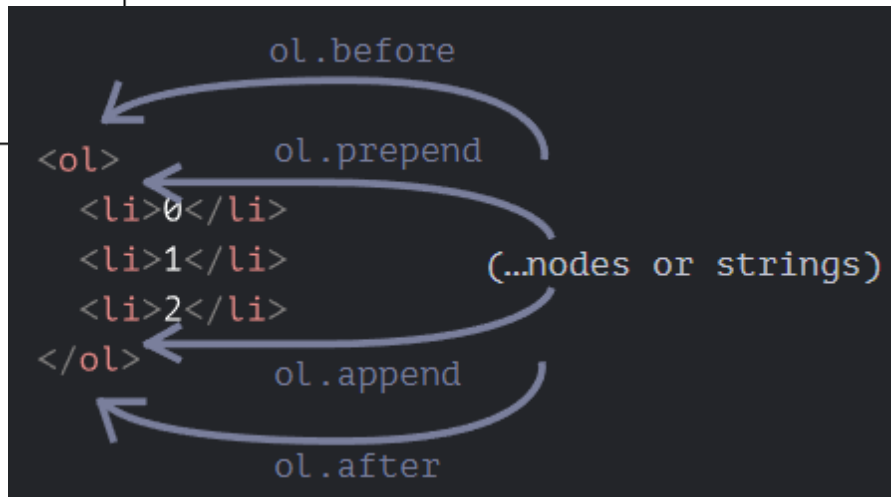
Створення вузлів

Для створення елементів використовуються такі методи:

document.createElement (tag)	Створює новий елемент із зазначеним тегом: let div = **document.createElement** ('div');
document.createTextNode (text)	Створює новий текстовий вузол з даним текстом: let textElem = **document.createTextNode** ('Тут був я ");
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

Додавання елемента

parentElem.**append** (elements or strings) ок	Додає елементи або рядки в кінець списку дочірніх елементів елемента parentElem
parentElem.**appendChild** (elem) Old	Додає elem в кінець дочірніх елементів parentElem
parentElem.**prepend** (elements or strings) ок	Додає елементи або рядки на початок списку дочірніх елементів елемента parentElem
elem.**before** (elements or strings) ок	Вставляє елементи або рядки перед елементом *elem*
elem.**after** (elements or strings) ок	Вставляє елементи або рядки після елемента *elem*
parentElem.**insertBefore** (elem, nextSibling) Old	Вставляє elem в колекцію дітей parentElem, перед елементом nextSibling
elem.**replaceWith** (elements or strings) ок	замінює вузол *elem* заданим списком вулів або рядків



https://uk.javascript.info/document

Видалення вузлів

element. remove() ок	Видаляє поточний елемент
parentElem.removeChild (elem) Old	Видаляє `elem` зі списку дочірніх елементів елемента parentElem.
parentElem.replaceChild (newElem, elem) Old	Серед дочірніх елементів елемента `parentElem` видаляє `elem` і вставляє на його місце `newElem`.

Задача. Створити 2 діви з рандомними числами

Задача. Створити 4 інпути для введення чисел

Задача. Створити таблицю з 3 ряками і 4 стовпцями з рандомними числами

Вміст елемента

innerHTML: вміст елемента	Властивість innerHTML дозволяє отримати/змінити HTML-вміст елемента у вигляді рядка.
`outerHTML: HTML елемента цілком`	Властивість outerHTML містить HTML елемента цілком document.body.children[0]. outerHTML
innerTEXT: вміст елемента як текст	Властивість innerTEXT дозволяє отримати/змінити вміст елемента як текст (навіть якщо там будуть теги, то вони будуть виводитись як текст)
nodeValue / data: вміст текстового вузла	Властивість innerHTML є тільки у вузлів-елементів. Вміст текстових вузлів або коментарів на читання і запис через властивість data. document.body.childNodes[0].data

Задача. Видалити вміст усіх div

Задача. Кожен заголовок h1 замінити на текст «Загловок»

Стилі

- Операції з стилями можна здійснювати з використання властивості елемента **style**

Використовуємо
camelCase
для назв властивостей

Приклади:

```
document.body.style.backgroundColor = 'red';
```

```
myDiv.style.color= ‘blue’
myDiv.style.width = ‘200px’
myDiv.style.height = ‘100px’
```

- Усі властивості, що задані у `style` як один рядок доступні через властивість `style.cssText`

  Приклад:

```
myDiv.style.cssText = “color: blue; width = 200px; height = 100px;”
```

- Поточний набір властивостей можна отримати з використанням `getComputedStyle(elem, [pseudo])` (`pseudo` – дає можливість отримати властивості псевдоелемента)

  Приклад:

```
getComputedStyle(myDiv)
```

Усі заголовки зробити червоними

Маніпуляція з класами елементів

`elem.className`	Відповідає усьому вмісту як одному рядку тексту (може містити декілька класів) – значення атрибуту `class`
	Приклад:
	`<div id = "myDiv" class= "container main" > … </div>`
	
	`myDiv.className = "container main"`

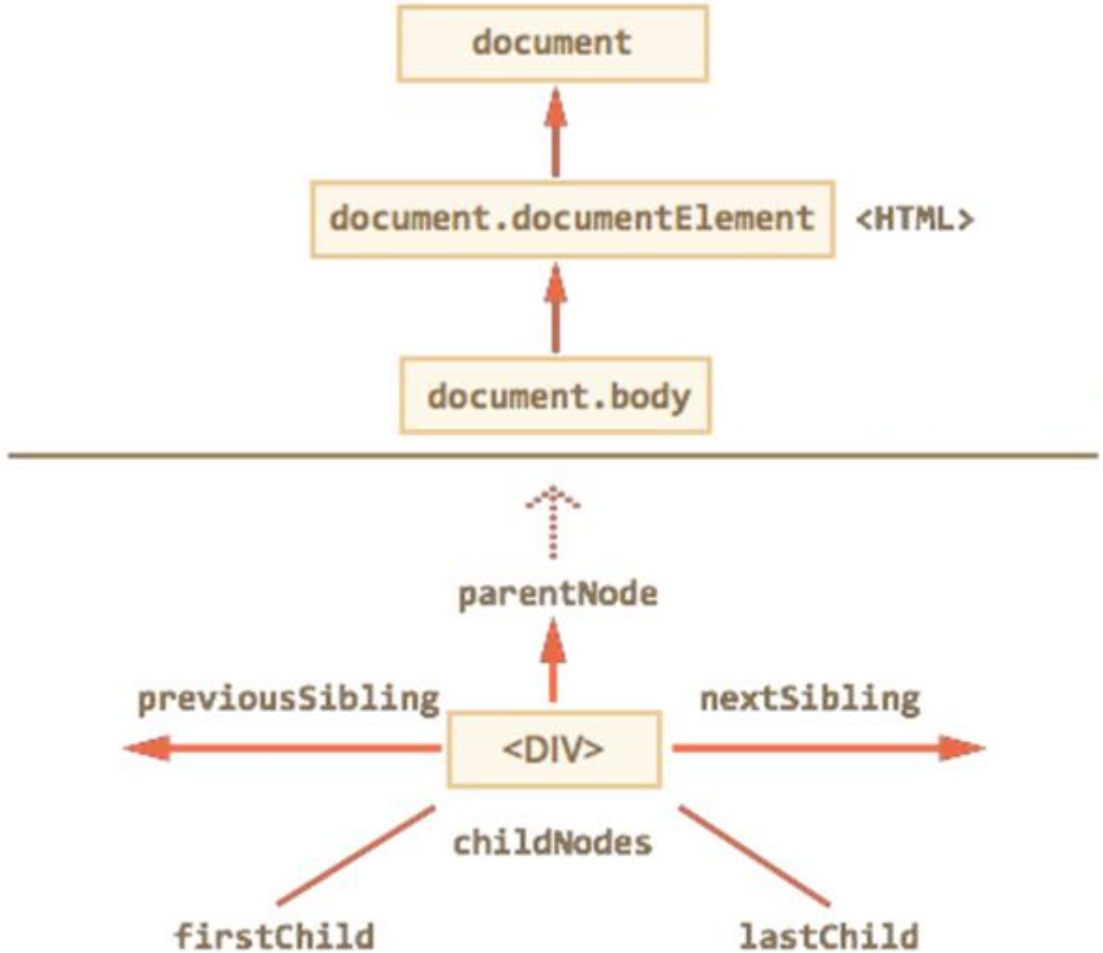
`elem.classList`	спеціальний об'єкт, який містить методи для • додавання - `elem.classList.add(`ім'я класу`)`, • видалення - `elem.classList.remove(`ім'я класу`)`, • перемикання - `elem.classList.toggle` (ім'я класу), (додає клас якщо не існує, якщо існує - видаляє) • перевірку наявності класу `elem.classList.contains(`ім'я класу`)`

Атрибути

`elem.setAttribute(name, value)`	Встановлює значення атрибуту
`elem.getAttribute(name)`	Читаємо значення атрибуту
`elem.removeAttribute(name)`	Видаляємо атрибут
`elem.hasAttribute(name)`	Перевіряємо чи існує атрибут

