Document and Element interface

Nội dung bài học

1. Document interface
2. Element interface
3. HTMLElement interface

1. Document interface

Properties

#	Name	Description
1	`document.documentElement`	Trả về node HTML (`<html>`)
2	`document.body`	Trả về node body (`<body>`)
3	`document.head`	Trả về node head (`<head>`)
4	`document.cookie`	(HTMLDocument) Lấy / đặt cookie
5	`document.title`	(HTMLDocument) Lấy tiêu đề của tài liệu hiện tại

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document Object Example</title>
</head>
<body>
  <h1>Hello, world!</h1>
  <p>This is a paragraph.</p>

  <script>
    // Truy cập các thuộc tính của đối tượng document
    const htmlNode = document.documentElement;
    const bodyNode = document.body;
    const headNode = document.head;
    const cookieInfo = document.cookie;
    const titleInfo = document.title;

    // In ra các thông tin thu được từ document
    console.log('HTML Node:', htmlNode);
    console.log('Body Node:', bodyNode);
    console.log('Head Node:', headNode);
    console.log('Cookie Info:', cookieInfo);
    console.log('Title Info:', titleInfo);
  </script>
```

```
</body>
</html>
```

Methods

Query elements

#	Name	Description
1	`document.getElementById(id)`	Lấy một phần tử bằng id
2	`document.getElementsByClassName(name)`	Lấy một NodeList sống bằng tên lớp
3	`document.getElementsByTagName(name)`	Lấy một NodeList sống bằng tên thẻ
4	`document.querySelector(selector)`	Lấy node đầu tiên khớp với selector
5	`document.querySelectorAll(selector)`	Lấy một NodeList tĩnh khớp với selector

Ví dụ:

```
<body>
  <h1 id="title">Hello, world!</h1>
  <div class="content">
    <p>This is a paragraph with <span class="highlight">highlighted</span> text.
</p>
    <p class="highlight">This paragraph is also highlighted.</p>
  </div>

  <script>
    // Lấy phần tử có id là "title"
    const titleElement = document.getElementById('title');
    console.log('Title Element:', titleElement);

    // Lấy tất cả các phần tử có class là "highlight"
    const highlightElements = document.getElementsByClassName('highlight');
    console.log('Highlight Elements:', highlightElements);

    // Lấy tất cả các phần tử <p>
    const paragraphElements = document.getElementsByTagName('p');
    console.log('Paragraph Elements:', paragraphElements);

    // Lấy phần tử đầu tiên có class là "highlight" sử dụng selector CSS
    const firstHighlightElement = document.querySelector('.highlight');
    console.log('First Highlight Element:', firstHighlightElement);

    // Lấy tất cả các phần tử có class là "highlight" sử dụng selector CSS
    const allHighlightElements = document.querySelectorAll('.highlight');
    console.log('All Highlight Elements:', allHighlightElements);
  </script>
</body>
```

Create

#	Name	Description
1	`document.createAttribute(name)`	Tạo và trả về một node Attr với tên được chỉ định
2	`document.createElement(tagName)`	Tạo và trả về một phần tử HTML với tagName được chỉ định

```
<body>
  <script>
    // Tạo một Attribute node
    const newAttribute = document.createAttribute('class');
    newAttribute.value = 'new-class';

    // Tạo một Element node
    const newElement = document.createElement('div');
    newElement.textContent = 'This is a new element';
    newElement.setAttributeNode(newAttribute); // Gắn attribute vào element

    // Thêm element vào body của tài liệu
    document.body.appendChild(newElement);
  </script>
</body>
```

2. Element interface

Properties

#	Name	Description
1	`Element.className`	Trả về chuỗi các lớp (classes) của phần tử
2	`Element.classList`	Đối tượng để quản lý danh sách các lớp của phần tử
3	`Element.innerHTML`	Nội dung HTML bên trong phần tử
4	`Element.outerHTML`	Đoạn mã HTML của phần tử cùng với nội dung bên trong
5	`Element.id`	ID của phần tử

ví dụ

```
 <div id="myDiv" class="content">
    <p id="myParagraph">This is a paragraph.</p>
  </div>
```

```
.highlight {
    color: red;
```

```
    font-weight: bold;
}
```

```
// Truy cập phần tử và lớp của nó
const divElement = document.getElementById('myDiv');
console.log('Class Name:', divElement.className);

// Thêm một lớp mới vào phần tử
divElement.classList.add('highlight');

