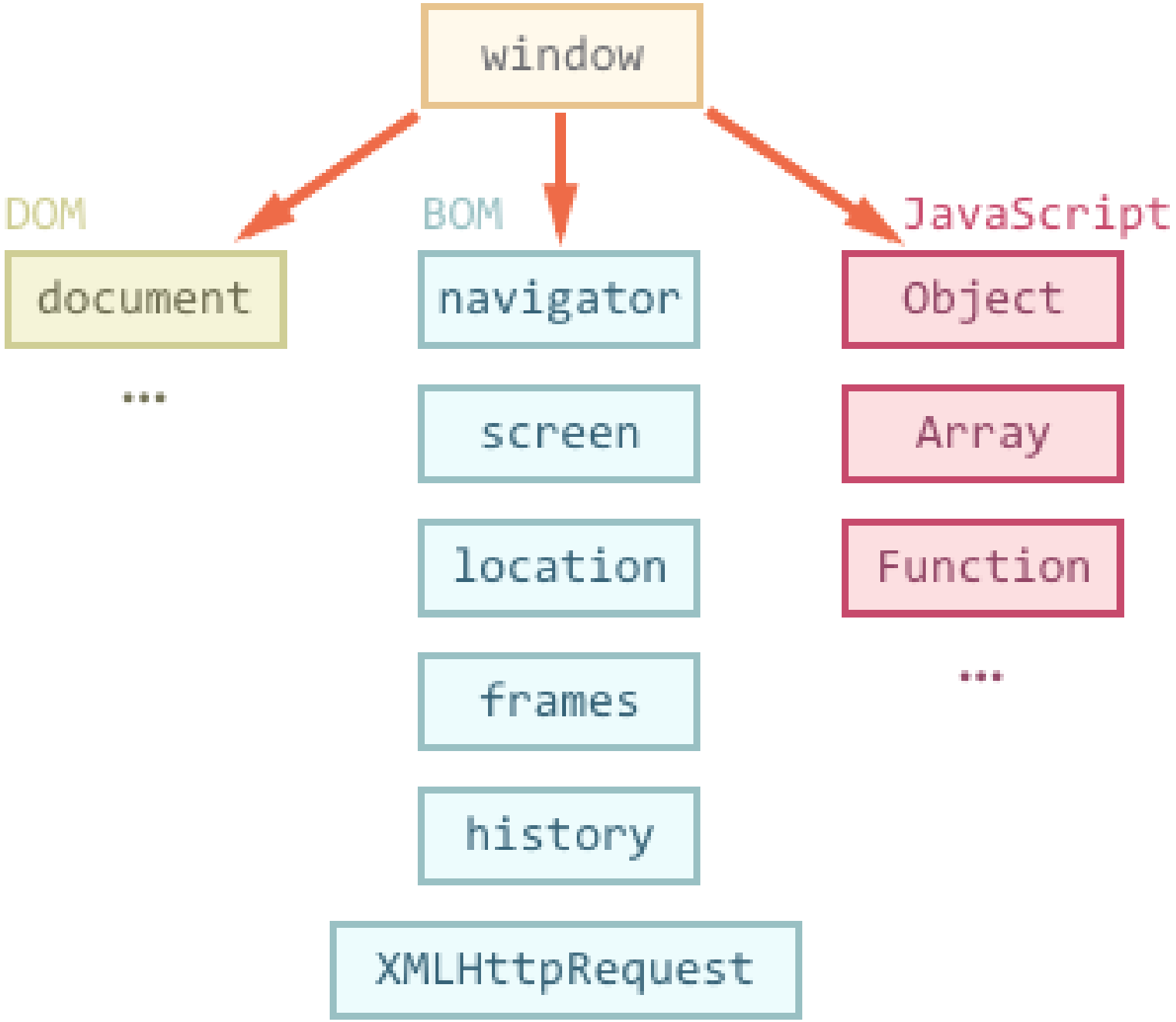
Тема: DOM

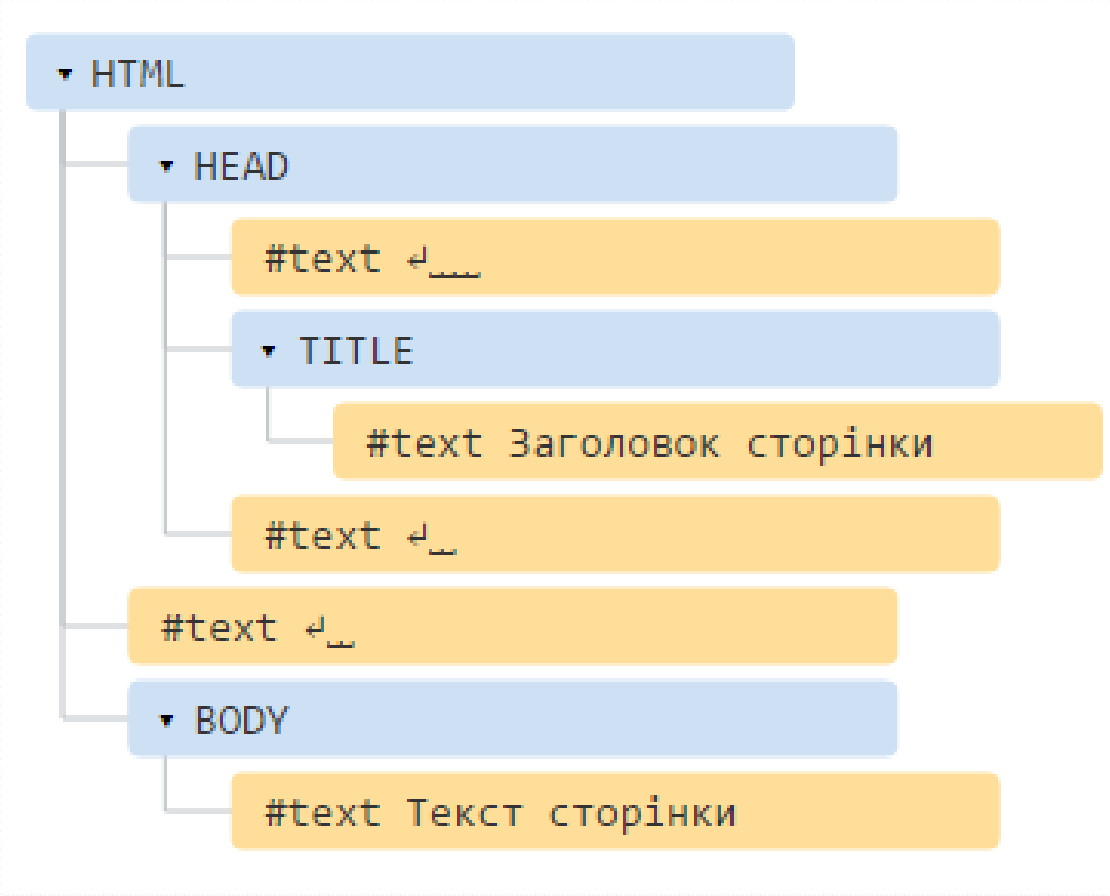
1. Загальні визначення

- **BOM (Browser Object Model) – об’єктна модель браузера;**
- **DOM (Document Object Model) – об’єктна модель документа;**
- Об’єктна модель представляє собою вкладені один в інший об’єкти. Головним об’єктом є об’єкт **window.**
- Відповідно до DOM кожний документ є деревом.
- Кожний тег HTML-документу утворює вузол дерева з типом «елемент». Вкладені у нього теги є дочірніми вузлами.
- Для представлення тексту створюються вузли типу «текст».
- DOM – представлення документа у вигляді дерева, доступне для роботи за допомогою мови JavaScript.


window
DOM
BOM
JavaScript
document
...
navigator
screen
location
frames
history
XMLHttpRequest
Object
Array
Function
...

