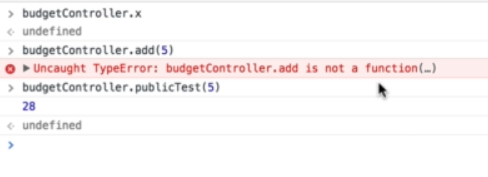
1. Bult with module pattern





c

Biến và hàm khai báo trong IIFE này là private

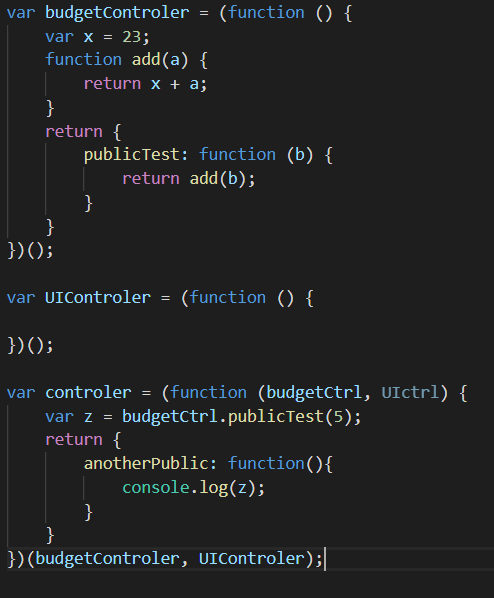
Với hàm và biến public chúng ta khai báo trong object và return nó

Nên: x, add là private

publicTest là public

nhưng bởi vì closure nên public test có thể sử dụng x, add ( closure là khi một hàm ngoài đà kết thúc, nhưng hàm trong vẫn sử dụng đk biến của hàm ngoài)

* ở trên là một module, ta sẽ tạo nhiều module như vậy để vận hành app, mỗi mudule là độc lập, ko liên quan đến nhau. Ta có thế đưa một module vào trong module khác bằng cách truyền đối số

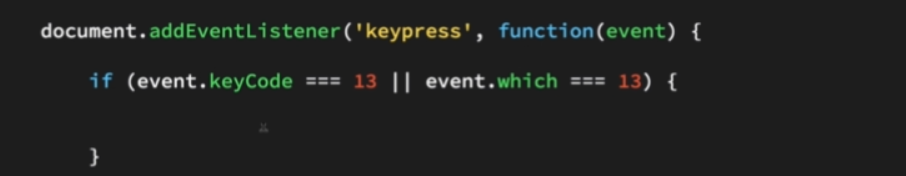


Budget và ui là nơi viết hàm để lấy dữ liệu, controler là nơi gọi hàm và thực thi hàm

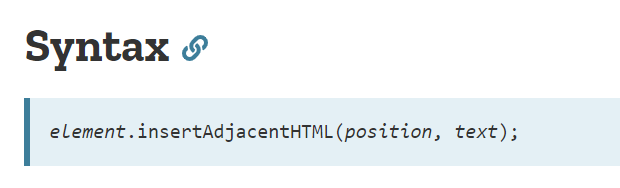
Tổ chức theo kiểu các module

Mỗi module sẽ có biến, và hàm private, và method public, medthod public sẽ sữ dụng biến và hàm private này nhờ closure, từ đó thực thi công việc trong module

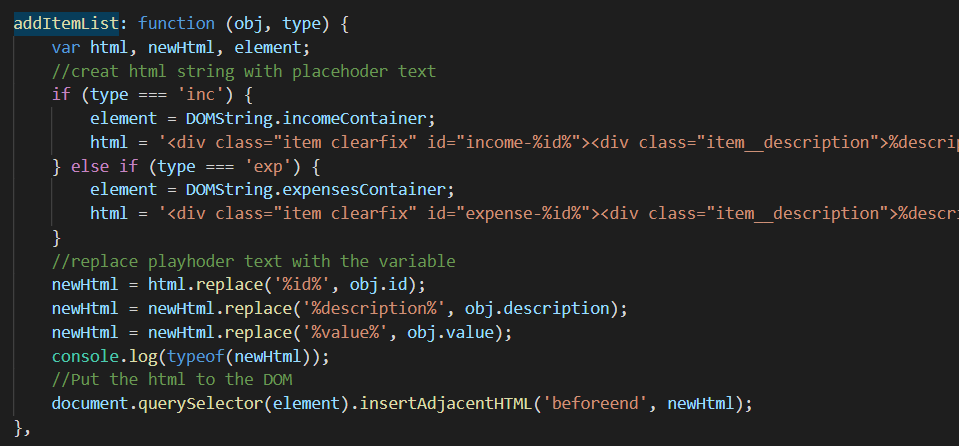
Có 1 module điều khiển: module này sử dụng các public của các module khác để thực hiện tổng công việc

