



DOM Basic p1

Trần Huy Hoàng

Email: huyhoang.tran6669@gmail.com

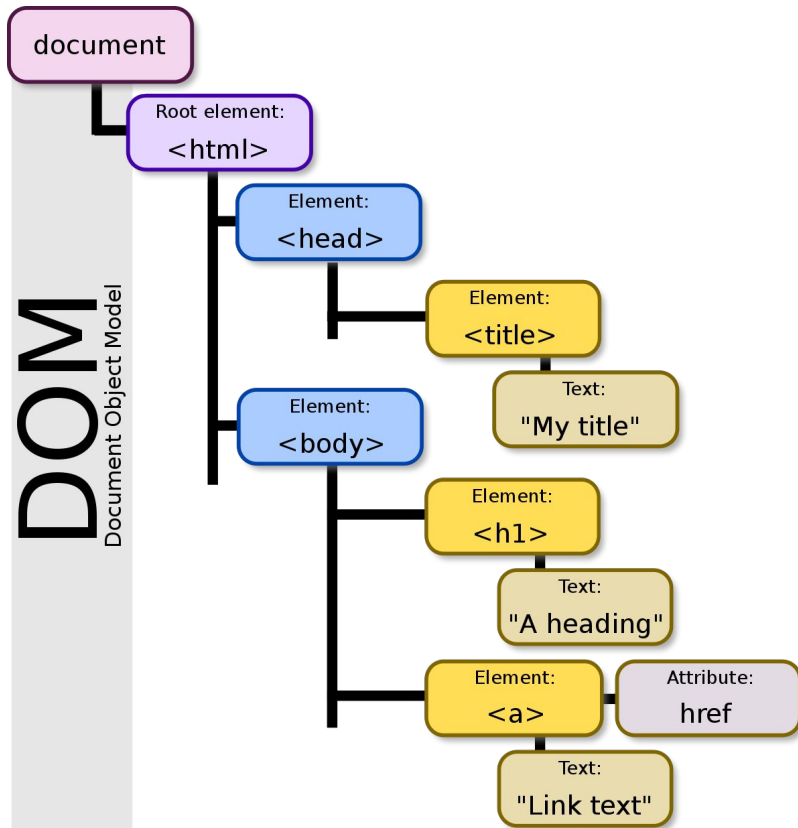
Phone: 0788.719.666

Nội dung



1. Giới thiệu về DOM
2. Node trong DOM
3. Các thao tác với phần tử DOM
 - 3.1. Truy cập vào một phần tử
 - 3.2. Lấy nội dung của một phần tử
 - 3.3. Chỉnh sửa nội dung của phần tử
 - 3.4. Thay đổi các thuộc tính của phần tử
 - 3.5. Lab1: Tạo Random màu (HexCode)
4. Thêm một phần tử DOM
5. Thay thế một phần tử DOM
6. Xóa phần tử DOM
7. CSS, Class List
8. Lab2: Thêm class, xóa class khỏi 1 phần tử DOM
9. Homeworks

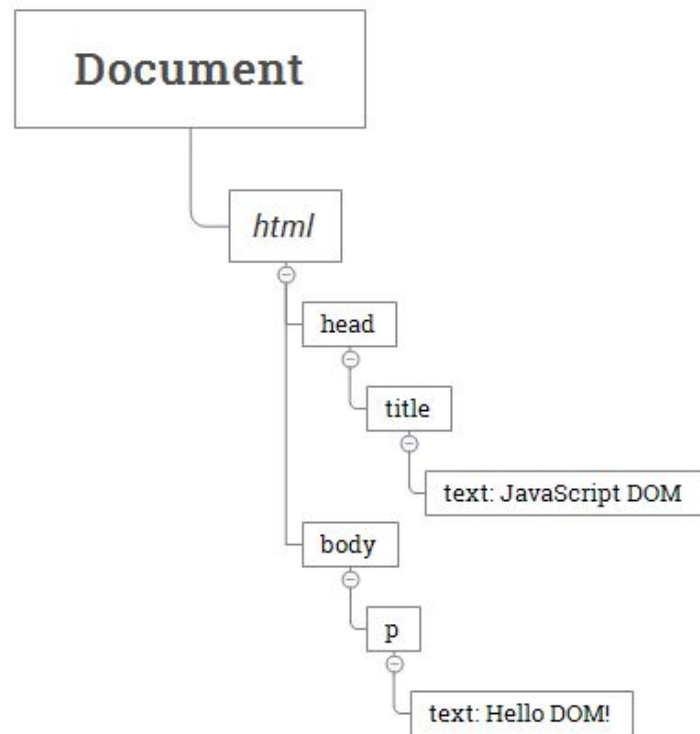
1. Giới thiệu về DOM



- ❖ **Document Object Model (DOM)** được tạo ra bởi trình duyệt khi trang web được tải
- ❖ **DOM** được tổ chức theo dạng cây (**DOM tree**), mỗi thành phần trên cây gọi là một **node**

