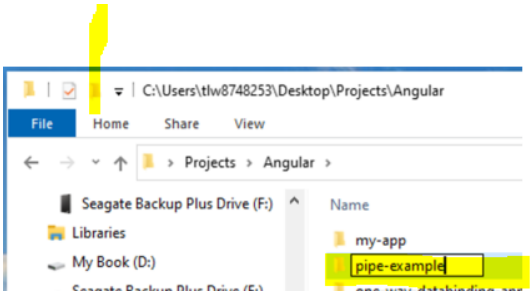
Table of Contents

Lecture and project for pipe-example

Copy project from parent-child-communication

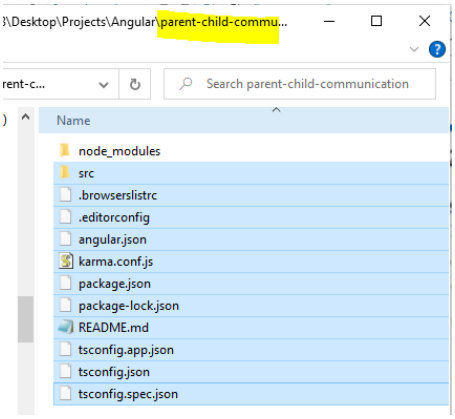
Create folder pipe-example

In your Angular project create a folder called pipe-example



Copy select files from last project

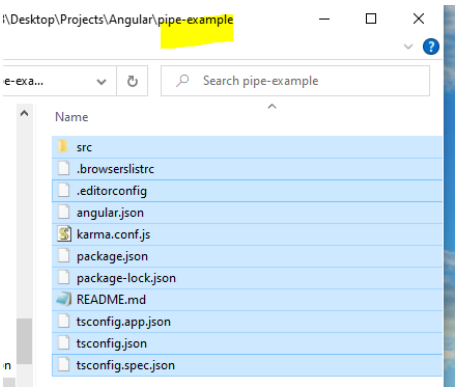
Select the following files from your parent-child-communication folder