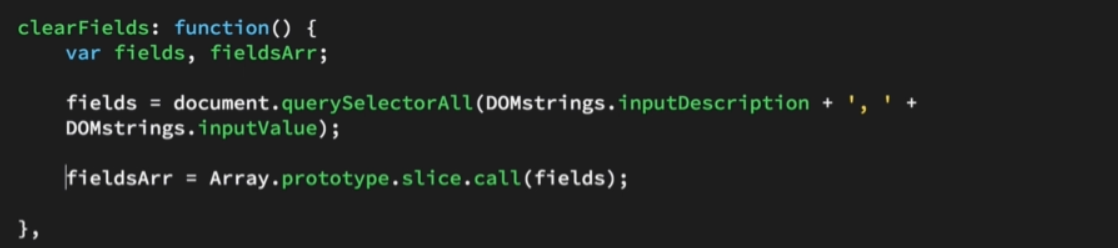


Button event



Đưa một đoạn mã html vào file html bằng js





Fields ở đây trả về 1 list, vì vậy không thể sử dụng và dùng các method của array

Slice method copy 1 array và trả về 1 array khác

Dùng câu lệnh Array.prototype.slice.call(fields) đề conver filds về array

* dùng fieldArr như array



Thực hiện xóa placehoder khi nhập

forEach thực hiện cho tất cả phần tử trong array, với

current là phần tử hiện tại

index là i

array là cái mảng hiện tại -> array[index];



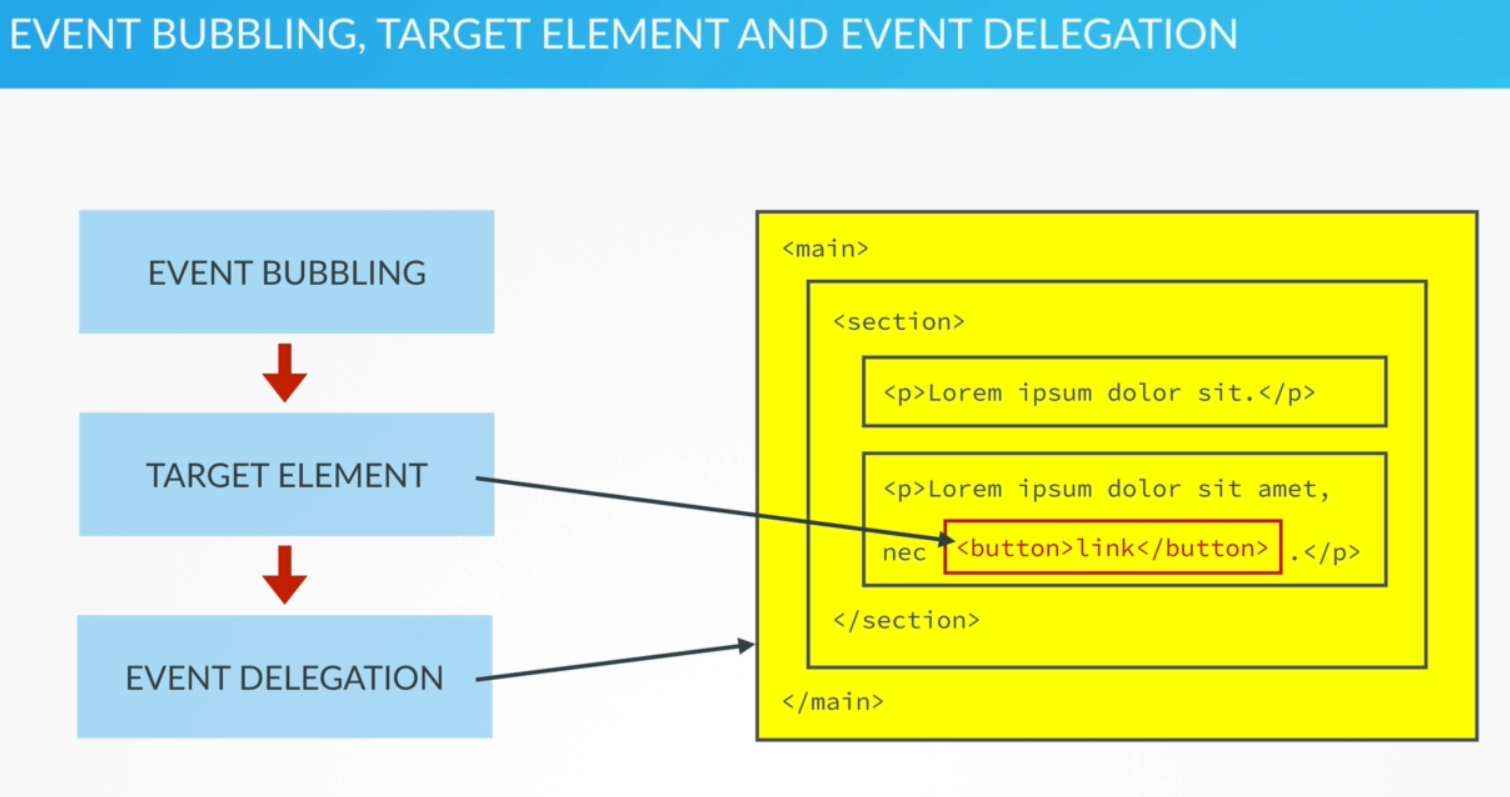
Fieldsarr[0].focus() đặt con trỏ chuột về lại fildarrr[0];