Навігація DOM-вузлами

Схема взаємозв'язку між вузлами

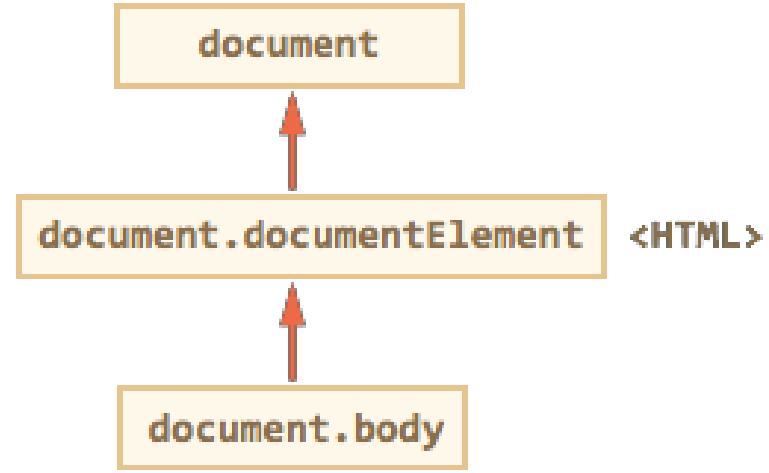


<HTML> = document.documentElement
<BODY> = document.body
<HEAD> = document.head

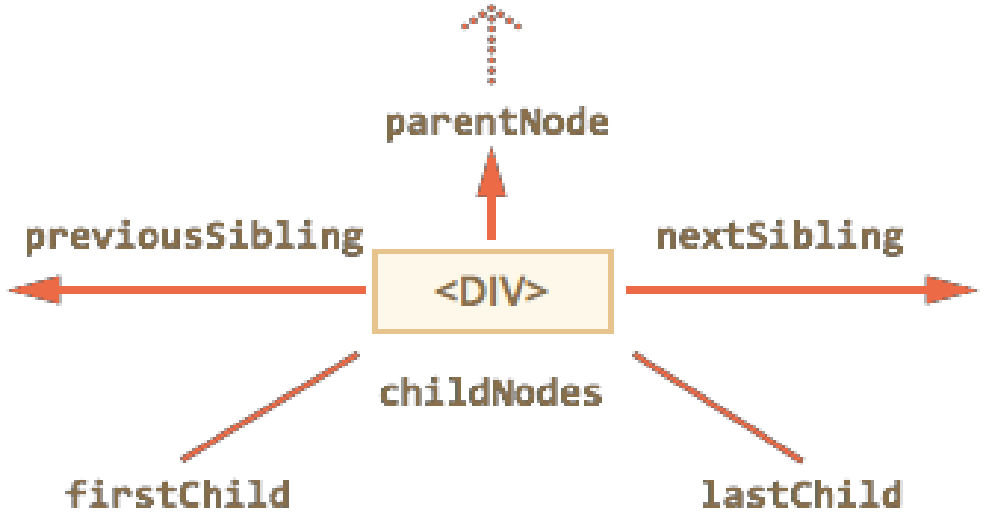
https://uk.javascript.info/dom-navigation

Якщо у DOM наступного/попереднього/батьківського/дочірнього вузла «немає» або «вузол не знайдений», використовується не undefined, **a null.**

childNodes	Дочірні вузли(або діти) -вузли, які лежать безпосередньо всередині даного (всі дочірні елементи, включаючи текстові)
firstChild	Перший дочірній вузол
lastChild	Останній дочірнійі вузол
previousSibling	Вузол, що знаходиться перед заданим
nextSibling	Вузол, що знаходиться після даного