Copy files ctrl-c

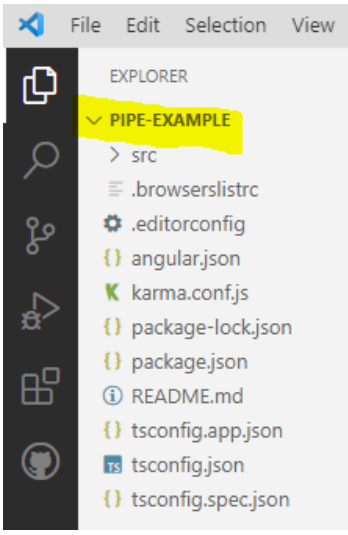
Move to pipe-example folder

Paste files ctrl-v



Update some project files

Open the project folder in VS Code



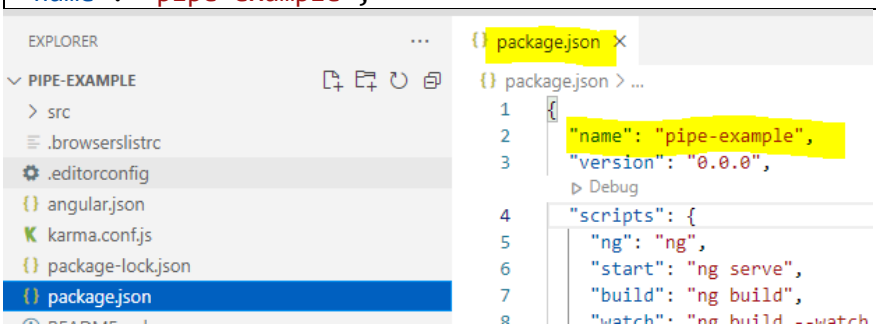
Update name in package.json file

From:

```
"name": "parent-child-communication",
```

To:

```
"name": "pipe-example",
```



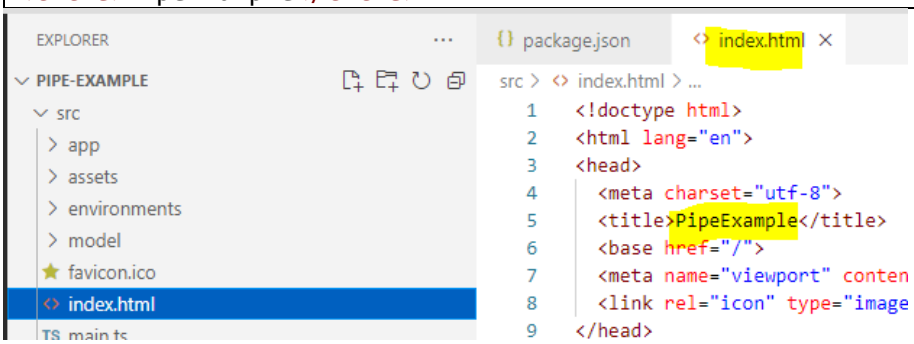
Update <title> in index.html file

From:

```
<title>ParentChildCommunication</title>
```

To:

```
<title>PipeExample</title>
```

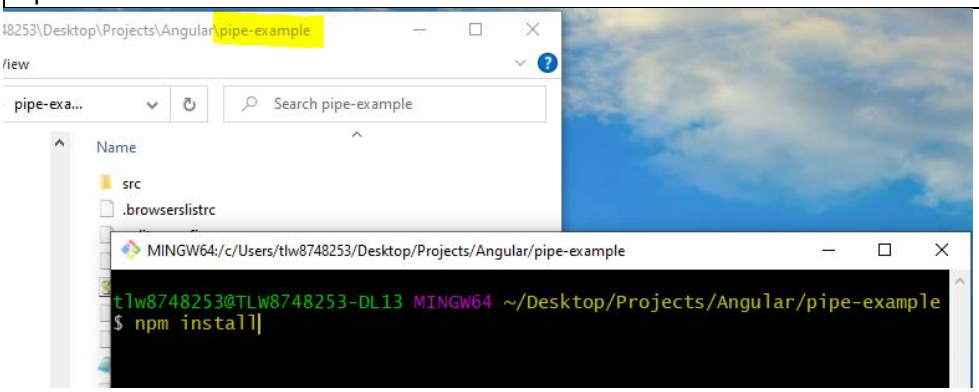


Install as Angular project

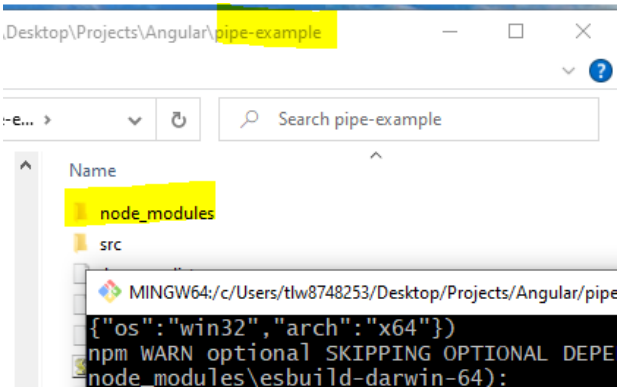
Open Git Bash window in project folder

Install project

```
npm install
```



Folder node_modules is created and project is installed



Project Refactoring

Start the project

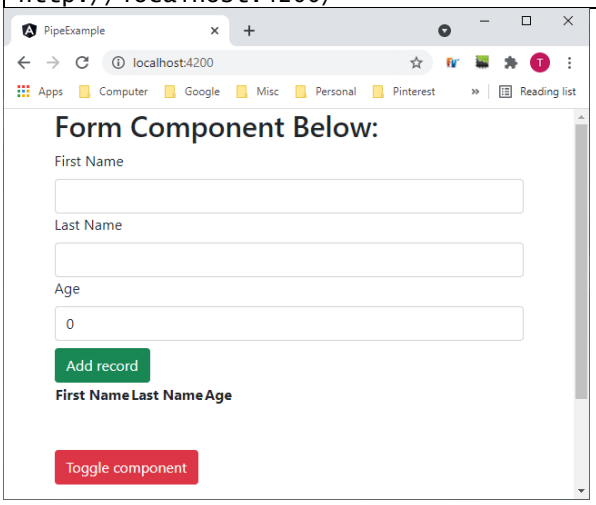
In Git Bash window

```
npm run start
```

```
MINGW64:/c/Users/tlw8748253/Desktop/Projects/Angular/pipe-example
tlw8748253@TLW8748253-DL13 MINGW64 ~/Desktop/Projects/Angular/pipe-example
$ npm run start

> pipe-example@0.0.0 start C:\Users\tlw8748253\Desktop\Projects\Angular\pi
pe-example
> ng serve

- Generating browser application bundles (phase: setup)...
Compiling @angular/core : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/forms : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
