Cơ bản về JS(2 tuần)

[I. DOM 2](#_Toc521575397)

[1. Định nghĩa 2](#_Toc521575398)

[2. Cây cấu trúc DOM 2](#_Toc521575399)

[2.1 Nút 2](#_Toc521575400)

[2.2 Quan hệ giữa các nút 2](#_Toc521575401)

[3. Thao tác với DOM 3](#_Toc521575402)

[3.1 Thuộc tính và các phương thức thường gặp 3](#_Toc521575403)

[3.2 Truy xuất 4](#_Toc521575404)

[3.2.1 Truy xuất trực tiếp 4](#_Toc521575405)

[3.2.2 Truy xuất trực tiếp 5](#_Toc521575406)

[3.3 Tạo thêm và di chuyển DOM vơi appendchild 5](#_Toc521575407)

[3.4 Xoá DOM vơi removeChild 6](#_Toc521575408)

[II. JSON 6](#_Toc521575409)

[1. Định nghĩa 6](#_Toc521575410)

[2. Các ví dụ 7](#_Toc521575411)

[III. Local Storage 8](#_Toc521575412)

[1. HTML Local Storage là gì ? 8](#_Toc521575413)

[2. Các đối tượng local storeage 9](#_Toc521575414)

[2.1 localStorage:lưu trữ dữ liệu mà ko có thời hạn 9](#_Toc521575415)

[2.2 sessionStorage:lưu trữ dữ liệu trong một session và sẽ mất khi tab hiện tại bị đóng 9](#_Toc521575416)

[3. Cách sử dụng 9](#_Toc521575417)

[3.1 Các loại trình duyện hỗ trợ 9](#_Toc521575418)

[3.2 Đối tượng localStorage: 9](#_Toc521575419)

[3.2.1 cú pháp sử dụng 10](#_Toc521575420)

[3.3 Đối tương SessionStorage 10](#_Toc521575421)

[IV. FETCH API 11](#_Toc521575422)

1. DOM
2. Định nghĩa

DOM là tên viết tắt của Document Object Model(Mô hình đối tượng tài liệu). Nó dùng để truy xuất và thao tác trên các tài liệu có cấu trúc dạng HTML hay XML bằng các ngôn ngữ lập trình thông dịch như Javascript,PHP hoặc Python.

1. Cây cấu trúc DOM
   1. Nút

Có 5 loại nút:

+ Nút gốc: chính là tài liệu HTML, thường biểu diễn bởi cặp thẻ <html></html>

+Nút phần tử: biểu diễn cho 1 phần tử HTML .

+Nút văn bản: mỗi đoạn kí tự trong tài liệu HTML, bên trong thẻ HTML đều là 1 nút văn bản.

+Nút thuộc tính: là các thuộc tính của một thẻ.

+Nút chú thích: là các comment line trong trang.

* 1. Quan hệ giữa các nút

Nút gốc là nút đầu tiên

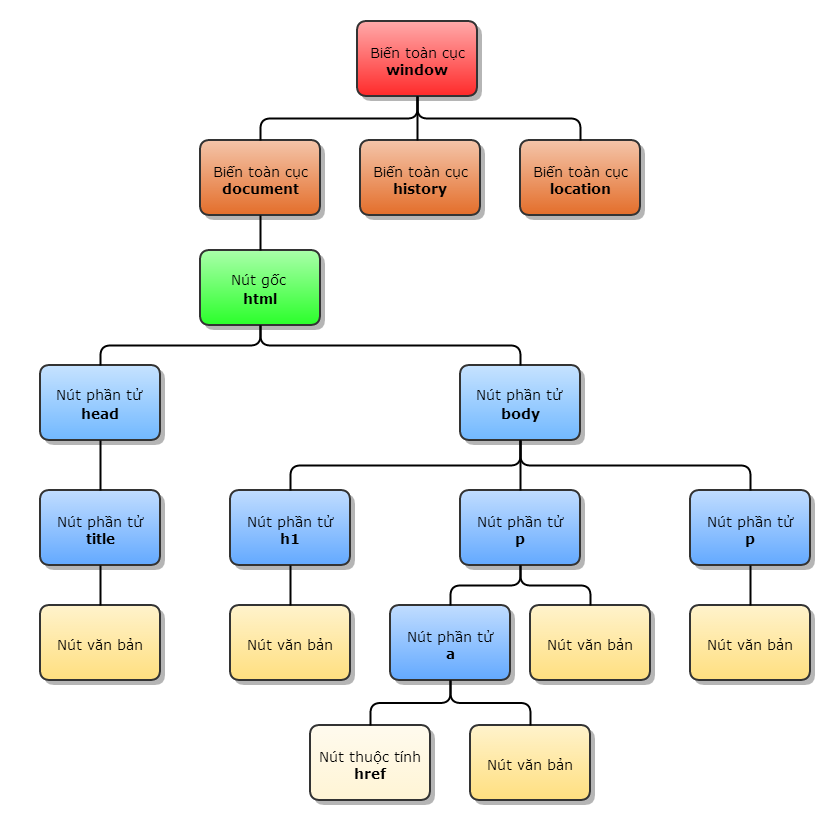
Tất cả các nút không phải nút gốc đều có 1 cha(parent).

