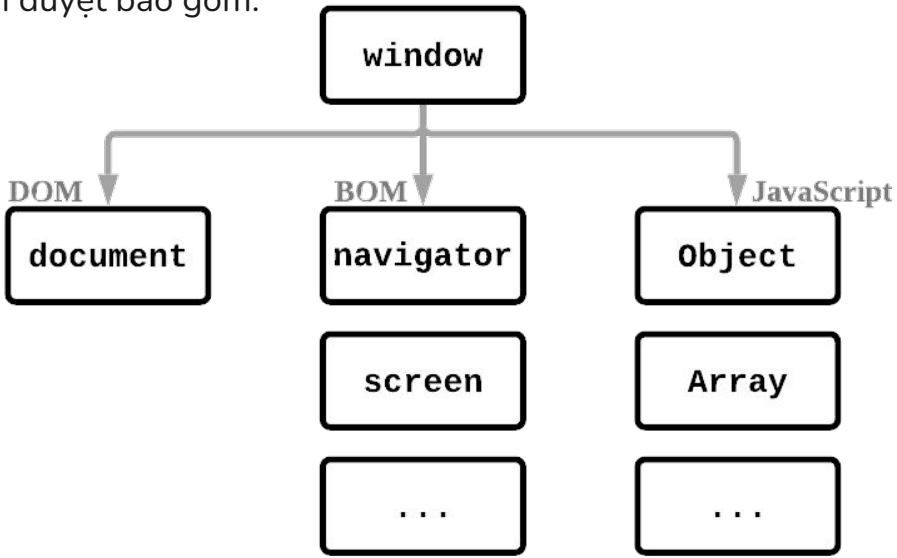
DOM

Ba Nguyễn

Introduction

JavaScript có thể chạy trên nhiều nền tảng / môi trường khác nhau (trình duyệt, máy chủ, ...). Mỗi môi trường cung cấp các chức năng riêng cho nó và ngôn ngữ JavaScript.

Trong môi trường trình duyệt bao gồm:



Concepts

window là đối tượng toàn cục - **globalThis** hay đối tượng gốc, đại diện cho cửa sổ trình duyệt

DOM (Document Object Model) là đối tượng đại diện cho toàn bộ nội dung trên trang, các thao tác thay đổi, thêm xóa các nội dung trên trang với JavaScript được thực hiện thông qua **DOM**

BOM (Browser Object Model) đại diện cho các đối tượng khác được cung cấp bởi trình duyệt, các đối tượng **BOM** cung cấp các phương thức quản lý/điều khiển trình duyệt

```
document.body.style.backgroundColor = "red";
```

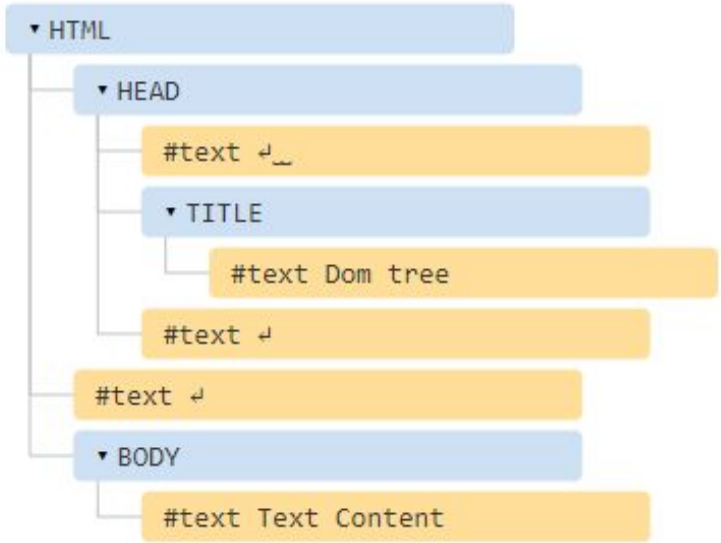
```
location.href = "https://google.com/";
```

