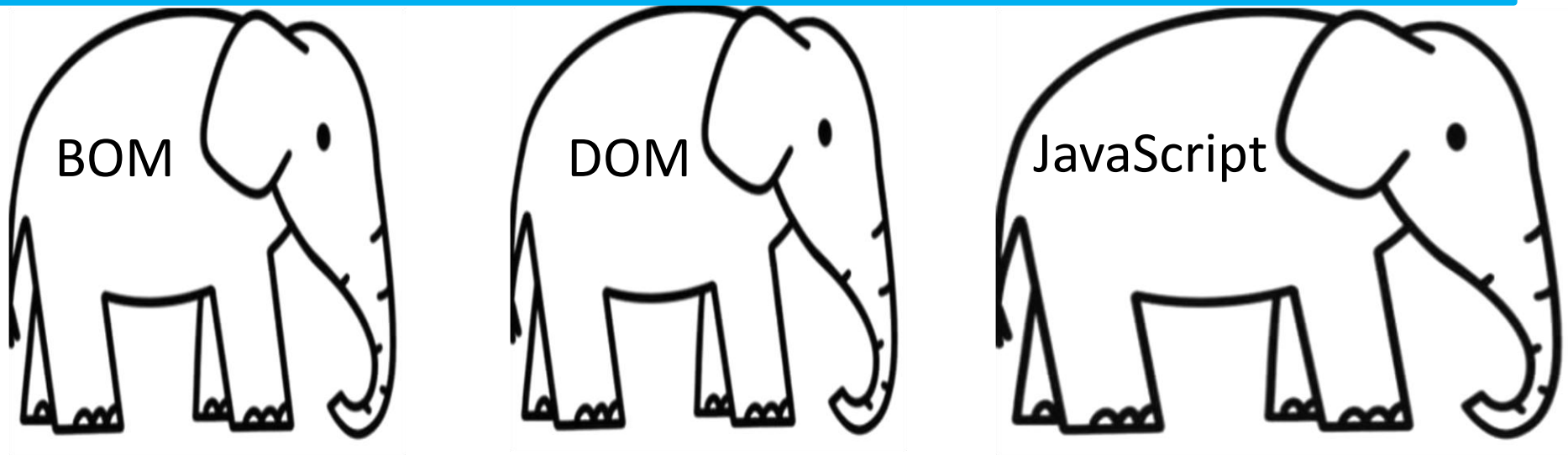
Window

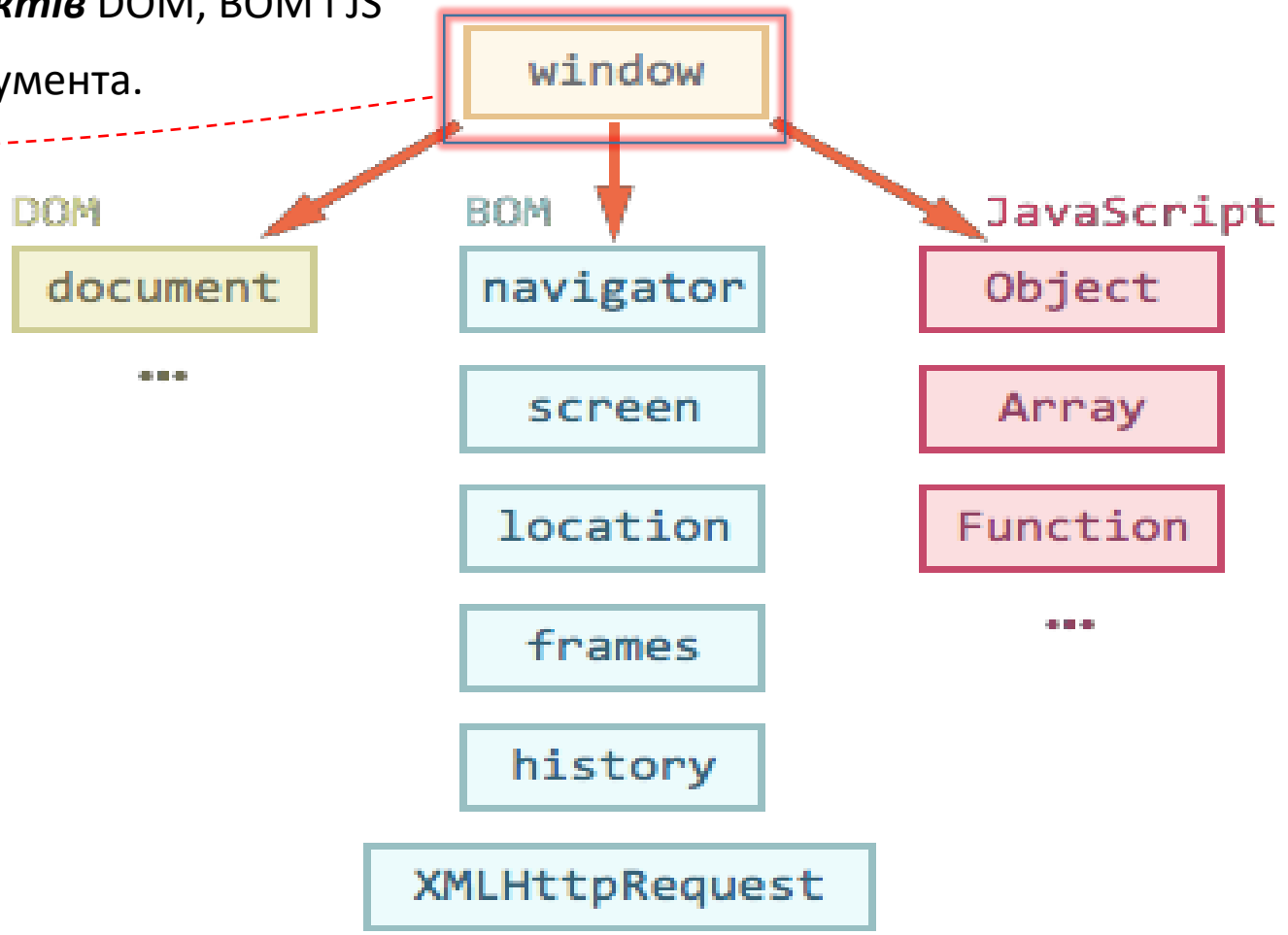


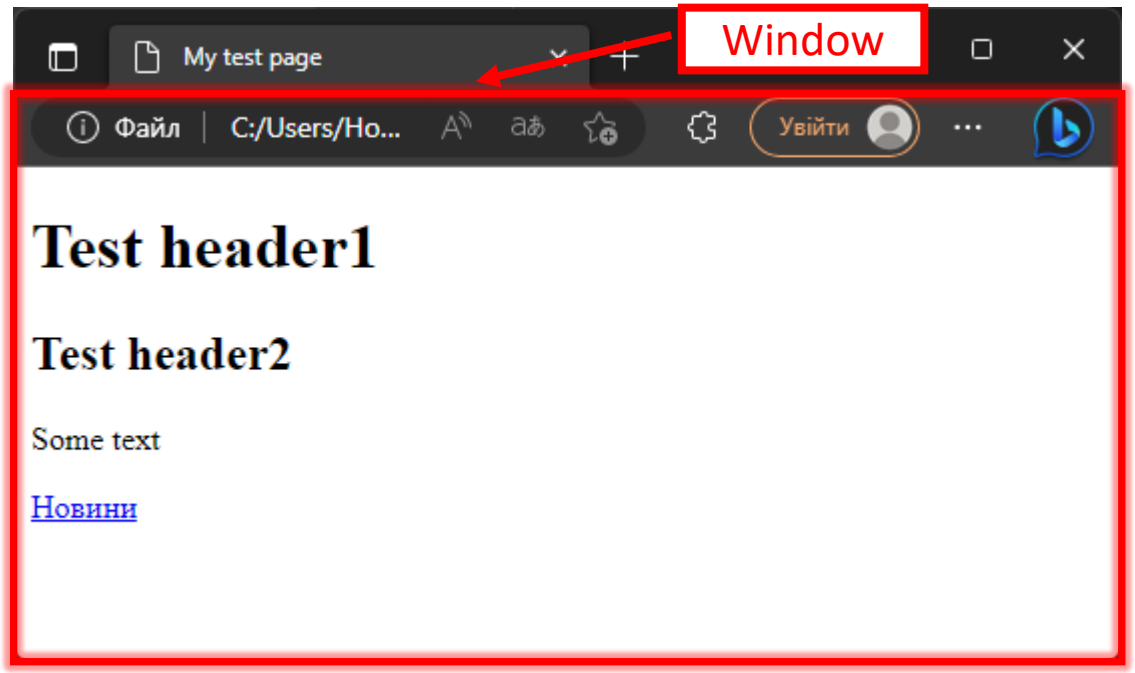
Структура браузерних об'єктів DOM, BOM і JS

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

Взаємозв'язок:

- Об'єкт **window** включає BOM та DOM як свої складові.
- DOM фокусується на вмісті веб-сторінки, тоді як BOM відповідає за взаємодію з самим браузером та його середовищем.

Структура браузерних об'єктів DOM, BOM і JS

Основною задачею браузера є відображення HTML документа.