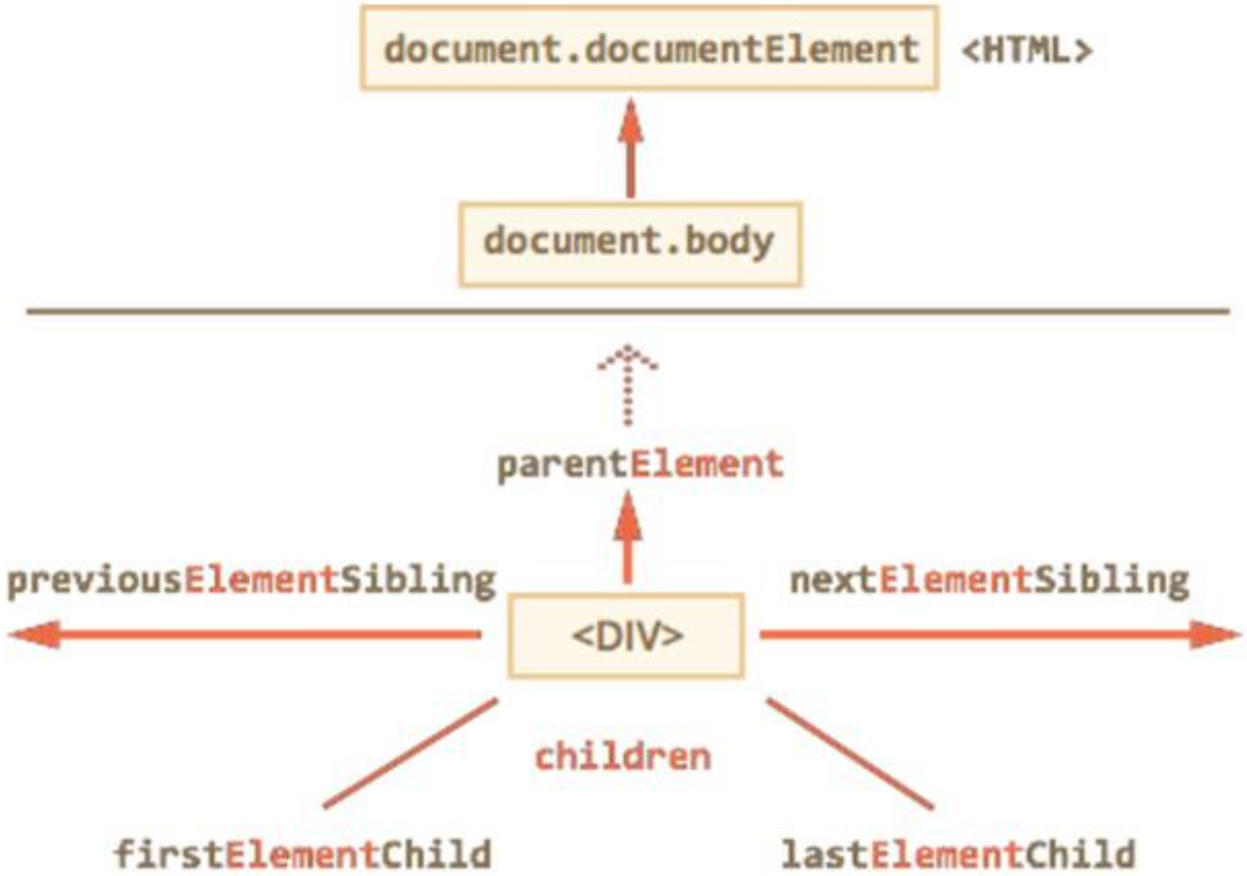


<HTML> = document.documentElement
<BODY> = document.body
<HEAD> = document.head

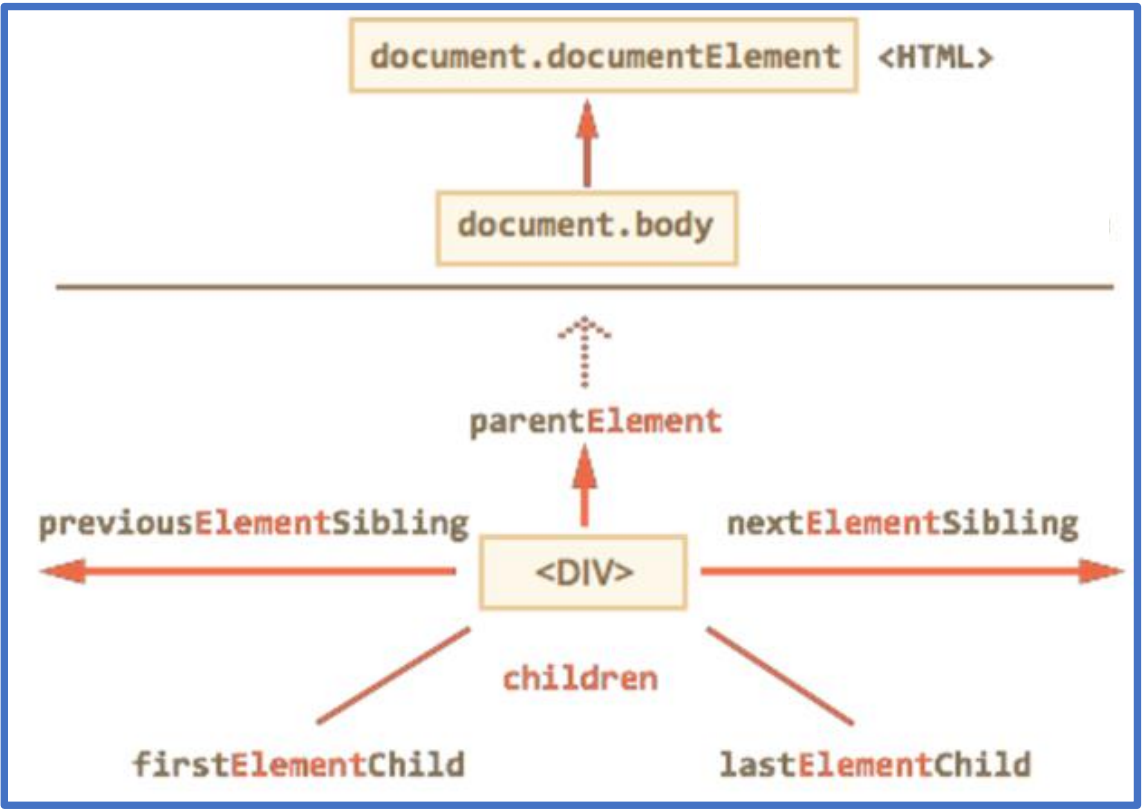


https://uk.javascript.info/dom-navigation

Навігація тільки за елементами



https://uk.javascript.info/dom-navigation



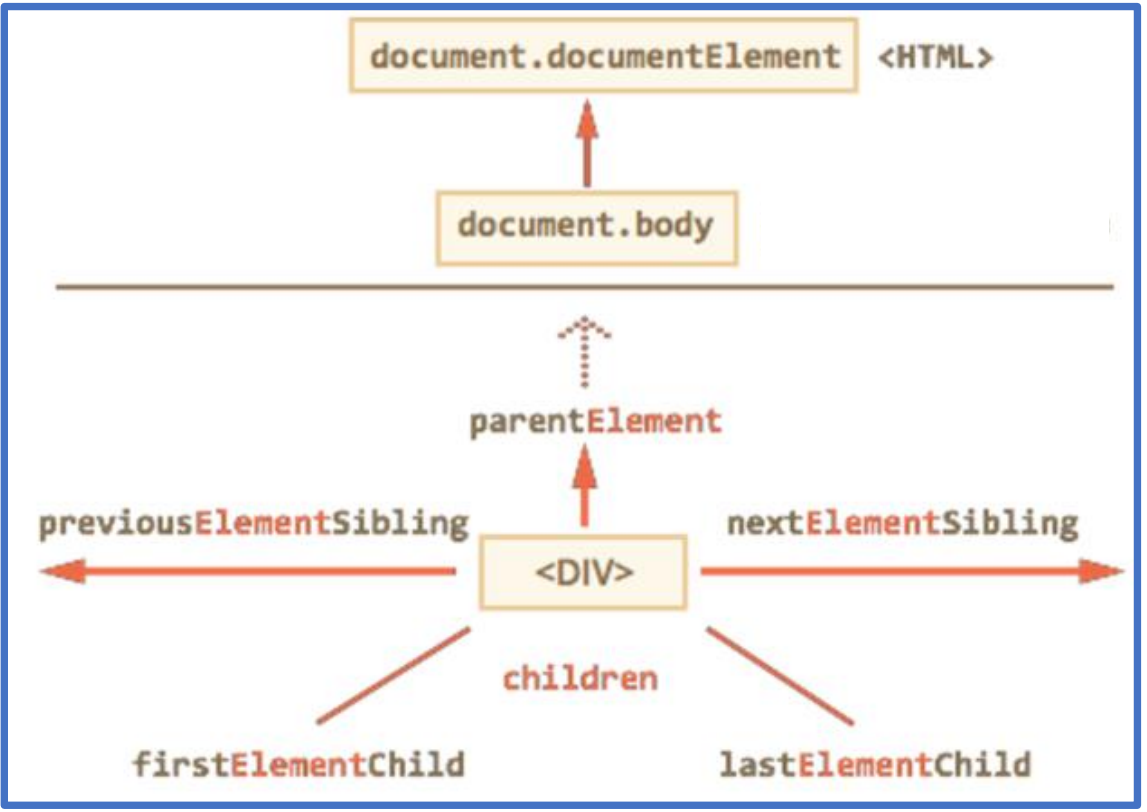
*візьмемо за приклад елемент "***ol***"*

const element=document.querySelector('ol')

Element

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Comp
    <meta name="viewport" conte
    <title>Document</title>
  </head>
  <body>
    <h2>Some header</h2>
    <ol>
      <li>00</li>
      <li>11</li>
      <li>22</li>
      <li>33</li>
    </ol>
    <div>
      Some content
    </div>
```

https://uk.javascript.info/dom-navigation



*візьмемо за приклад елемент "***ol***"*

const element=document.querySelector('ol')

Element

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Comp
    <meta name="viewport" conte
    <title>Document</title>
  </head>
  <body>
    <h2>Some header</h2>
    <ol>
      <li>00</li>
      <li>11</li>
      <li>22</li>
      <li>33</li>
    </ol>
    <div>
      Some content
    </div>
```

element.parentElement

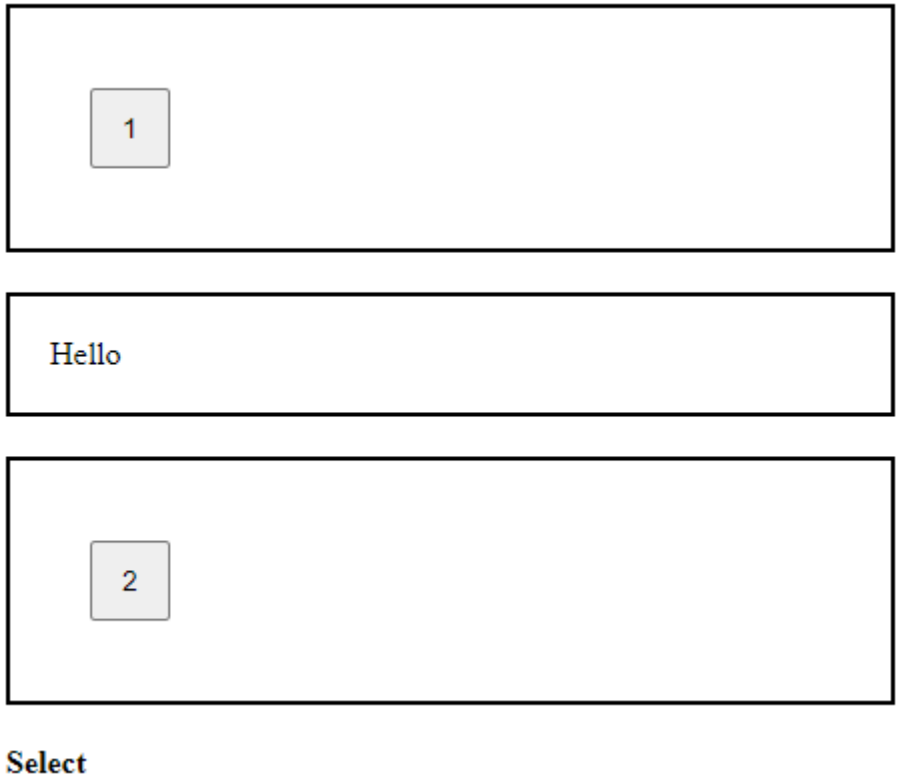
parentElement

*Найближчий «батьківський» елемент, всередині якого міститься "***ol***"*

https://uk.javascript.info/dom-navigation

nav1. Дано елементи div. При натисненні на «Select» підсвічувати div, у якому знаходяться кнопки зеленою рамкою

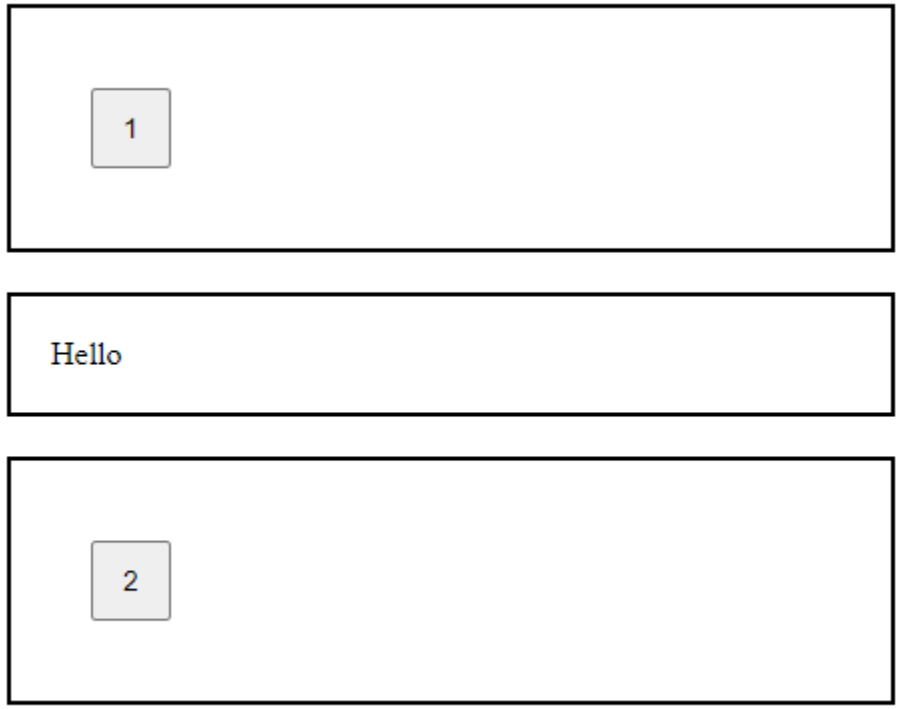
```
<body>
    <div>
        <button>1</button>
    </div>

    <div>
        <span>Hello</span>
    </div>

    <div>
        <button>2</button>
    </div>

    <strong id="operation">Select</strong>
</body>
```



nav1. Дано елементи div. При натисненні на «Select» підсвічувати div, у якому знаходиться кнопки зеленою рамкою

```
<body>
    <div>
        <button>1</button>
    </div>

    <div>
        <span>Hello</span>
    </div>

    <div>
        <button>2</button>
    </div>

    <strong id="operation">Select</strong>
</body>
```

let btn =document.querySelector('button')



