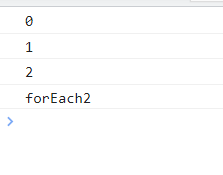
1. Khi tạo( tự thêm ) một prototype vào trong một object( Array, Number..)

a.khi nào dùng for in thì nó cũng tính cái tên thuộc tính mới thêm vào là một key, nếu in ra key( index ) thì sẽ thấy được nó cũng in ra tên thuộc tính vừa thêm vào

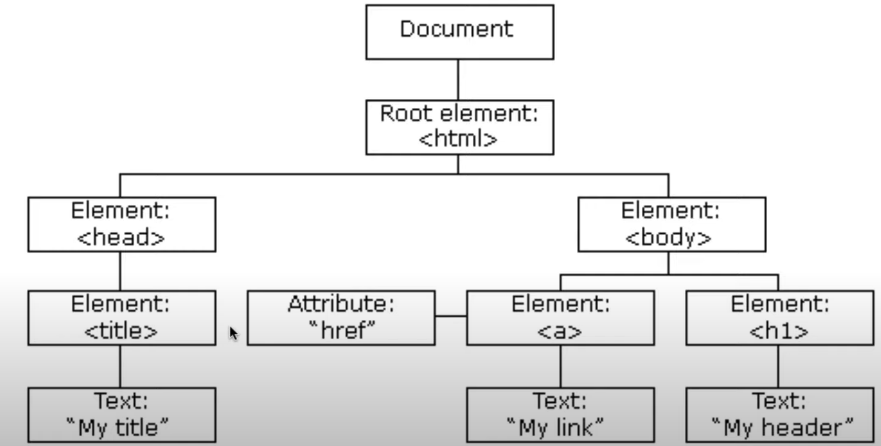


a1. Cách duyệt key chính quy ( bỏ qua các key prototype thêm vào)

sử dụng this.hasOwnProperty(index); ( this ở đây chính là đối tượng object mẹ)

🡪Return về boolen

2.Dom



a. document object model: mô hình tài liệu được thể hiện ở mô hình đối tượng

b. gồm ba thành phần

b1. Element

b2.attribute

b3.Text

c.node :điểm lồi lên của một mô hình ( chỉ chung chung có thể là element , attribute, text)

d. ý nghĩa của html DOM:cho phép chỉnh sửa từ text, element , nói chung là mọi thành phần của file html

3. root : nguồn gốc

4. html dom không phải là js :

a. js chỉ cung cấp các công cụ để truy cập là html dom

5. access: truy cập

6. document là cái bao quát , đứng đầu là mẹ trong toàn cái web

7 các cách get element trong dom(id)

Id, class, tag , css selection , html collection

🡪 getElementById()

Return về một element( vì id chỉ độc lập theo tiêu chuẩn w3c)

8. getElementsByClassName : (class)

🡪 return về HTML collection (có tính chất tương tự như array) truy xuất giống array nhưng không sử dụng được các phương thức như reduce , filter đồ…

9, getElementsByTagName: (tag)

🡪 giống y như class , nhưng là biến cần truyền vào là tên các thẻ ( h1, p, head, footer..)

Return về một HTML colection

10.querySelector( Css selector)

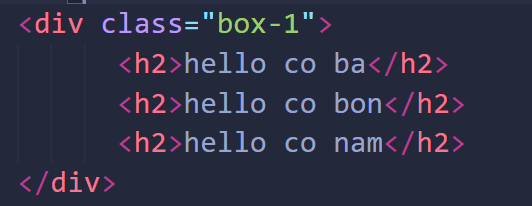
a.lưu ý ,cái này chỉ select được một cái đầu tiên thôi , dù css selector chọn được nhiều nhưng mà nó chỉ lấy cái đầu tiên được chọn