- **window**:
  - **location** – рядок адреси
  - **navigator** – інформація про браузер
  - **history** – журнал відвіданих посилань
  - **XMLHttpRequest** – об’єкт для асинхронних запитів на сервер
  - **document** – об’єкт, який надає доступ до HTML-документу
    - **all []** – масив усіх тегів
    - **forms []** – масив форм
    - **links []** – масив посилань
    - **anchors []** – масив якорів
    - **images []** – масив зображень
    - **embeds []** – масив Flash- або інших вбудованих елементів
    - **scripts []** – масив усіх тегів script
    - **styleSheets []** – масив таблиць стилів
    - **body** – відповідає тегу body
  - **event** – об’єкт, який містить параметри подій
  - **screen** – інформація про роздільну здатність екрана

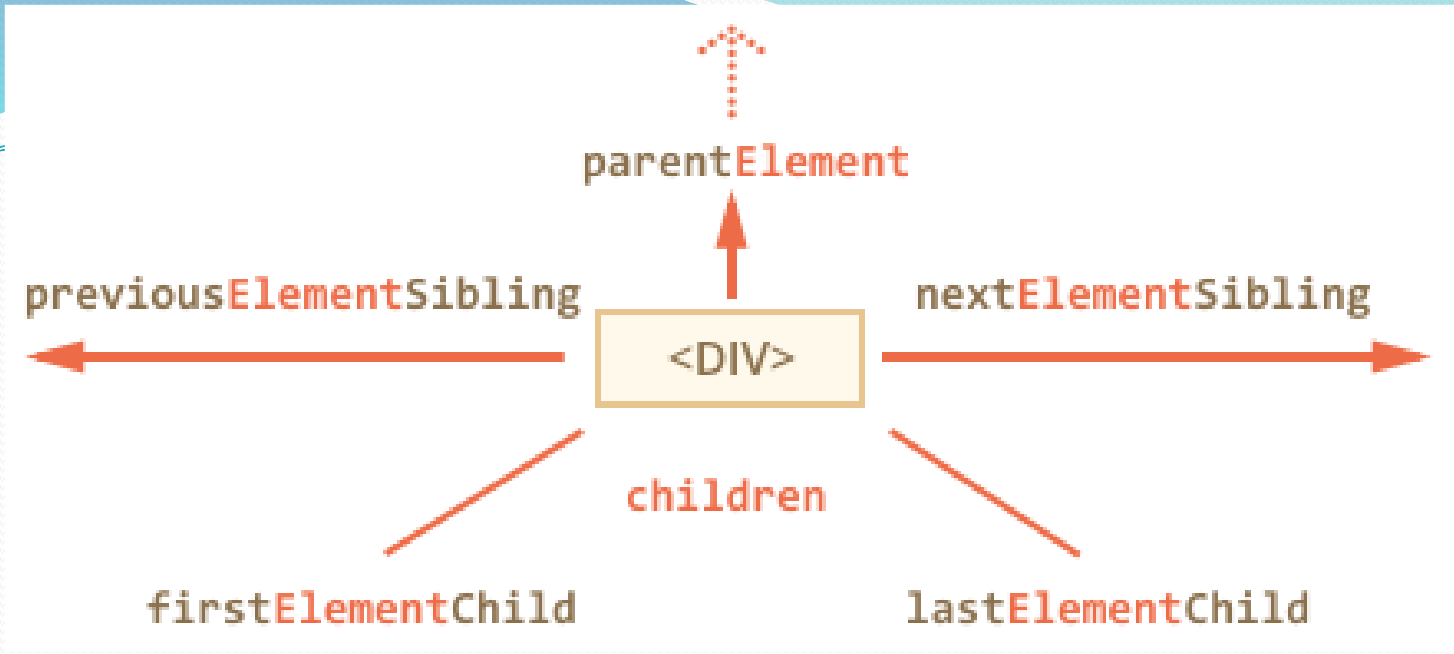
```
<!DOCTYPE HTML>
<html>
<head>
  <title>Заголовок сторінки</title>
</head>
<body>
  Текст сторінки
</body>
</html>
```