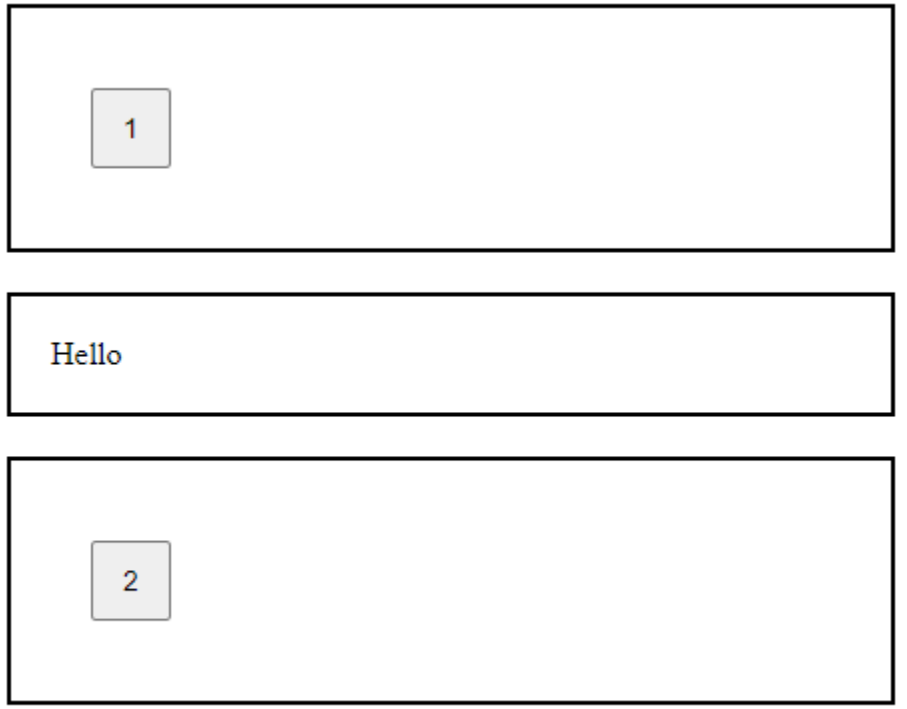
nav1. Дано елементи div. При натисненні на «Select» підсвічувати div, у якому знаходиться кнопки зеленою рамкою

```
<body>
    <div>
        <button>1</button>
    </div>

    <div>
        <span>Hello</span>
    </div>

    <div>
        <button>2</button>
    </div>

    <strong id="operation">Select</strong>
</body>
```

btn.parentElement

parentElement

let btn =document.querySelector('button')



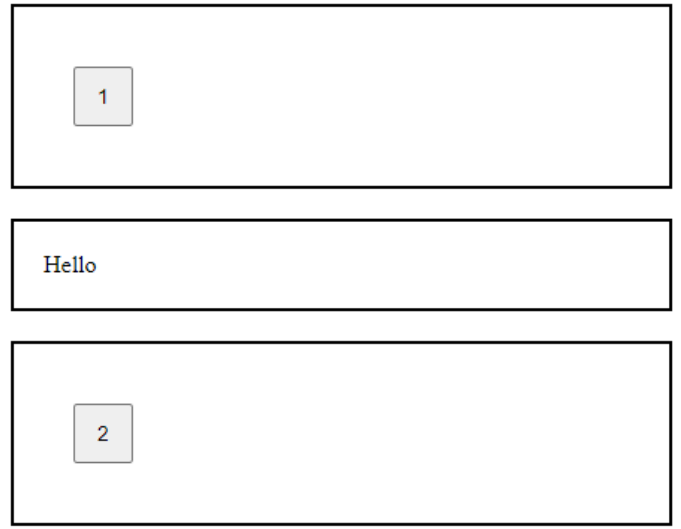
nav1. Дано елементи div. При натисненні на «Select» підсвічувати div, у якому знаходиться кнопки зеленою рамкою

```
<body>
    <div>
        <button>1</button>
    </div>

    <div>
        <span>Hello</span>
    </div>

    <div>
        <button>2</button>
    </div>

    <strong id="operation">Select</strong>
</body>
```

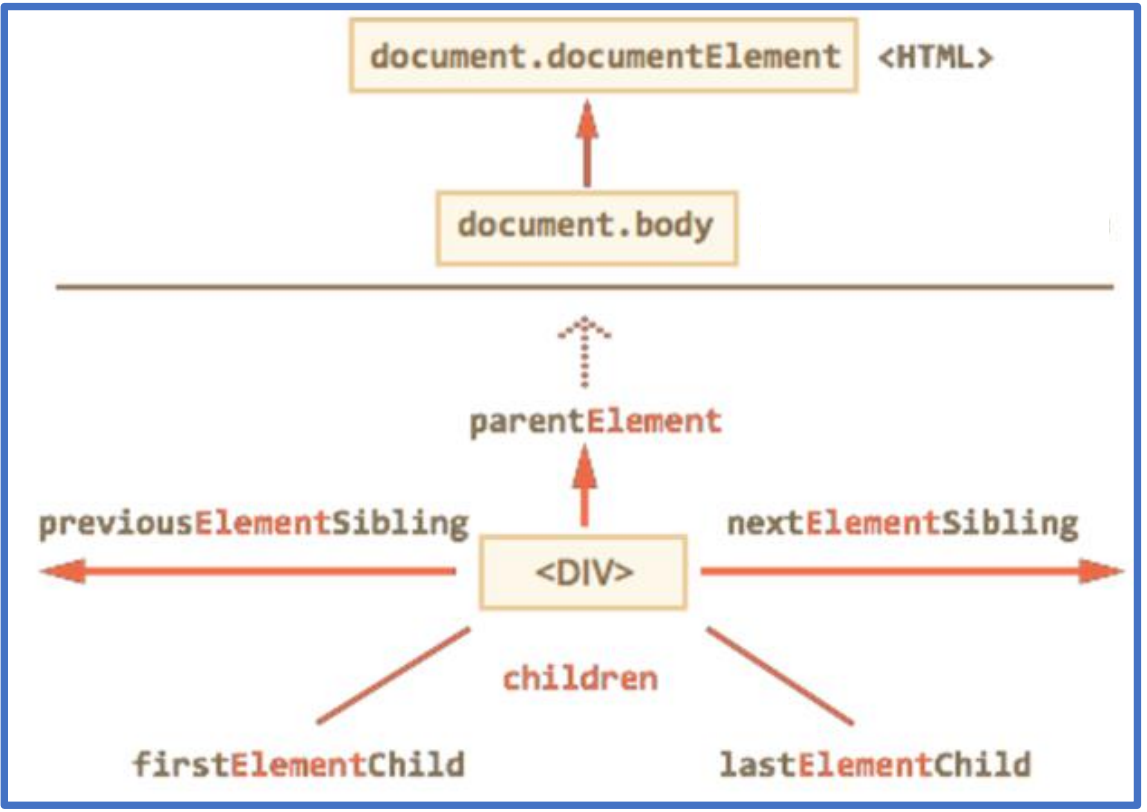
btn.parentElement

parentElement

let btn =document.querySelector(‘button’)




```
const btnList = document.querySelectorAll('button')
for (const btn of btnList) {
    btn.parentElement.style.border = '4px solid green'
}
```



*візьмемо за приклад елемент "***ol***"*

const element=document.querySelector('ol')

Element

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Comp
    <meta name="viewport" conte
    <title>Document</title>
  </head>
  <body>
    <h2>Some header</h2>
    <ol>
      <li>00</li>
      <li>11</li>
      <li>22</li>
      <li>33</li>
    </ol>
    <div>
      Some content
    </div>
```

попередній сусідній елемент

previousElementSibling

element.previousElementSibling

наступний сусідній елемент

element.nextElementSibling

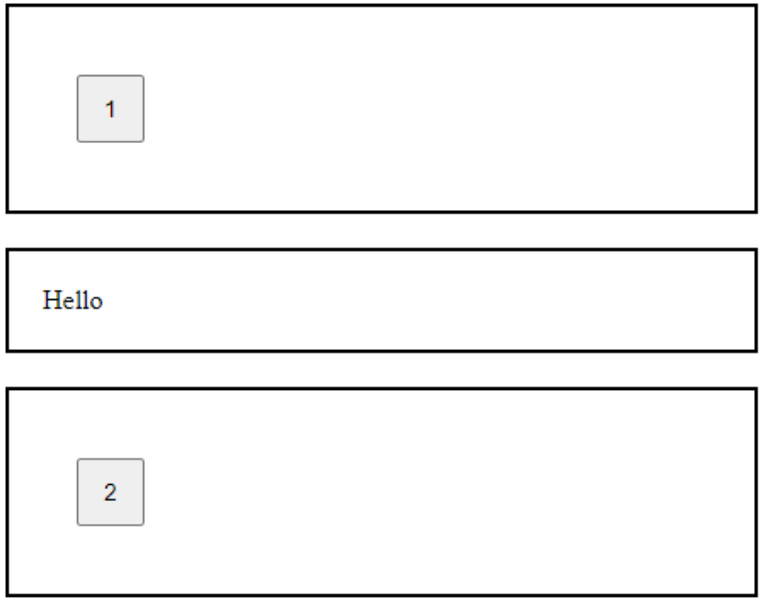
nextElementSibling

https://uk.javascript.info/dom-navigation

nav2. Дано елементи div. Знайти div, у якому span та:
1) наступний після нього зафарбувати у червоний колір (фон)
2) попередній після нього зафарбувати у зелений колір (фон)