// Truy cập và thay đổi nội dung HTML của phần tử
const paragraphElement = document.getElementById('myParagraph');
console.log('Inner HTML:', paragraphElement.innerHTML);
paragraphElement.innerHTML = 'This is a <span class="highlight">highlighted</span>
paragraph.';

// In ra đoạn mã HTML của phần tử
console.log('Outer HTML:', divElement.outerHTML);

// Truy cập và thay đổi ID của phần tử
divElement.id = 'newId';
console.log('New ID:', divElement.id);
```

Element size

#	Name	Description
1	`Element.clientHeight`	Chiều cao nội bộ của phần tử
2	`Element.clientTop`	Chiều rộng của viền trên của phần tử
3	`Element.clientWidth`	Chiều rộng nội bộ của phần tử
4	`Element.clientLeft`	Chiều rộng của viền trái của phần tử

```
<div id="myDiv">
    This is a div element.
</div>
```

```
 #myDiv {
    width: 200px;
    height: 100px;
    border: 2px solid black;
    padding: 10px;
    margin: 20px;
}
```

```
// Truy cập phần tử div
const divElement = document.getElementById('myDiv');

// Lấy và in ra kích thước nội bộ của phần tử (không tính viền và padding)
console.log('Client Height:', divElement.clientHeight);
console.log('Client Width:', divElement.clientWidth);

// Lấy và in ra kích thước của viền trên và viền trái của phần tử
console.log('Client Top:', divElement.clientTop);
console.log('Client Left:', divElement.clientLeft);
```

Scroll

#	Name	Description
1	`Element.scrollHeight`	Chiều cao của khu vực cuộn của phần tử
2	`Element.scrollTop`	Số lượng pixel mà đỉnh của tài liệu được cuộn theo chiều dọc
3	`Element.scrollLeft`	Lệch cuộn bên trái của phần tử
4	`Element.scrollWidth`	Chiều rộng của khu vực cuộn của phần tử

```
#container {
    width: 200px;
    height: 100px;
    border: 1px solid black;
    overflow: auto; /* Enable scrolling */
}
#content {
    width: 400px; /* Content width is larger than container */
    height: 400px; /* Content height is larger than container */
    background-color: #f0f0f0;
}
```

```
<div id="container">
    <div id="content">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin non finibus magna. Integer maximus vehicula turpis nec laoreet. Cras non arcu eget elit varius bibendum. Duis posuere ante id eros accumsan, in bibendum neque rhoncus. Vivamus porta sapien non purus laoreet, at lacinia dolor ultricies. Sed vestibulum bibendum turpis nec efficitur.
    </div>
</div>
```

```
// Truy cập phần tử container
const container = document.getElementById('container');

// In ra chiều cao của khu vực cuộn của phần tử
console.log('Scroll Height:', container.scrollHeight);

// In ra số lượng pixel mà đỉnh của tài liệu được cuộn theo chiều dọc
console.log('Scroll Top:', container.scrollTop);

// In ra lệch cuộn bên trái của phần tử
console.log('Scroll Left:', container.scrollLeft);

// In ra chiều rộng của khu vực cuộn của phần tử
console.log('Scroll Width:', container.scrollWidth);
```

Traversal

#	Name	Description
1	`Element.children`	Trả về một HTMLCollection trực tiếp chứa tất cả các phần tử con trực tiếp.
2	`Element.firstElementChild`	Trả về phần tử con đầu tiên của một phần tử.
3	`Element.lastElementChild`	Trả về phần tử con cuối cùng của một phần tử.
4	`Element.previousElementSibling`	Trả về phần tử anh em trước đó, hoặc null nếu nó là phần tử con đầu tiên.
5	`Element.nextElementSibling`	Trả về phần tử anh em kế tiếp, hoặc null nếu nó là phần tử con cuối cùng.

```
<div id="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
</div>
```

```
.highlight {
    background-color: yellow;
}
```

```
// Truy cập phần tử container
const container = document.getElementById('container');

// Lấy tất cả các phần tử con trực tiếp và thêm lớp 'highlight' cho từng phần tử
```

```
con
const children = container.children;
for (let i = 0; i < children.length; i++) {
    children[i].classList.add('highlight');
}

// Lấy phần tử con đầu tiên và thêm lớp 'first-child'
const firstChild = container.firstElementChild;
firstChild.classList.add('first-child');

// Lấy phần tử con cuối cùng và thêm lớp 'last-child'
const lastChild = container.lastElementChild;
lastChild.classList.add('last-child');