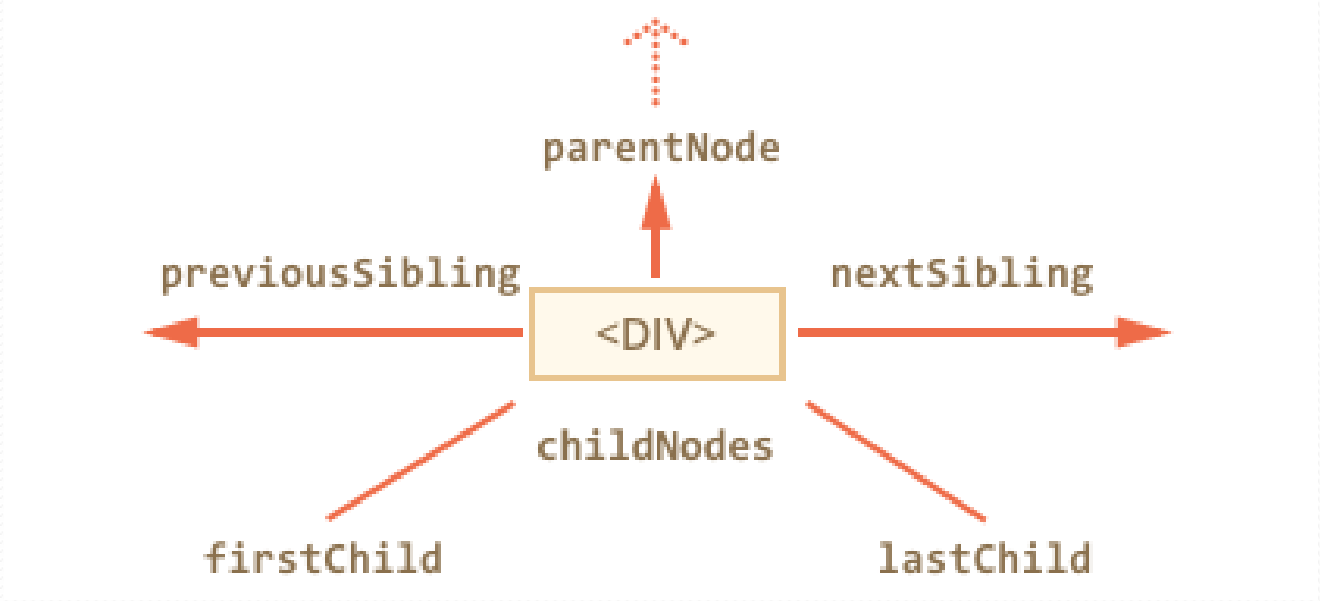


2. Навігація по DOM-структурі

- Отримання DOM-об’єкта за ідентифікатором:
  `document.getElementById("ідентифікатор")`
- Отримання DOM-елементів, що мають заданий атрибут name:
  `document.getElementsByName("ім’я")`
- Пошук DOM-об’єктів за тегом:
  `elem.getElementsByTagName("тег")`
- Пошук DOM-елементів за класом:
  `elem.getElementsByClassName("клас")`
- Пошук DOM-елементів за CSS-селектором (усі, що знайдені):
  `elem.querySelectorAll("CSS-селектор")`
- Пошук DOM-елементу за CSS-селектором (першого):
  `elem.querySelector("CSS-селектор")`

Повертаються DOM-елементи за виключенням текстових	
тег <html>	`document.documentElement`
тег <body>	`document.body`
колекція нащадків	`elem.children`
батьківський елемент	`elem.parentNode`
перший нащадок	`elem.firstElementChild`
останній нащадок	`elem.lastElementChild`
правий сусід	`elem.nextElementSibling`
лівий сусід	`elem.previousElementSibling`
Повертаються усі DOM-елементи, включаючи текстові	
колекція нащадків	`elem.childNodes`
перший нащадок	`elem.firstChild`
останній нащадок	`elem.lastChild`
правий сусід	`elem.nextSibling`
лівий сусід	`elem.previousSibling`


parentElement
previousElementSibling
nextElementSibling
<DIV>
children
firstElementChild
lastElementChild


parentNode
previousSibling
nextSibling
<DIV>
childNodes
firstChild
lastChild

Властивості DOM-вузлів (можна лише зчитувати):