√ Browser application bundle generation complete.

Initial Chunk Files   | Names         |      Size
vendor.js             | vendor        |   2.39 MB
styles.css, styles.js | styles        | 543.80 kB
polyfills.js          | polyfills     | 510.58 kB
main.js               | main          |  28.19 kB
runtime.js            | runtime       |   6.62 kB

| Initial Total |   3.46 MB

Build at: 2021-10-27T14:29:20.868Z - Hash: eebd1139ab76895628ed - Time: 27678m
s

** Angular Live Development Server is listening on localhost:4200, open your b
rowser on http://localhost:4200/ **


√ Compiled successfully.
√ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2021-10-27T14:29:22.669Z - Hash: fdc1f133cd3d817ff158 - Time: 1165ms

√ Compiled successfully.
```

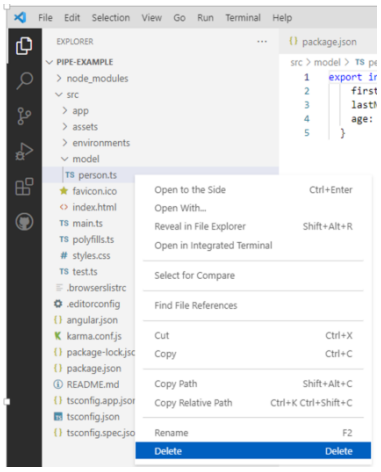
Open project in the browser

```
http://localhost:4200/
```



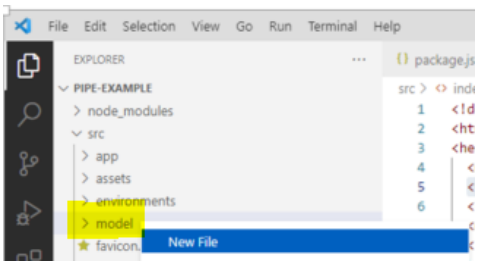
Refactor project files for this project

Delete person.ts



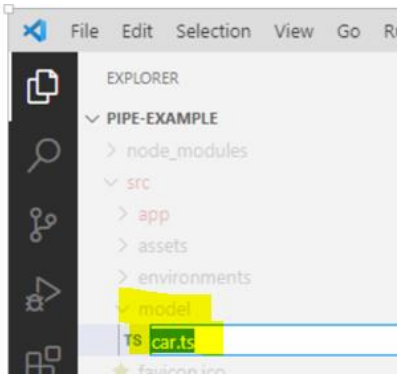
Create car.ts

Right mouse click on folder model



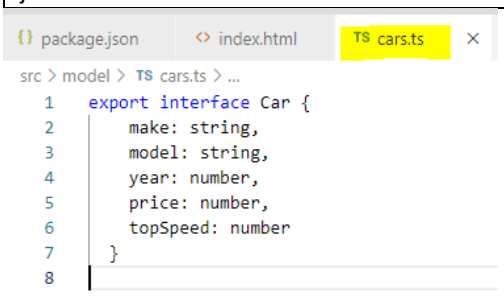
Select “New File”

Enter “car.ts”



Add the following to the cars.ts file

```
export interface Car {
  make: string,
  model: string,
  year: number,
  price: number,
  topSpeed: number
}
```



Refactor form.component.ts

Change the following lines of code:

From:

```
import { Person } from '../../model/person';
@Output('addPerson')
addPerson: EventEmitter<Person> = new EventEmitter();
```

To:

```
import { Car } from '../../model/car';
@Output('addCar')
addCar: EventEmitter<Car> = new EventEmitter();
```

```
TS form.component.ts 5 ×
src > app > form > TS form.component.ts > FormComponent
1  import { Component, EventEmitter, OnInit, Output } from '@angular/core';
2
3  import { Car } from '../../model/car';
4
5  @Component({
6    selector: 'app-form',
7    templateUrl: './form.component.html',
8    styleUrls: ['./form.component.css']
9  })
10 export class FormComponent implements OnInit {
11
12   @Output('addCar')
13   addCar: EventEmitter<Car> = new EventEmitter();
```

Replace the following lines of code:

Replace:

```
firstNameInputValue: string = "";
lastNameInputValue: string = "";
ageInputvalue: number = 0;
```

With:

```
makeInputValue: string = "";
modelInputValue: string = "";
yearInputvalue: number = 0;
priceInputvalue: number = 0;
topSpeedInputvalue: number = 0;
```

```
@Output('addCar')
addCar: EventEmitter<Car> = new EventEmitter();

makeInputValue: string = "";
modelInputValue: string = "";
yearInputvalue: number = 0;
priceInputvalue: number = 0;
topSpeedInputvalue: number = 0;
```

Update the addRecord() function

From:

```
addRecord() {
  let person: Person = {
```

```
      'firstName': this.firstNameInputValue,
      'lastName': this.lastNameInputValue,
      'age': this.ageInputvalue
    }

    this.addPerson.emit(person);
  }
```

To:

```
  addRecord() {
    let car: Car = {
      'make': this.makeInputValue,
      'model': this.modelInputValue,
      'year': this.yearInputvalue,
      'price': this.priceInputvalue,
      'topSpeed': this.topSpeedInputvalue
    }

    this.addCar.emit(car);
  }
```

```
addRecord() {
  let car: Car = {
    'make': this.makeInputValue,
    'model': this.modelInputValue,
    'year': this.yearInputvalue,
    'price': this.priceInputvalue,
    'topSpeed': this.topSpeedInputvalue
  }

  this.addCar.emit(car);
}
```

Refactor form.component.html

Update the component.html to reflect the car record.

Replace the existing HTML with:

```
<div>
    <label class="form-label">Make</label>
    <input [(ngModel)]="makeInputValue" class="form-control" type="text" />
  </div>
  <div>
    <label class="form-label">Model</label>
    <input [(ngModel)]="modelInputValue" class="form-control" type="text" />
  </div>
  <div>
    <label class="form-label">Year</label>
    <input [(ngModel)]="yearInputvalue" class="form-control" type="number" />
  </div>
  <div>
    <label class="form-label">Price</label>
    <input [(ngModel)]="priceInputvalue" class="form-control" type="number" />
  </div>
  <div>
    <label class="form-label">Top Speed (mph)</label>
    <input [(ngModel)]="topSpeedInputvalue" class="form-control" type="number" />
  </div>
  <div>
    <button (click)="addRecord()" class="btn btn-success mt-2">Add record</button>
  </div>
```

form.component.html

src > app > form > form.component.html > div

```
1  <div>
2      <label class="form-label">Make</label>
3      <input [(ngModel)]="makeInputValue" class="form-control" type="text" />
4  </div>
5  <div>
6      <label class="form-label">Model</label>
7      <input [(ngModel)]="modelInputValue" class="form-control" type="text" />
8  </div>
9  <div>
10     <label class="form-label">Year</label>
11     <input [(ngModel)]="yearInputvalue" class="form-control" type="number" />
12 </div>
13 <div>
14     <label class="form-label">Price</label>
15     <input [(ngModel)]="priceInputvalue" class="form-control" type="number" />
16 </div>
17 <div>
18     <label class="form-label">Top Speed (mph)</label>
19     <input [(ngModel)]="topSpeedInputvalue" class="form-control" type="number" />
20 </div>
21 <div>
22     <button (click)="addRecord()" class="btn btn-success mt-2">Add record</button>
23 </div>
```