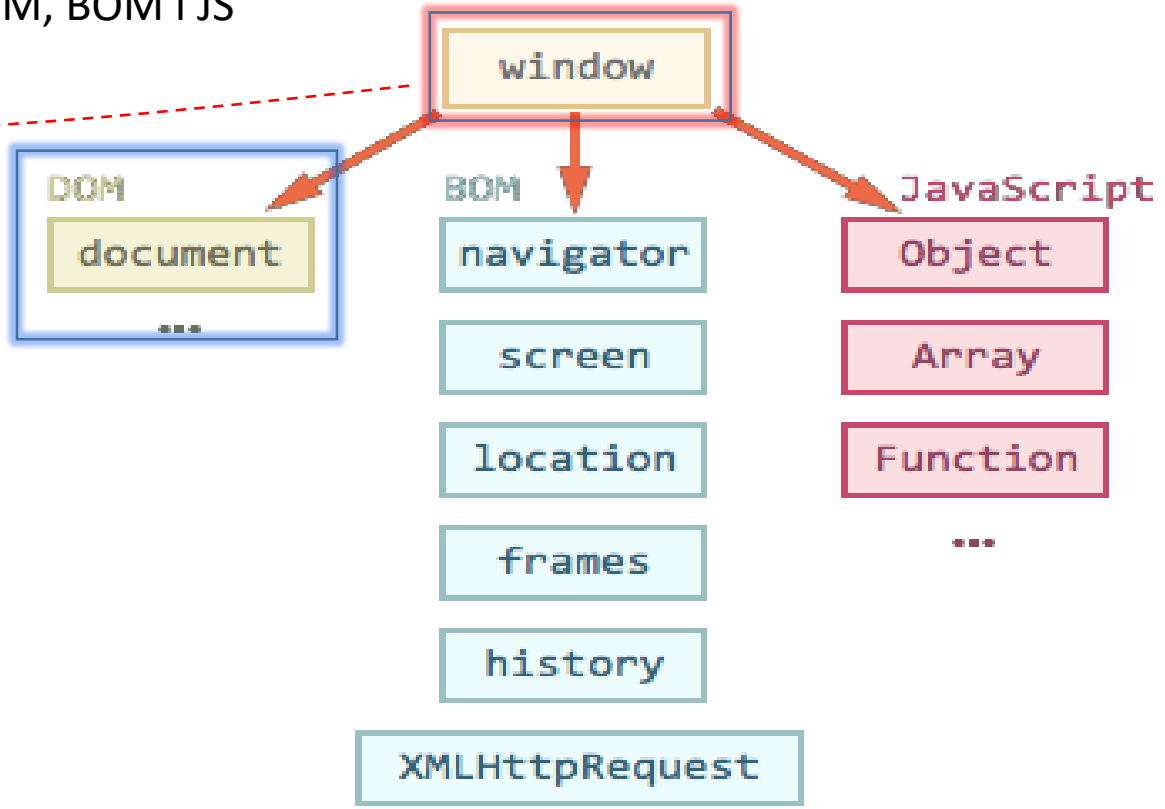
Об'єкт ***window***

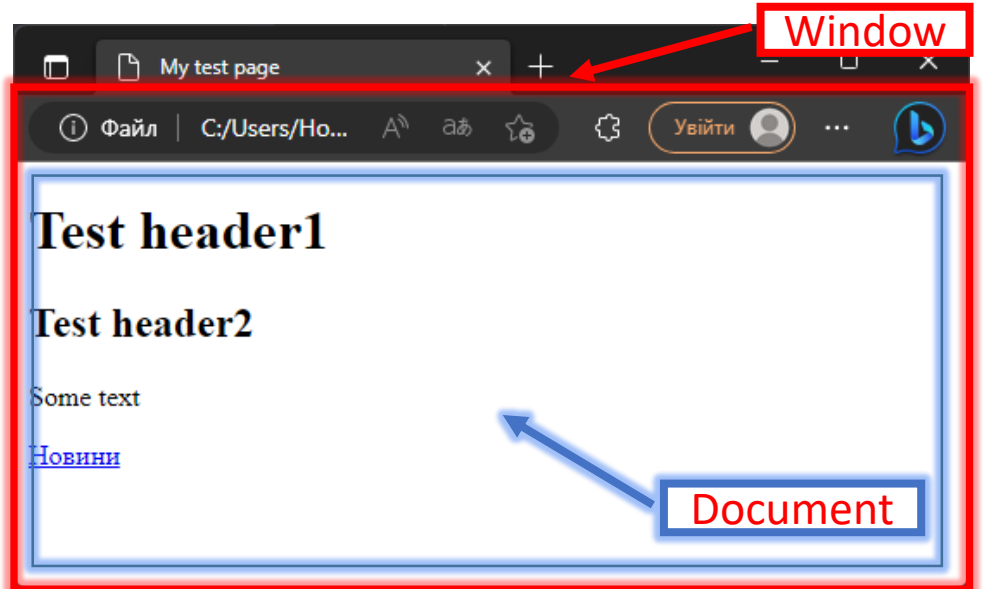
- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів

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.

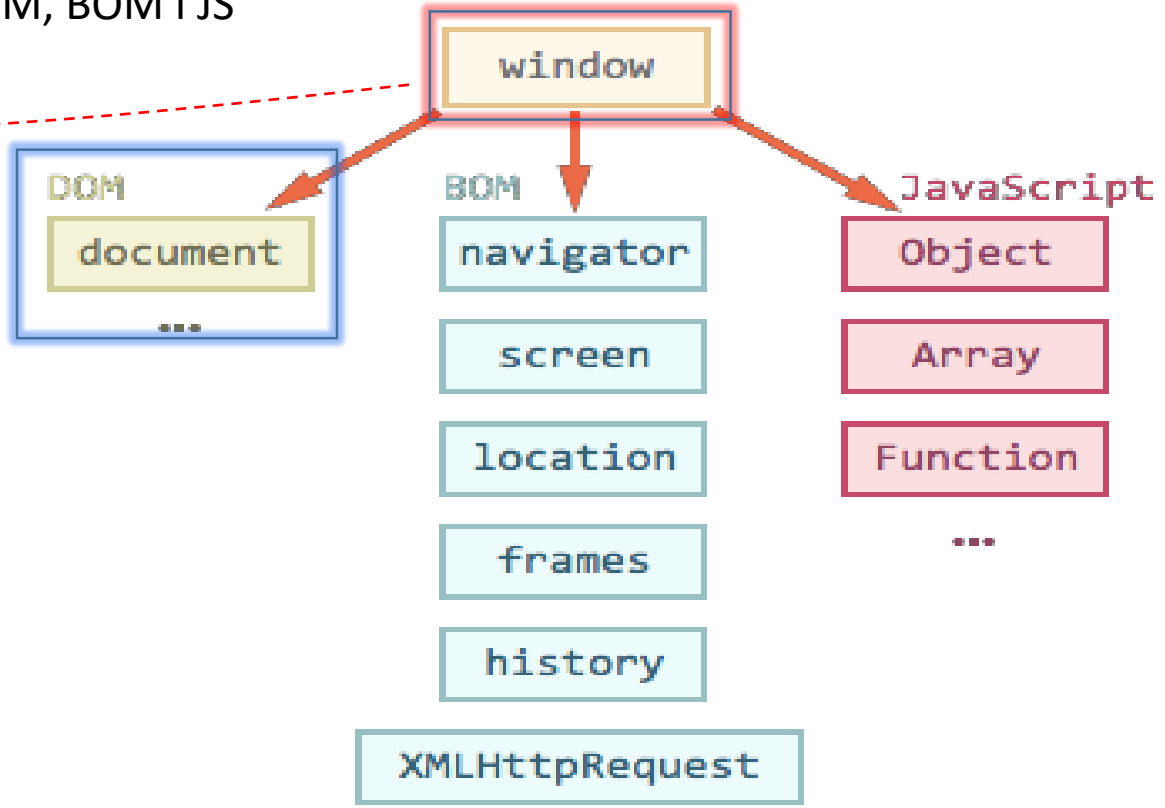
Об'єкт ***window***

- 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

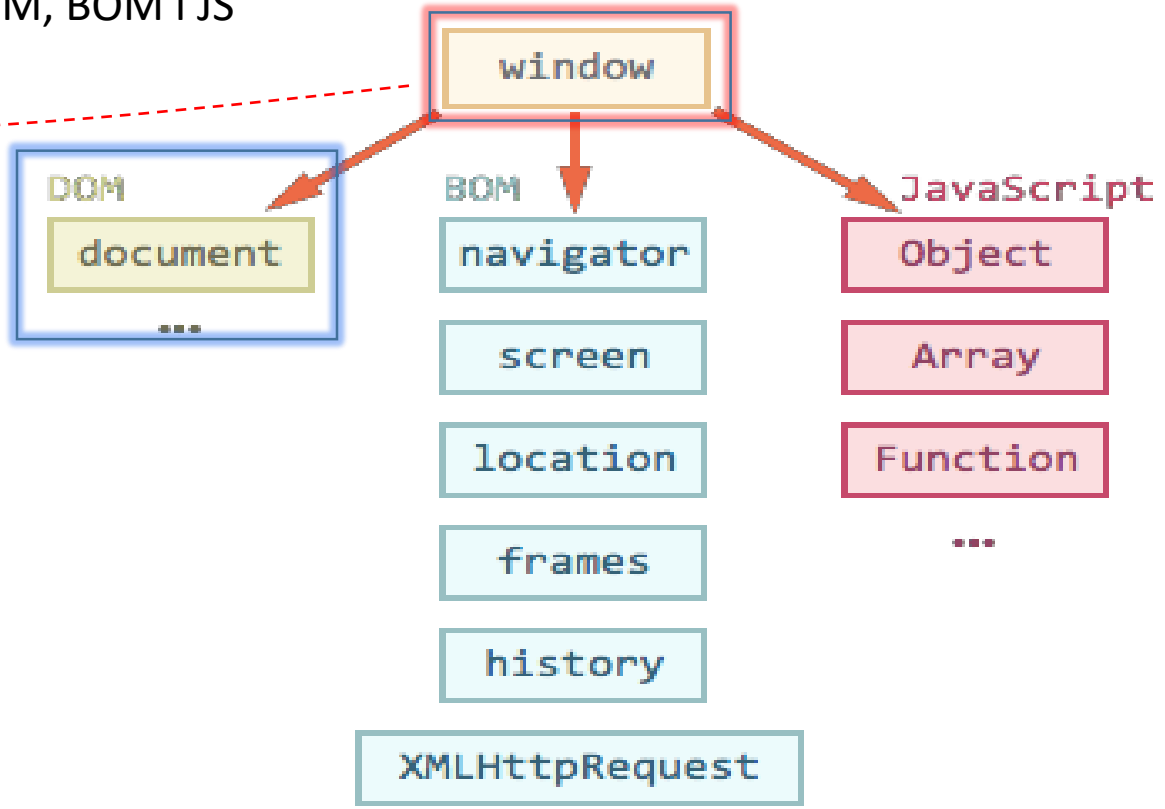
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>My test page</title>
  </head>
  <body>
    <h1>Test header1</h1>
    <div class="container">
      <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.