DOM đại diện cho một tài liệu HTML dưới dạng cây các nút cho phép bạn **thêm, sửa, xóa** các thành phần của tài liệu một cách hiệu quả

1. Giới thiệu về DOM



Ví dụ về DOM Tree

```
<html>
  <head>
    <title>JavaScript DOM</title>
  </head>
  <body>
    <p>Hello DOM!</p>
  </body>
</html>
```

2. Node trong DOM

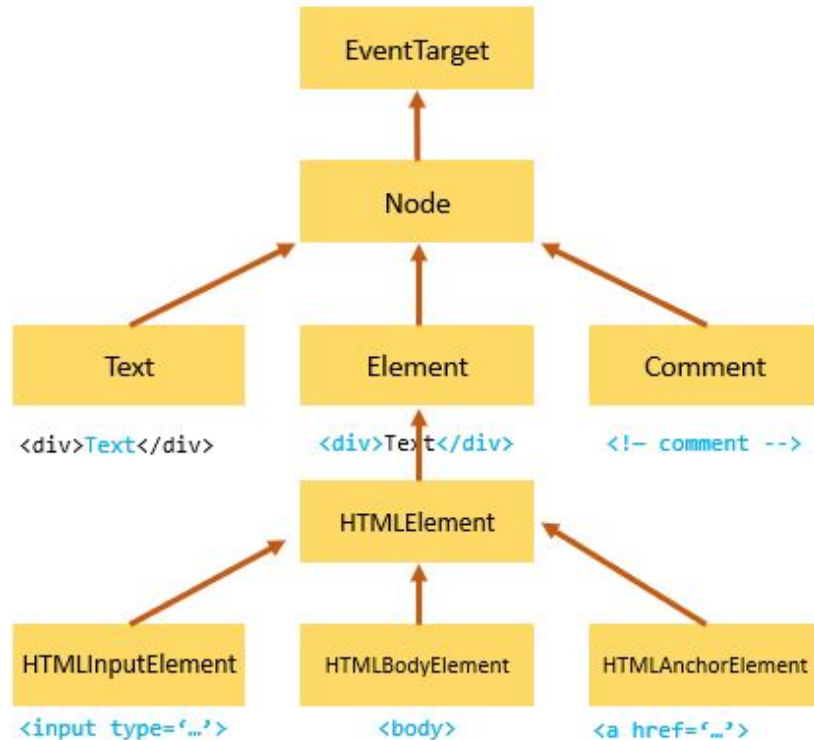
Cấu tạo của một Node trong DOM Tree

Node:

- ❖ **nodeName:** # thường có giá trị giống tag name
- ❖ **nodeValue:** # mặc định thường là null

```
if (node.nodeType == Node.ELEMENT_NODE) {  
    let name = node.nodeName; // tag name like <p>  
}
```

2. Node trong DOM



1. Giới thiệu về DOM

Bảng giá trị của các loại Node được quy định

Constant	Value	Description
Node.ELEMENT_NODE	1	An Element node like <p> or <div>.
Node.TEXT_NODE	3	The actual Text inside an Element or Attr.
Node.CDATA_SECTION_NODE	4	A CDATASection, such as <![CDATA[[...]]>.
Node.PROCESSING_INSTRUCTION_NODE	7	A ProcessingInstruction of an XML document, such as <?xml-stylesheet ... ?>.
Node.COMMENT_NODE	8	A Comment node, such as <!-- ... -->.
Node.DOCUMENT_NODE	9	A Document node.
Node.DOCUMENT_TYPE_NODE	10	A DocumentType node, such as <!DOCTYPE html>.
Node.DOCUMENT_FRAGMENT_NODE	11	A DocumentFragment node.