Một nút có thể không có, có một hoặc có nhiều con.

Những nút có cùng nút cha được gọi là nút anh em(siblings).

Trong các nút anh em. Nút đầu tiên được gọi là anh cả(firstChild) và nút cuối được gọi là em út(lastChild).

Cụ thẻ trong hình bên dưới



1. Thao tác với DOM
   1. Thuộc tính và các phương thức thường gặp

|  |  |
| --- | --- |
| **Thuộc tính** | **Ý nghĩa** |
| id | Định danh – là duy nhất cho mỗi phần tử nên thường được dùng để truy xuất DOM trực tiếp và nhanh chóng. |
| className | Tên lớp – Cũng dùng để truy xuất trực tiếp như id, nhưng 1 className có thể dùng cho nhiều phần tử. |
| tagName | Tên thẻ HTML. |
| innerHTML | Trả về mã HTML bên trong phần tử hiện tại. Đoạn mã HTML này là chuỗi kí tự chứa tất cả phần tử bên trong, bao gồm các nút phần tử và nút văn bản. |
| outerHTML | Trả về mã HTML của phần tử hiện tại. Nói cách khác, outerHTML = tagName + innerHTML. |
| textContent | Trả về 1 chuỗi kí tự chứa nội dung của tất cả nút văn bản bên trong phần tử hiện tại. |
| attributes | Tập các thuộc tính như id, name, class, href, title… |
| style | Tập các thiết lập định dạng của phần tử hiện tại. |
| value | Lấy giá trị của thành phần được chọn thành một biến. |
| **Phương thức** | **Ý nghĩa** |
| getElementById(id) | Tham chiếu đến 1 nút duy nhất có thuộc tính id giống với id cần tìm. |
| getElementsByTagName(tagname) | Tham chiếu đến tất cả các nút có thuộc tính tagName giống với tên thẻ cần tìm, hay hiểu đơn giản hơn là tìm tất cả các phần tử DOM mang thẻ HTML cùng loại. Nếu muốn truy xuất đến toàn bộ thẻ trong tài liệu HTML thì hãy sử dụng document.getElementsByTagName('\*'). |
| getElementsByName(name) | Tham chiếu đến tất cả các nút có thuộc tính name cần tìm. |
| getAttribute(attributeName) | Lấy giá trị của thuộc tính. |
| setAttribute(attributeName, value) | Sửa giá trị của thuộc tính. |
| appendChild(node) | Thêm 1 nút con vào nút hiện tại. |
| removeChild(node) | Xóa 1 nút con khỏi nút hiện tại. |

Mặt khác, các phần tử DOM đều là các nút trên cây cấu trúc DOM. Chúng sở hữu thêm các thuộc tính quan hệ để biểu diễn sự phụ thuộc giữa các nút với nhau. Nhờ các thuộc tính quan hệ này, chúng ta có thể truy xuất DOM gián tiếp dựa trên quan hệ và vị trí của các phần tử.

|  |  |
| --- | --- |
| **Thuộc tính quan hệ** | **Ý nghĩa** |
| parentNode | Nút cha |
| childNodes | Các nút con |
| firstChild | Nút con đầu tiên |
| lastChild | Nút con cuối cùng |
| nextSibling | Nút anh em liền kề sau |
| previousSibling | Nút anh em liền kề trước |

* 1. Truy xuất
     1. Truy xuất trực tiếp

Mỗi nút trên cây DOM đều có 6 thuộc tính quan hệ để giúp bạn truy xuất gián tiếp theo vị trí của nút:

* **Node.parentNode**: tham chiếu đến nút cha của nút hiện tại, và nút cha này là duy nhất cho mỗi nút. Do đó, nếu bạn cần tìm nguồn gốc sâu xa của 1 nút, bạn phải nối thuộc tình nhiều lần, ví dụ Node.parentNode.parentNode.
* **Node.childNodes**: tham chiếu đến các nút con trực tiếp của nút hiện tại, và kết quả là 1 mảng các đối tượng. Lưu ý rằng, các nút con không bị phân biệt bởi loại nút, nên kết quả mảng trả về có thể bao gồm nhiều loại nút khác nhau.
* **Node.firstChild**: tham chiếu đến nút con đầu tiên của nút hiện tại, và tương đương với việc gọi Node.childNodes[0].
* **Node.lastChild**: tham chiếu đến nút con cuối cùng của nút hiện tại, và tương đương với việc gọi Node.childNodes[Element.childNodes.length-1].
* **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.
  + 1. Truy xuất trực tiếp

document.getElementById('id\_cần\_tìm')

document.getElementsByTagName('div')

document.getElementsByName('tên\_cần\_tìm')

* 1. Tạo thêm và di chuyển DOM vơi appendchild

Như đã biết, mọi nút trên cây cấu trúc DOM đều bắt nguồn sâu xa từ nút gốc và bắt buộc phải có 1 nút cha. Do đó, về bản chất, khi 1 DOM mới được tạo ra, nó cô đơn 1 mình và không thể sử dụng được như các phần tử DOM thông thường. Chỉ sau khi bạn tìm 1 nút khác trên cây DOM để làm cha đứa bé (sử dụng [nút\_cha.appendChild(nút\_con)](https://thachpham.com/web-development/javascript/dom-trong-javascript-can-ban.html#help-appendChild)) thì quá trình tạo thêm DOM hoàn tất. Nếu đã sẵn sàng để tạo vài em bé thì hãy cùng nhau tạo vài nút mới với 2 phương thức sau:

* document.createElement(tên\_thẻ\_html): tạo 1 nút phần tử từ nút gốc.
* document.createTextNode(chuỗi\_kí\_tự): tạo 1 nút văn bản từ nút gốc.
* Một lưu ý khi tạo thêm DOM: nút mới được tạo sẽ vẫn có các thuộc tính và phương thức như 1 nút DOM thực thụ. Tuy nhiên, các thuộc tính của nút mới sẽ ở trạng thái rỗng hoặc mặc định, nên bạn sẽ phải thiết lập thêm các thuộc tính cần thiết.

|  |  |
| --- | --- |
|  | var babyDom = document.createElement('div');  babyDom.id = 'Whatsapp';  babyDom.class = 'OTP-Messenger';    var fatherDom = document.getElementById('Facebook');  // Facebook tậu ứng dụng nhắn tin Whatsapp  fatherDom.appendChild(babyDom); |

* Phương thức appendChild còn được sử dụng để tổ chức “tái định cư” cho các nút DOM. Nếu như 1 nút DOM đang có 1 nút cha, nhưng lại được 1 nút cha khác “nhận nuôi” thông qua appendChild thì nó sẽ cắt đút quan hệ với nút cha cũ để về bên gia đình mới.

|  |  |
| --- | --- |
|  | var child = document.getElementById('Motorola');  var oldFather = document.getElementById('Google');  // Google mua Motorola  oldFather.appendChild(child);    var newFather = document.getElementById('Lenovo');  // Lenovo mua lại Motorola  newFather.appendChild(child); |

* 1. Xoá DOM vơi removeChild

Khi loại bỏ DOM, nút cha sẽ sử dụng phương thức removeChild() để từ mặt 1 hoặc nhiều nút con.

|  |  |
| --- | --- |
|  | // Cách 1: Loại bỏ 1 nút con khi biết nút cha và nút con  var google = document.getElementById('Google');  var motorola = document.getElementById('Motorola');  google.removeChild(motorola);    // Cách 2: Loại bỏ 1 nút con khi chỉ biết nút con  var whatsapp = document.getElementById('Whatsapp');  if (whatsapp.parentNode) {    whatsapp.parentNode.removeChild(whatsapp);  }    // Cách 3: Loại bỏ toàn bộ nút con  var body = document.getElementsByTagName('body')[0];  while (body.firstChild) {    body.removeChild(body.firstChild);  } |

1. JSON

1. Định nghĩa

**JSON** là chữ viết tắt của **J**avascript **O**bject **N**otation, đây là một dạng dữ liệu tuân theo một quy luật nhất định mà hầu hết các ngôn ngữ lập trình hiện nay đều có thể đọc được, bạn có thể sử dụng lưu nó vào một file, một record trong CSDL rất dễ dàng.