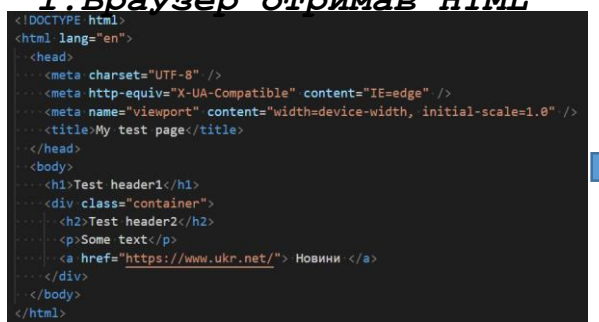
Об'єкт ***window***

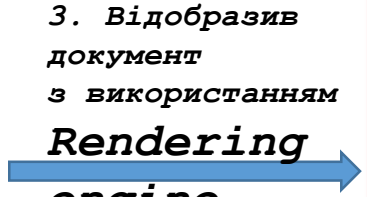
- представляє вікно браузера (фрейма) і містить властивості і методи для управління ним
- є глобальним об'єктом JavaScript, через який програмними засобами можна звернутися до всіх інших об'єктів

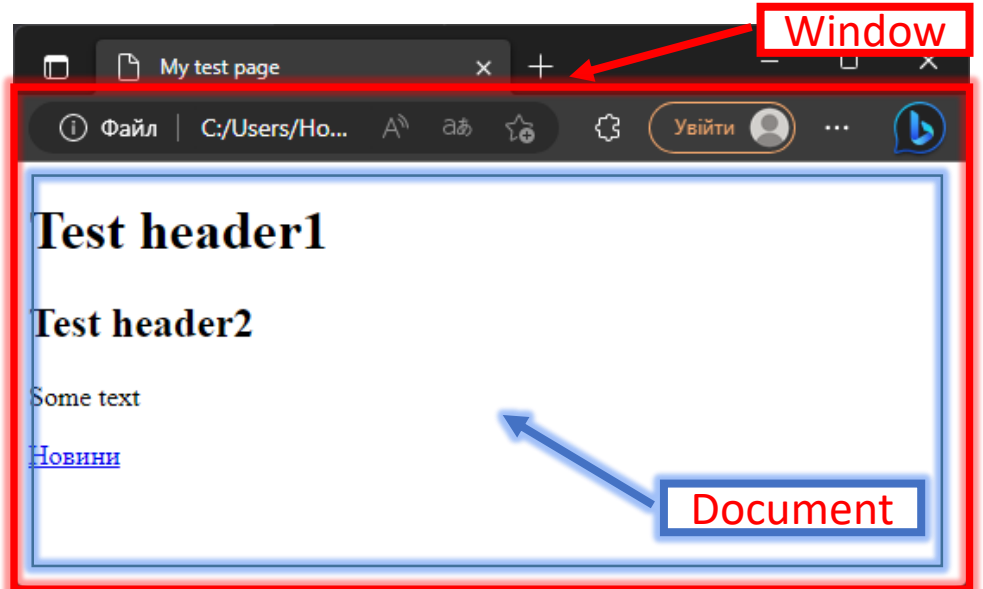
Об'єкт ***document***

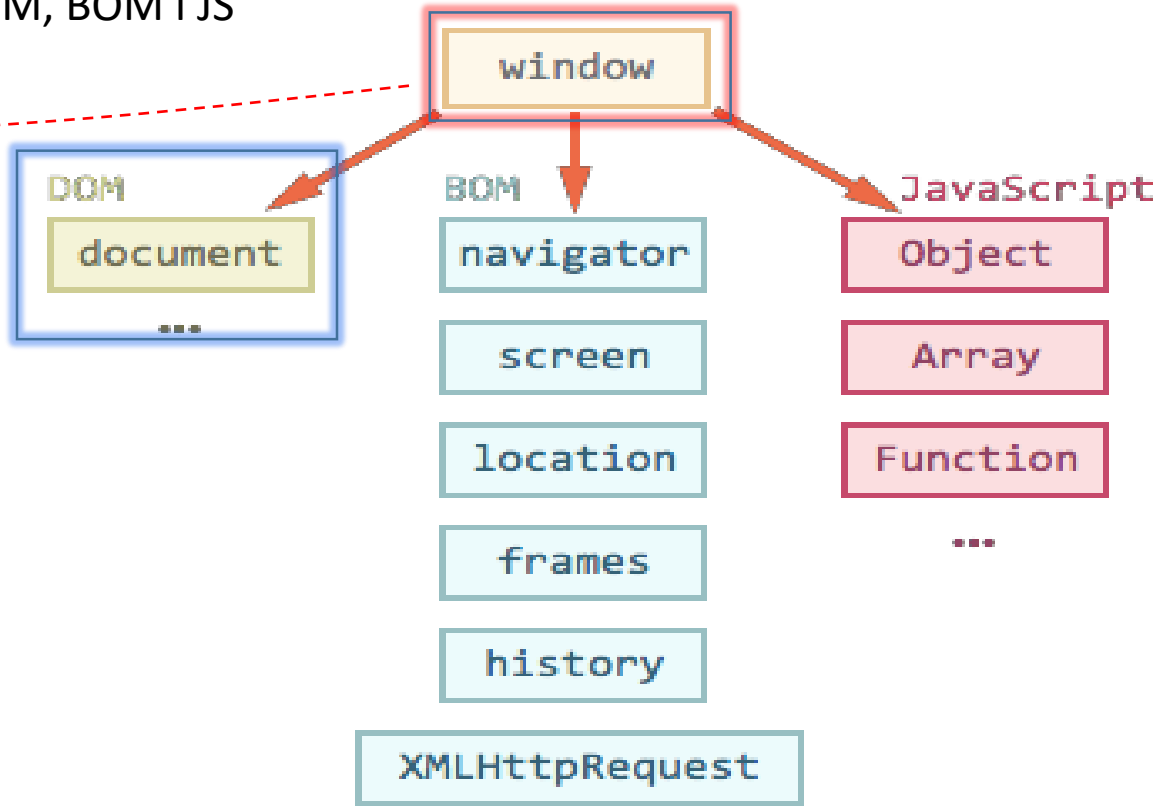
- представляє HTML документ, що відображається у вікні браузера (у `window`)
- є об'єктом JavaScript, через який програмними засобами можна звернутися до вмісту веб-сторінки і маніпулювати ним

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.

