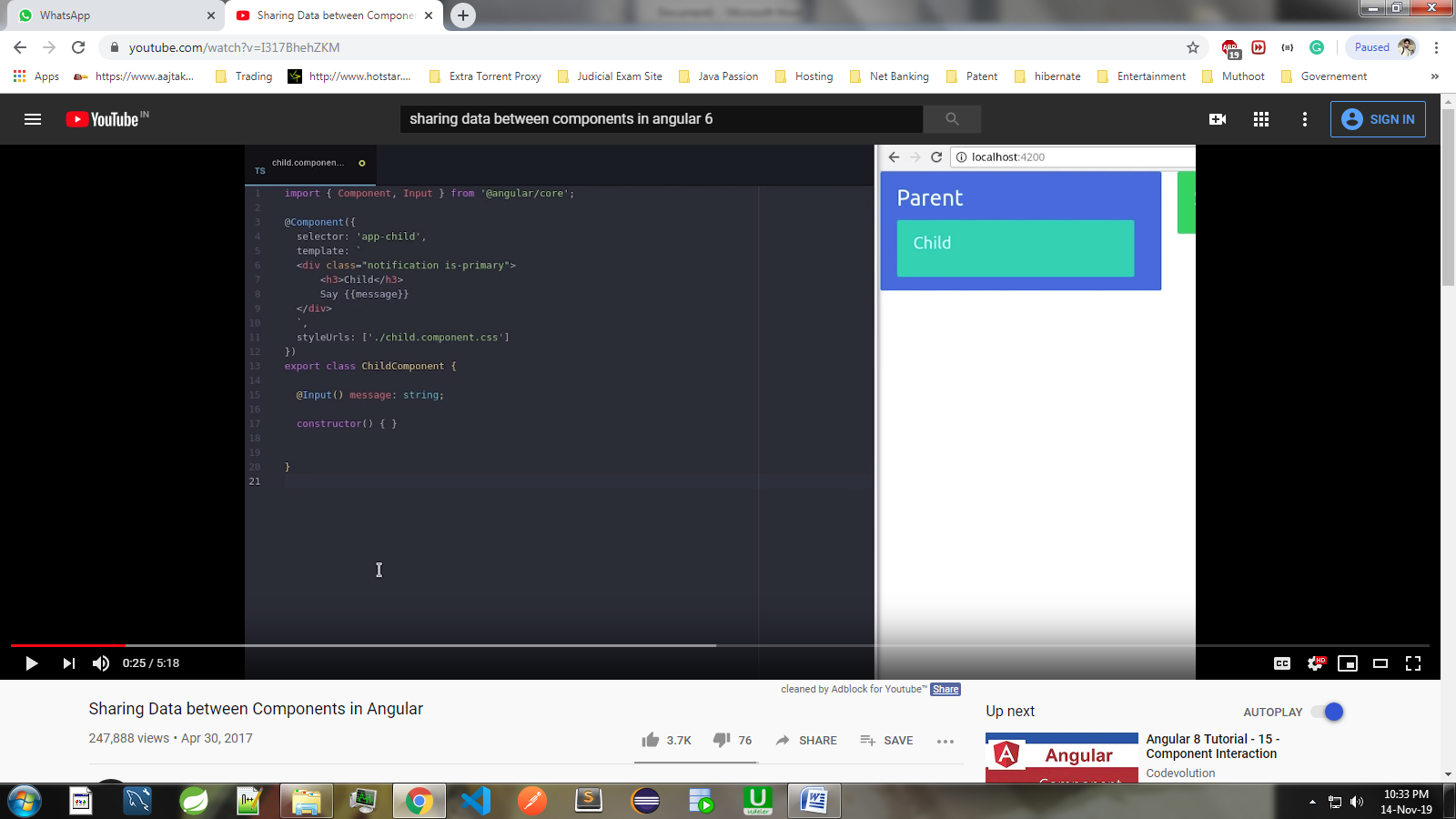