Covert string to float



isNaN();

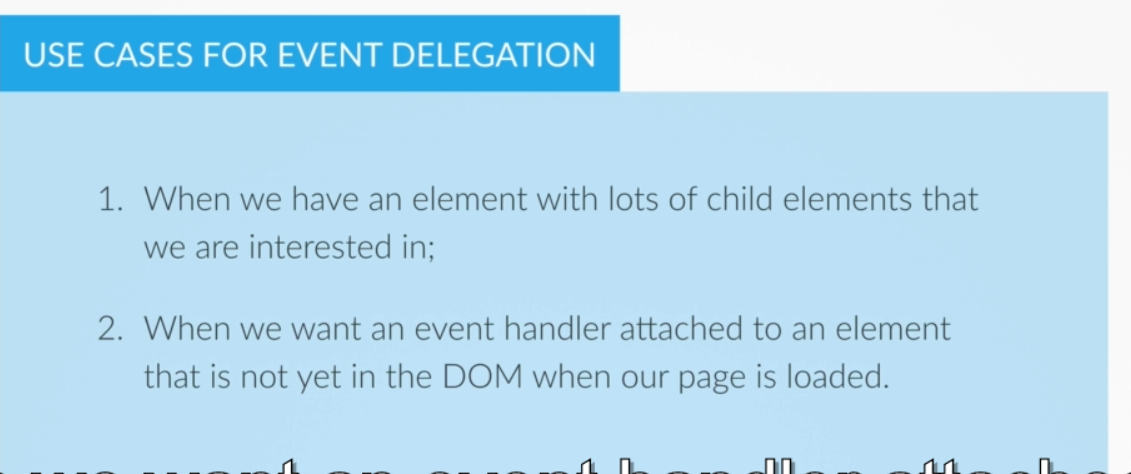


Khi kích hoạt một dom element ví dụ button như trên thì event đó sẽ kích hoạt toàn bộ parent elements theo kiểu bubling. Button -> link -> section -> main

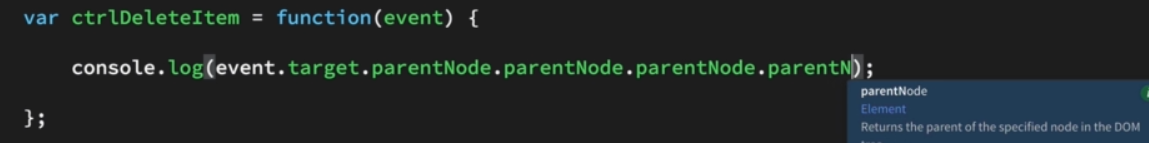
Taget element là event kích hoạt đầu tiên -> button

Event delegation: add event hadler to the parent element and wait for the event to bubble up

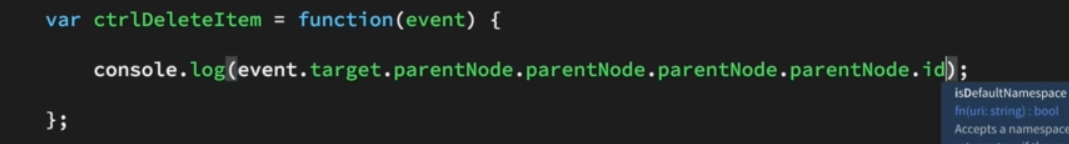
* thực hiện việc cần làm sau khi thực hiên taget element -> thực hiện việc cần làm sau khi nhấn nút



