Методи маніпуляції елементами в JS

```
let myElement = document.querySelector('.dev')
```

Класи

Властивість **classList** повертає псевдомасив, що містить усі класи елемента.

Методи **classList:**

myElement.classList.add(String [,String]) – додає елементу вказані класи

myElement.classList.remove(String [,String]) – видаляє в елемента вказані класи

myElement.classList.toggle(String [, Boolean]) – перемикач

myElement.classList.contains(String) – перевіряє наявність класу в елемента

Методи для роботи з атрибутами

myElement.getAttibute(attributeName) – повертає значення атрибута ***attributeName***

myElement.setAttribute(attributeName, value) – встановлює значення ***value*** атрибуту ***attributeName***

myElement.removeAttribute(attributeName) – видаляє атрибут ***attributeName***

Додавання inline-стилів

```
myElement.style.color = 'blue';
// додає вказаний стиль

myElement.style.cssText = 'color: blue; background: white';
// додає декілька стилів

myElement.setAttribute ('style', 'color: blue; background: white');
// додає декілька стилів
```

*Зазвичай стилі JavaScript аналогічні стилям в CSS, за винятком того, що стилі, написані через дефіс, змінені на **camelCase**. Наприклад, **background-color** стає **backgroundColor**.*

Звернення до елементів

```
let myElement = document.querySelector('.dev')
```

myElement.children – повертає колекцією елементів DOM, які є дочірніми елементами

myElement.firstElementChild – повертає перший дочірній елемент ***myElement*** або значення null

myElement.lastElementChild – повертає останній дочірній елемент ***myElement*** або значення null

myElement.previousElementSibling – повертає елемент безпосередньо перед зазначеним у дочірньому списку його батьків або null, якщо вказаний елемент є першим у списку

myElement.nextElementSibling – повертає елемент безпосередньо після зазначеного у дочірньому списку його батьків або null, якщо вказаний елемент є останнім у списку

Створення нових елементів

```
let myNewElement = document.createElement('div')
let myNewTextNode = document.createTextNode('some text')
```

Методи .innerHTML та .textContent

```
myElement.innerHTML = '
  <div>
    <h2>New content</h2>
    <p>some text</p>
  </div>
'
```

Видалення

```
myElement.innerHTML = null
```

Додавання

```
myElement.innerHTML += '
  <a href="about.html">continue reading...</a>
  <hr>
'
```

Методи вставки

myElement.append(...nodes or strings) – додає вузли або рядки в кінець ***myElement***

myElement.appendChild(childNode) – додає ***childNode*** в кінець ***myElement***

myElement.prepend(...nodes or strings) – додає вузли або рядки на початок ***myElement***

myElement.before(...nodes or strings) – додає вузли або рядки до ***myElement***

myElement.after(...nodes or strings) – додає вузли або рядки після ***myElement***

myElement.replaceWith(...nodes or strings) – заміняє ***myElement*** заданими вузлами або рядками

myElement.insertBefore(newNode, referenceNode) – додає ***newNode*** в ***myElement*** перед ***referenceNode*.**

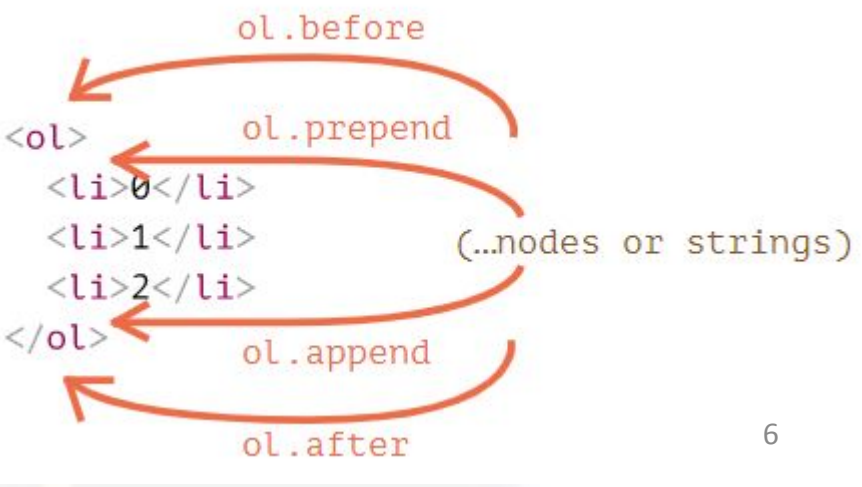
Методи вставки (приклад)

```
// файл HTML
<ol id="ol">
  <li>0</li>
  <li>1</li>
  <li>2</li>
</ol>
// файл JavaScript
<script>
  ol.before('before'); // вставити рядок "before" перед <ol>
  ol.after('after'); // вставити рядок "after" після <ol>

  let liFirst = document.createElement('li');
  liFirst.innerHTML = 'prepend';
  ol.prepend(liFirst); // вставити liFirst на початок <ol>

  let liLast = document.createElement('li');
  liLast.innerHTML = 'append';
  ol.append(liLast); // вставити liLast в кінець <ol>
</script>
```

```
before
1.prepend
2.0
3.1
4.2
5.append
after
```



Створення нових елементів

```
let link = document.createElement('a')
let text = document.createTextNode('continue reading...')
let hr = document.createElement('hr')

link.href = 'about.html'
link.appendChild(text)

myElement.appendChild(link)
myElement.appendChild(hr)
```

Видалення елементів

myParentElement.removeChild(myElement) – видаляє myElement з myParentElement в DOM та повертає посилання на myElement

```
// файл HTML
<div id="top" align="justify">
      <div id="nested"></div>
</div>
// файл JavaScript
let d = document.getElementById("top");
let d_nested = document.getElementById("nested");
let removedNode = d.removeChild(d_nested);
```

myElement.remove() – видаляє елемент з дерева, якому він належить

```
// файл HTML
<div id="div1">Here is div1</div>
<div id="div2">Here is div2</div>
<div id="div3">Here is div3</div>
// файл JavaScript
let myElement = document.getElementById('div2');
myElement.remove(); // видалить div з ідентифікатором div2
```

Задання функцій через «стрілку» =>

```
let func = x => 2*x;
```

```
let func = function(x) { return 2*x; };
```

```
let getTime = () => {
  let date = new Date();
  let hours = date.getHours();
  let minutes = date.getMinutes();
  return (hours + ':' + minutes);
};
alert( getTime() ); // поточний час
```

- Без фігурних дужок повертають вираз ***expr: (args) => expr***.
- З фігурними дужками вимагають явного **return**.
- Не мають своїх **this** і **arguments**, при зверненні одержують їх з навколишнього контексту.
- Не можна використовувати як конструктори, з new.

Обробники подій

```
<button onclick="alert('Hello!')">Click here</button>
```

```
// файл HTML
<button id="btn"> Click here </button>
// файл JavaScript
const btn = document.querySelector('#btn');
btn.onclick = () => alert('Hello!');
```

```
// файл HTML
<button id="btn"> Click here </button>
// файл JavaScript
const btn = document.querySelector('#btn');
btn.addEventListener('click', () => {
  alert ("Hello!");
});
```