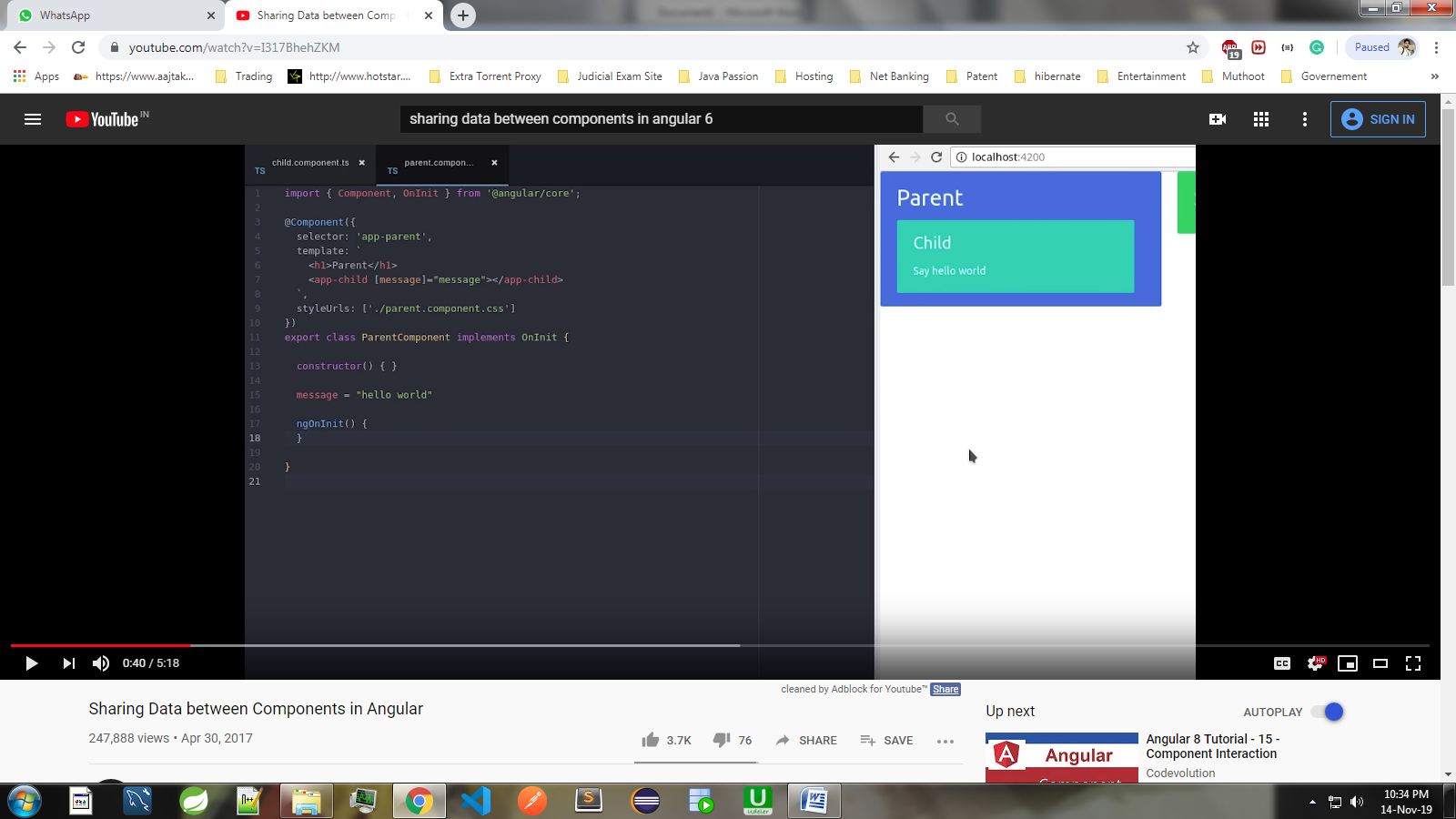
**Angular Communication:**