💡 Lưu ý đặt thẻ **<script></script>** ở cuối hoặc thêm thuộc tính **defer**

DOM

Trong DOM, mọi thứ trong HTML đều là một **object** (hay một **node**), kể cả các thẻ rỗng, thẻ lồng nhau, comments, hay nội dung text bên trong các thẻ cũng là một object riêng biệt và đều có thể truy cập và chỉnh sửa thông qua DOM

```
<!DOCTYPE html>
<html>
    <head>
        <title>Dom tree</title>
    </head>
    <body>
        Text Content
    </body>
</html>
```



DOM Concepts

- **Node**: Mọi thứ trong HTML đều trở thành một **node** trong **DOM**, bất kể là phần tử HTML, hay text bên trong, dấu xuống dòng (dấu cách) giữa các phần tử, comment, ...
- **Child nodes**: Chỉ bao gồm các **node là con trực tiếp**
- **Element** chỉ bao gồm các **node là phần tử HTML**
- **Document**, **node** và **element** cung cấp các thuộc tính và phương thức đặc biệt để thao tác với nội dung trên trang, **và tự động cập nhật khi một node được thêm, xóa khỏi trang**
- Các thuộc tính và phương thức của **document**: api/document
- Các thuộc tính và phương thức của **node**: api/node
- Các thuộc tính và phương thức của **element**: api/element
- Một số **element** đặc biệt như **form**, **input**, **table** cung cấp một số thuộc tính và phương thức dành riêng phù hợp, tham khảo thêm: web/api

Collection vs List

Các thuộc tính và phương thức trả về kết quả là một hợp các phần tử được đặt trong một **HTML Collection** hoặc một **Node List**.

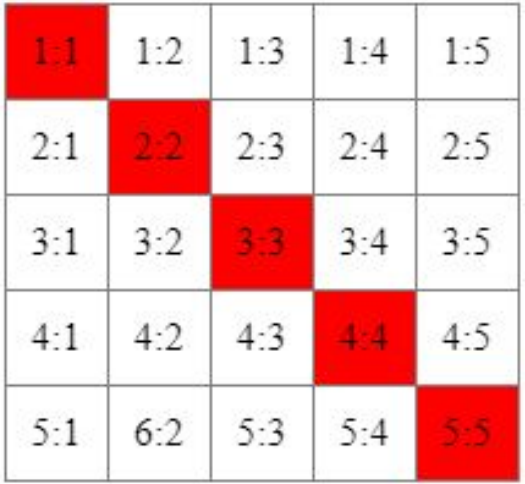
Các cấu trúc này không phải một mảng và không cung cấp các phương thức của mảng. Để chuyển đổi và dùng được các phương thức của mảng, sử dụng **Array.from(collection/list)**

```
let collection = document.body.children;

let elems = Array.from(collection);

elems.forEach(function (elem) {
    // code
});
```

Exercise

Tạo một bảng, sử dụng JS để thay đổi màu cho các ô theo đường chéo trong bảng (sử dụng **td.style.backgroundColor = “red”** để đổi màu nền)

💡 Tham khảo các thuộc tính dành riêng cho **table**: table, row



Searching

Để tìm kiếm / lựa chọn một phần tử bất kỳ trên trang, **DOM** cung cấp các phương thức:

```
// Tìm kiếm và trả về phần tử theo id
document.getElementById("id");

// Tìm kiếm phần tử đầu tiên khớp với bộ chọn CSS
document.querySelector(".css-selector");
// Tìm kiếm TẤT CẢ phần tử khớp với bộ chọn CSS
document.querySelectorAll(".css-selector");
```

💡 **getElementBy*** tự cập nhật theo DOM (live collection) còn **querySelector*** thì không (static collection)