Cú pháp của JSON là mỗi thông tin dữ liệu sẽ có 2 phần đó là key và value, điều này tương ứng trong CSDL là tên field và giá trị của nó ở một record nào đó. Tuy nhiên nhìn qua thì đơn giản nhưng nếu ta mổ xẻ nó ra thì có một vài điều như sau:

Chuỗi JSON được bao lại bởi dấu ngoặc nhọn {}

Các key, value của JSON bắt buộc phải đặt trong dấu nháy kép {"}, nếu bạn đặt nó trong dấu nháy đơn thì đây không phải là một chuỗi JSON đúng chuẩn. Nên trường hợp trong value của bạn có chứa dấu nháy kép thì hãy dùng dấu (\) để đặt trước nó nhé, ví dụ học \"json là gì\" tại website freetuts.net.

Nếu có nhiều dữ liệu (nhiều cặp key => value) thì ta dùng dấu phẩy (,) để ngăn cách

Các key của JSON bạn nên đặt chữ cái không dấu hoặc số, dấu \_ và không có khoảng trắng., ký tự đầu tiên không nên đặt là số. Điều này rất giống với nguyên tắc đặt tên biến trong PHP.

1. Các ví dụ

**Ví dụ 1**: Có 3 sinh viên  gồm các thông tin sau: Nguyễn Văn Cường - 21 tuổi, Nguyễn Văn Kính - 22 tuổi, Nguyễn Văn Chính - 23 tuổi. Hãy xây dựng chuỗi JSON lưu trữ thông tin danh sách sinh viên đó.1

Nếu xem ví dụ ở phần 1 JSON là gì thì bạn thấy không thể lưu được, vì nếu lưu thì ta sẽ lưu thế này:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | {      "sinhvien1\_ten" : "Nguyễn Văn Cường",      "sinhvien1\_tuoi" : "21 Tuổi",      "sinhvien2\_ten" : "Nguyễn Văn Kính",      "sinhvien2\_tuoi" : "22 Tuổi",      "sinhvien3\_ten" : "Nguyễn Văn Chính",      "sinhvien4\_tuoi" : "22 Tuổi"  } |

Nhìn vào không hay đúng không nào, vì thông thường mỗi đối tượng chúng ta sẽ gom lại một nơi lưu trữ để tiện cho việc tìm kiếm và phân chia, nhưng chuỗi này không lưu được như vậy. Bây giờ ta sẽ tìm hiểu một cách khác nhé.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [      {          "name" : "Nguyễn Văn Cường",          "age" : "21 tuổi"      },      {          "name" : "Nguyễn Văn Kính",          "age" : "22 tuổi"      },      {          "name" : "Nguyễn Văn Chính",          "age" : "23 tuổi"      }  ] |

Nhìn gọn hơn rồi đúng không nào, chúng ta sử dụng dấu ([]) để gom nhóm lại, và bên trong là danh sách các chuỗi JSON con được cách nhau bởi dấu phẩy (,). Lưu ý là các bạn phải tuân theo những quy tắc JSON mà phần 1 JSON là gì đã trình bày nhé.

Ví dụ 2: Giả sử chúng ta có bài toán như sau, ở một trường đại học lưu trữ điểm của sinh viên, mỗi sinh viên sẽ có một mã số sinh viên  sẽ đăng ký học các tín chỉ khác nhau, hãy tổ chức cấu trúc chuỗi JSON để lưu trữ danh sách sinh viên và danh sách các môn học của sinh viên đó.2

**Giả sử**:

* Nguyễn Văn Cường có ID là sv0001
* Nguyễn Văn Kính có ID là sv0002

Bài này hơi khó rồi đấy nhỉ, các bạn tham khảo lời giải sau và rút ra bài học nhé.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | {      "sv0001" : {          "toan" : "Môn Toán",          "ly" : "Môn Lý"      },      "sv0002" : {          "toan" : "Môn Toán",          "anh" : "Môn Anh"      }  } |

Các bạn thấy khác với ví dụ 1 mình không sử dụng dấu [] để bao ngoài cùng mà thay bằng cặp dấu {}, tại sao? Đơn giản là vì cặp [] dùng để bao các chuỗi JSON con, mà chuỗi JSON thì bắt buộc có cặp {} nên trong trường hợp trên không sử dụng được (các cặp con ở dạng key => {value})