1. chọn 1 taget element trong một parent nhiều elemet, khi thực hiện taget element thì toàn bộ element trong đó được thực hiện theo
2. khi một page chưa load xong, nhấn vào event element thì sẽ không đủ chức năng -> delegation -> truy xuất lên parent -> đủ chức năng

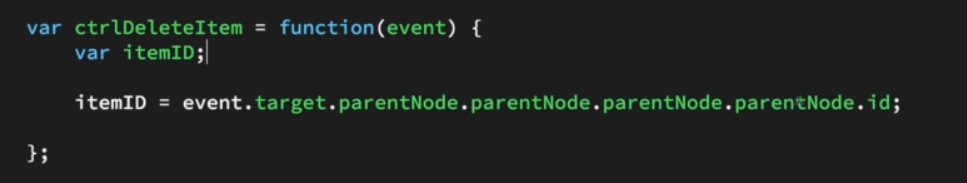


Console.log event, nhấn vào nút nhấn, thấy được đối tượng, nhưng ta cần xóa toàn parent của đối tượng đó nên dùng .parentNode để move up lên parrent

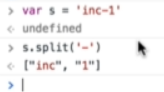


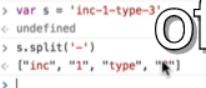


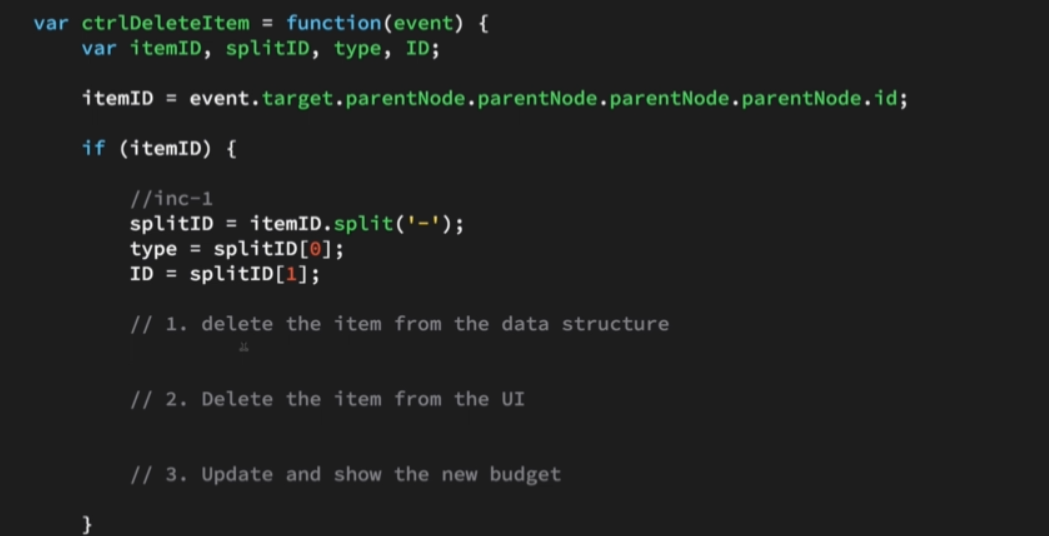
.id

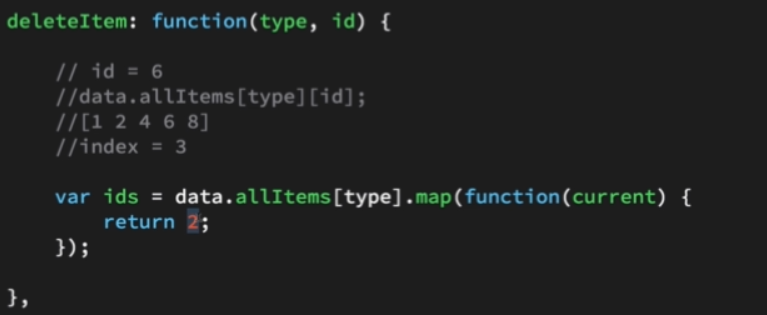


* cái này gọi là traversing
* split







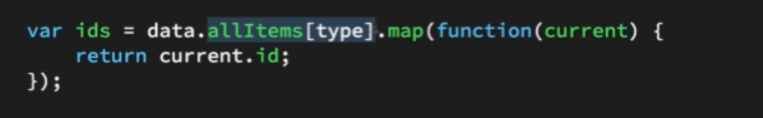


Id !== index vì khi xóa -> id giữ nguyên -> index thay đổi -> nếu gọi data.allItems[type][id] -> gọi phần tử thứ id của mảng, không phải phần tử với id

array.map()