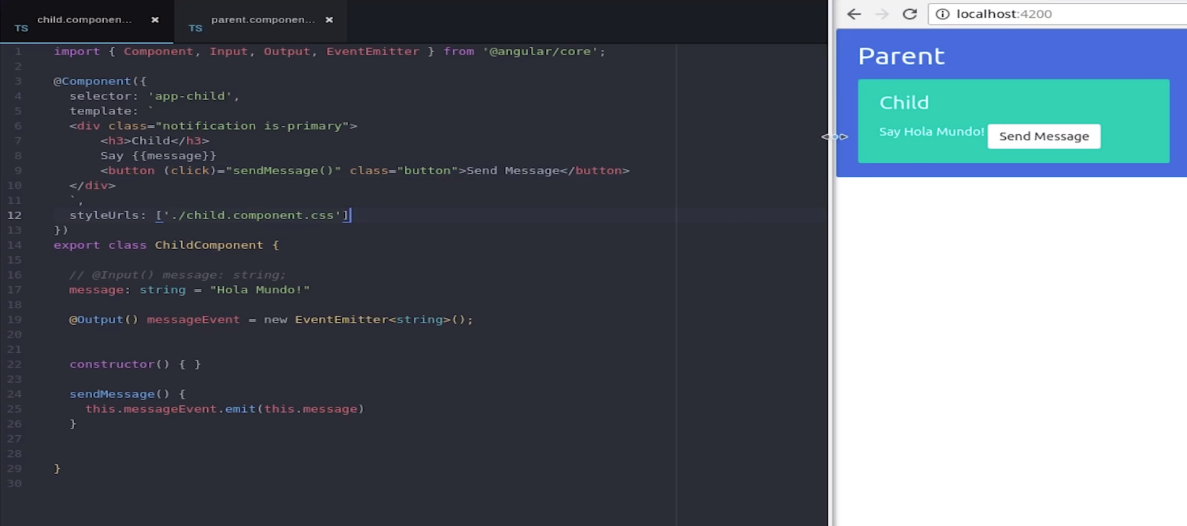
1. **Using @Input (Parent ----->>>>>Child)**

