**INTRODUZIONE**

**Angular js segue un pattern simile all’MVC. Questo pattern si può definire come MV\*, in cui \* può essere un controller, una view, o qualsiasi altra cosa.**

**ESEMPIO BASE**

**CODICE HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>

    <title>Prima pagina angularJS</title>

</head>

<body>

    <div ng-app>

        <div ng-init="quantity=1; price=100;">

            QUANTITY: <input type="text" name="qty" ng-model="quantity"> <br><br>

            PRICE: <input type="text" name="prc" ng-model="price"> <br><br>

            <p>Total price is: {{quantity \* price}} </p>

        </div>

        <div ng-init="nome='Tizio';cognome='Caio';">

            NOME: <input type="text" name="name" ng-model="nome"> <br><br>

            COGNOME: <input type="text" name="surname" ng-model="cognome"> <br><br>

            <p>Full name is: {{nome + " " + cognome}} </p>

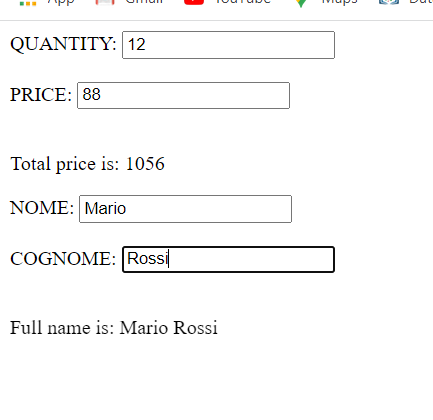
        </div>

    </div>

</body>

</html>

**BROSWER:**

****

**NB. Il linguaggio che si inserisce tra le doppie perentesi graffe è JAVASCRIPT!**

**Finora abbiamo considerato le variabili a se stanti. Consideriamo ora gli oggetti javascript:**

**CODICE HTML:**

<body>

    <div ng-app>

        <div ng-init="anagrafica = {nome:'Alfonso',cognome:'Pascucci'}">

            NOME: <input type="text" name="name" ng-model="anagrafica.nome">

            COGNOME: <input type="text" name="name" ng-model="anagrafica.cognome">

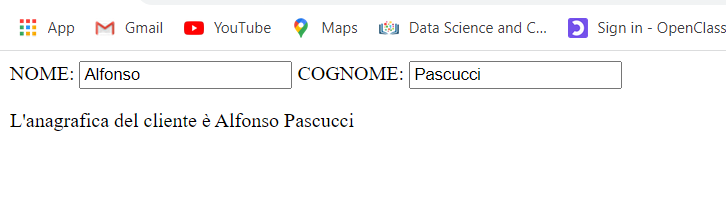
            <p>L'anagrafica del cliente è {{anagrafica.nome + " " + anagrafica.cognome}} </p>

        </div>

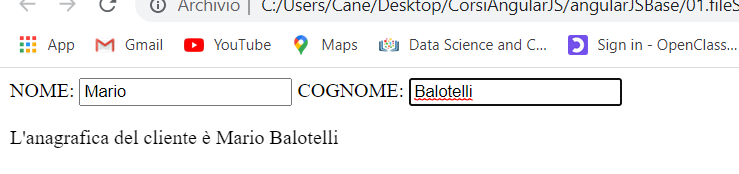
    </div>

</body>

**BROSWER:**

****

**Cambiando i nome all’interno degli input:**

****

**Vediamo ora un esempio con un array:**

**CODICE HTML:**

<body>

    <div ng-app>

        <div ng-init="numeri = [4,32,44,2,12]">

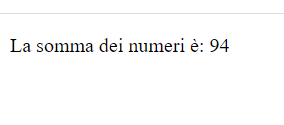
           <p>La somma dei numeri è: {{numeri[0]+ numeri[1] + numeri[2]+ numeri[3] + numeri[4]  }}</p>

        </div>

    </div>

</body>

**BROSWER:**

****

**Proviamo ora con un array di oggetti: Consideriamo ora i voti scritto e orale di 3 materie per lo studente**

**X: le materie sono italiano, matematica, storia.**

**CODICE HTML:**

<body>

    <div ng-app>

        <div ng-init="votiCarlo = {

            italiano: {scritto:9,orale:8},

            matematica: {scritto:9,orale:8},

            storia: {scritto:3,orale:1}

        }">

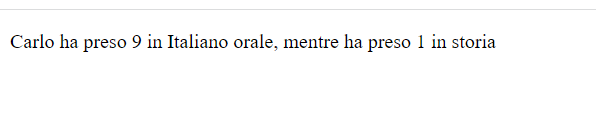
           <p>Carlo ha preso {{votiCarlo.italiano.scritto}} in Italiano orale, mentre ha preso {{votiCarlo.storia.orale}} in storia </p>

        </div>

    </div>

</body>

**BROSWER:**

****

**NB. E’ obbligatorio inizializzare le variabili all’interno dell’ng-init!!**

**Cosa succede se non lo facciamo?**

**Vediamo un esempio.**

**Consideriamo un esempio in cui c’è l’inizializzazione delle variabili:**

**CODICE HTML:**

<body>

    <div ng-app>

        <div ng-init="number1=3;number2=10;number3=4">

            INSERT N.1: <input type="number" name="num1" ng-model="number1">

            INSERT N.2: <input type="number" name="num1" ng-model="number2">

            INSERT N.3: <input type="number" name="num1" ng-model="number3">

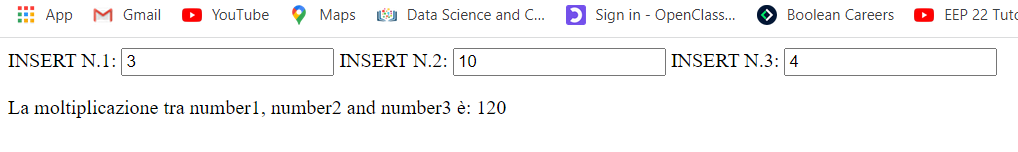
            <p>La moltiplicazione tra number1, number2 and number3 è: {{number1\*number2\*number3}}</p>