Ví dụ array = [1,2,4,6,8] qua câu lệnh treeng 🡪 id=[2, 2 ,2 ,2, 2]

Khác nhau giữa map và forEach là map trả về 1 mảng

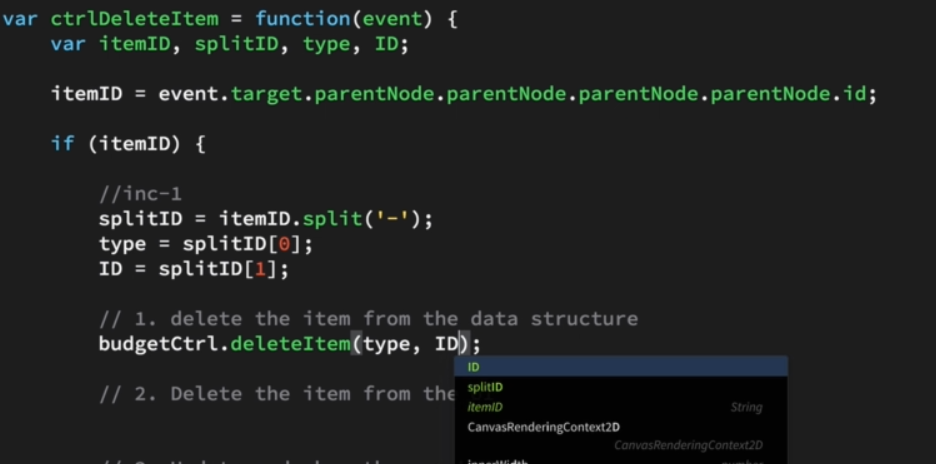


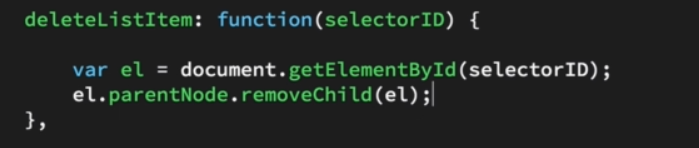
Ids = [id1,id2,id3,..]





Array.splice(vị trí bắt đầu xóa,số phần tử cần xóa)