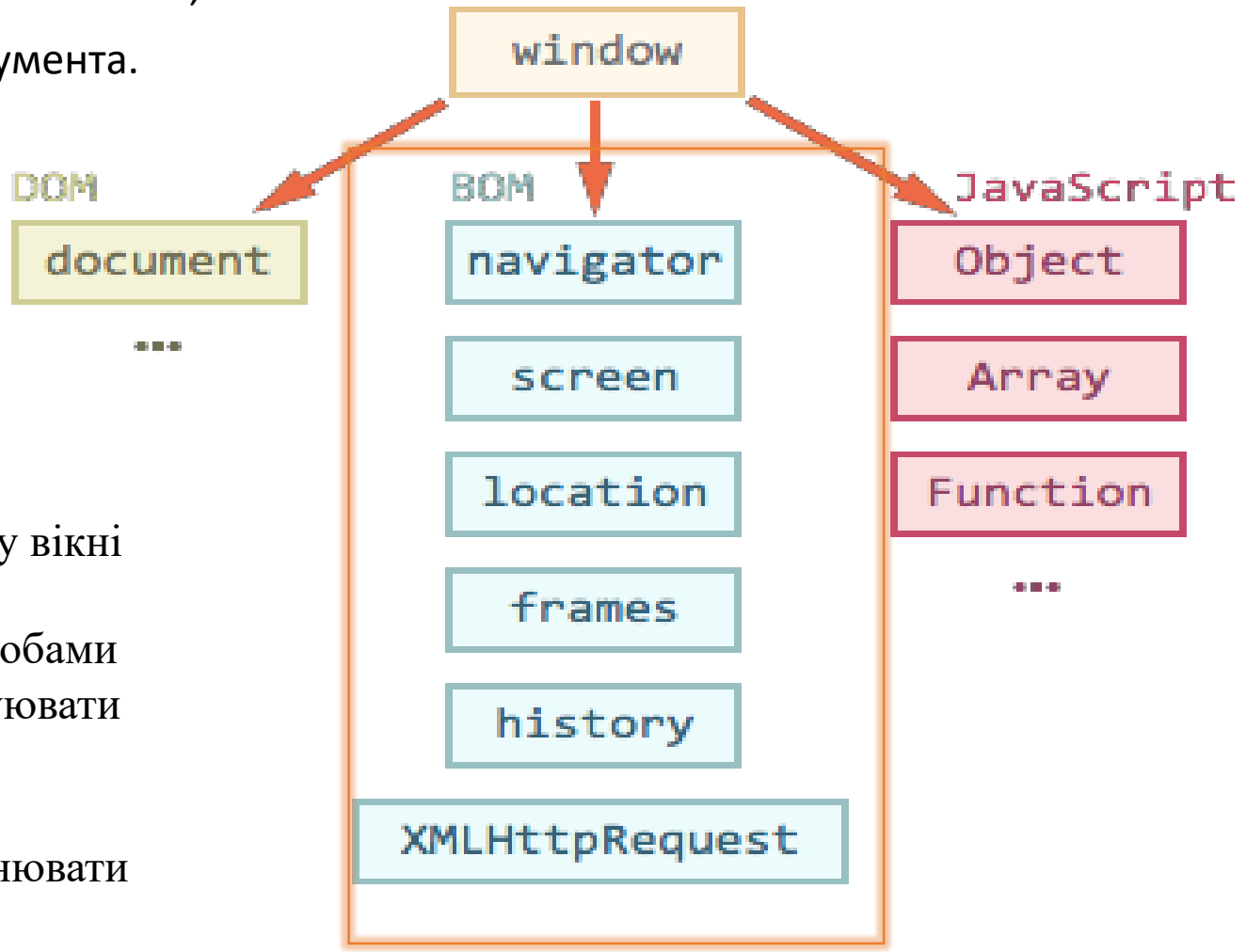
Об'єкт ***window***

- 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

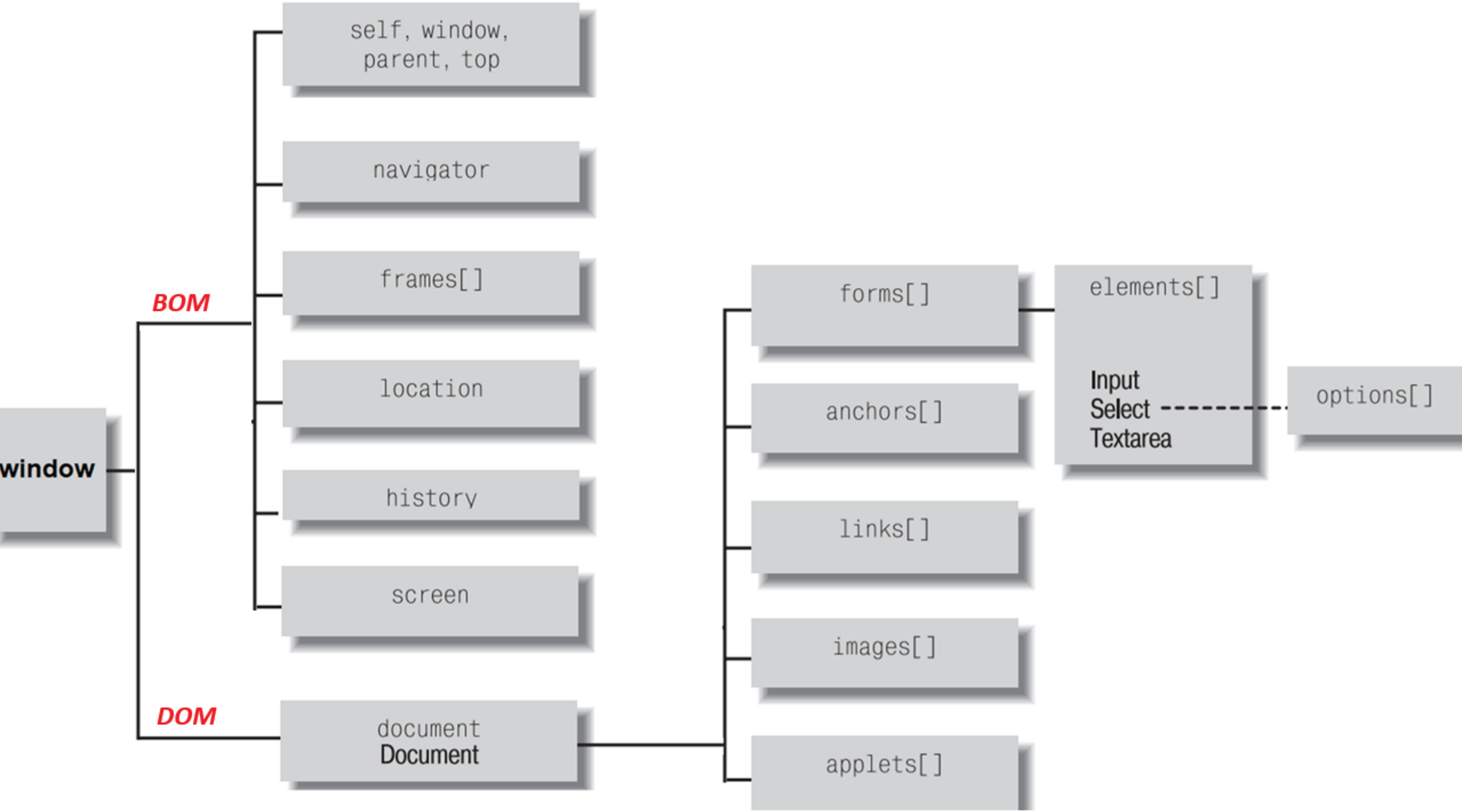
DOM – це програмний інтерфейс (API) , що дозволяє взаємодіяти з структурою HTML документа та здійснювати маніпуляції з його елементами



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


self, window,
parent, top
navigator
frames[]
location
history
screen
BOM
window
DOM
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

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

Розмір вікна

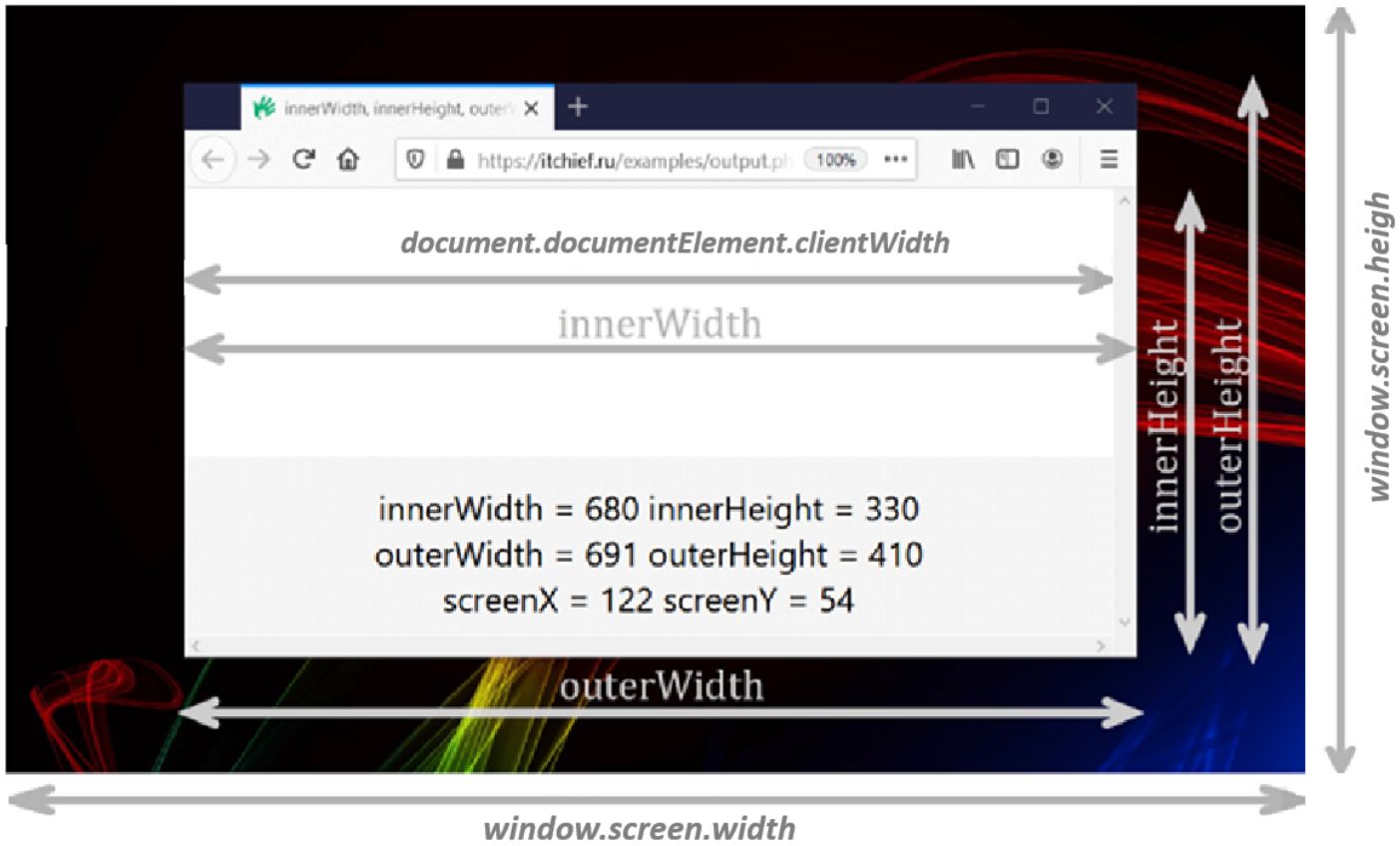
Об’єкт ***window***

- 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.

documentElement.clientWidth – ширина області контенту (без полоси прокрутки якщо така є)

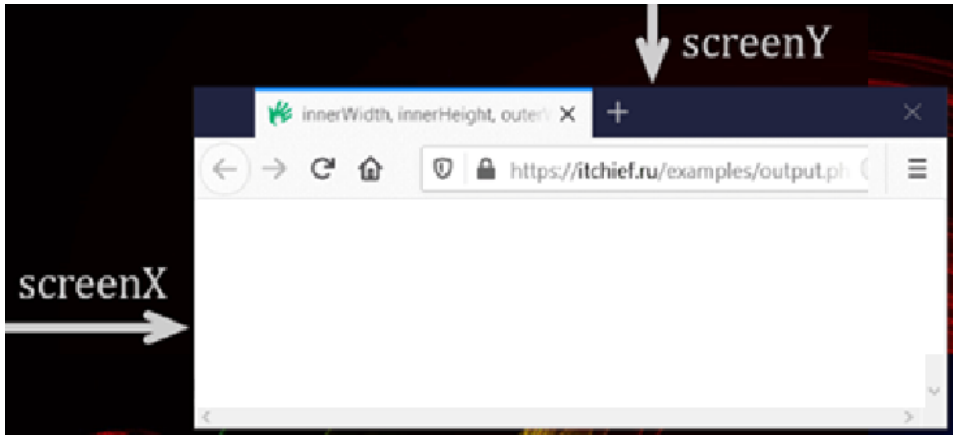
documentElement.clientHeight – висота області контенту



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

Положення вікна на екрані

Назва властивості/ методу	Пояснення
`window.screenX`	Визначає горизонтальну координату верхнього лівого кута вікна відносно екрану.
`window.screenY`	Визначає вертикальну координату верхнього лівого кута вікна відносно екрану.
`window.moveTo(x, y)`	Переміщує вікно до вказаних координат `x` (горизонталь) та `y` (вертикаль).
`window.moveBy(x, y)`	Переміщує вікно на певну відстань відносно поточного положення.



Властивості `screenLeft` та `screenTop` також стосуються положення вікна на екрані, але вони використовуються здебільшого в старих браузерах або для сумісності з попередніми версіями:

- `window.screenLeft`: Визначає горизонтальну координату верхнього лівого кута вікна відносно екрана.
- `window.screenTop`: Визначає вертикальну координату верхнього лівого кута вікна відносно екрана.

Сучасні браузери зазвичай надають перевагу властивостям `screenX` та `screenY`, оскільки вони стандартні для більшості платформ. Однак `screenLeft` і `screenTop` все ще можуть бути корисними для роботи з браузерами, які підтримують ці властивості.