```
<body>
    <div>
        <button>1</button>
    </div>

    <div>
        <span>Hello</span>
    </div>

    <div>
        <button>2</button>
    </div>

    <strong id="operation">Select</strong>
</body>
```

const el =
document.querySelector('div:has(span)')



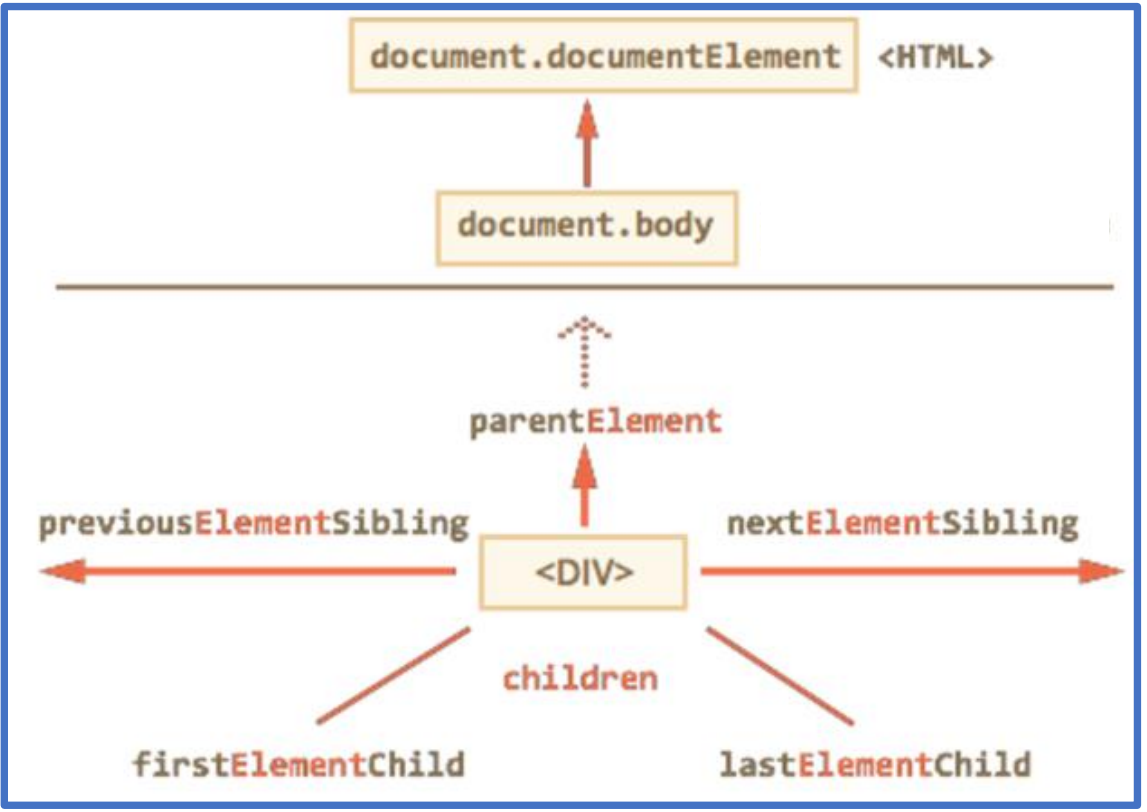
nav2. Дано елементи div. Знайти div, у якому span та:
1) наступний після нього зафарбувати у червоний колір (фон)
2) попередній після нього зафарбувати у зелений колір (фон)

```
<body>
    <div>
        <button>1</button>
    </div>

    <div>
        <span>Hello</span>
    </div>

    <div>
        <button>2</button>
    </div>

    <strong id="operation">Select</strong>
</body>
```

el.**previousElementSibling**.
style.backgroundColor = 'green'

const el =
document.querySelector('div:has(span)')

el.**nextElementSibling**.
style.backgroundColor = 'red'



Element

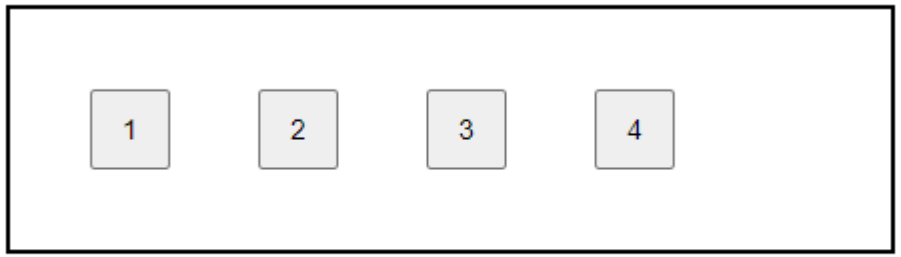
```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Comp
    <meta name="viewport" conte
    <title>Document</title>
  </head>
  <body>
    <h2>Some header</h2>
    <ol>
      <li>00</li>
      <li>11</li>
      <li>22</li>
      <li>33</li>
    </ol>
    <div>
      Some content
    </div>
```

children

дочірні елементи (тобто елементи, які знаходяться всередині даного

https://uk.javascript.info/dom-navigation

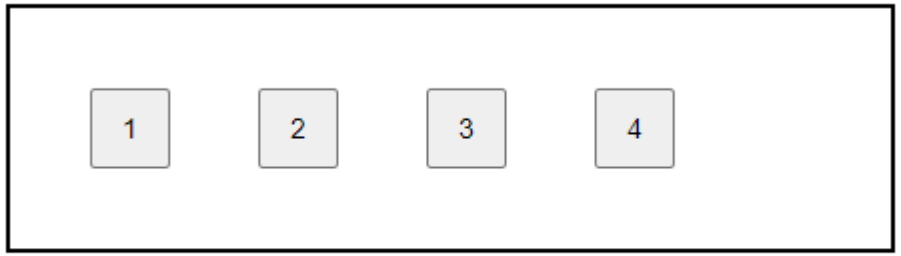
nav3. Усім дочірнім елементам div зробити зелений фон:




```
<body>
    <div>
        <button>1</button>
        <button>2</button>
        <button>3</button>
        <button>4</button>
    </div>

    <strong id="operation">Select</strong>
```

const el = document.querySelector('div')

nav3. Усім дочірнім елементам div зробити зелений фон:

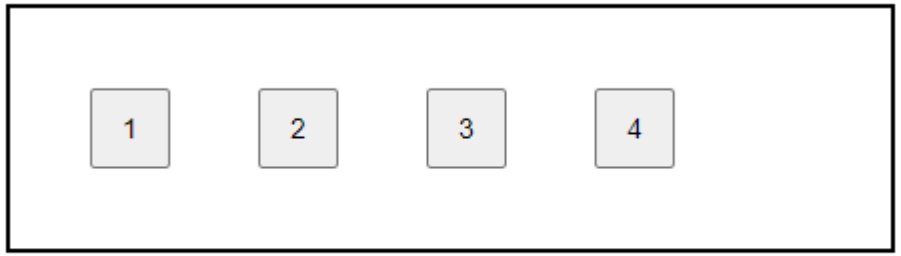



```
<body>
    <div>
        <button>1</button>
        <button>2</button>
        <button>3</button>
        <button>4</button>
    </div>

    <strong id="operation">Select</strong>
```

const el = document.querySelector('div')

el.**children**

nav3. Усім дочірнім елементам div зробити зелений фон:

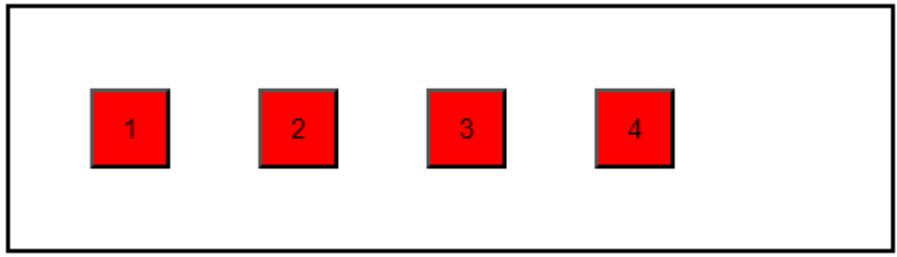



```
<body>
    <div>
        <button>1</button>
        <button>2</button>
        <button>3</button>
        <button>4</button>
    </div>