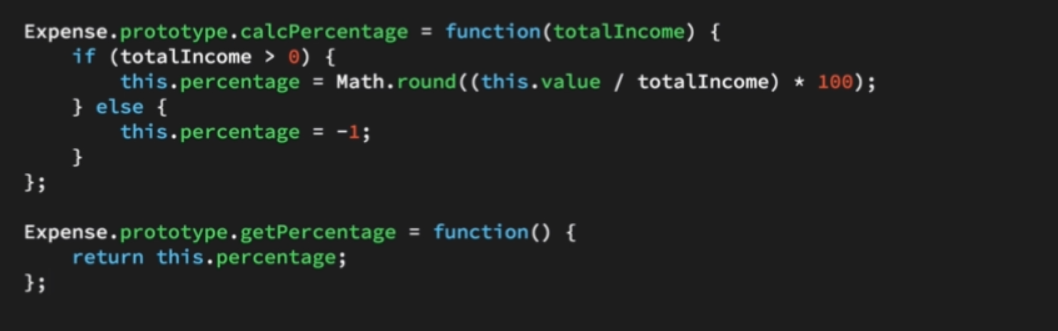




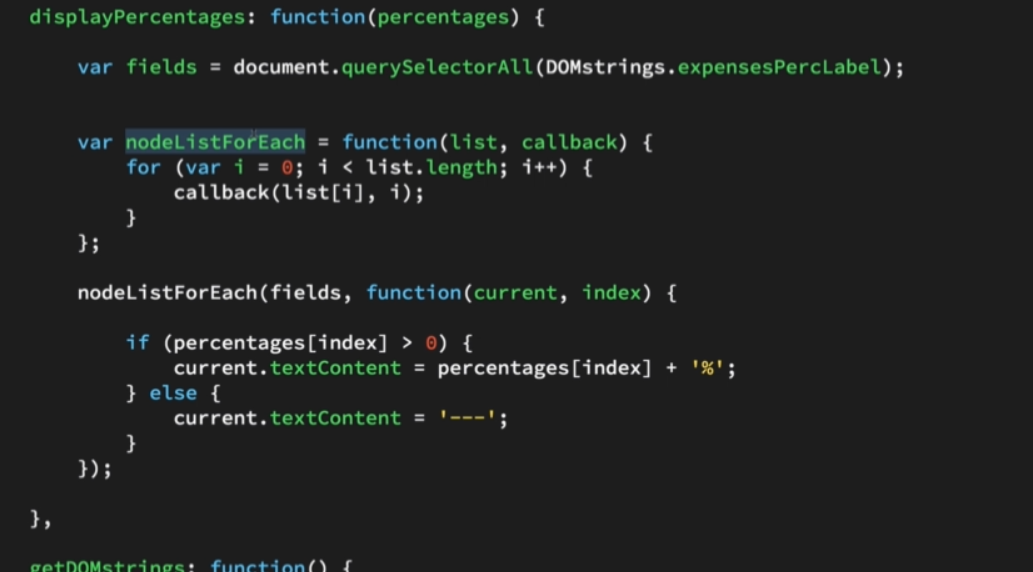
Trong js không cho phép xóa element mà phải dùng parent xóa child: parent.removeChild(child)

Hoặc

Child.parentNode.removeChild(child)



Tính nhiều cái lần rồi return



Callback là một hàm được truyền vào với vai trò thông số của hàm khác. Được dùng khi việc khai báo định nghĩa một hàm chưa thực hiện hoàn thành công việc, mà công việc sẽ được viết tiếp ở phần gọi hàm. Callback được dùng khi khai báo một hàm cần tham số của hàm callback để thực hiện sau hoặc truyền vào event.

Fields sẽ trả về list, trong list không thể sử dụng foreach -> dùng hack đưa về array

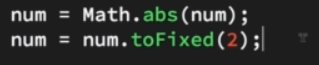
Hoặc tạo hàm forEach cho list:

List.length: lặp từ phần tử 0-> phần tử cuối cùng

Hàm callback gọi phần từ đó và index

Khi gọi hàm call back phần tử đó và index sẽ nằm trong đó -> sử dụng

(sử dụng for thẳng luôn cũng đk)

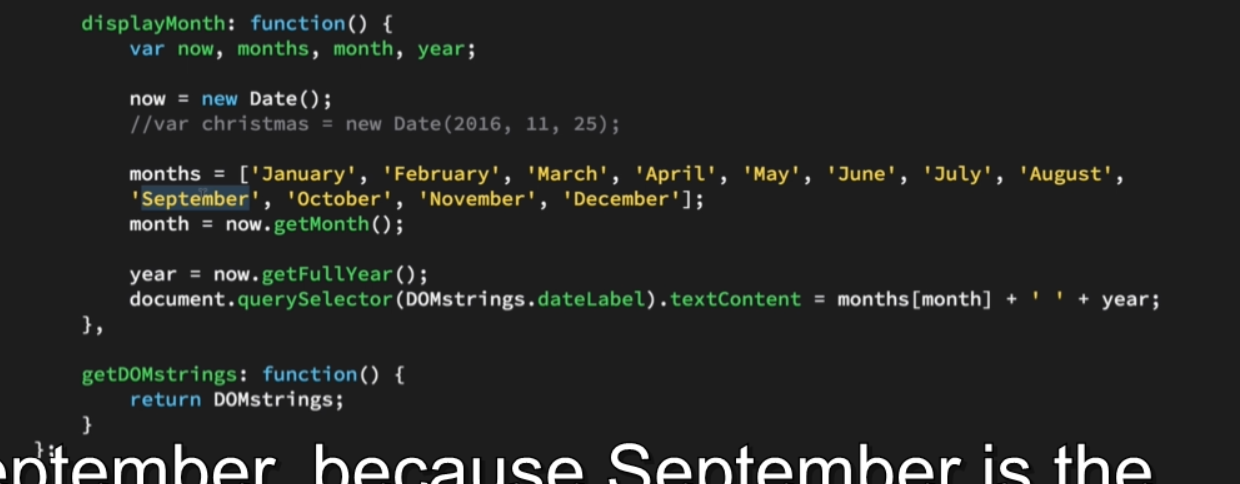


Trị tuyệt đối, 2 chữ số thập phân: trả về string



Substr(vị trí, số phần tử)





Contructor Date