**Parent Component Child Component**

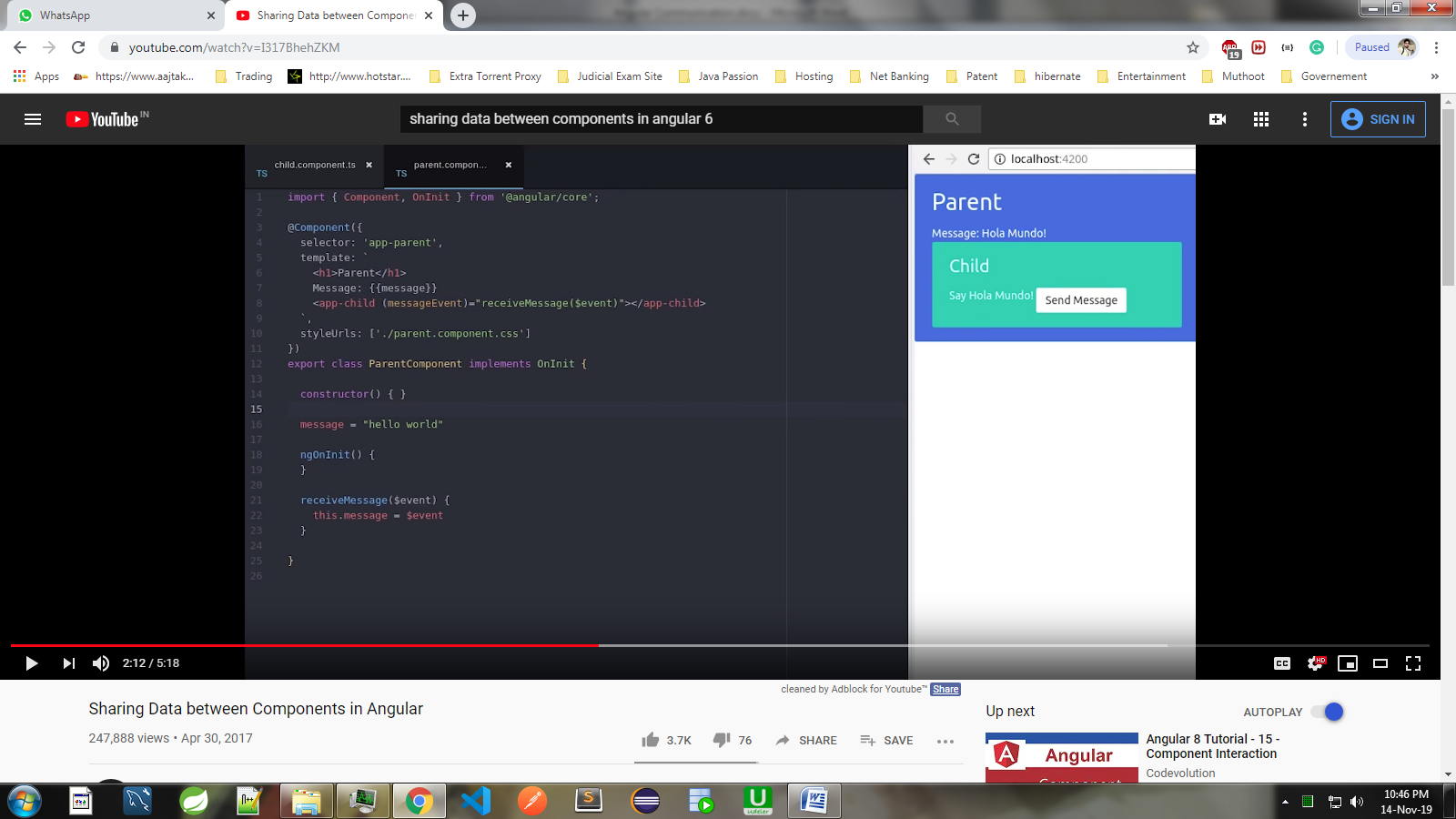
****

1. **Using @Output + EventEmitter<string> (Child--->>>Parent)**