Прокрутка вікна

Властивості, що стосуються прокрутки вікна

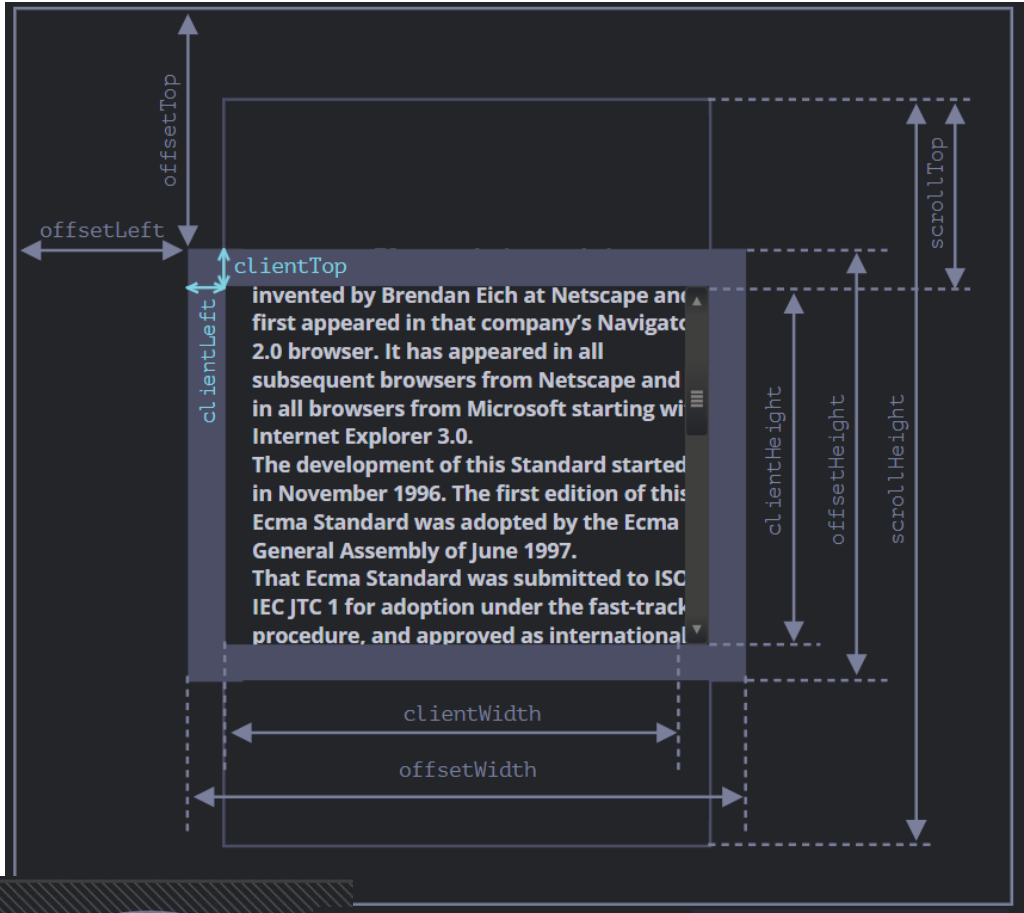
Назва властивості	Пояснення
`window.scrollX`	Повертає кількість пікселів, на яку сторінка прокручена горизонтально (вліво). Синонім `window.pageXOffset`.
`window.scrollY`	Повертає кількість пікселів, на яку сторінка прокручена вертикально (вниз). Синонім `window.pageYOffset`.
`window.pageXOffset`	Синонім `window.scrollX`. Повертає горизонтальну прокрутку сторінки в пікселях. Використовується для сумісності зі старими браузерами.
`window.pageYOffset`	Синонім `window.scrollY`. Повертає вертикальну прокрутку сторінки в пікселях. Використовується для сумісності зі старими браузерами.

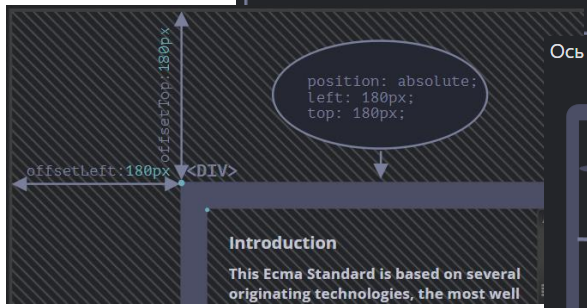
Методи, що стосуються прокрутки вікна

Назва методу	Пояснення
`window.scrollTo(x, y)`	Прокручує сторінку до заданих координат (x — горизонтальна, y — вертикальна). Наприклад, `window.scrollTo(0, 100)` прокрутить на 100 пікселів вниз.
`window.scrollBy(x, y)`	Прокручує сторінку на задану кількість пікселів відносно поточної позиції. Наприклад, `window.scrollBy(0, 50)` прокрутить на 50 пікселів вниз від поточного місця.
`window.scroll()`	Аналогічний `scrollTo()`, але підтримує додаткові опції. Наприклад, `window.scroll({ top: 100, behavior: 'smooth' })` забезпечує плавну прокрутку.

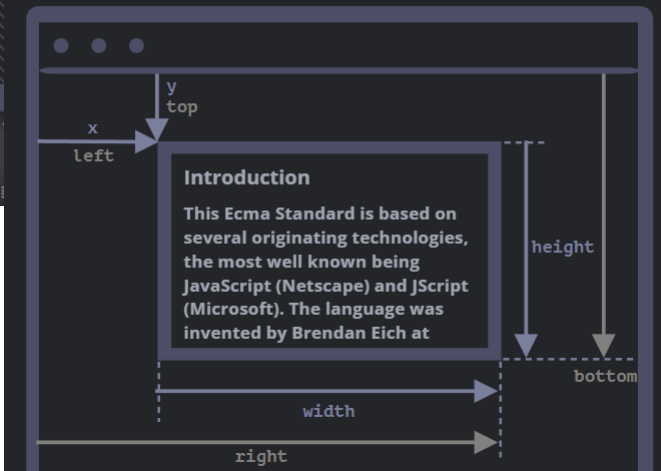
Розміри і положення елемента

Назва властивості	Пояснення
`Element.offsetLeft`	Повертає відстань у пікселях від лівого краю елемента до лівого краю його найближчого позиціонованого батьківського елемента (з `position: relative`, `absolute`, або `fixed`). Якщо такого немає, то відносно `body`. Це позиція на сторінці.
`Element.offsetTop`	Повертає відстань у пікселях від верхнього краю елемента до верхнього краю його найближчого позиціонованого батьківського елемента (з `position: relative`, `absolute`, або `fixed`). Якщо такого немає, то відносно `body`. Це позиція на сторінці.
`Element.offsetWidth`	Повертає повну ширину елемента в пікселях, включаючи `padding`, але без урахування `margin`. Це зовнішній розмір елемента.
`Element.offsetHeight`	Повертає повну висоту елемента в пікселях, включаючи `padding`, але без урахування `margin`. Це зовнішній розмір елемента.
`Element.clientLeft`	Повертає ширину лівої рамки (`border-left-width`) елемента в пікселях. Це відстань між зовнішнім лівим краєм і внутрішнім лівим краєм (початком `padding`).
`Element.clientTop`	Повертає ширину верхньої рамки (`border-top-width`) елемента в пікселях. Це відстань між зовнішнім верхнім краєм і внутрішнім верхнім краєм (початком `padding`).
`Element.clientWidth`	Повертає внутрішню ширину елемента в пікселях, включаючи `padding`, але без урахування `border` і `margin`. Якщо є смуга прокрутки, її ширина віднімається.
`Element.clientHeight`	Повертає внутрішню висоту елемента в пікселях, включаючи `padding`, але без урахування `border` і `margin`. Якщо є смуга прокрутки, її висота віднімається.
`Element.scrollHeight`	Повертає повну висоту вмісту елемента, включаючи невидиму частину через прокрутку. Це розмір усього вмісту, незалежно від видимої області.





Ось зображення з результатом виклику `elem.getBoundingClientRect()`:



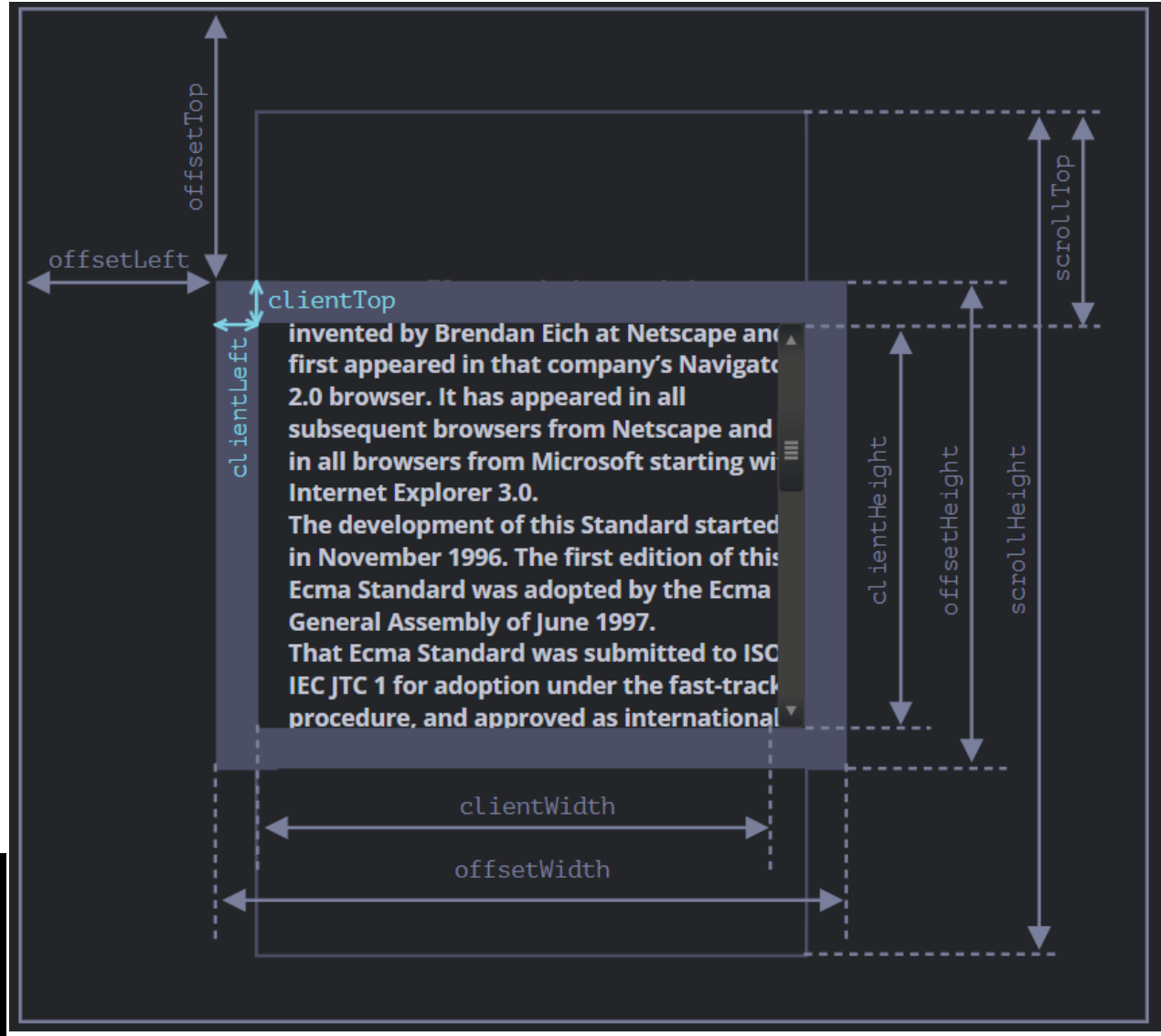
Прокрутка елемента

Властивості, що стосуються прокрутки елемента

Назва властивості	Пояснення
`Element.scrollTop`	Повертає або встановлює кількість пікселів, на яку елемент прокручений вертикально. Наприклад, для `document.documentElement` може використовуватися для всієї сторінки.
`Element.scrollLeft`	Повертає або встановлює кількість пікселів, на яку елемент прокручений горизонтально. Наприклад, для `document.documentElement` може використовуватися для всієї сторінки.
`Element.scrollHeight`	Повертає повну висоту вмісту елемента, включаючи невидиму частину через прокрутку. Для `document.documentElement` показує повну висоту сторінки.
`Element.scrollWidth`	Повертає повну ширину вмісту елемента, включаючи невидиму частину через прокрутку. Для `document.documentElement` показує повну ширину сторінки.

Методи, що стосуються прокрутки елемента

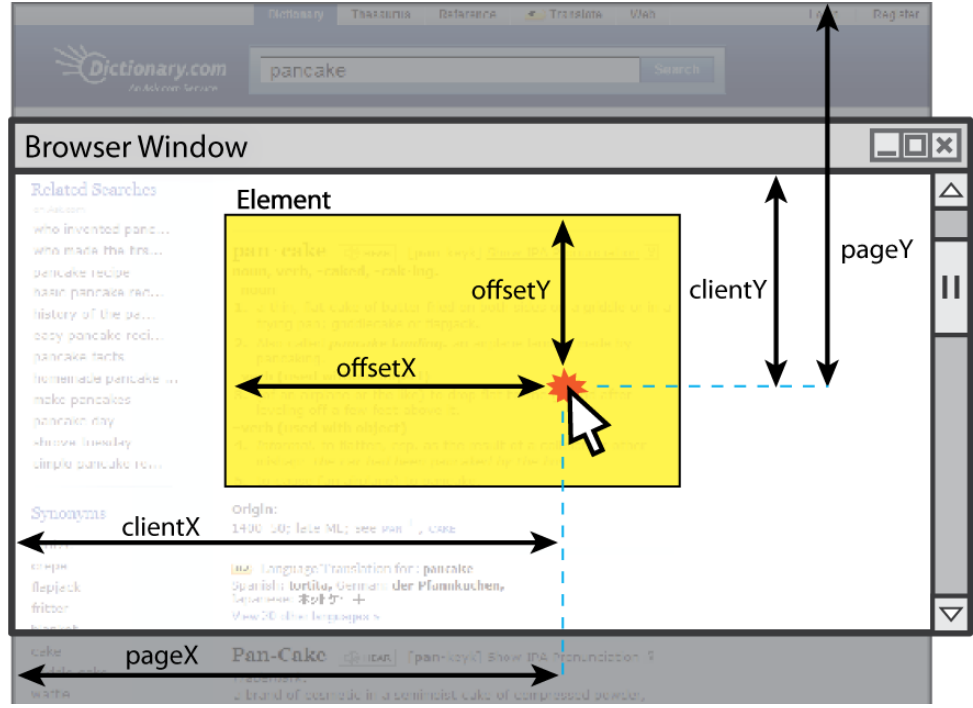
Назва методу	Пояснення
`Element.scrollTo(x, y)`	Прокручує елемент до заданих координат (x — горизонтальна, y — вертикальна). Наприклад, `element.scrollTo(0, 50)` прокрутить елемент на 50 пікселів вниз.
`Element.scrollBy(x, y)`	Прокручує елемент на задану кількість пікселів відносно поточної позиції. Наприклад, `element.scrollBy(0, 10)` прокрутить елемент на 10 пікселів вниз.
`Element.scroll()`	Аналогічний `scrollTo()`, але підтримує додаткові опції. Наприклад, `element.scroll({ top: 50, behavior: 'smooth' })` забезпечує плавну прокрутку елемента.



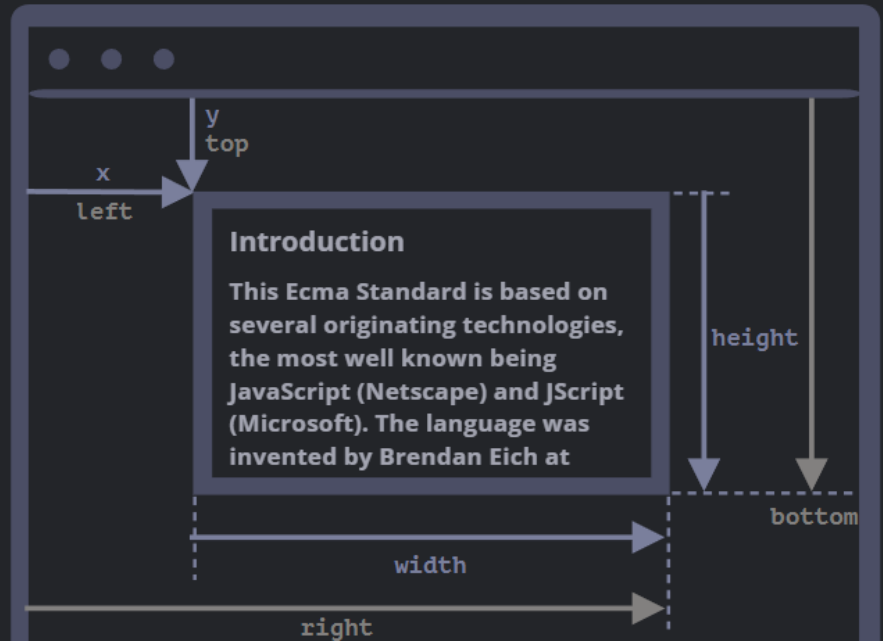
https://uk.javascript.info/size-and-scroll

Координати

Властивість	Пояснення
`clientX`	Встановлює або отримує горизонтальну координату відносно вікна перегляду.
`clientY`	Встановлює або отримує вертикальну координату відносно вікна перегляду.
`pageX`	Встановлює або отримує горизонтальну координату відносно сторінки.
`pageY`	Встановлює або отримує вертикальну координату відносно сторінки.
`screenX`	Встановлює або отримує горизонтальну координату відносно екрана.
`screenY`	Встановлює або отримує вертикальну координату відносно екрана.
`offsetX`	Встановлює або отримує горизонтальну координату відносно елемента.
`offsetY`	Встановлює або отримує вертикальну координату відносно елемента.



Ось зображення з результатом виклику `elem.getBoundingClientRect()`:



https://uk.javascript.info/coordinates

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 = 'https://www.ukr.net/'` • 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.

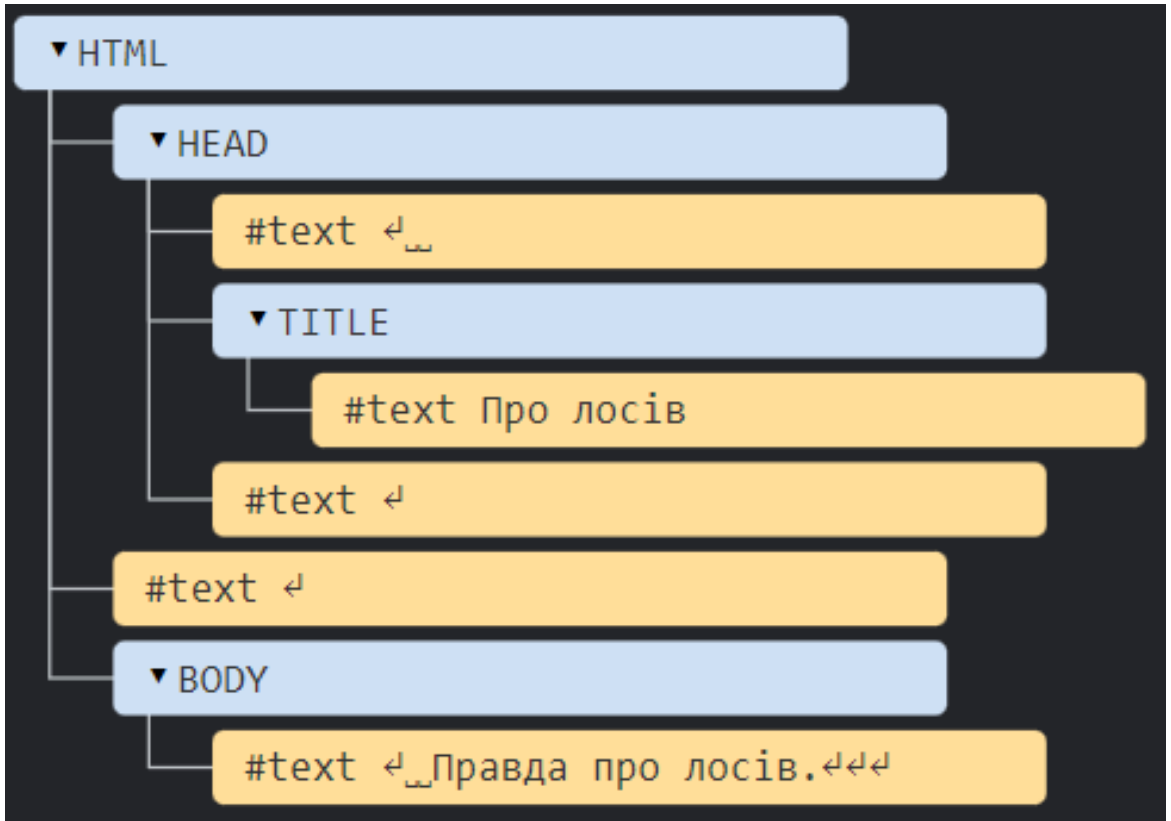
Об’єкт ***document***

- 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».