    <strong id="operation">Select</strong>
```

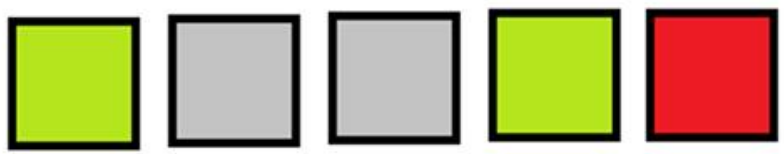
const el = document.querySelector('div')

el.children

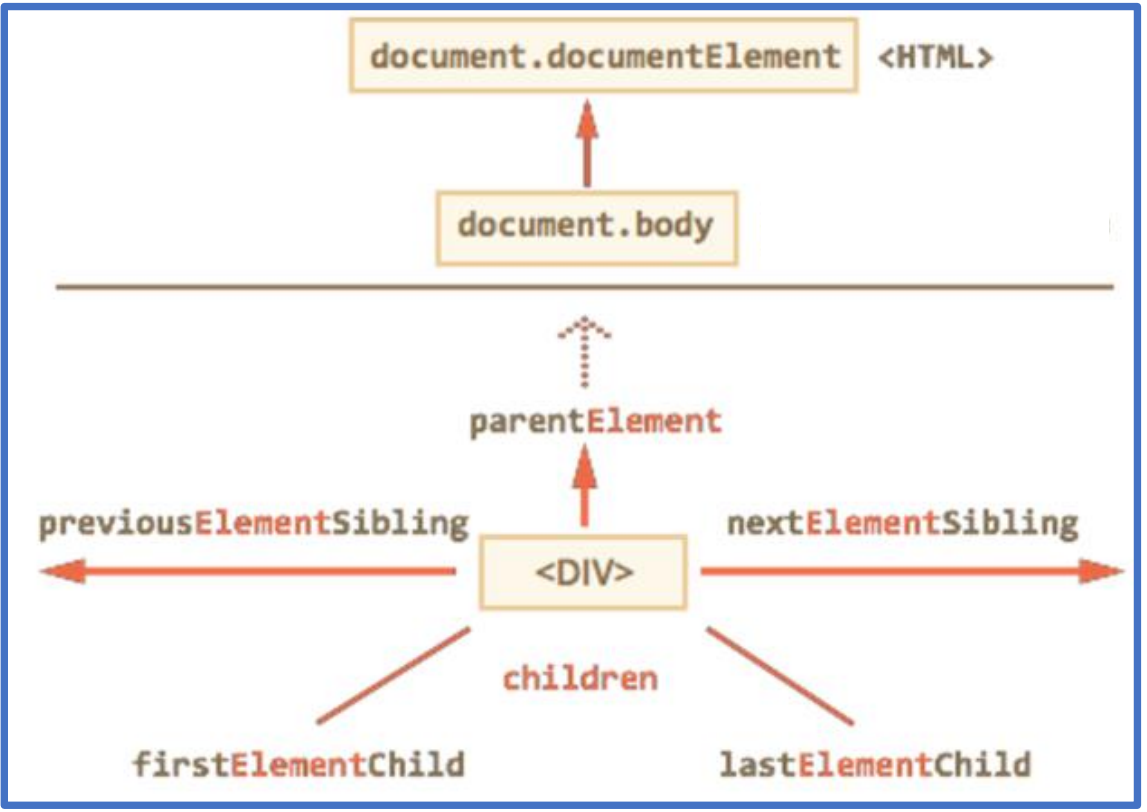
```
for (const btn of el.children) {
    btn.style.backgroundColor = 'red'
}
```



Задача. Однорядковий сапер. Однорядкова таблиця, до клітинок якої додано інформацію про наявність міни (використати атрибути). Спочатку клітинки сірі. При натисненні на клітинку аналізується чи є там міна і тоді колір стає червоним, якщо немає – зеленим. Додати можливість відкриття усіх сусідніх незамінованих клітинок при відкритті незамінованої клітинки.



```
<table>
  <tr>
    <td mine="0"></td>
    <td mine="1"></td>
    <td mine="1"></td>
    <td mine="0"></td>
    <td mine="1"></td>
  </tr>
</table>
```



перший дочіній елемент — `firstElementChild`

Element

останній дочіній елемент — `lastElementChild`

```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Comp
    <meta name="viewport" conte
    <title>Document</title>
  </head>
  <body>
    <h2>Some header</h2>
    <ol>
      <li>00</li>
      <li>11</li>
      <li>22</li>
      <li>33</li>
    </ol>
    <div>
      Some content
    </div>
```

children

https://uk.javascript.info/dom-navigation

nav4. Для першого елемента списоку у кінець додати «-first», для останнього – «last»

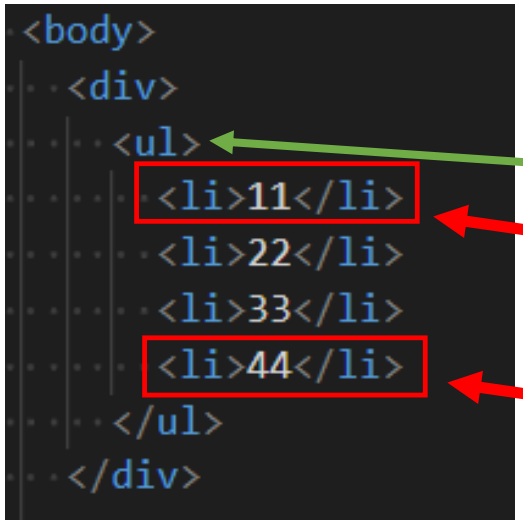
```
<body>
  <div>
    <ul>
      <li>11</li>
      <li>22</li>
      <li>33</li>
      <li>44</li>
    </ul>
  </div>
```

const el = document.querySelector('ul')

- 11
- 22
- 33
- 44

Select

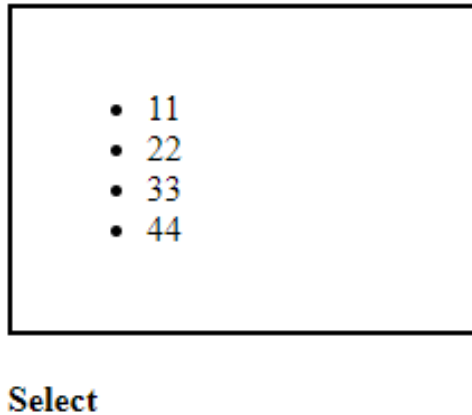
nav4. Для першого елемента списоку у кінець додати «-first», для останнього – «last»



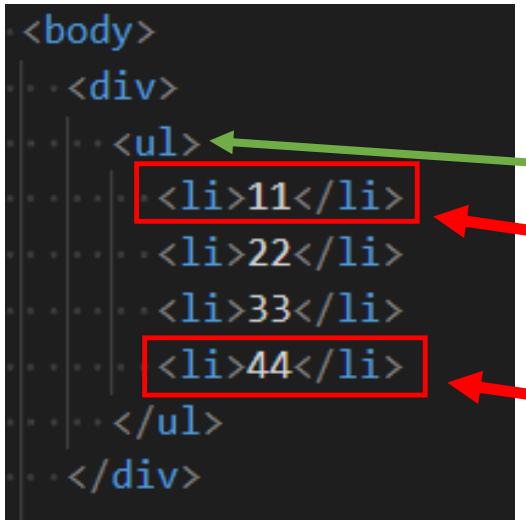
const el = document.querySelector('ul')

el.firstElementChild

lastElementChild



nav4. Для першого елемента списоку у кінець додати «-first», для останнього – «last»



const el = document.querySelector('ul')

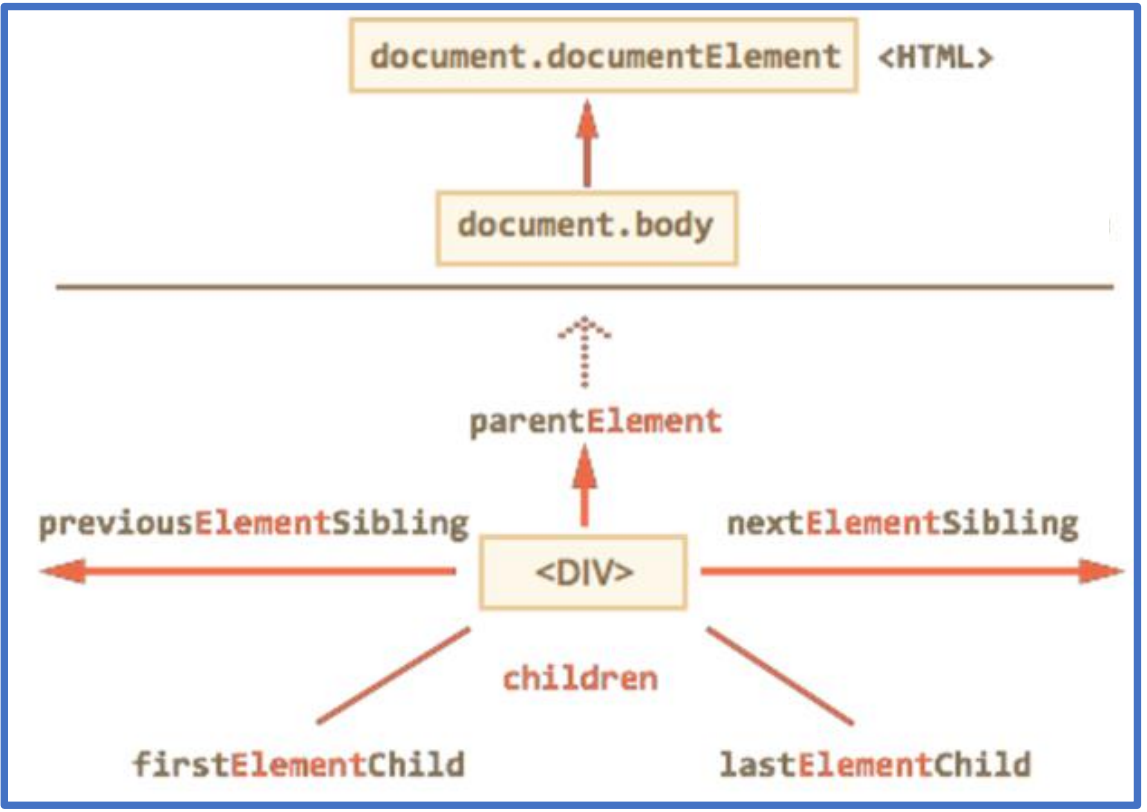
el.firstElementChild

lastElementChild

- 11
- 22
- 33
- 44