// Lấy phần tử anh em trước đó của phần tử cuối cùng và thêm lớp 'previous-
sibling'
const previousSibling = lastChild.previousElementSibling;
if (previousSibling) {
    previousSibling.classList.add('previous-sibling');
}

// Lấy phần tử anh em kế tiếp của phần tử đầu tiên và thêm lớp 'next-sibling'
const nextSibling = firstChild.nextElementSibling;
if (nextSibling) {
    nextSibling.classList.add('next-sibling');
}
```

Methods

Add/Remove/Replace

#	Name	Description
1	`Element.before(...nodes)`	Chèn một tập hợp các node trước phần tử hiện tại.
2	`Element.after(...nodes)`	Chèn một tập hợp các node sau phần tử hiện tại.
3	`Element.prepend(...nodes)`	Chèn một tập hợp các node trước phần tử con đầu tiên.
4	`Element.append(...nodes)`	Chèn một tập hợp các node sau phần tử con cuối cùng.
5	`Element.remove()`	Xóa phần tử hiện tại khỏi cây DOM.
6	`Element.replaceWith(...nodes)`	Thay thế phần tử hiện tại bằng một tập hợp các node mới.

```
<div id="container">
    <div id="first">First</div>
    <div id="second">Second</div>
    <div id="third">Third</div>
</div>
```

```
// Truy cập các phần tử trong DOM
const container = document.getElementById('container');
const first = document.getElementById('first');
const second = document.getElementById('second');
const third = document.getElementById('third');

// Chèn một phần tử trước phần tử "first"
const newElementBefore = document.createElement('div');
newElementBefore.textContent = 'Before First';
first.before(newElementBefore);

// Chèn một phần tử sau phần tử "third"
const newElementAfter = document.createElement('div');
newElementAfter.textContent = 'After Third';
third.after(newElementAfter);

// Chèn một phần tử vào trước phần tử con đầu tiên của "container"
const newElementPrepend = document.createElement('div');
newElementPrepend.textContent = 'Prepend';
container.prepend(newElementPrepend);

// Chèn một phần tử vào sau phần tử con cuối cùng của "container"
const newElementAppend = document.createElement('div');
newElementAppend.textContent = 'Append';
container.append(newElementAppend);

// Loại bỏ phần tử "second" khỏi DOM
second.remove();

// Thay thế phần tử "third" bằng một phần tử mới
const newElementReplace = document.createElement('div');
newElementReplace.textContent = 'Replaced Third';
third.replaceWith(newElementReplace);
```

Attributes

#	Name	Description
1	`Element.getAttribute(name)`	Lấy giá trị của một thuộc tính theo tên.
2	`Element.hasAttribute(name)`	Kiểm tra xem một thuộc tính có tồn tại hay không.
3	`Element.setAttribute(name, value)`	Thiết lập giá trị cho một thuộc tính theo tên.
4	`Element.removeAttribute(name)`	Xóa một thuộc tính theo tên.
5	`Element.toggleAttribute(name)`	Chuyển đổi trạng thái của một thuộc tính.
6	`Element.attributes`	Danh sách trực tiếp tất cả các thuộc tính của phần tử

```
<div id="container" class="box" data-info="example">
<p id="paragraph" style="color: red;">Lorem ipsum dolor sit amet.</p>
```

```
</div>
```

```
// Truy cập các phần tử trong DOM
const container = document.getElementById('container');
const paragraph = document.getElementById('paragraph');

// Lấy giá trị của thuộc tính 'class' của phần tử "container"
const containerClass = container.getAttribute('class');
console.log('Container class:', containerClass);

// Kiểm tra xem phần tử "container" có thuộc tính 'data-info' hay không
const hasDataInfo = container.hasAttribute('data-info');
console.log('Container has data-info attribute:', hasDataInfo);

// Thiết lập giá trị của thuộc tính 'title' cho phần tử "container"
container.setAttribute('title', 'Container Element');
console.log('Container title attribute:', container.getAttribute('title'));

// Xóa thuộc tính 'style' khỏi phần tử "paragraph"
paragraph.removeAttribute('style');

// Chuyển đổi trạng thái của thuộc tính 'disabled' của phần tử "container"
container.toggleAttribute('disabled');
console.log('Container disabled attribute:', container.hasAttribute('disabled'));