Refactor app.component.html

Change the following lines of HTML:

From:

```
<app-form (addPerson)="onAddPerson($event)" *ngIf="formComponentShouldBeDisplayed"></app-form>
<app-table [myPeople]="people"></app-table>
```

To:

```
<app-form (addCar)="onAddCar($event)" *ngIf="formComponentShouldBeDisplayed"></app-form>
<app-table [myCars]="cars"></app-table>
```

app.component.html ×

src > app > app.component.html > app-switch-example

```
1  <div class="container">
2    <h1 [ngStyle]="{ 'visibility': formComponentShouldBeDisplayed ? 'visible' : 'hidden' }">Form Component Below:</h1>
3    <app-form (addCar)="onAddCar($event)" *ngIf="formComponentShouldBeDisplayed"></app-form>
4    <app-table [myCars]="cars"></app-table>
```

Refactor app.component.ts

Change the following lines of code:

From:

```
import { Person } from '../model/person';
people: Person[] = [];
```

To:

```
import { Car } from '../model/car';
cars: Car[] = [];
```

TS app.component.ts 3 ×

src > app > TS app.component.ts > ...

```
1  import { Component } from '@angular/core';
2
3  import { Car } from '../model/car';
4
5  @Component({
6    selector: 'app-root',
7    templateUrl: './app.component.html',
8    styleUrls: ['./app.component.css']
9  })
10 export class AppComponent {
11
12   cars: Car[] = [];
13
```

Update the onAddPerson event handler

From:

```
onAddPerson(event: Person) {
  this.people.push(event); // This event object will be a Person object

  console.log(this.people);
}
```

To:

```
onAddCar(event: Car) {
  this.cars.push(event); // This event object will be a Person object
}
```

```
onAddCar(event: Car) {
  this.cars.push(event); // This event object will be a Person object
}
```

Refactor table.component.ts

Update the following lines of code:

From:

```
import { Person } from '../../model/person';
@Input('myPeople')
myPeople: Person[] = [];
```

To:

```
import { Car } from '../../model/car';
@Input('myCars')
myCars: Car[] = [];
```

TS table.component.ts ×

src > app > table > TS table.component.ts > TableComponent > ngOnInit

```
1  import { Component, Input, OnInit } from '@angular/core';
2
3  import { Car } from '../../model/car';
4
5  @Component({
6    selector: 'app-table',
7    templateUrl: './table.component.html',
8    styleUrls: ['./table.component.css']
9  })
10 export class TableComponent implements OnInit {
11
12   @Input('myCars')
13   myCars: Car[] = [];
```

Refactor table.component.html

Replace the HTML with:

```
<table>
  <thead>
    <tr>
      <th>Make</th>
      <th>Model</th>
      <th>Year</th>
      <th>Price</th>
      <th>Top Speed</th>
     </tr>
  </thead>
  <tbody *ngFor="let car of myCars">
    <tr>
      <td>{{ car.make }}</td>
      <td>{{ car.model }}</td>
      <td>{{ car.year }}</td>
      <td>{{ car.price }}</td>
      <td>{{ car.topSpeed }}</td>
    </tr>
  </tbody>
</table>
```