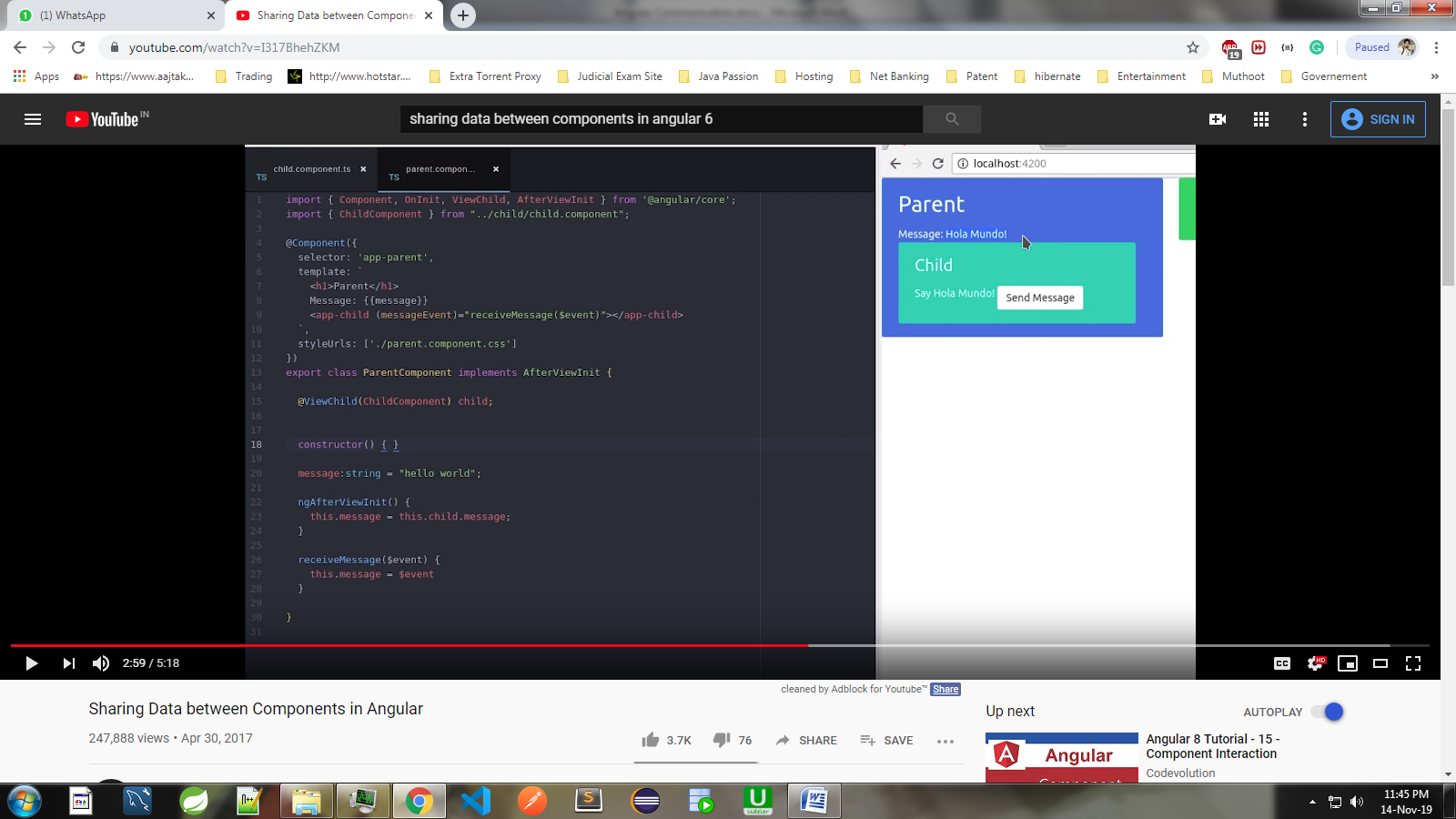
**a) Child Component**

****

**b) Parent Component**

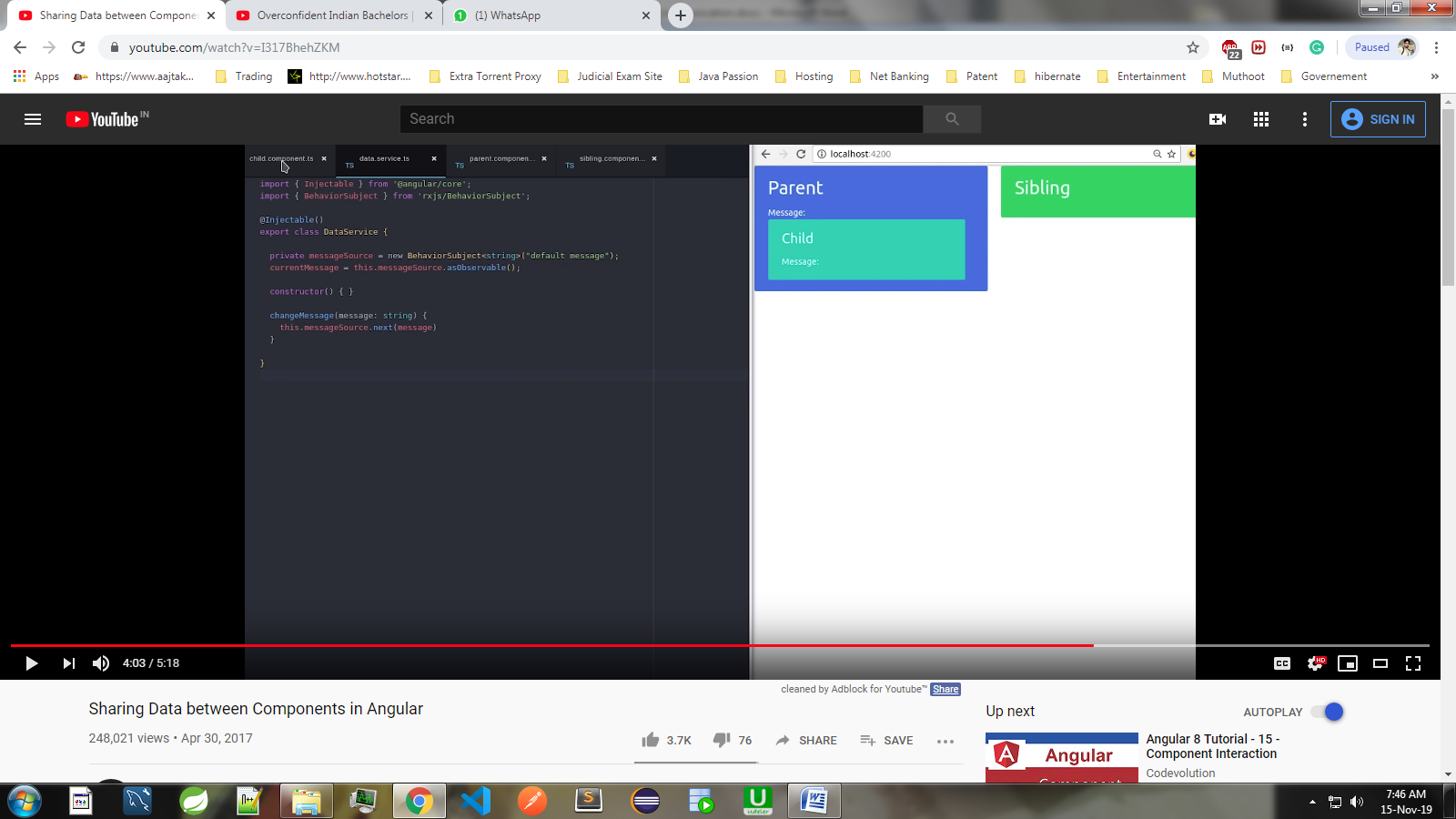