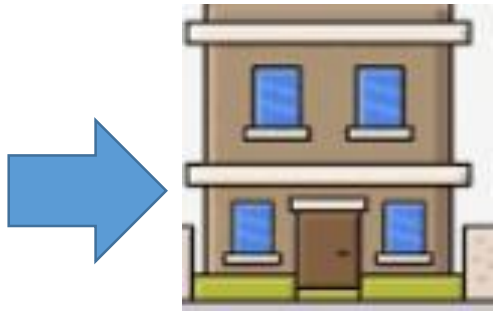
```
<!DOCTYPE HTML>
<html>
<head>
  <title>Про лосів</title>
</head>
<body>
  Правда про лосів.
</body>
</html>
```

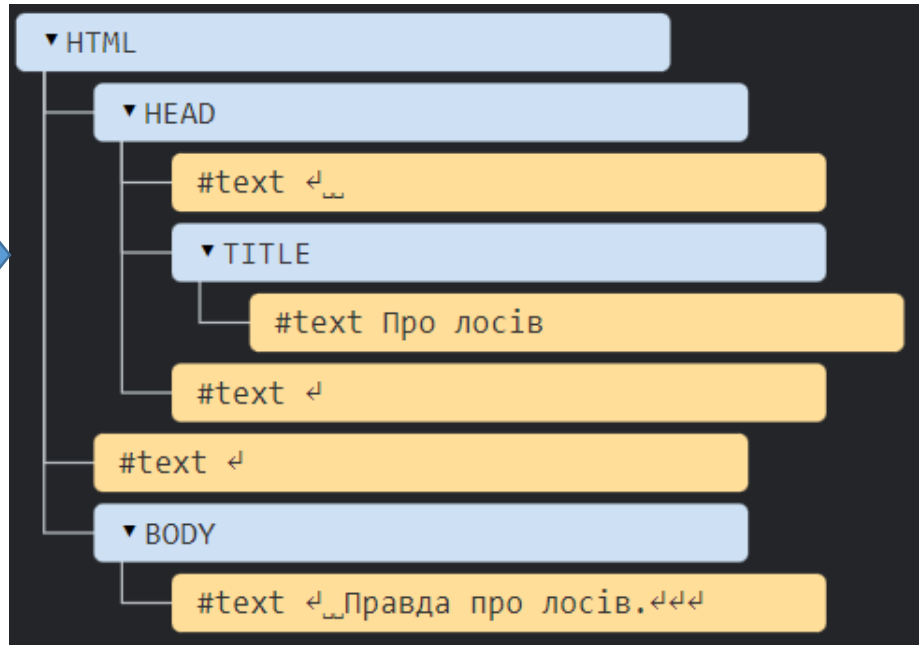


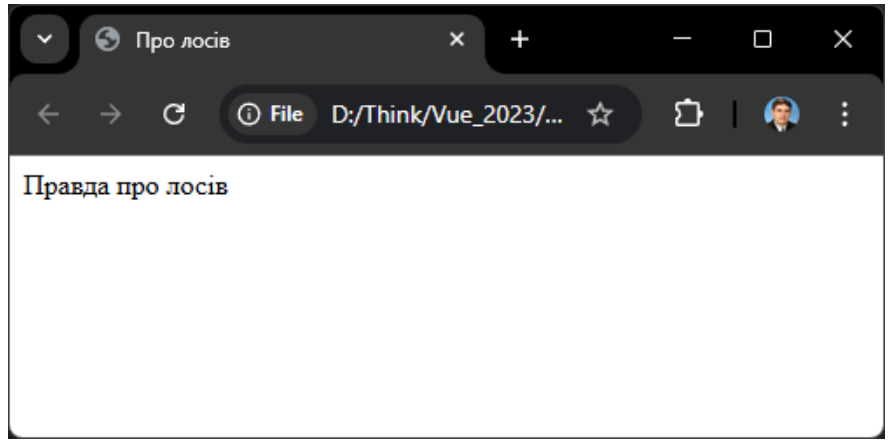
https://uk.javascript.info/document

```
const hotel = {
  rooms: {
    room1: {
      bathroom: {
        hasBathtub: false,
        sink: true,
        toilet: true,
      },
      bedroom: {
        bed: {…
        nightstand: {…
      },
      sofa: {…
      table: {…
      chair: {…
    },
    room2: {
      bathroom: {
        hasShower: false,
        hasBathtub: true,
        sink: true,
        toilet: true,
      },
      bedroom: {
        bed: {…
        nightstand: {…
      },
      sofa: {…
      table: {…
```

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







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

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)
```

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>
```

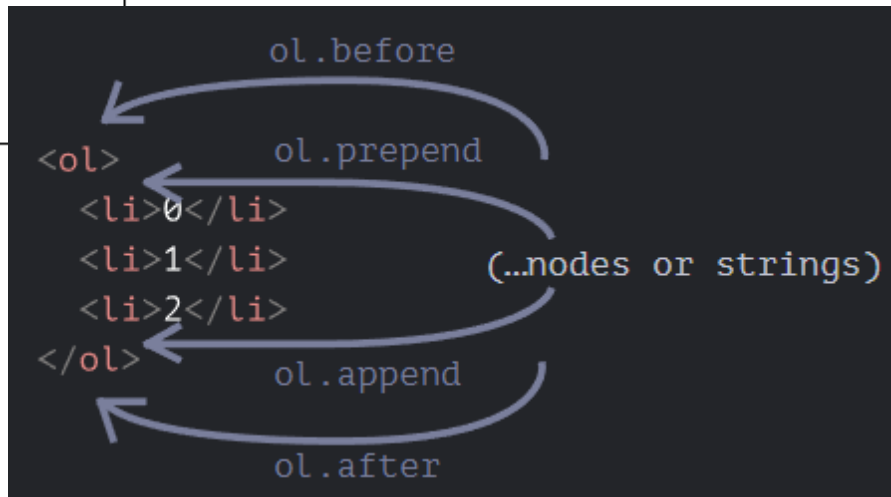
Створення вузлів

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

Атрибути vs Властивості

Атрибути:

- Атрибути — це те, що визначено в HTML-коді. Наприклад, у тегу `<input type="text" id="myInput" value="Hello">` атрибутами є `type`, `id`, `value`.
- Атрибути доступні через методи `getAttribute()`, `setAttribute()`, `removeAttribute()`.
- Вони завжди є рядками (strings), оскільки HTML — це текстовий формат.
- Зміна атрибута через `setAttribute()` змінює HTML-код, але не завжди впливає на поведінку елемента в DOM.

Властивості:

- Властивості — це частина DOM-об'єкта, який створюється браузером на основі HTML. Наприклад, для `<input>` об'єкт `HTMLInputElement` має властивості `type`, `id`, `value`.
- Властивості доступні напряму через об'єкт елемента, наприклад, `element.id`, `element.value`.
- Властивості можуть бути не лише рядками, а й іншими типами даних (наприклад, `element.checked` — це boolean).
- Зміна властивості зазвичай впливає на поведінку елемента в DOM, але не завжди оновлює HTML-атрибут.

Атрибути vs Властивості

Атрибут	Відповідна властивість	Рекомендація	Примітки
`id`	`element.id`	Використовуйте властивість	Синхронізується з атрибутом.
`class`	`element.className`	Використовуйте `element.classList`	`classList` зручніший для роботи з класами (додавання/видалення).
`value` (для `<input>`)	`element.value`	Використовуйте властивість	Атрибут задає початкове значення, властивість — поточне.
`href` (для `<a>`)	`element.href`	Використовуйте властивість	Властивість повертає повний URL, атрибут — значення з HTML.
`disabled`	`element.disabled`	Використовуйте властивість	Властивість — boolean, атрибут — рядок (`"disabled"` або відсутній).
`data-*` (наприклад, `data-id`)	`element.dataset.id`	Використовуйте `dataset`	`data-*` атрибути доступні через `element.dataset`.
`role` (ARIA)	Немає властивості	Використовуйте `setAttribute()`	ARIA-атрибути не мають відповідних властивостей.
`style`	`element.style`	Використовуйте властивість	`element.style` — об’єкт для роботи з CSS, наприклад, `element.style.color`.

ПОДІЇ

Всі зміни, які відбуваються на Web-сторінки, пов'язані з роботою браузера або маніпуляціями користувача з клавішами миші або клавіатури, називаються подіями. Для вказівки дій, які необхідно зробити в зв'язку з появою того або іншої події, використовуються обробники подій.