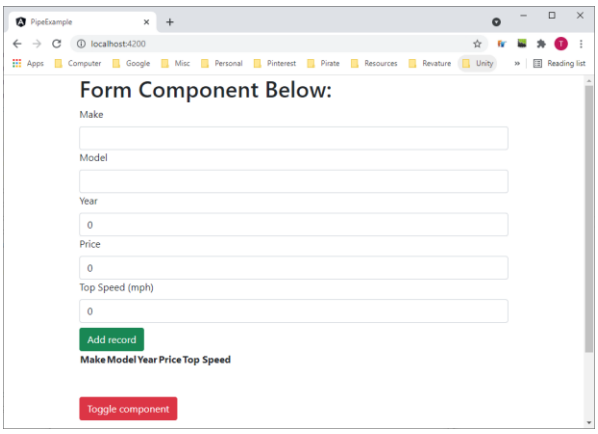
table.component.html

src > app > table > table.component.html > table

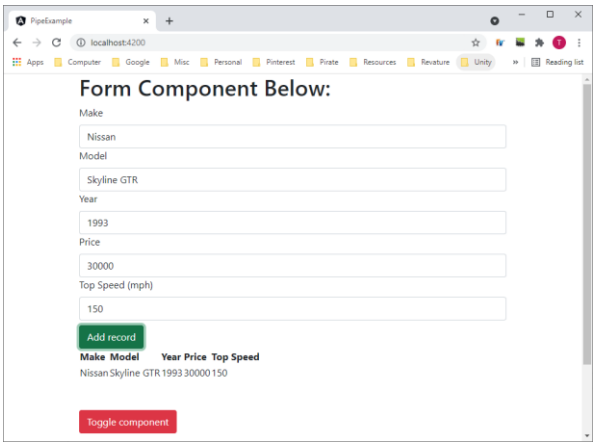
```html
<table>
  <thead>
    <tr>
      <th>Make</th>
      <th>Model</th>
      <th>Year</th>
      <th>Price</th>
      <th>Top Speed</th>
    </tr>
  </thead>
  <tbody *ngFor="let car of myCars">
    <tr>
      <td>{{ car.make }}</td>
      <td>{{ car.model }}</td>
      <td>{{ car.year }}</td>
      <td>{{ car.price }}</td>
      <td>{{ car.topSpeed }}</td>
    </tr>
  </tbody>
</table>
```

Refactoring is now complete

The page should now look as follows:



Test the page:



Transforming Data Using Pipes

https://angular.io/guide/pipes

CurrencyPipe

https://angular.io/api/common/CurrencyPipe

CurrencyPipe PIPE

Transforms a number to a currency string, formatted according to locale rules that determine group sizing and separator, decimal-point character, and other locale-specific configurations.

See more...

```
{{ value_expression | currency [ : currencyCode [ : display [ : digitsInfo [ : locale ] ] ] ] }}
```

Add pipe to price in table.component.html

Change the following line of HTML:

From:

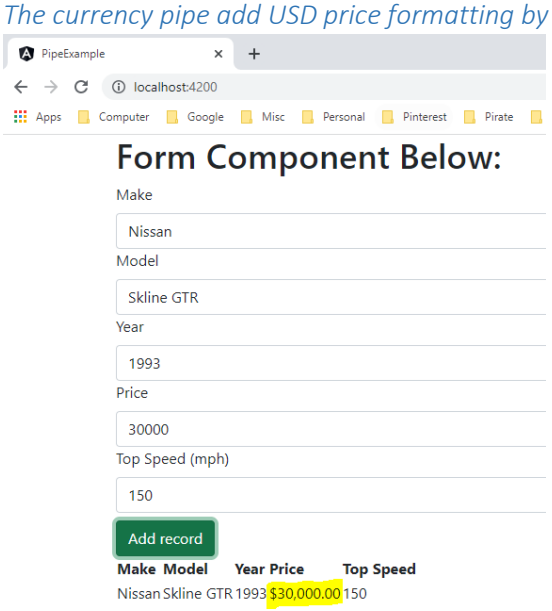
```
<td>{{ car.price }}</td>
```

To:

```
<td>{{ car.price | currency }}</td>
```

```
<tbody *ngFor="let car of myCars">
  <tr>
    <td>{{ car.make }}</td>
    <td>{{ car.model }}</td>
    <td>{{ car.year }}</td>
    <td>{{ car.price | currency }}</td>
    <td>{{ car.topSpeed }}</td>
  </tr>
</tbody>
```