****

1. **Using @ViewChild (Child--->>>Parent)**

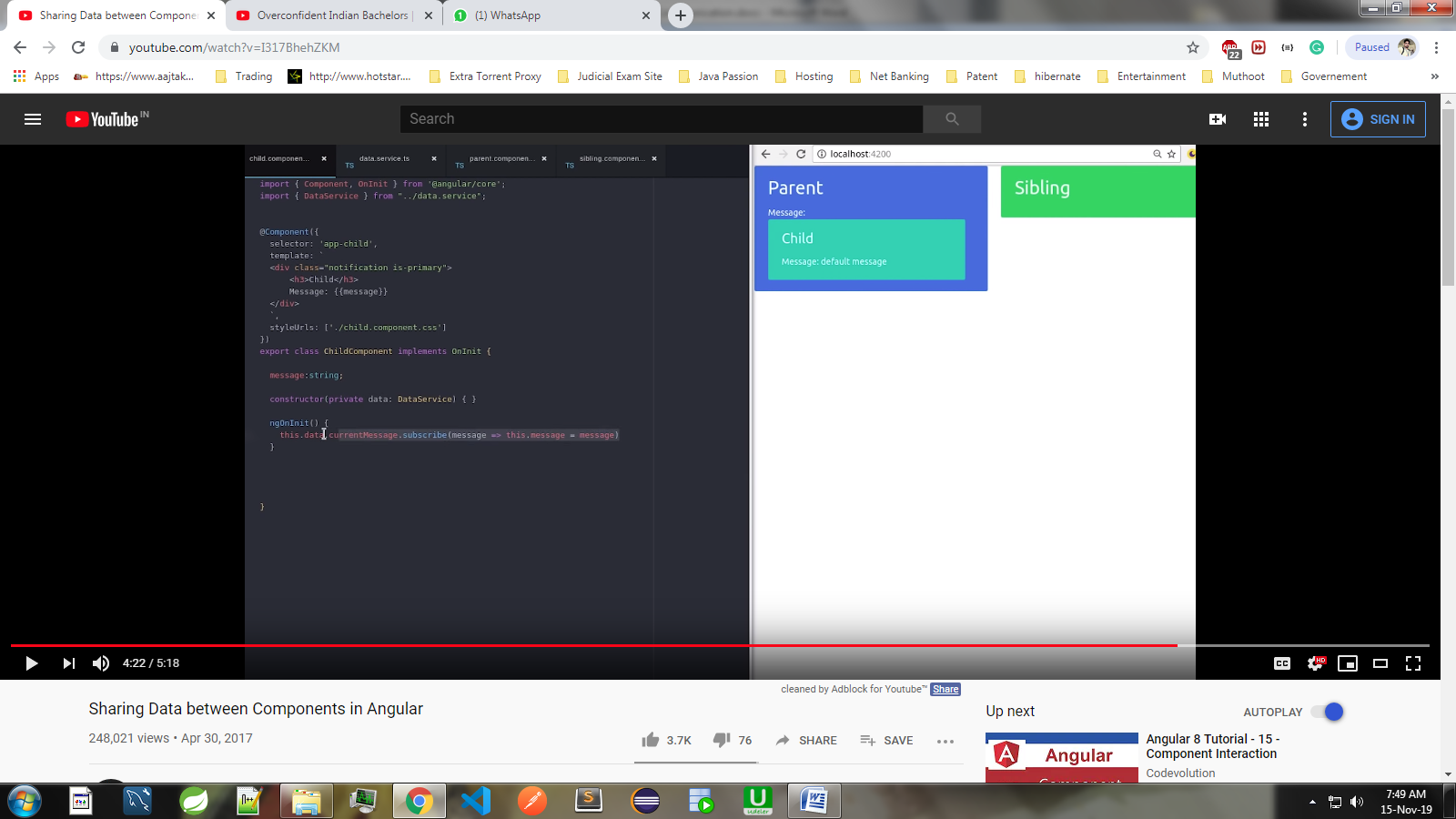
****

**4. Using BehaviourSubject<string> (Comp1 -->> Comp2) or (Comp2 -->> Comp1)**

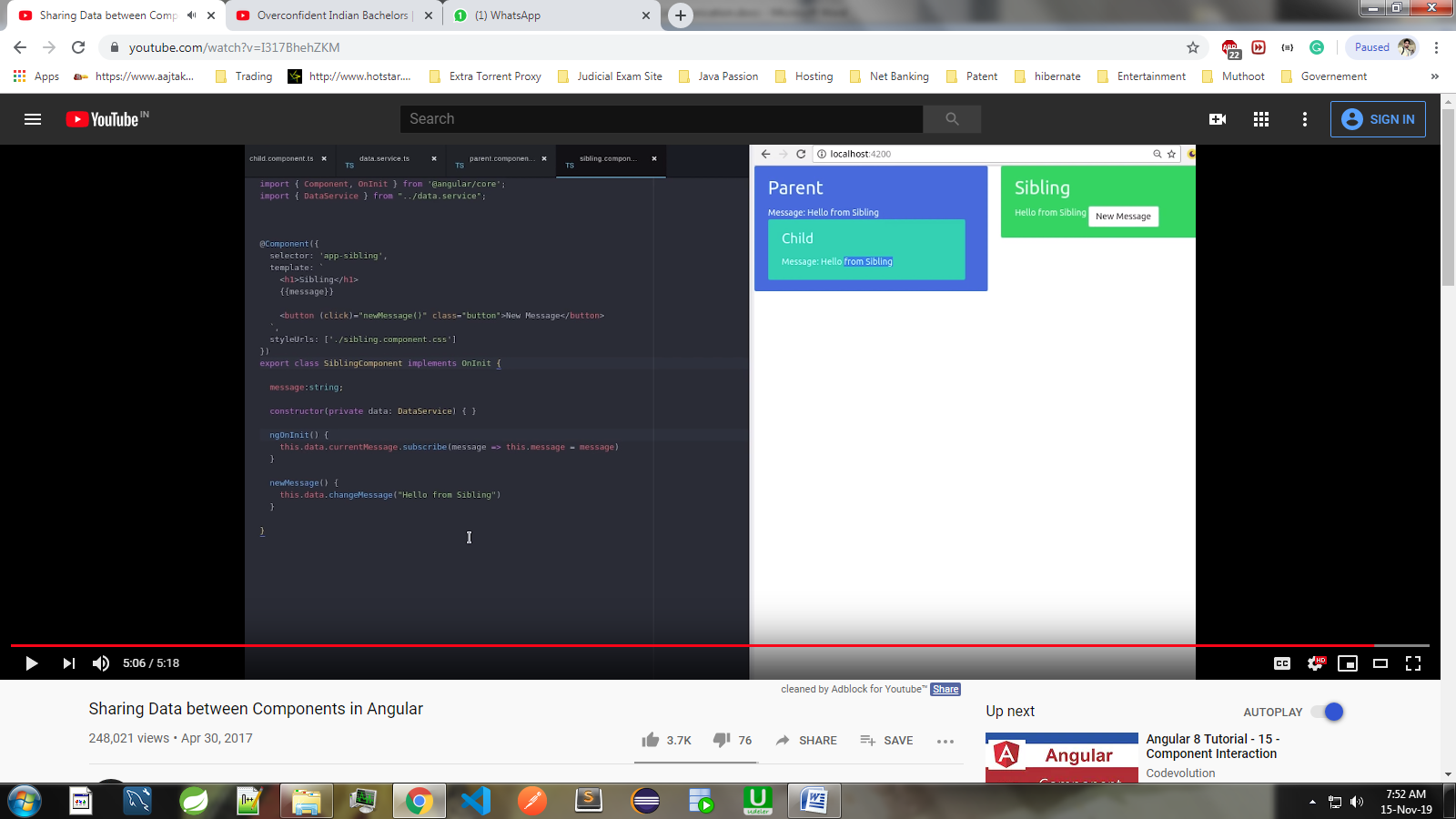
**We make common service of BehaviourSubject<string>**