- **`tagName`** – є тільки у тегів (**`elem. nodeType == Node.ELEMENT_NODE`**)
- **`nodeName`** – для тегів містить **`tagName`**, а для усіх інших елементів містить рядок з типом вузла;
- **`textContent`** – вміст тегу у текстовому вигляді (конкатенація усіх текстових вузлів)

Властивості DOM-вузлів (можна зчитувати і записувати):

- **`innerHTML`** – містить вміст тегу (лише для тегів - рядок з HTML-кодом);
- **`data`** – вміст DOM-елементів (для усіх DOM-елементів окрім тегів, застосовують для, наприклад текстового вузла, або коментаря)

Отримання класів тегу:

- **`className`** – значення атрибуту class (рядок);
- **`classList`** – класи у вигляді колекції;
  - **`elem.classList.contains("class")`** – повертає true/false, в залежності від того, чи є у елемента клас **`class`**.
  - **`elem.classList.add/remove("class")`** – додає/видаляє клас **`class`**
  - **`elem.classList.toggle("class")`** – якщо класу **`class`** немає, додає його, якщо є – видаляє.

Методи для роботи з атрибутами тегу:

`elem.hasAttribute(name)`	перевіряє наявність атрибуту
`elem.getAttribute(name)`	повертає значення атрибуту
`elem.setAttribute(name, value)`	встановлює значення атрибуту
`elem.removeAttribute(name)`	видаляє атрибут

- Створення DOM-елемента:

  `document.createElement("тег")`
- Створення текстового вузла:

  `document.createTextNode("текст")`
- Додавання вузла у DOM як дочірнього тега:

  `parentElem.appendChild(elem)`

  Тут: **`parentElem`** – батьківський елемент, у який додається елемент **`elem`**. Елемент додається у кінець.
- Додавання вузла у DOM як лівого сусіда:

  `parentElem.insertBefore(elem, nextElem)`

Тут: **`parentElem`** – батьківський елемент, у який додається елемент **`elem`**; **`nextElem`** – елемент, перед яким додається елемент **`elem`**

Приклад 1.

HTML:

```
<div id="container">
   <p>Абзац 1</p>
   <p>Абзац 2</p>
</div>
```

Створюється в пам'яті:

```
<div class="alert big">
    Помилка при введенні e-mail
</div>
```

Додається у контейнер:

```
<div id="container">
  <p>Абзац 1</p>
  <p>Абзац 2</p>
  <div class="alert big">
    Помилка при введенні e-mail
  </div>
</div>
```

JavaScript:

```
var div = document.createElement('div');
div.className = "alert big";
div.innerHTML = "Помилка при введенні e-mail";
document.getElementById("container").appendChild(div)
```

Приклад 2.

HTML:

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

Створюється в пам’яті:

```
<li>Новий пункт!</li>
```

Додається у DOM-структуру:

```
<ol id="list">
  <li>0</li>
  <li>Новий пункт!</li>
  <li>1</li>
  <li>2</li>
</ol>
```

JavaScript:

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

- Створення копії DOM-вузла (разом з вкладеними вузлами):
  `elem.cloneNode(true)`
- Створення копії DOM-вузла (без вкладених вузлів):
  `elem.cloneNode(false)`
- Видалення DOM-вузла:
  `parentElem.removeChild(elem)`
- Заміна дочірнього елементу:
  `parentElem.replaceChild(newElem, elem)`

Приклад.

```
var div = document.createElement('div');
div.className = "alert alert-success";
div.innerHTML = "<strong>Ура!</strong> Є нове повідомлення.";

document.body.insertBefore(div, document.body.firstChild);

// створити копію вузла
var div2 = div.cloneNode(true);

// в попію можна вносити зміни
div2.querySelector('strong').innerHTML = 'Супер!';

// і додати її у DOM-стуктуру
div.parentNode.insertBefore(div2, div.nextElementSibling);
```