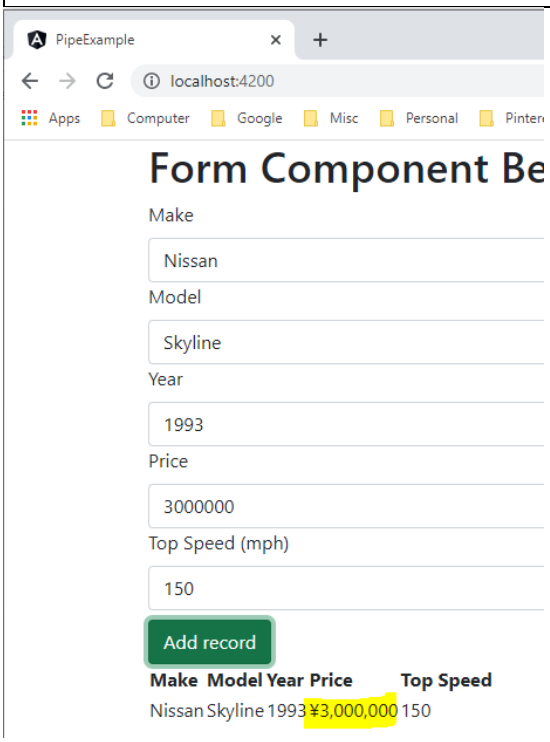
The currency pipe add USD price formatting by default



Update the pipe to price in table.component.html

Add currency change to the following line of HTML:

```
From:
        <td>{{ car.price | currency }}</td>
To:
        <td>{{ car.price | currency: 'JPY' }}</td>
```



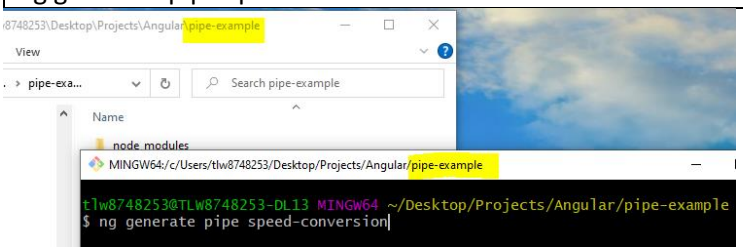
Custom Pipe: SpeedConversion

Create a custom pipe for conversion of speed

Open a Git Bash window in the project folder

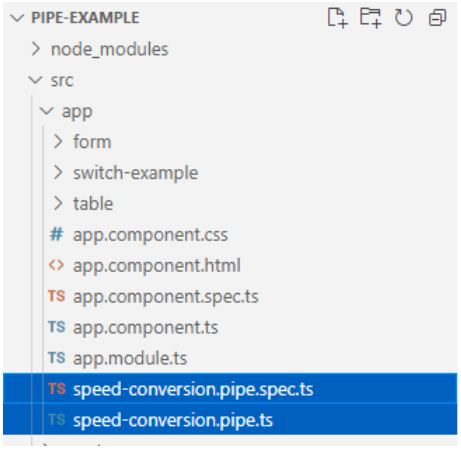
Enter the following command:

```
ng generate pipe speed-conversion
```




```
$ ng generate pipe speed-conversion
CREATE src/app/speed-conversion.pipe.spec.ts (224 bytes)
CREATE src/app/speed-conversion.pipe.ts (235 bytes)
UPDATE src/app/app.module.ts (729 bytes)
```

Speed conversion files created



Update default code in speed-conversion.pipe.ts

Add conversion code for the speed conversion desired

Replace:

```
transform(value: unknown, ...args: unknown[]): unknown {
  return null;
}
```

With:

```
transform(value: number, ...args: string[]): number {
  if (args[0] === 'kph') {
    return 1.60934 * value;
  } else if (args[0] === 'm/s') {
    return 0.44704 * value;
  } else if (args[0] === 'ft/s') {
    return 1.46667 * value;
  } else return value;
}
```

table.component.html speed-conversion.pipe.ts

src > app > speed-conversion.pipe.ts > ...