        </div>

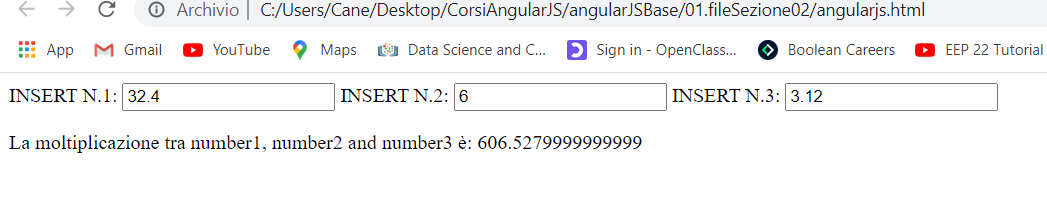
    </div>

</body>

**BROSWER:**

****

**Ovviamente, se cambiamo I valori negli input, cambierà anche il risultato:**

****

**Supponiamo ora di non inizializzare le variabili:**

**CODICE HTML:**

<body>

    <div ng-app>

        <div ng-init="number1;number2;number3">

            INSERT N.1: <input type="number" name="num1" ng-model="number1">

            INSERT N.2: <input type="number" name="num1" ng-model="number2">

            INSERT N.3: <input type="number" name="num1" ng-model="number3">

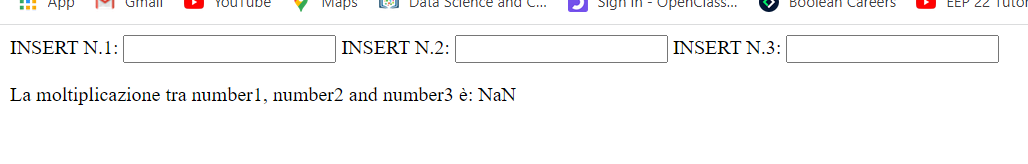
            <p>La moltiplicazione tra number1, number2 and number3 è: {{number1\*number2\*number3}}</p>

        </div>

    </div>

</body>

**BROSWER:**

****

**A differenza di JS, il risultato non è un’eccezione ma è Not A Number.**

**Se inserissimo anche solo uno degli input, il risultato darà 0.**

**STRUTTURE DI CONTROLLO**

**L’unica struttura di controllo accettata in angular è l’operatore ternario, in quanto il suo risultato può essere immagazzinato in una variabile. L’if e lo switch non vengono presi in considerazione.**

**Esempio (sempre quello dei numeri):**

**CODICE HTML:**

<body>

    <div ng-app>

        <div ng-init="number1=5;number2=4;number3=2">

            INSERT N.1: <input type="number" name="num1" ng-model="number1">

            INSERT N.2: <input type="number" name="num1" ng-model="number2">

            INSERT N.3: <input type="number" name="num1" ng-model="number3">

            <p>  {{(number1>number2) && (number1>number3) ? number1\*number2\*number3 : number1+number2+number3;}} </p>

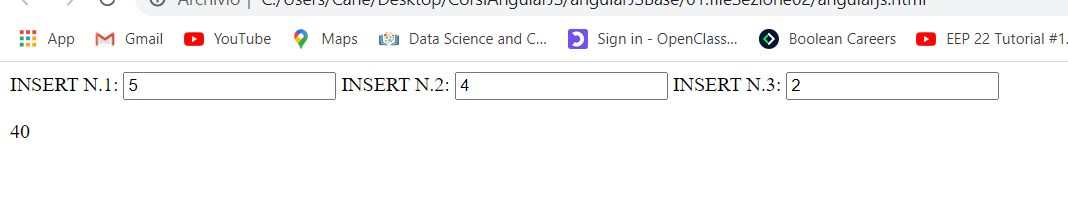
        </div>

    </div>

    <script src="app.js"></script>

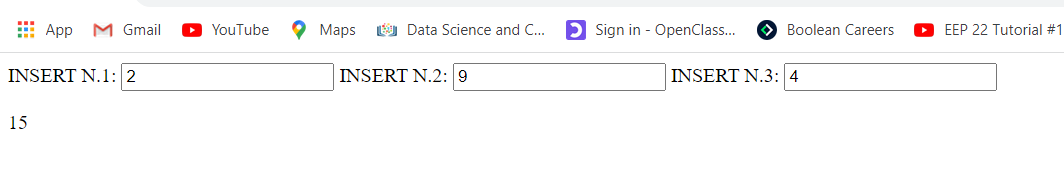
</body>

**BROSWER:**

****

**Poiché, di default, n1 è maggiore degli altri 2 🡺 Il programma restituisce la moltiplicazione tra di loro.**

**Provando a cambiare i numeri:**

****

**Ora il primo numero non è maggiore degli altri 🡺 Il programma restituisce la somma.**

**Perché angularjs non permette i control flow? Perché si basa sul pattern MVC. I controlli, i loop etc vengono inseriti nel controller o nel service.**