1. Local Storage

1. HTML Local Storage là gì ?

Với local storeage web applications có thể lưu trữ dữ liệu tại trình duyệt của user,đây là một tính năng hưu ích trong HTML5 Trước HTML 5,dữ liệu của ứng dụng phải lưu trữ trong cookies,Local Storage thì bảo mật hơn và có thể lưu trữ nhiều dữ liệu mà không ảnh hưởng tới hiệu năng của website. Không giống như cookies,giới hạn lưu trữ dữ liệu của nó lớn hơn(it nhất là 5Mb) và thông tin không bao giờ được gửi tới server Local storage được sử dụng cho mỗi domain hoặc protocal.Tất cả các trang từ một ứng dung có thể lựu trữ và truy cập cung một data giống nhau

1. Các đối tượng local storeage

2.1 localStorage:lưu trữ dữ liệu mà ko có thời hạn

2.2 sessionStorage:lưu trữ dữ liệu trong một session và sẽ mất khi tab hiện tại bị đóng

1. Cách sử dụng

3.1 Các loại trình duyện hỗ trợ gồm có:Google 4.0,Microsoft Edge 8.0,Firefox 3.5,Safari 4.0,Opera 11.5.kiểm tra bằng dòng lệnh xem trình duyệt có hỗ trợ không bằng câu lệnh sau

if (typeof(Storage) !== "undefined") {

// Code for localStorage/sessionStorage.

} else {

// Sorry! No Web Storage support..

}

3.2 Đối tượng localStorage:dữ liệu sẽ vẫn còn cho dù trình duyệt đã bị đóng

3.2.1 cú pháp sử dụng: -Để lưu trữ tại localStorage ta sử dụng chức năng setItem với hai tham số key và value,hoặc dùng Array hay object

localStorage.setItem('name', 'thanh');

localStorage.name = 'thanh';

localStorage['name'] = 'thanh';

Để lấy ra giá trị đã lưu trữ ta sử dụng chức năng getItem()

var name = localStorage.getItem('name');

Để xóa dữ liệu đã lưu trữ ta dùng cú pháp sau.

localStorage.removeItem('name');

Áp dụng:

Có một giới hạn lưu trữ lượng dữ liệu mà bạn có thể lưu ở localStorage,nó phụ thược vào nhà cung cấp trình duyệt.Nếu ứng dụng của ban chỉ lưu trữ khoảng 2,5mb data tại localStorage thì có thể xem xet lưu trữ,còn nếu nhiều hơn thì phải xem xét lại.

3.3 Đối tương SessionStorage -cách sử dụng sessionStorage thì cũng tương tự như localStorage chỉ khác là khi bạn tắt tab đang hiện tại.Cú pháp

Storing Data

sessionStorage.setItem('name', 'thanh');

Retrieving Data

var name = sessionStorage.getItem('name');

Deleting Data

sessionStorage.removeItem('name');

Retrieving an Item Key

sessionStorage.key(name);

Clearing the Datastore

sessionStorage.clear();

1. FETCH API
   1. Fetch là gì?

Fetch là một API cho phép gửi request và nhận response bằng javascript. Bằng việc sử dụng nó thì việc gửi các yêu cầu tới web và xử lý phản hồi nhanh hơn so với AJAX. Fetch tương thích với hầu hết các browser phiên bản sau năm 2015.

* 1. Tạo một fetch cơ bản để lấy dữ liệu từ một Json API

Code mẫu:

fetch('http://jsonplaceholder.typicode.com/users’)

.then(function(response) {

// Làm một việc gì đó với response

})

.catch(function(error) {

console.log('Dường như có vấn đề gì đó xảy ra: \n', error);

});

Giải tích đoạn code trên: Phương thức fetch sẽ lấy url(trong ví dụ là link của json api) dưới dạng parameter để lấy dữ liệu. Sau đó fetch sẽ trả về một promise. Với trạng thái promise là resolve thì fetch của chúng ta đã lấy dữ liệu thành công và trả về response. Khi resolve, response sẽ được truyền tới .then. Đây là nơi mà chúng ta có thể thao tác với response. Khi reject tức là việc fetch không thành công, tất cả sẽ quy về .catch với tham số là lỗi tương ứng.

Đọc response trả về:

Responses có phương thức truy cập vào nội dung trả về. Ví dụ, Response.json() trả về một promise resolves dạng JSON. Thêm bước này vào ví dụ hiện tại sẽ cập nhật code thành:

fetch('examples/example.json')

.then(function(response) {

// trả về response dưới dạng json.

return response.json();

})

.then(function(responseAsJson) {

// Ghi ra console json ở trên

console.log(responseAsJson);

})

.catch(function(error) {

console.log('Looks like there was a problem: \n', error);

});