```
const el = document.querySelector('ul')
console.log(el.firstElementChild)
el.firstElementChild.innerText += '-first'
el.lastElementChild.innerText += '-last'
```

- 11-first
- 22
- 33
- 44-last




```
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Comp
    <meta name="viewport" conte
    <title>Document</title>
  </head>
  <body>
    <h2>Some header</h2>
    <ol>
      <li>00</li>
      <li>11</li>
      <li>22</li>
      <li>33</li>
    </ol>
    <div>
      Some content
    </div>
```

parentElement

previousElementSibling

firstElementChild

Element

lastElementChild

children

nextElementSibling

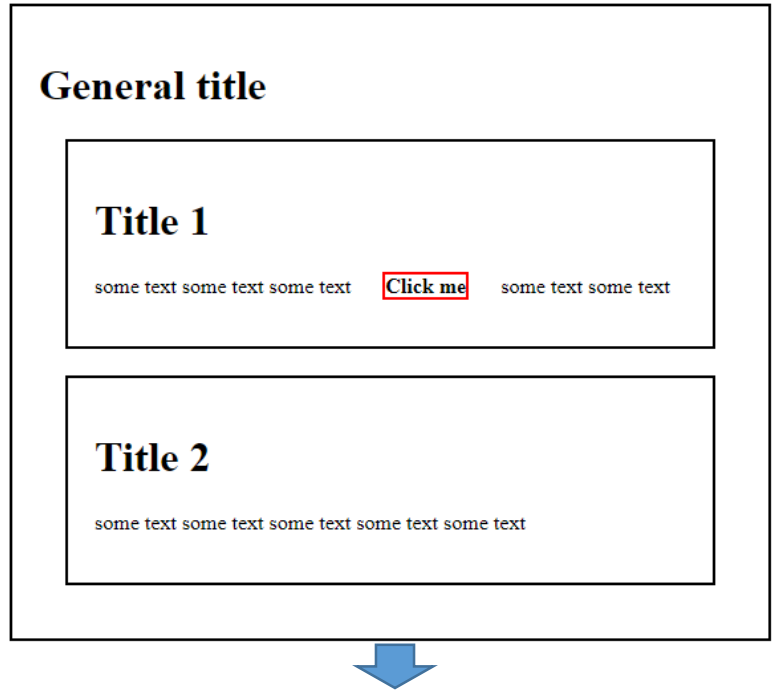
https://uk.javascript.info/dom-navigation

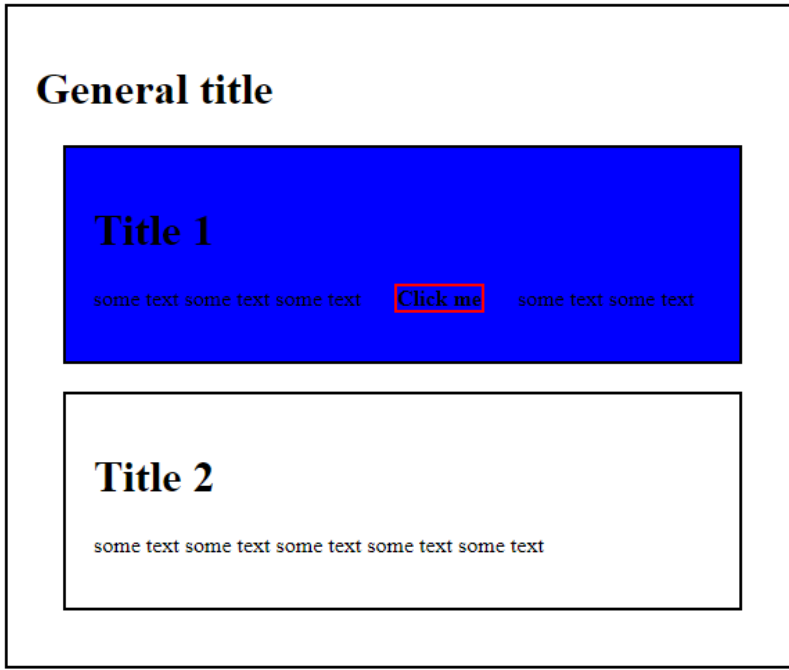
Пошук найближчого предка за css селектором. `closest`

`elem.closest(css_селектор)`	Метод **elem.closest (css_селектор)** шукає найближчий елемент вище по ієрархії DOM, що підходить під CSS-селектор css. Сам елемент теж включається в пошук.Інакше кажучи, метод closest біжить від поточного елемента вгору по ланцюжку батьків і перевіряє, чи підходить елемент під вказаний CSS-селектор. Якщо підходить - зупиняється і повертає його.

При кліку на strong зробити для першого вверх батьківського div з класом "section" синій колір

```html
<div class="section">
  <h1>General title</h1>
  <div class="section">
    <div>
      <h1>Title 1</h1>
      <p>
        some text some text some text <strong>Click me</strong> some text
        some text
      </p>
    </div>
  </div>
  <div class="section">
    <div>
      <h1>Title 2</h1>
      <p>some text some text some text some text some text</p>
    </div>
  </div>
</div>
```

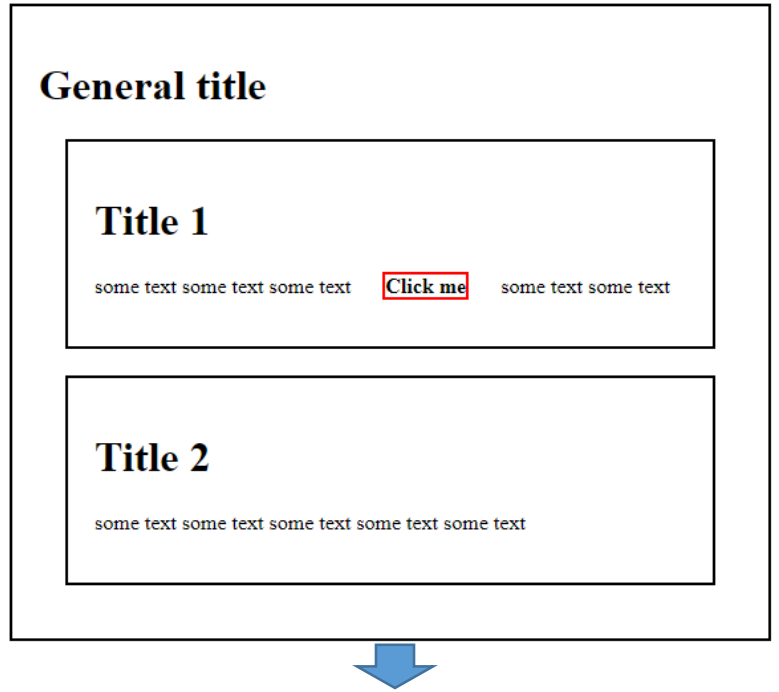


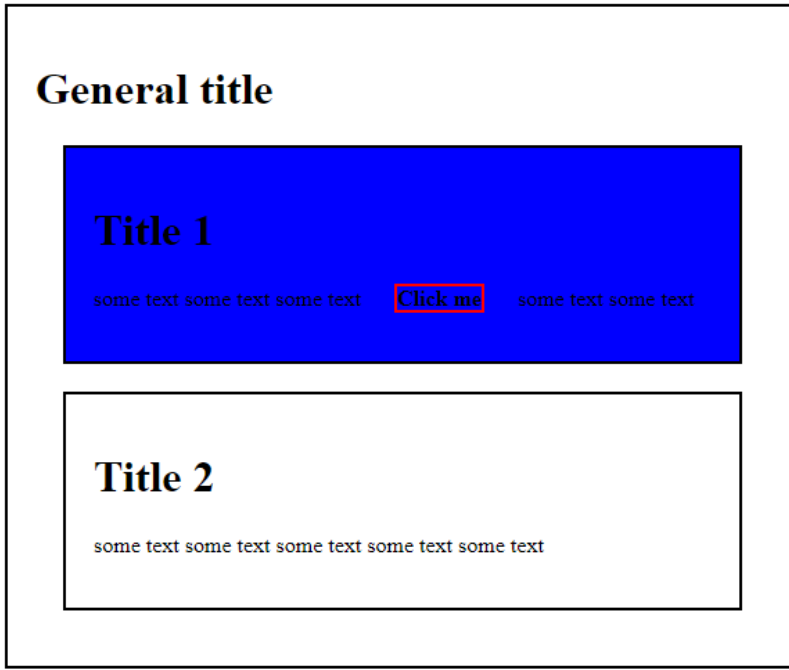


При кліку на strong зробити для першого вверх батьківського div з класом "section" синій колір

```html
<div class="section">
  <h1>General title</h1>
  <div class="section">
    <div>
      <h1>Title 1</h1>
      <p>
        some text some text some text <strong>Click me</strong> some text
        some text
      </p>
    </div>
  </div>
  <div class="section">
    <div>
      <h1>Title 2</h1>
      <p>some text some text some text some text some text</p>
    </div>
  </div>
</div>
```