`node.nodeType`

2. Node trong DOM



Bởi vì các node trong DOM tree có mối quan hệ với nhau, thay vì truy cập trực tiếp chúng ta có thể truy cập gián tiếp thông qua mối quan hệ này.

- **Node.parentNode:** tham chiếu đến nút cha của nút hiện tại
- **Node.childNodes:** tham chiếu đến các nút con trực tiếp của nút hiện tại
- **Node.firstChild:** tham chiếu đến nút con đầu tiên của nút hiện tại
- **Node.lastChild:** tham chiếu đến nút con cuối cùng của nút hiện tại
- **Node.nextSibling:** tham chiếu đến nút anh em nằm liền kề sau với nút hiện tại.
- **Node.previousSibling:** tham chiếu đến nút anh em nằm liền kề trước với nút hiện tại.

3.1 Truy cập vào một phần tử

● **document.getElementById:** Truy cập thông qua ID

```
<script>
  let color_1_element = document.getElementById("color_1");
  console.log(color_1_element);
</script>
```

```
<div class="grid grid-cols-6 gap-0">
  <div id="color_1" class="color_review h-96 bg-red-400 relative">
    <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
  </div>
</div>
```

```
▼ <div id="color_1" class="color_review h-96 bg-red-400 relative">
  <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
</div>
```

3.1 Truy cập vào một phần tử

- **document.getElementsByTagName:** Truy cập thông qua Tag , trả về danh sách các phần tử

```
<div class="mt-16 ">
  <div class="grid grid-cols-6 gap-0">
    <div id="color_1" class="color_review h-96 bg-red-400 relative">
      <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
    </div>
    <div id="color_2" class="color_review h-96 bg-red-400 relative">
      <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
    </div>
    <div id="color_3" class="color_review h-96 bg-red-400 relative">
      <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
    </div>
    <div id="color_4" class="color_review h-96 bg-red-400 relative">
      <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
    </div>
    <div id="color_5" class="color_review h-96 bg-red-400 relative">
      <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
    </div>
    <div id="color_6" class="color_review h-96 bg-red-400 relative">
      <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
    </div>
  </div>
</div>
```

```
let p_tags = document.getElementsByTagName("p");
console.log(p_tags);
```

3.1 Truy cập vào một phần tử

```
HTMLCollection(6) [p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gr
▼ p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700, p.absolute.
  p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700, p.absolute.
    ► 0: p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700
    ► 1: p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700
    ► 2: p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700
    ► 3: p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700
    ► 4: p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700
    ► 5: p.absolute.bottom-0.text-center.w-full.font-bold.text-2xl.mb-8.text-gray-700
      length: 6
    ► [[Prototype]]: HTMLCollection
```

3.1 Truy cập vào một phần tử

- **document.getElementsByClassName:** Truy cập thông qua tên Class , trả về danh sách các phần tử

```
let color_review_items = document.getElementsByClassName("color_review");  
console.log(color_review_items)
```

```
HTMLCollection(6) [div#color_1.color_review.h-96.bg-red-400.relative, c  
▼, div#color_5.color_review.h-96.bg-red-400.relative, div#color_6.color  
div#color_2.color_review.h-96.bg-red-400.relative, color_3: div#color_3  
div#color_5.color_review.h-96.bg-red-400.relative, ...] ⓘ  
  ▶ 0: div#color_1.color_review.h-96.bg-red-400.relative  
  ▶ 1: div#color_2.color_review.h-96.bg-red-400.relative  
  ▶ 2: div#color_3.color_review.h-96.bg-red-400.relative  
  ▶ 3: div#color_4.color_review.h-96.bg-red-400.relative  
  ▶ 4: div#color_5.color_review.h-96.bg-red-400.relative  
  ▶ 5: div#color_6.color_review.h-96.bg-red-400.relative  
  ▶ color_1: div#color_1.color_review.h-96.bg-red-400.relative  
  ▶ color_2: div#color_2.color_review.h-96.bg-red-400.relative  
  ▶ color_3: div#color_3.color_review.h-96.bg-red-400.relative  
  ▶ color_4: div#color_4.color_review.h-96.bg-red-400.relative  
  ▶ color_5: div#color_5.color_review.h-96.bg-red-400.relative  
  ▶ color_6: div#color_6.color_review.h-96.bg-red-400.relative  
    length: 6  
  ▶ [[Prototype]]: HTMLCollection
```