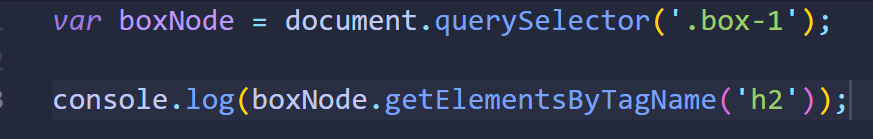
retrun về một element

🡪 để select hết thì dùng : querySelectorAll()

Cái này return về Nodellist ( cũng tương tự html list)

11 selection Element DOM (new look)





- Đầu tiên là nó lấy cái box-1, từ cái box-1 trở thành một document, từ đây ta có thể sử dụng các thuộc tính select để nhanh tìm con của box-1

12. Khi Dom được tải xong thì nếu dùng document.write('text') trong thẻ js, thì nó sẽ ghi đè text lên toàn bộ web , mấy hết tất cả web cũ thay bằng text

13. cách thêm một attribute vào html bằng js( setter)

A, thêm một atttribute hợp lệ với thẻ tag:

a. - lấy ra element ( dùng các phương thức ở trên )🡪truy cập vào tag từ document

b. - từ element được lấy ra element.attribute(id, className , href, link, src..) )('tên của attribute' )

vd: var heading = getElementsTagName('h1')

heading.id = 'footer-heading'

B, thêm một atttribute không có sẳn hoặc tự sáng tạo :

a. giống a ở trên

b.element.setAttribute('id',footer-heading')

14.cách lấy ra tên của attribute có sẳn(hay đã được thêm nhờ js như lưu ý 13 ở trên) (getter)

- Chỉ lấy ra các thẻ hợp lệ với thẻ tag

a.giống 13a

b. element.className ( class thì thẻ nào mà chẵn hợp lệ )

-Lấy ra chung cho các thẻ hợp lệ và không hợp lệ với các thẻ tag

a.giống 13.a.

b.element.getAttribute('class') ===footer-heading

🡪 lưu ý ở mỗi bước a trong node này chỉ được dùng các thuộc tính trả về element (còn trả về html selection hay node selection không được)

/////

Truy cập vào text

I . truy cập

15.truy cập vảo text (geter)

a.tương tự như truy cập vào atttribute , document 🡪element 🡪 atttribute

b. innerText , textContent

🡪 từ element hoặc attribute đã truy cập 🡪 móc nối với một trong hai thuộc tính trên

Ex:

Var heading = document.querySelector('h1');

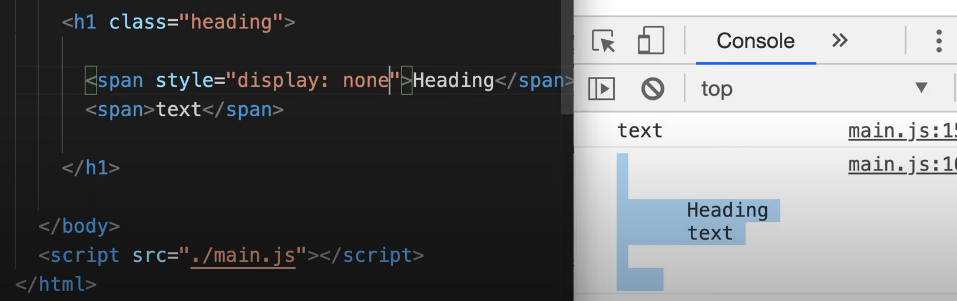
🡪Console.log(Heading.innerText)

16. chỉnh sửa text (seter)

a.tương tự như a trên (cần truy cập vào element hoặc attribute chứa text)

b. heading.innerText = 'hello' 🡪 chèn đè lên heading gốc text= "hello"

17. sự khác biệt của innerText và textContent:



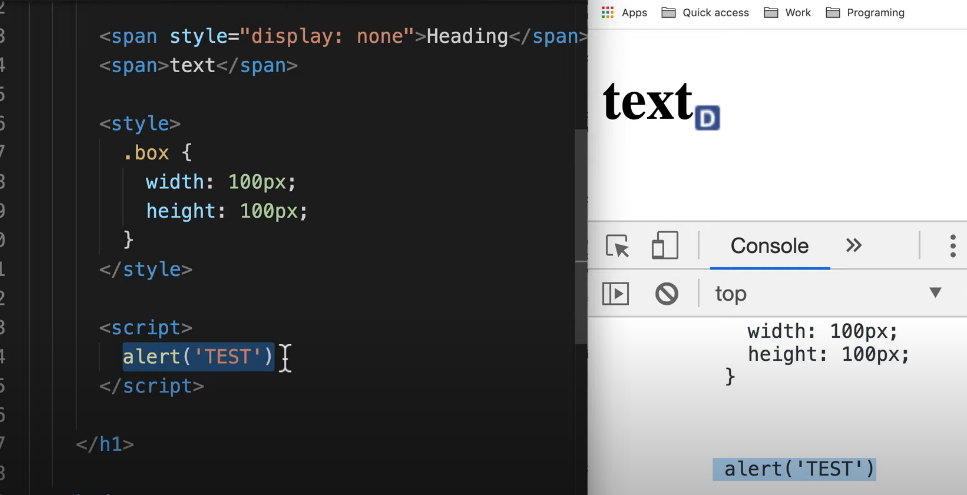
🡪 innerText ( # là thuộc tính của khi và chỉ khi đang truy cập đến element node, còn truy cập thẳng vào text node hay đi như nào thì sẽ không dùng được)

a. chỉ quan tâm đến text + không quan tâm cách trình bày text trong element

🡪 textContent(# là thuộc tính của cả khi truy cập đến element node hay là text node, dùng thuộc tính này vẫn được)

a. quan tâm đến bố trí của text trong elementt , không bị chi phối bởi các modifed gây ảnh hưởng đến text

b.lưu ý quan trọng :



Mặc dù alert('test') là mã hiện thị lên web , nhưng mà trong đây nó tuân thủ nghiêm ngặt trên biểu đồ DOM, là chỉ gồm các thành phần document, element( các tag), attribute , text , thằng alert nằm trong một node , và không nằm trên 3 loại trên thì cũng bị cho thànnh text

🡪 lưu ý chung cho cả hai phương thức :

+innerText: có text kiểu"

Heading

"

Thì nó sẽ hiện thị ra bên ngoài màn hình y như trên cũng enter như này , dấu enter được mã hoá bằng thẻ <br></br>

+ thằng textContent sẽ đọc luôn khoảng trắng

🡪 lưu ý chung cho hai thuộc tính:

Nếu bên trong text ta đưa một đoạn string = ' <h1> hello </h1> ' thì cái thẻ mở và thẻ đóng của h1 nó được mã hoá để trở thành string

//////

Thêm một element node vào trong một element node

InnerHTML, outerHTML

innerHTML có thể :

-thêm text

-thêm element node

- thêm attribute vào element node

Lưu í là nó ghi đè bên trong element truy cập

Để get nó 🡪 return về string element node

outerHTML cũng thêm node element nhưng mà ghi đề lên thẻ element mẹ truy cập từ document;

🡪 lúc dùng innerHTML console ra thì nó sẽ hiện lên còn thẻ con cũ + thẻ mới chèn vào, nhưng thật ra toàn bộ thẻ mẹ và thẻ con đã bị ghi đè mới outer thêm vào

🡪 dùng outerHTML console ra thì nó sẽ hiện lên thẻ mẹ + thẻ con cũ + thẻ mới outher vào

//////

DOM CSS

18.Style

a.lấy ra các CSS inline chỉ trong các element node

b.cũng là một object

c.

🡪 quy ước chung ở đây element = biến đã truy cập từ document vào element

setter:

c1. Element.style.width = 200px;(được thêm vào css inline)

getter:

c2. Console.log(Element.style.width ) 🡪200px;

d. Hợp nhất các style vào một funciton:

assign( phân công)

Object.assign.(element.style,{

Width:200px ,

Color:red ,

BackgroundColor: green ,

}

)

19. cái cách viết nhấp nhô hoa trong đặt tên biến ( currentValue) gọi là camelCase)