```
const el = document.querySelector('strong')
```





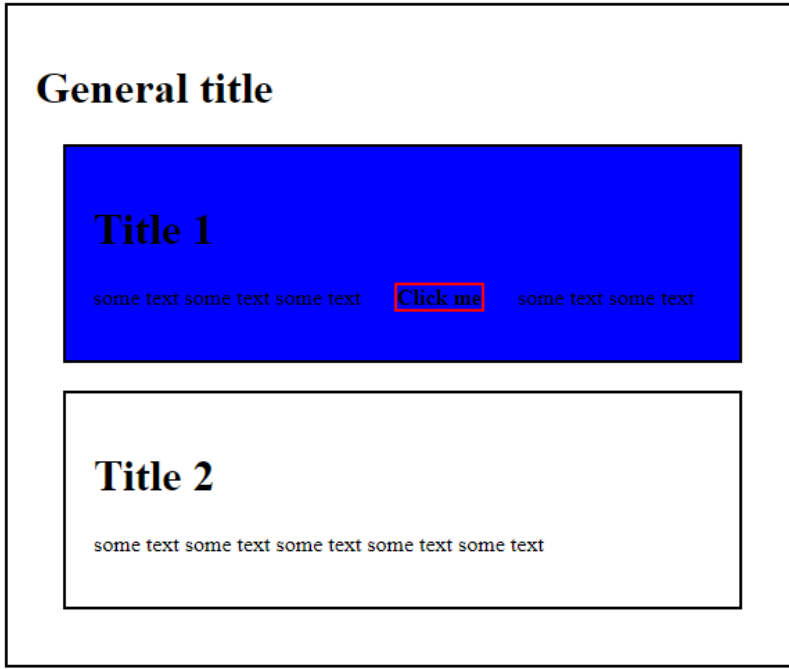
При кліку на strong зробити для першого вверх батьківсього div з класом "section" синій колір

el.closest('.section')

```html
<div class="section">
  <h1>General title</h1>
  <div class="section">
    <div>
      <h1>Title 1</h1>
      <p>
        some text some text some text <strong>Click me</strong> some text
        some text
      </p>
    </div>
  </div>
  <div class="section">
    <div>
      <h1>Title 2</h1>
      <p>some text some text some text some text some text</p>
    </div>
  </div>
</div>
```

перехід до батьківського ✖

const el = document.querySelector('strong')



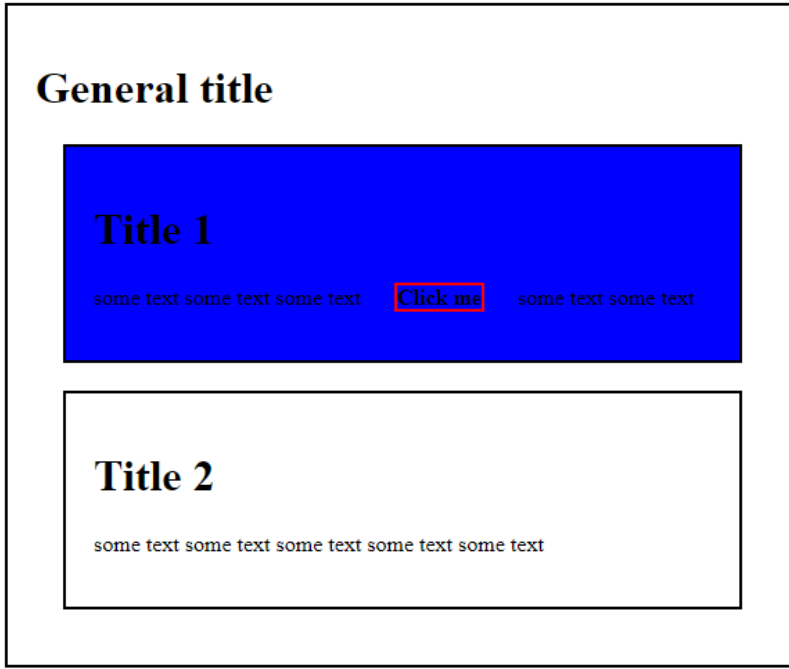
При кліку на strong зробити для першого вверх батьківсього div з класом "section" синій колір

el.closest('.section')

```html
<div class="section">
  <h1>General title</h1>
  <div class="section">
    <div>
      <h1>Title 1</h1>
      <p>
        some text some text some text <strong>Click me</strong> some text
        some text
      </p>
    </div>
  </div>
  <div class="section">
    <div>
      <h1>Title 2</h1>
      <p>some text some text some text some text some text</p>
    </div>
  </div>
</div>
```

перехід до батьківського ✖

const el = document.querySelector('strong')



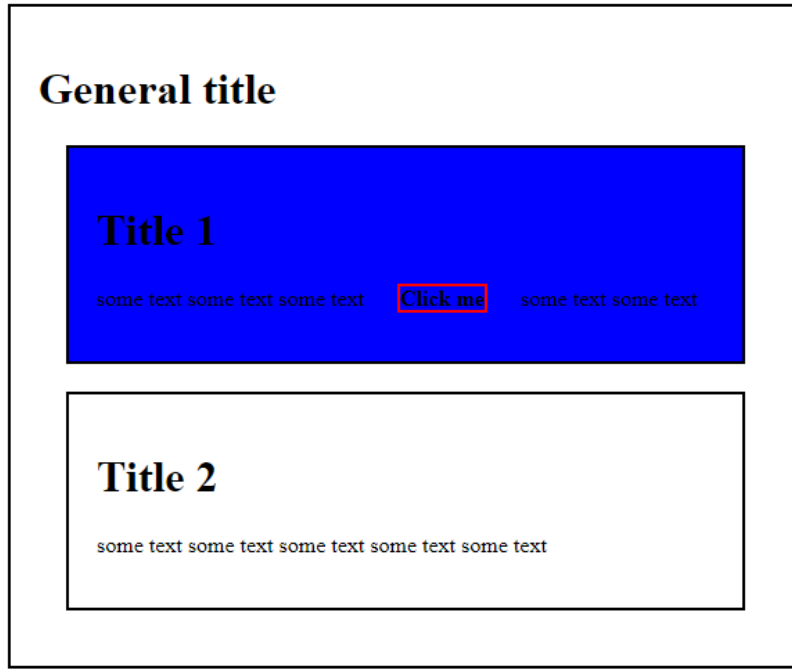
При кліку на strong зробити для першого вверх батьківсього div з класом "section" синій колір

el.closest('.section')

```
<div class="section">
  <h1>General title</h1>
  <div class="section">
    <div>
      <h1>Title 1</h1>
      <p>
        some text some text some text <strong>Click me</strong> some text
        some text
      </p>
    </div>
  </div>
  <div class="section">
    <div>
      <h1>Title 2</h1>
      <p>some text some text some text some text some text</p>
    </div>
  </div>
</div>
```

перехід до батьківського ✚

const el = document.querySelector('strong')



При кліку на strong зробити для першого вверх батьківсього div з класом "section" синій колір

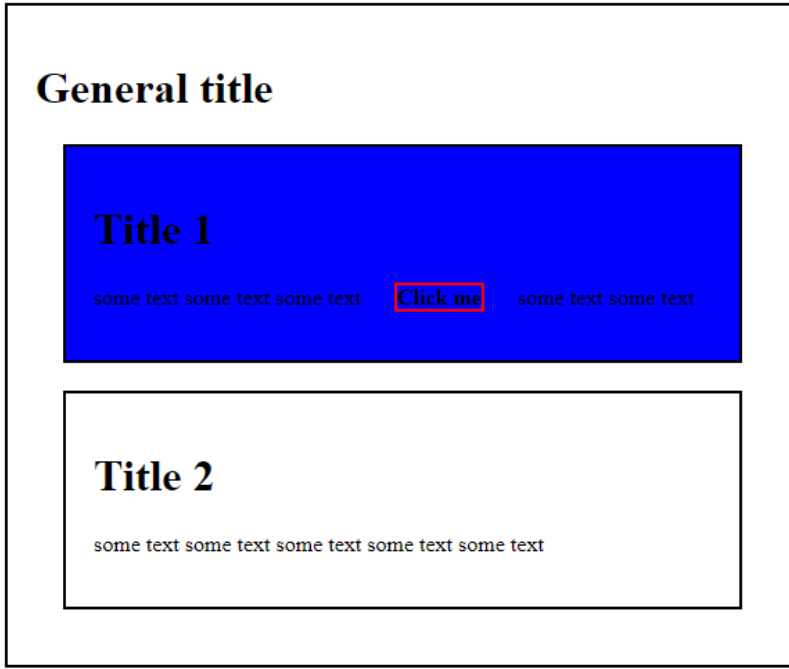
el.closest('.section')

```html
<div class="section">
  <h1>General title</h1>
  <div class="section">
    <div>
      <h1>Title 1</h1>
      <p>
        some text some text some text <strong>Click me</strong> some text
        some text
      </p>
    </div>
  </div>
  <div class="section">
    <div>
      <h1>Title 2</h1>
      <p>some text some text some text some text some text</p>
    </div>
  </div>
</div>
```

перехід до батьківського

const el = document.querySelector('strong')

```
const el = document.querySelector('strong')
el.onclick = function () {
  el.closest('.section').style.backgroundColor = 'blue'
}
```



Приклад. Реалізувати додвання та вилучення задач. (використати ***closest*** для пошуку контейнера задачі, що потрбіно видалити)