3.1 Truy cập vào một phần tử

- **document.getElementById:** Truy cập thông qua ID
- **document.getElementsByTagName:** Truy cập thông qua Tag , trả về danh sách các phần tử
- **document.getElementsByClassName:** Truy cập thông qua tên Class , trả về danh sách các phần tử
- **document.querySelector:** Truy cập thông qua CSS Selector, trả về phần tử đầu tiên tìm thấy
- **document.querySelectorAll:** Truy cập thông qua CSS Selector, trả về danh sách các phần tử

3.2 Lấy nội dung của một phần tử

👉 Node.innerHTML

❖ Thao tác innerHTML sẽ trả về mã code HTML của node đó

👉 Node.innerText

❖ Thao tác innerText sẽ trả về text bên trong của node đó

👉 Node.textContent

❖ Thao tác textContent sẽ trả về text bên trong của node đó (phần hiển thị ra ngoài)

```
let color_2_element = document.getElementById("color_2");
console.log(color_2_element.innerHTML);
console.log(color_2_element.innerText);
console.log(color_2_element.textContent);
```

3.2 Lấy nội dung của một phần tử

```
let color_2_element = document.getElementById("color_2");  
console.log(color_2_element.innerHTML);  
console.log(color_2_element.innerText);  
console.log(color_2_element.textContent);
```

```
<p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
```

#217262

#217262

3.3 Chỉnh sửa nội dung của phần tử



- `Node.innerHTML = value` HTML String '`<div>abc</div>`'
- `Node.innerText = value` TEXT String "xyz"
- `Node.textContent = value` TEXT String "xyz"

3.4 Thay đổi các thuộc tính của phần tử

Phần tử DOM được chọn

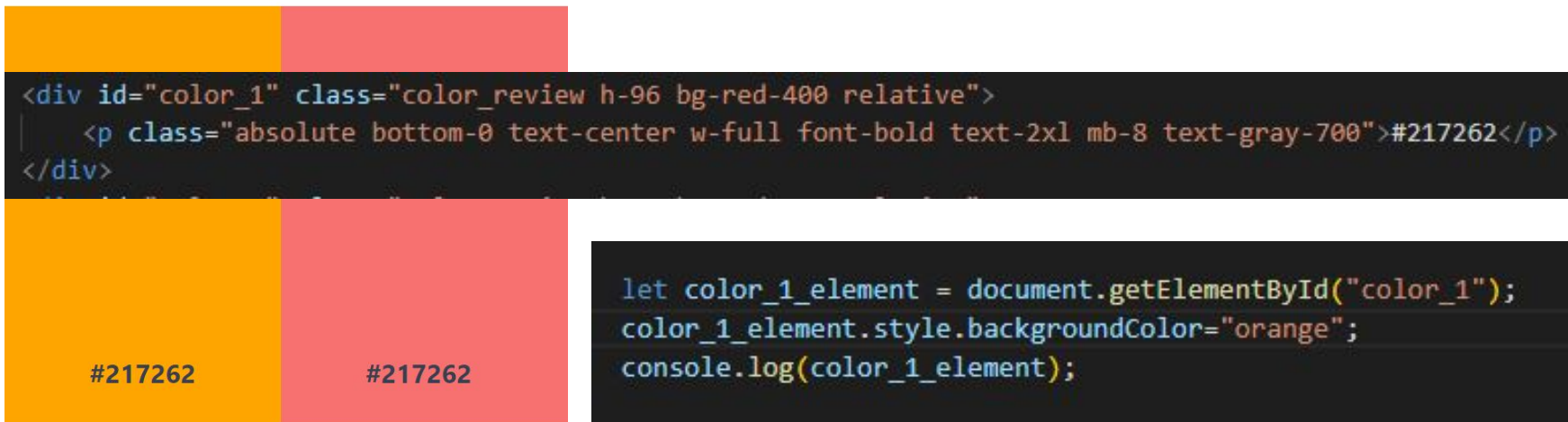
style mới cần thêm

element.style.property = new style

**Property có 2 từ viết theo dạng camelCase.
Ví dụ: marginLeft, paddingLeft**

Để thay đổi style của 1 phần tử

3.4 Thay đổi các thuộc tính của phần tử



```
<div id="color_1" class="color_review h-96 bg-red-400 relative">
|   <p class="absolute bottom-0 text-center w-full font-bold text-2xl mb-8 text-gray-700">#217262</p>
| </div>
```

```
let color_1_element = document.getElementById("color_1");
color_1_element.style.backgroundColor="orange";
console.log(color_1_element);
```

Chú ý: Khi property có 2 từ trở lên cần viết theo dạng camelCase

Ví dụ: backgroundColor, marginLeft, paddingLeft,...