Типи подій

Можна виділити декілька груп подій, у залежності від того як вони генеруються. Розглянемо деякі з них

Події документу

load	Закінчено завантаження документа (Frame, Image, Layer)
unload	Користувач залишає сторінку, наприклад, завантажуючи інший документ (Frame, window)
resize	Користувач змінив розмір вікна (Frame, window)
error	Виникла помилка підчас завантаження (Image, window)
move	Користувач перемістив вікно (Frame, window)

`DOMContentLoaded` – коли HTML завантажений і опрацьований, DOM документу повністю побудований і доступний

Події миші

click	Користувач натискає кнопку миші (Button, Checkbox, document, Link, Radio, Reset, Submit , і т.д. – майже всі об'єкти
mousedown	Користувач натиснув кнопку миші. (Button, document, Link)
mouseup	Користувач відпустив конпку миші (Button, document, Link)
dblclick	Кристувач двічі натискає на кнопку миші (Button, Checkbox, document, FileUpload, Hidden, Link, Password, Radio, Reset, Select, Submit)
dragdrop	Користувач перетягує об'єкт у вікно
mousemove	Користувач перемістив курсор миші (Виникає лише при явному заданні обробника).
mouseout	Користувач перемістив курсор за межі об'єкта(Area, Layer, Link)
mouseover	Користувач помістив курсор миші над об'єктом (Area, Layer, Link)

contextmenu – відбувається при виклику констекстного меню (права кнопка миші)

Події клавіатури

keydown	Користувач натиснув клавішу (document, Image, Link, Text, Textarea)
keyup	Користувач відпустив клавішу (document, Image, Link, Text, Textarea)
keypress	Користувач натиснув клавішу і відпустив. (document, Image, Link)

Події елемента

abort	Користувач зупиняє завантаженнязображення (Image)
blur	Користувач прибирає фокус із об'єкта (Button,Checkbox,FileUpload, Frame, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window)
change	Користувач змінює зміст елемента форми (the FileUpload, Select, Text)
error	Виникла помилка підчас завантаження (Image, window)
focus	Користувач перемістив фокус на об'єкт (Button, Checkbox, FileUpload, Frame, Layer, Password, Radio, Reset, Select, Submit, Text, Textarea, window)
select	Користвуач виділив текст (Text, Textarea)

ОБРОБКА ПОДІЙ

Під обробкою подій розуміють виконання деякої команди або ж функції у відповідь на настання події. Обробка подій може здійснювати декількома способами

Задання обробника з використанням атрибуту елемента HTML

У цьому способі необхідно у HTML розмітці елемента додати атрибут, який починається з «on» і містить назву події (наприклад: **on**click, **on**change, **on**select,...). Значенням атрибуту може бути або деяка команда або виклик функції. Як завжди значення атрибутів вказуємо у лапках.

Загальна форма	<елемент **on**подія= "виклик_обробника">
Приклад (вказівка команди)	`<input type="button" name="name" value="Press" onclick="alert("Hello")" />`
Приклад (виклик функції)	`<input type="button" name="name" value="Press" onclick="myFunc()" />`

Приклад. Зчитування значення з текстового поля

---------- у тексті сторінки ----------

```
    <input type="text" id="MyText" value="" /><br>
```

---------- звертання до елемента -------

```
            let name = document.getElementById("MyText").value;

            alert("Hello " + name);
```

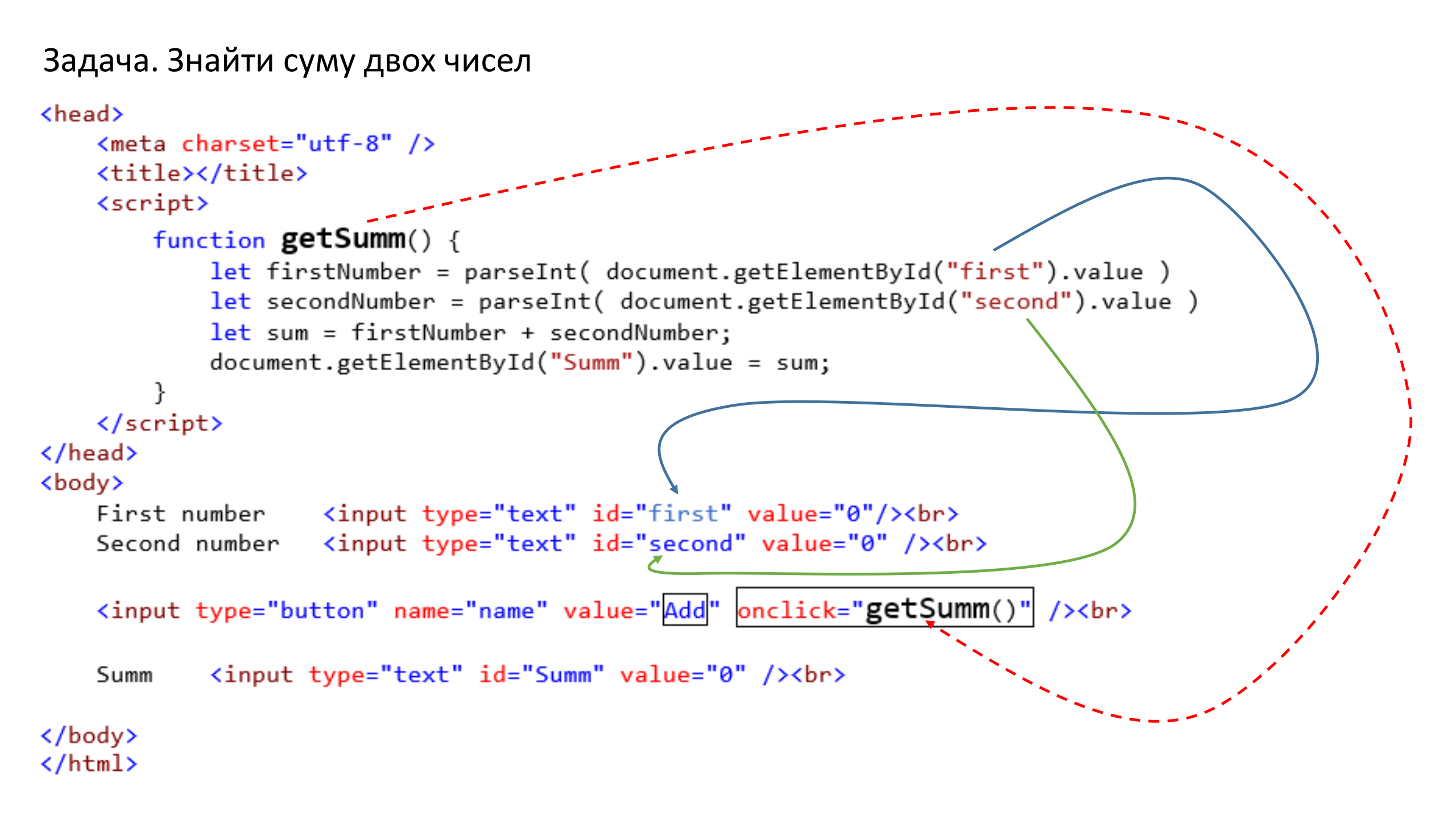
Задача. Знайти суму двох чисел

```
<head>
    <meta charset="utf-8" />
    <title></title>
    <script>
        function getSumm() {
            let firstNumber = parseInt( document.getElementById("first").value )
            let secondNumber = parseInt( document.getElementById("second").value )
            let sum = firstNumber + secondNumber;
            document.getElementById("Summ").value = sum;
        }
    </script>
</head>
<body>
    First number    <input type="text" id="first" value="0"/><br>
    Second number   <input type="text" id="second" value="0" /><br>

    <input type="button" name="name" value="Add" onclick="getSumm()" /><br>

    Summ    <input type="text" id="Summ" value="0" /><br>

</body>
</html>
```



Задання обробника з використанням властивості елемента

У цьому способі необхідно використати властивість елемента як об'єкта JavaScript. При цьому ім'я властивості, як і атрибут має вигляд **on**подія. На відміну від атрибутів елементів, які можуть бути вказані як у нижньому так і верхньому регістрах (onclick, ONCLICK, Onclick,onClick), властивості елемента вказують у нижньому регістрі (onclick).

Загальна форма	елемент . **on**подія= функція_обробника ; елемент ["**on**подія "]= функція_обробника ;
Приклад	``` ------------ Елемент в HTML ------------ <input type="button" id="myButton" value="Press" /> ------------ Встановлення обробника в JavaScript ------------ <script> //------ Варіант 1 ----- myButton.onclick = myFunc; // myFunc – функція обробник //------ Варіант 2 ----- myButton["onclick"] = myFunc; // myFunc – функція обробник </script> ```

```
function функція (){
    ...
}
window.onload = function () {
    document.getElementById("btn").onclick = функція
}
```

```
        function getSum() {
            const num1 = parseInt(document.getElementById("num1").value)
            const num2 = parseInt(document.getElementById("num2").value)
            const s = num1 + num2
            document.getElementById("res").value = s
        }

        window.onload = function () {
            document.getElementById("btn").onclick = getSum
        }

        alert("hello")
    </script>

</head>

<body>
    <label>         Number 1        <input type="number" id="num1" value="2">    </label> <br>
    <label>         Number 2        <input type="number" id="num2" value="3">    </label> <br>
    <button id="btn">         Get sum    </button> <br>
```

```html
<html lang="en">
<head>
    <title>Document</title>
    <script>
        function getSum() {
            const num1 = parseInt(document.getElementById("num1").value)
            const num2 = parseInt(document.getElementById("num2").value)
            const s = num1 + num2
            document.getElementById("res").value = s
        }

        window.onload = function () {
            document.getElementById("btn").onclick = getSum
        }

        alert("hello")
    </script>

</head>

<body>
    <label>        Number 1        <input type="number" id="num1" value="2">    </label> <br>
    <label>        Number 2        <input type="number" id="num2" value="3">    </label> <br>
    <button id="btn">        Get sum    </button> <br>
    <label>        Sum         <input type="number" name="" id="res" value="">
    </label>
</body>
</html>
```

Задача. Конвертер валют (курс і кількість валюти, що треба обміняти задаються)

Приклад. Перевірка стану чекбокса

---------- у тексті сторінки ----------

```
<input type="checkbox" id="MyCheckbox" value="10" /><br>
```

---------- звертання до елемента -------

```
if (document.getElementById("MyCheckbox").checked) {
                alert("Is checked")
            }
            else {
                alert("Is not checked");

            }
```

Задача. Знайти загальну суму обіду

---- напої ---

- чай -10 грн.
- сік – 20 грн
- кава – 35 грн

-----перше ---

- суп – 45 грн
- борщ - 37 грн.

---- друге –

- паста – 60 грн.
- картопля з катлетою – 55 грн.
- гречка з грибами – 49 грн.

Приклад. Аналіз значення з випадаючого списку

---------- у тексті сторінки ----------

```
    <select id="MySelect">
        <option value="1">text1</option>
        <option value="2">text2</option>
        <option value="3">text3</option>

    </select>
```

---------- звертання до елемента -------

```
let value = document.getElementById("MySelect").value;
```

Задача. Знайти загальну вартість поїздки у поїзді:
тип вагону :

- плацкарт – 200грн.
- купе – 800грн.

акційні пакети (вибір тільки одного)

- сніданок – акційна ціна 50 грн.
- безкоштовний чай
- безкоштовна кава