HTML:

```
<body>
  <h3>Заголовок</h3>
</body>
```

5. Змінена копія div-блоку додається у DOM-структуру

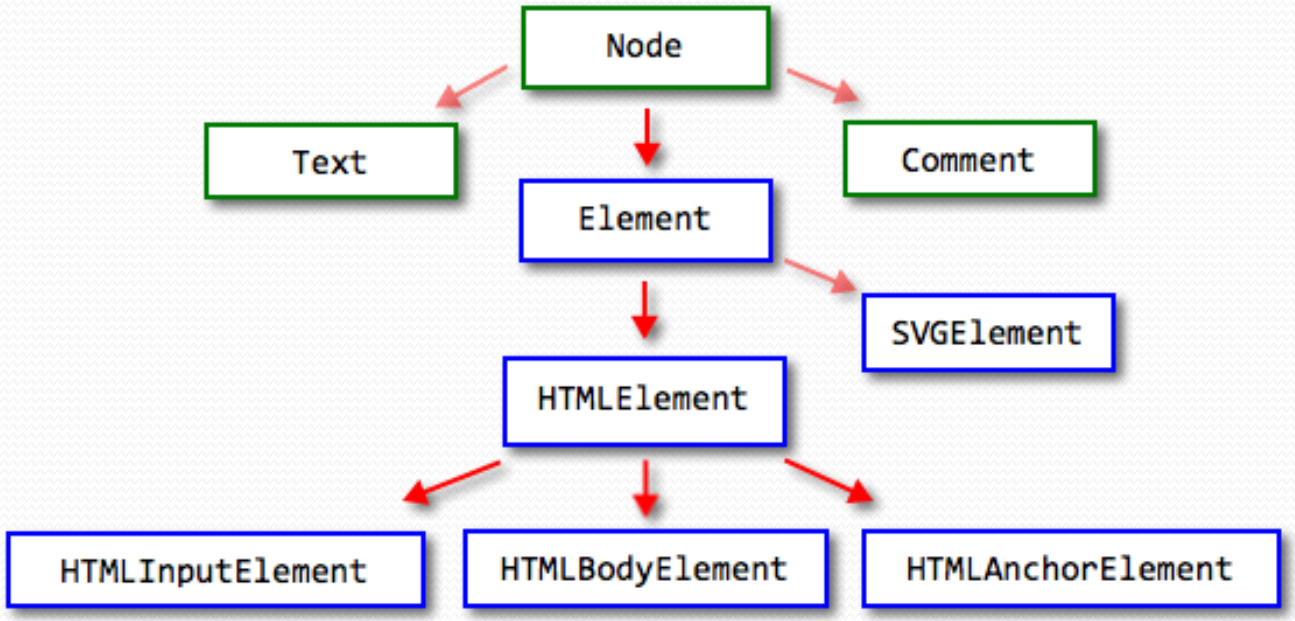
```
<body>
  <div class="alert alert-success">
    <strong>Ура!</strong> Є нове повідомленн
  </div>
  <div class="alert alert-success">
    <strong>Супер!</strong> Є нове
повідомлення.
  </div>
  <h3>Заголовок</h3>
</body>
```

У полі **elem.nodeType** кожного DOM-вузла зберігається тип вузла, який може мати одне із значень:

Значення	Константа	Пояснення
1	ELEMENT_NODE	Тег
2	ATTRIBUTE_NODE	Атрибут тега
3	TEXT_NODE	Текстовий вузол
8	COMMENT_NODE	Вузол, що відповідає за коментар
9	DOCUMENT_NODE	для об’єкту document

```
var childNodes = elem.childNodes;
 for (var i = 0; i < childNodes.length; i++) {
      // знайти DOM-елементи, які відповідаються тегам
      if (childNodes[i].nodeType != Node.ELEMENT_NODE)
      {
         …
      }
}
```

DOM-вузли є об’єктами різних класів:




```
console.log(document.body instanceof HTMLBodyElement); // true
console.log(document.body instanceof HTMLElement); // true
console.log(document.body instanceof Element); // true
console.log(document.body instanceof Node); // true
```