**ANGULAR DATA BINDING**

So, **what is data binding**? Hãy thử tưởng tượng rằng công ty X có anh nhân viên A, anh này có vợ là chị B, bằng một cách nào đó cứ đến kỳ chuyển lương thì chị B sẽ nhận được thông báo về điện thoại của mình rằng số tiền trong tài khoản của anh A đã được công ty X trả Y triệu. Tương tự như thế trong lập trình khi chúng ta có data D thuộc object O sẽ được ràng buộc với data G thuộc object OP, bằng một cách nào đó, mỗi khi data D thay đổi thì G cũng sẽ biết sự thay đổi và thay đổi theo. Đó là một dạng của data binding. Trong Angular chúng ta đã được Angular làm nhiệm vụ đồng bộ những dữ liệu bị ràng buộc lại với nhau, chúng ta chỉ cần update data/state cần thiết. Vậy có những dạng data binding nào?

**INTERPOLATION**

Bây giờ hãy nhớ lại, một component trong Angular chỉ là một TypeScript (TS) class thông thường, và nó đi kèm với một decorator để găn thêm meta-data như template mà nó định nghĩa là gì. Như vậy, TS class và template này hoàn toàn không biết đến nhau, mà nó sẽ được Angular xử lý để gắn chúng lại. Câu hỏi đặt ra là làm thế nào để tôi hiển thị một dữ liệu nào đó (tên, tuổi, ngày tháng năm sinh, một string, number bất kỳ, hay bất kỳ thứ gì (object) có thể hiển thị được? Đây chính là nơi tỏa sáng của cặp đôi hoàn cảnh (mà chúng ta gọi là interpolation) {{ expression }}. Nó có thể hiểu là hãy tính toán cái expression này, nếu có trả về cái gì thì phun (display) nó ra ngay vị trí chỗ dấu {{}} này cho tôi. Chỉ đơn giản thế. Giờ các bạn có thể phun data về tên tuổi của một người thành cái profile đơn giản như sau:

import { Component } from "@angular/core";

@Component({

selector: "app-hello",

template: `

<h2>Hello there!</h2>

<h3>Your name: {{ user.name }}</h3>

<p>Your name: {{ user.age }}</p>

`,

})

export class HelloComponent {

user = {

name: "Tiep Phan",

age: 30,

};

}

**PROPERTY BINDING**

Một số tag khi sử dụng bạn cần phải truyền vào data cho một property nào đó. Đối với DOM chúng ta có 2 khái niệm khác nhau là property và attribute. À mà khoan DOM là gì? Khi browser load trang web của bạn, nó sẽ parse phần HTML và xây dựng nên một cây Document Object Model (DOM) từ đó để biểu diễn tương ứng những gì HTML đang được dựng, cho phép chúng ta có thể tương tác với phần HTML như đọc, sửa HTML bằng JavaScript. Giả sử khi bạn có phần HTML: Sau khi parse xong sẽ có một object (node) thuộc type HTMLInputElement được tạo ra. Ở đây type=”text” hay value=”something” là các HTML attribute. Mỗi tag HTML có thể có nhiều attribute khác nữa (xin mời bạn Search Google). Object được tạo tương ứng sẽ có dạng

obj = {

type: ‘text’,

value: ‘something’,

attributes: [] // thuộc type NamedNodeMap, dạng như một array

// … các thuộc tính, method khác

}

Như bạn có thể thấy attribute được để chỉ những gì các bạn đặt vào phần opening tag của một tag sẽ là attributes, còn những gì là property của object (node) sẽ được gọi là property. Các attributes sẽ được lưu trữ vào property attributes của node tương ứng. Có những attribute sẽ được map sang thành property tương ứng, ví dụ như type và value ở trên, nhưng có những attribute không được map sang ví dụ class sẽ thành className, chúng không có quan hệ 1-1 với nhau. Thực tế khi sử dụng Angular, chúng ta sẽ muốn ứng dụng trở nên linh động hơn (ví dụ: cùng là 1 template nhưng có thể thay đổi data để hiển thị). Lúc này chúng ta cần update các properties của DOM để làm cho nó đáp ứng chẳng hạn. Nhưng nếu cần phải dùng document.querySelector/jQuery để manipulate DOM thì Angular quá vô vị. Phải có cách gì đó hay ho hơn chứ nhỉ? Đó chính là đất diễn của property binding. Chúng ta chỉ cần sử dụng một cú pháp để báo cho Angular biết chúng ta cần binding một property như sau ở template.

<input type="text" [value]="user.name" />

Trong dự án Angular, bạn hoàn toàn có thể hiểu type="text" lúc này cũng là một property binding, thay vì [type]="'text'" (để ý có 1 cặp nháy đơn và 1 cặp nhảy đôi), nó đang biểu diễn binding cho một hằng giá trị (literal), lúc này bạn hoàn toàn có thể bỏ qua mấy dấu vuông vuông kia đi. Nhưng trong hầu hết các trường hợp còn lại, bạn sẽ dùng dấu vuông vuông [property]="expression" để thực hiện khai báo property binding. Trong ví dụ ở trên chúng ta đã binding từ TS class ra ngoài template, và khi dữ liệu ở class thay đổi, Angular sẽ tự động làm nhiệm vụ update template để hiển thị tương ứng sự thay đổi đó cho chúng ta.

**Lưu ý**: ngoài property binding cho các phần tử HTML, chúng ta cũng có thể áp dụng property binding cho các component.

**EVENT BINDING**

JavaScript sử dụng rất nhiều đến khái niệm event. Khi một điều gì đó xảy ra, chúng ta muốn thực hiện một số task nào đó. Ví dụ, khi người dùng click vào button, tôi muốn hiển thị alert cho người dùng nhìn thấy. Angular có cách nào chính thống cho việc này không đây, hay chúng ta sẽ dùng addEventListener như thường? Câu trả lời chính là Event binding. Để gắn event listener vào một phần tử nào đó ở trên template, chúng ta có thể sử dụng cú pháp ngoặc tròn tròn như sau:

@Component({

selector: "app-hello",

template: `

<h2>Hello there!</h2>

<button (click)="showInfo()">Click me!</button>

`,

})

export class HelloComponent {

showInfo() {

alert("Inside Angular Component method");

}

}

Chỉ là lúc này chúng ta sẽ trỏ đến method bên trong Component. Khá là giống như chúng ta sử dụng inline event listener đó.

<button onclick="showInfo()">Click me!</button>

**TWO-WAY BINDING**

Trong thực tế two-way binding chính là kết hợp của binding dữ liệu từ class ra template thông qua property binding, và từ template vào class thông qua event binding. Nó chứa cú pháp ngắn gọn dạng vuông vuông tròn tròn như sau:

<input type="text" [(ngModel)]="user.name" />

Để sử dụng ngModel bạn cần imports FormsModule, nhưng trong bài này chúng ta chỉ cần hiểu, nó là cách viết tắt của dạng tương ứng là:

<input type="text" [ngModel]="user.name" (ngModelChange)="user.name = $event" />

**Tóm lại 2 binding: là thay đổi giá trị của 1 biến ở html thì giá trị của biến đó**

**ở ts cũng thay đổi và ngược lại.**

**Local Storage: Dùng để lưu trữ dữ liệu của trang web – dưới dạng key-value**

***Key: string, value: string***

**Bộ nhớ 5mb**

## Session Storage

## Dùng như local storage nhưng khác nhau kích thước bộ nhớ và khả năng lưu trữ.

***Key: string, value: string***

Xử lý realtime = signalR – cho asp.net core – angular