// Lấy tất cả các thuộc tính của phần tử "container"
const containerAttributes = container.attributes;
console.log('Container attributes:');
for (let attribute of containerAttributes) {
    console.log(attribute.name, '=', attribute.value);
}
```

Query elements

#	Name	Description
1	`Element.getElementsByClassName(name)`	Trả về một NodeList trực tiếp chứa tất cả các phần tử con trùng với tên lớp.
2	`Element.getElementsByTagName(name)`	Trả về một NodeList trực tiếp chứa tất cả các phần tử con trùng với tên thẻ.
3	`Element.querySelector(selector)`	Trả về phần tử đầu tiên trong phần tử hiện tại mà phù hợp với trình chọn CSS.
4	`Element.querySelectorAll(selector)`	Trả về một NodeList tĩnh chứa tất cả các phần tử con của phần tử hiện tại mà phù hợp với trình chọn CSS.

ví dụ

```
.highlight {
    background-color: yellow;
}
```

```
<div class="container">
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <div id="target">Target Element</div>
</div>
```

```
// Truy cập phần tử container
const container = document.querySelector('.container');

// Lấy tất cả các phần tử con có lớp 'item' và làm nổi bật chúng
const itemsByClass = container.getElementsByClassName('item');
for (let item of itemsByClass) {
    item.classList.add('highlight');
}

// Lấy tất cả các phần tử con là thẻ <div> và làm nổi bật chúng
const divsByTagName = container.getElementsByTagName('div');
for (let div of divsByTagName) {
    div.classList.add('highlight');
}

// Lấy phần tử có id là 'target' và làm nổi bật chúng
const targetElement = document.querySelector('#target');
targetElement.classList.add('highlight');

// Lấy tất cả các phần tử con có class 'item' bằng cách sử dụng querySelectorAll
và làm nổi bật chúng
const itemsByQuery = document.querySelectorAll('.container .item');
itemsByQuery.forEach(item => {
    item.classList.add('highlight');
});
```

Others

#	Name	Description
1	`Element.getBoundingClientRect()`	Trả về một đối tượng DOMRect cung cấp thông tin về kích thước và vị trí của phần tử.
2	`Element.scroll()`	Cuộn phần tử đến một tập hợp cụ thể các tọa độ.
3	`Element.scrollTo()`	Cuộn phần tử đến một tập hợp cụ thể các tọa độ.

#	Name	Description
4	`Element.scrollBy()`	Cuộn một phần tử theo một lượng cụ thể được chỉ định.

```css
#container {
    height: 2000px;
    background-color: #f0f0f0;
}
#target {
    width: 100px;
    height: 100px;
    background-color: #ff0000;
    position: absolute;
}
```

```html
<div id="container">
    <div id="target"></div>
</div>
```

```javascript
// Lấy phần tử "target"
const target = document.getElementById('target');

// Lấy thông tin về kích thước và vị trí của phần tử "target"
const rect = target.getBoundingClientRect();
console.log('Bounding rect:', rect);

// Cuộn phần tử "container" đến vị trí của phần tử "target"
document.getElementById('container').scroll({
    top: rect.top,
    behavior: 'smooth'
});

// Hoặc có thể sử dụng phương thức scrollTo() để cuộn
// document.getElementById('container').scrollTo({
//   top: rect.top,
//   behavior: 'smooth'
// });

// Hoặc có thể cuộn một lượng cụ thể bằng phương thức scrollBy()
// document.getElementById('container').scrollBy({
//   top: 100,
//   behavior: 'smooth'
// });
```

> Source https://developer.mozilla.org/en-US/docs/Web/API/Element

3. HTMLElement interface

Properties

#	Name	Description
1	`HTMLElement.dataset`	Đọc/Ghi dữ liệu tùy chỉnh từ các thuộc tính dữ liệu (data-*).
2	`HTMLElement.hidden`	Xác định xem phần tử có được ẩn đi không.
3	`HTMLElement.innerText`	Nội dung văn bản "được hiển thị" của một nút và các nút con của nó.
4	`HTMLElement.style`	Đại diện cho các khai báo của thuộc tính kiểu của phần tử.
5	`HTMLElement.title`	Văn bản hiển thị trong hộp popup khi di chuột qua phần tử.
6	`HTMLElement.tabIndex`	Vị trí của phần tử trong thứ tự tab.