3.5 Lab1: Tạo Random màu (HexCode)

Viết 1 file html để thực hiện chức năng như hình vẽ

--- Random Generate Color ---

Generate

#E2C28C

#B2F511

#E0AA06

#B52F67

#9F2AAE

#40793C

3.5 Lab1: Tạo Random màu (HexCode)

Hàm tạo mã hex random

```
const hex_characters = '0123456789ABCDEF'
function generate_hex_code(){
  let hexCode = '#'
  for (let i=0; i<6; i++){
    hexCode += hex_characters[Math.floor(Math.random()*16)]
  }
  return hexCode
}
```

4. Thêm một phần tử DOM



Để thêm phần tử vào DOM, chúng ta sẽ sử dụng 1 số phương thức:

- **appendChild()** : Thêm phần tử vào cuối của phần tử cha
- **prepend()**: Thêm phần tử vào đầu của phần tử cha
- **insertBefore()** : Thêm phần tử vào trong phần tử cha và trước phần tử được chỉ được

4. Thêm một phần tử DOM

```
<body>
  <h1 id="text">Xin chào các bạn</h1>
  <div class="box">
    <p>Box item</p>
  </div>

  <script>
    const para2 = document.createElement('p');
    para2.innerText = "New para 2";
    document.body.appendChild(para2);
  </script>
</body>
```

Mã HTML

Phương thức tạo phần tử HTML.

Gán nội dung cho thẻ p

Thêm thẻ p mới tạo vào phần cuối của thẻ body



Xin chào các bạn

Box item

New para 2

Kết quả

Thêm phần tử vào cuối của phần tử cha

4. Thêm một phần tử DOM

```
<body>
  <h1 id="text">Xin chào các bạn</h1>
  <div class="box">
    <p>Box item</p>
  </div>
  <script>
    Phương thức tạo phần tử HTML
    ↓
    const para = document.createElement('p');
    para.innerText = "New para 1";
    document.body.prepend(para);
  </script>
</body>
```

Mã HTML

Gán nội dung "New para 1" cho thẻ p

Thêm thẻ p mới tạo vào phần đầu của thẻ body



New para 1

Xin chào các bạn

Box item

Kết quả

Thêm phần tử vào đầu của phần tử cha

4. Thêm một phần tử DOM

```
<h1 id="text">Xin chào các bạn</h1>

<div class="box">
  <p>Box item</p>
</div>

<script>

  const heading = document.createElement('h2');
  heading.innerHTML = "Các bạn có khỏe không?";

  const box = document.querySelector('.box');
  document.body.insertBefore(heading, box);

</script>
</body>
```

Chèn thẻ mới (heading) vào trước class "box" nằm trong body.



Xin chào các bạn

Các bạn có khỏe không?

Box item

Kết quả

Thêm phần tử vào **trong phần tử cha** và **trước phần tử được chỉ định**. Xác định mình sẽ thêm vào vị trí nào: Ở đây vị trí muốn thêm là trước class "box".

4. Thêm một phần tử DOM

Một số phương thức khác để thêm nội dung, phần tử DOM

- `targetElement.insertAdjacentHTML(position, text);`
- `targetElement.insertAdjacentElement(position, element);`
- `targetElement.insertAdjacentText(position, text);`

Trong đó:

- ***beforebegin***: Vị trí trước ***targetElement***.
- ***afterbegin***: Bên trong ***targetElement***, trước first child.
- ***beforeend***: Bên trong ***targetElement***, sau last child.
- ***afterend***: Vị trí trước ***targetElement***.

```
<!-- beforebegin -->
<p>
  <!-- afterbegin -->
  foo
  <!-- beforeend -->
</p>
<!-- afterend -->
```

5. Thay thế một phần tử DOM

```
<body>

<div id="parent">
  <p id="child">Thay thế phần tử trong JS DOM</p>
</div>

<script>
  const parent = document.getElementById('parent');
  const child = document.getElementById('child');

  let newElement = document.createElement('h1');
  newElement.innerText = "Xin chào các bạn";

  parent.replaceChild(newElement, child);

  child.parentNode.replaceChild(newElement, child);
</script>

</body>
```

Mã HTML

Truy cập phần tử cha có id = "parent" và phần tử con có id = "child".