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'speedConversion'
})
export class SpeedConversionPipe implements PipeTransform {

  transform(value: number, ...args: string[]): number {
    if (args[0] === 'kph') {
      return 1.60934 * value;
    } else if (args[0] === 'm/s') {
      return 0.44704 * value;
    } else if (args[0] === 'ft/s') {
      return 1.46667 * value;
    } else return value;
  }

}
```

Add pipe to speed in table.component.html

Change / add the following line of HTML:

Change:

```
<th>Top Speed</th>
```

To:

```
<th>Top Speed (mph)</th>
```

Add:

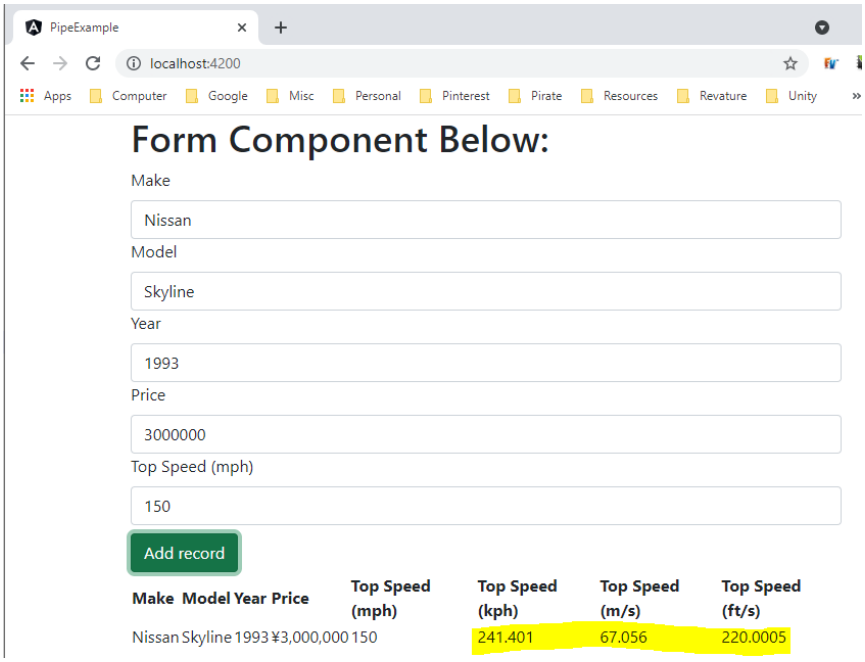
```
<th>Top Speed (kph)</th>
<th>Top Speed (m/s)</th>
<th>Top Speed (ft/s)</th>

<td>{{ car.topSpeed | speedConversion: 'kph' }}</td>
<td>{{ car.topSpeed | speedConversion: 'm/s' }}</td>
<td>{{ car.topSpeed | speedConversion: 'ft/s' }}</td>
```

table.component.html

src > app > table > table.component.html > ...

```
<table>
  <thead>
    <tr>
      <th>Make</th>
      <th>Model</th>
      <th>Year</th>
      <th>Price</th>
      <th>Top Speed (mph)</th>
      <th>Top Speed (kph)</th>
      <th>Top Speed (m/s)</th>
      <th>Top Speed (ft/s)</th>
    </tr>
  </thead>
  <tbody *ngFor="let car of myCars">
    <tr>
      <td>{{ car.make }}</td>
      <td>{{ car.model }}</td>
      <td>{{ car.year }}</td>
      <td>{{ car.price | currency: 'JPY' }}</td>
      <td>{{ car.topSpeed }}</td>
      <td>{{ car.topSpeed | speedConversion: 'kph' }}</td>
      <td>{{ car.topSpeed | speedConversion: 'm/s' }}</td>
      <td>{{ car.topSpeed | speedConversion: 'ft/s' }}</td>
    </tr>
  </tbody>
</table>
```

Test the custom pipe for speed conversion



Brief discussion on json pipe

For debugging can be useful to use the json pipe.

In this project we can add to the table.component.html file

```
{{ car | json }}
```

```
    <td>{{ car.topSpeed | speedConversion: 'ft/s' }}</td>
  </tr>

  {{ car | json }}

</tbody>
```

Produce the following on the webpage

Make	Model Year Price	Top Speed (mph)	Top Speed (kph)	Top Speed (m/s)	Top Speed (ft/s)
Nissan	Skyline 1993 ¥3,000,000	150	241.401	67.056	220.0005

{ "make": "Nissan", "model": "Skyline", "year": 1993, "price": 3000000, "topSpeed": 150 }