```
<div id="myElement" data-info="example" title="Hover me!" tabindex="0">Hello,
World!</div>
```

```
.hidden {
    display: none;
}
```

```
// Lấy tham chiếu đến phần tử HTML
const element = document.getElementById('myElement');

// Đọc và ghi dữ liệu từ các thuộc tính dữ liệu tùy chỉnh (data-*)
console.log('Custom data attribute:', element.dataset.info);
element.dataset.newInfo = 'new value';

// Ẩn và hiện phần tử
setTimeout(() => {
    element.hidden = true;
}, 2000);
setTimeout(() => {
    element.hidden = false;
}, 4000);

// Thay đổi văn bản hiển thị
element.innerText = 'Goodbye, World!';

// Thay đổi kiểu dáng của phần tử
element.style.color = 'blue';
element.style.fontSize = '20px';

// Đặt tiêu đề cho phần tử
element.title = 'This is a div element';
```

```
// Đặt vị trí của phần tử trong thứ tự tab
element.tabIndex = 1;
```

Methods

#	Name	Description
1	`HTMLElement.blur()`	Loại bỏ trọng tâm bàn phím khỏi phần tử hiện tại.
2	`HTMLElement.focus()`	Đưa trọng tâm bàn phím vào phần tử hiện tại.
3	`HTMLElement.click()`	Gửi một sự kiện nhấp chuột đến phần tử.

```
<input type="text" id="myInput" value="Click me and focus!">
<button id="myButton">Click me</button>
```

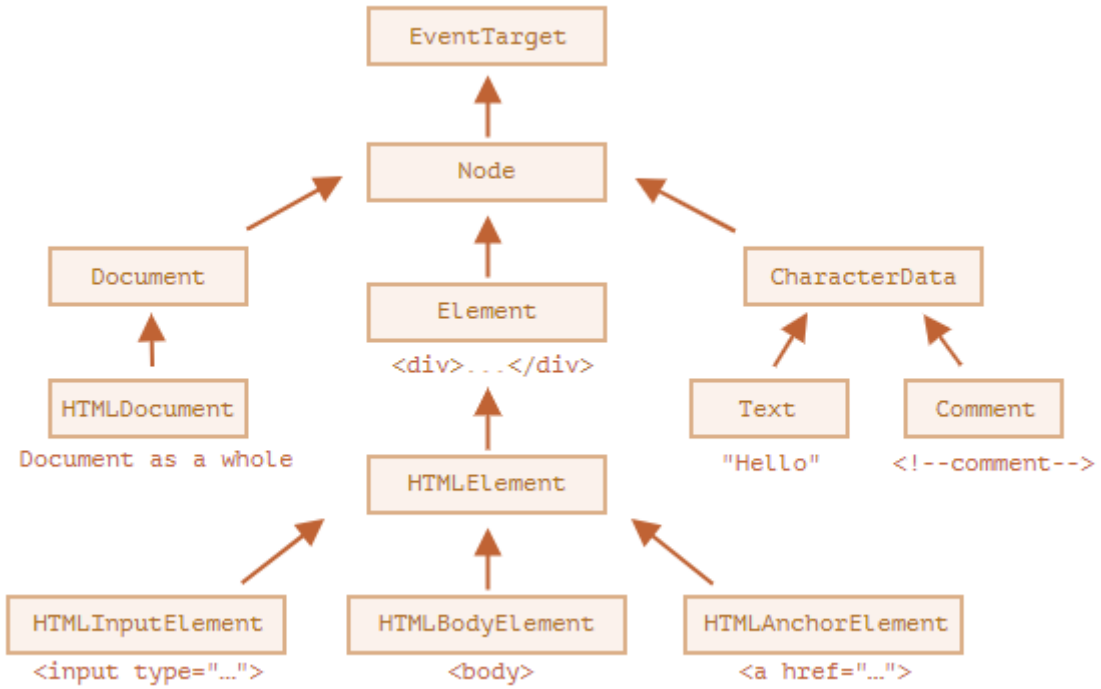
```
// Lấy tham chiếu đến các phần tử HTML
const inputElement = document.getElementById('myInput');
const buttonElement = document.getElementById('myButton');

// Blur phần tử input sau 2 giây
setTimeout(() => {
    inputElement.blur();
}, 2000);

// Focus vào phần tử input khi trang được tải
window.onload = function() {
    inputElement.focus();
};

// Gửi sự kiện click đến phần tử button khi click vào input
inputElement.onclick = function() {
    buttonElement.click();
};

// Thay đổi giá trị của phần tử input khi click vào button
buttonElement.onclick = function() {
    inputElement.value = 'Button clicked!';
};
```



> Source: https://javascript.info/basic-dom-node-properties

Tham khảo

- https://javascript.info/basic-dom-node-properties
- https://developer.mozilla.org/en-US/docs/Web/API/Element