Tạo thẻ h1 và chèn nội dung cho thẻ

C1: Từ phần tử cha, dùng method `replaceChild` để thay thế phần tử

C2: Sử dụng method `parentNode` để truy cập phần tử cha. Gọi method `replaceChild` để thay thế phần tử.



Xin chào các bạn

Kết quả

Thay thế phần tử DOM có id = "child"

6. Xóa phần tử DOM

```
<div id="parent">
  <p id="para1">Para 1</p>
  <p class="para2">Para 2</p>
  <p>Para 3</p>
  <p>Para 4</p>
  <p>Para 5</p>
</div>
```

Mã HTML

```
<script>
  const para1 = document.getElementById('para1');
  const para2 = document.querySelector('.para2');
  const parent = document.getElementById('parent');

  parent.removeChild(para1);
  parent.removeChild(para2);

  const para3 = document.querySelector('#parent p:nth-child(1)');
  para3.parentNode.removeChild(para3);
</script>
```

Truy cập thẻ p có id = "para1", id = "para2" và phần tử cha có id = "parent"

Gọi method `removeChild()` để xóa phần tử.

Có thể xóa phần tử mà không cần truy cập vào phần tử cha

Sử dụng method `parentNode()` để lấy ra cha của phần tử muốn xóa. Gọi method `removeChild()`

7. CSS, Class List



classList là thuộc tính read-only của một phần tử trả về một tập hợp các class CSS và cho phép chúng ta sử dụng một số phương thức để quản lý và cập nhật các class đó.

Thuộc tính `classList` cung cấp một số phương thức như:

👉 `add()`: Thêm các class được chỉ định

👉 `remove()`: Loại bỏ các class được chỉ định

👉 `contains()`: Kiểm tra xem class được chỉ định có tồn tại trên phần tử hay không

👉 `toggle()`: toggles class được chỉ định

7. CSS, Class List

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Class List</title>
  <style>
    .text-red {
      color: red;
    }
    .text-bold {
      font-weight: bold;
    }
    .text-big {
      font-size: 40px;
    }
    .text-center {
      text-align: center;
    }
  </style>
</head>
<body>
  <h1 class="text-red">Truy cập phần tử DOM</h1>
</body>
</html>
```

Đầu tiên chúng ta sẽ truy cập vào phần tử muốn thao tác

```
const para = document.getElementById('para');
```

```
para.classList.add('text-big');
```

```
para.classList.add('text-big', 'text-bold');
```

```
para.classList.remove('text-big');
```

```
para.classList.remove('text-big', 'text-bold');
```

```
para.classList.contains('text-red'); // true
para.classList.contains('text-big'); // false
```

8. Lab2

Viết 1 file html để thực hiện chức năng khi click vào nút “Add Random Color” sẽ thêm 1 màu vào cuối danh sách

--- Random Generate Color ---

Add Random Color

#6F8BAB

#E3B98F

#792474

#78D1BD

#64F64F

#02522D

#E29C10

#E9EC8E

9. Homeworks

Bài 1: Tạo 1 thẻ p có **id="text"**, có nội dung bất kỳ (có thể tạo bằng HTML hay Javascript - tùy chọn). Yêu cầu

- ◆ Đặt màu văn bản thành **#777**
- ◆ Đặt kích thước phông chữ thành **2rem**
- ◆ Đặt nội dung HTML thành **"Tôi có thể làm bất cứ điều gì tôi muốn với JavaScript"**.

Bài 2: Sử dụng vòng lặp để đặt màu chữ cho tất cả thẻ li thành màu **blue** (tạo thẻ *ul-li* bằng html)

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
```

9. Homeworks

Bài 1: Cho mã HTML có nội dung như sau (tạo thẻ ul-li bằng html):

```
<ul id="list">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
  <li>Item 6</li>
  <li>Item 7</li>
</ul>
```

Sử dụng javascript để thực hiện những công việc sau:

- Thêm 3 thẻ **** có nội dung **Item 8**, **Item 9**, **Item 10** vào cuối danh sách
- Sửa nội dung cho thẻ ** 1** thành màu đỏ (color)
- Sửa background cho thẻ ** 3** thành màu xanh (background-color)
- Remove thẻ ** 4**

The End