HTML Attributes vs DOM Properties

Hầu hết các HTML attributes **tiêu chuẩn** sẽ được chuyển đổi thành thuộc tính **DOM** tương ứng

```
<p id="para" title="title" class="para" type="button">
    Content
</p>

<script>
    let p = document.getElementById(p);
    p.id; // para
    p.title; // title
    p.className; // para
    p.type; // undefined
</script>
```

HTML Attributes vs DOM Properties

Attributes và Propeties được **đồng bộ** với nhau, khi một attribute trong HTML thay đổi, property tương ứng trong **DOM** cũng thay đổi theo, và ngược lại (trừ một số ít trường hợp, VD: **input.value**)

💡 Giá trị của các thuộc tính cũng có thể có kiểu dữ liệu khác nhau

Để gắn các thuộc tính tùy chỉnh vào **DOM**, thêm cho các thuộc tính tiền tố **data-**

```
<p id="para" data-type="type">Content</p>

<script>
    let p = document.getElementById("para");

    p.dataset.type; // type
</script>
```

DOM Contents

Ngoài các properties tương ứng với attributes, các **element** còn có các properties khác để thao tác với nội dung trong nó như văn bản, mã HTML

```
// ...
p.textContent; // Content
p.textContent = "LoL"; // Thay đổi nội dung

p.innerHTML; // LoL
// Thay thế nội dung HTML trong thẻ p
p.innerHTML = "<i>Replace</i>";
// Thêm nội dung HTML vào trong thẻ p
p.innerHTML += "<b>Append</i>";
```

Modifying the document

Ngoài các thao tác với những node/element có sẵn, **DOM** cũng cung cấp các phương thức cho phép tạo mới và chèn/di chuyển/xóa các node/element từ JavaScript

```
// Tạo một phần tử HTML
// document.createElement("tagname")

let img = document.createElement("img");
img.src = "images/demo.png";
img.alt = "Anh demo";

let txt = document.createTextNode("Content");
```

Modifying the document

Các phương thức để thêm một node vào **DOM**

```
<!-- ul.before() -->
<ul>
    <!-- ul.prepend() -->
    <li></li>
    <!-- ul.append() -->
</ul>
<!-- ul.after() -->
```

💡 Chuỗi không được phân tích thành HTML như **innerHTML**

💡 Nếu **node** đã tồn tại trong **DOM**, các phương thức sẽ di chuyển vị trí của **node** thay vì chèn thêm

Modifying the document

💡 Để thêm nhiều node giống nhau, có thể sử dụng phương thức **el.cloneNode(true)** để sao chép một node và thêm vào **DOM**

Xóa một **node** khỏi **DOM**

```
node.remove();
// hoặc
parent.removeChild(node);
```

Styles & Classes

Có 2 cách tạo kiểu CSS cho phần tử bằng JS

- Tạo một class trong CSS và thêm class vào phần tử
- Thêm trực tiếp CSS vào thuộc tính style (inline CSS)

Ưu tiên thêm class hơn thêm trực tiếp css. Các thuộc tính/phương thức **DOM** thường dùng:

```
elem.className; // Chuỗi class
elem.classList; // Collection
elem.classList.add("class"); // Thêm class
elem.classList.remove("class"); // Xóa class
elem.classList.toggle("class"); // Bật - tắt class
elem.classList.contains("class"); // Kiểm tra
```

Styles & Classes

Trong các trường hợp thay đổi trực tiếp giá trị thuộc tính CSS của phần tử, lưu ý:

- Đối với thuộc tính có dấu gạch nối, chuyển sang dạng **camelCase**
- Các thuộc tính có prefix chuyển sang dạng **capitalize**

```
el.style.width = "100px";
el.style.backgroundColor = "red";
el.style.borderRadius = "5px";
```

- Để xóa (reset) một thuộc tính, gán cho nó một giá trị rỗng
- Phải đặt đúng đơn vị (với những giá trị như **pixel**, **%**, **rem**, ...)
- Đặt Full CSS: **style.cssText** hoặc **elem.setAttribute("style", "...")**