****

**Child Component**

****

**Parent Component:**

****

**Var , Let and Const:**

constructor(){

var a; Note: I can declare first and later i can initialised

a = 10;

let b;

b = 10;

const c = 30; Note: constant has to be initialised while declaring

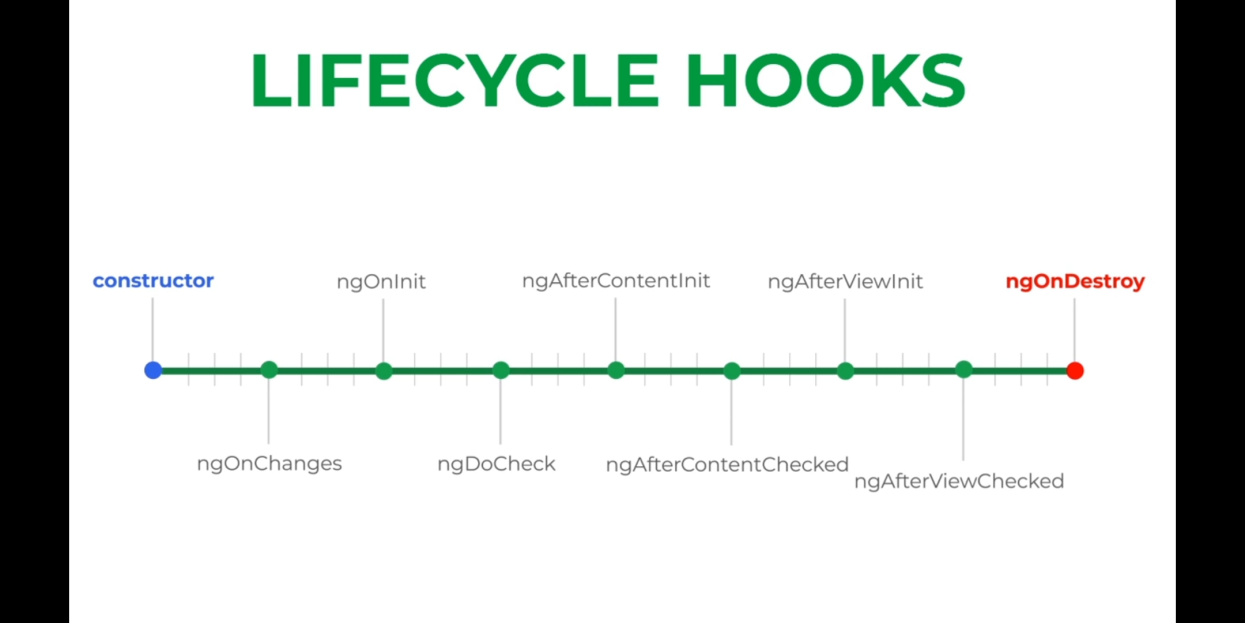
}

***Re-initialising*:**

|  |  |  |
| --- | --- | --- |
| **Var** | **Let** | **Const** |
| constructor(){  var a=10;  console.log(a) ; // 10  a = 20;  console.log(a) ; // 20  } | constructor(){  let a=10;  console.log(a) ; // 10  a = 20;  console.log(a) ; // 20  } | constructor(){  const a=10;  console.log(a) ; // 10  a = 20;  //Error ,you cannot re-initialised  console.log(a) ;  } |
| **showData()**  **{**  **var a=10;**  **console.log(a); //10**  **if(true)**  **{**  **var a=20;**  **console.log(a); //20**  **a=30;**  **console.log(a) // 30**  **}**  **console.log(a); //30**  **}** | **showData()**  **{**  **let a=10;**  **console.log(a); //10**  **if(true)**  **{**  **let a=20;**  **console.log(a); //20**  **a=30;**  **console.log(a) // 30**  **}**  **console.log(a); //10**  **}** |  |

Component lifecycle hooks overview:

* Directive and component instances have a lifecycle as Angular creates, updates, and destroys them
* To see component lifecycle by implementing one or more of the lifecycle hook  interfaces in the Angular core library.
* Each interface has a single hook method whose name is the interface name prefixed with ng.
* No directive or component will implement all of the lifecycle hooks.



* Angular only calls a directive/component hook method if it is defined.
* The JavaScript language doesn't have interfaces.
* Angular is a platform and framework for building client applications in HTML and TypeScript.
* Angular is written in TypeScript.
* Both components and services are simply classes, with decorators that mark their type and provide metadata that tells Angular how to use them.
* Angular instead inspects directive and component classes and calls the hook methods if they are defined.
* Angular finds and calls methods like ngOnInit(), with or without the interfaces.

.