



Todo – App

Master/ detail

Ref : angular Hero App



- Display the list of selectable todos
- Add the functionality to selected the todo and show the todo in Detail view
- Use of built in directives ngIf, ngFor in the component's template

Todo - App

My Todos

- 1 Create Angular2 todo app
- 2 Take Rest WS Test
- 3 Create PPT for Angular todo
- 4 Check mails
- 5 Take Interview
- 6 Talk to Rakeshji
- 7 Prepare Questions for
- 8 Prepare Test
- 9 Talk to narendra sir
- 10 Plan for team lunch

Todo - App

My Todos

- 1 Create Angular2 todo app
- 2 Take Rest WS Test
- 3 Create PPT for Angular todo
- 4 Check mails
- 5 Take Interview
- 6 Talk to Rakeshji
- 7 Prepare Questions for
- 8 Prepare Test
- 9 Talk to narendra sir
- 10 Plan for team lunch

selected todo, in diff style

Take Rest WS Test - details!

id: 2

name: Take Rest WS Test

selected todo's detail
view

Write below command in the command line

```
npm start
```

This command runs the TypeScript compiler in "watch mode", recompiling automatically when the code changes. The command simultaneously launches the app in a browser and refreshes the browser when the code changes.

You can keep building the Todo – App without pausing to recompile or refresh the browser.

To display the list of todos, you need to add todos to the view template.

Create todos

Create an array of ten todos.

```
src/app/app.component.ts (Todo Array - named as TODOS)
-----
const TODOS : Todo[]=[
{ id:1, name : 'Create Angular2 todo app'},
{ id:2, name : 'Take Rest WS Test'},
{ id:3, name : 'Create PPT for Angular todo'},
{ id:4, name : 'Check mails'},
{ id:5, name : 'Take Interview'},
{ id:6, name : 'Talk to Rakeshji'},
{ id:7, name : 'Prepare Questions for Interview'},
{ id:8, name : 'Prepare Test'},
{ id:9, name : 'Talk to narendra sir'},
{ id:10, name : 'Plan for team lunch'}
];
```

Expose todos

Create a public property in *AppComponent* that exposes the todos for binding

```
src/app/app.component.ts (todos array property)
-----
export class AppComponent {
  title='Todo - App';
  todos=TODOES;
  todo: Todo = {
    id:1,
    name:'Create Angular 2 Todo app'
  }
}
```

Expose todos

Create a public property in *AppComponent* that exposes the todos for binding

```
src/app/app.component.ts (todos array property)
-----
export class AppComponent {
  title='Todo - App';
  todos=TODOES;
  todo: Todo = {
    id:1,
    name:'Create Angular 2 Todo app'
  }
}
```

The todo data is separated from the class implementation because ultimately the todo names will come from a data service.

Display todo names in a template

To display the todo names in an unordered list, insert the following chunk of HTML below the title and above the todo details.

```
src/app/app.component.ts (todos template)
```

```
-----  
@Component({  
  selector: 'my-app',  
  template: `  
    <h1>{{title}}</h1>  
    <h2>My Todos</h2>  
    <ul class="todos">  
      <li>  
        <!-- each todo will go here -->  
      </li>  
    </ul>  
    <h2>{{todo.name}} - details!</h2>  
    <div><label>id: </label>{{todo.id}}</div>  
    <div>  
      <label>name: </label>  
      <input [(ngModel)]="todo.name" placeholder="add task">  
    </div>  
  `,  
})
```

Now you can fill the template with todo names

List todos with ngFor

The goal is to bind the array of todos in the component to the template, iterate over them, and display them individually.

Modify the `` tag by adding the built-in directive `*ngFor`.

```
src/app/app.component.ts (ngFor)
-----
<li *ngFor="let todo of todos">
```

The (*) prefix to ngFor is a critical part of this syntax. It indicates that the `` element and its children constitute a master template.

The ngFor directive iterates over the component's todos array and renders an instance of this template for each todo in that array.

The `let todo` part of the expression identifies `todo` as the template input variable, which holds the current todo item for each iteration. You can reference this variable within the template to access the current todo's properties.

List todos with ngFor

Within the `` tags, add content that uses the `todo` template variable to display the `todo`'s properties.

```
src/app/app.component.ts (ngFor)
-----
<li *ngFor="let todo of todos">
  <span class="badge">{{todo.id}}</span>{{todo.name}}
</li>
```

Save the changes and check your browser.

Put some style for todos

Users should get a visual cue of which todo they are hovering over and which todo is selected.

To add styles to your component, set the styles property on the @Component decorator to the following CSS classes:

```
src/app/app.component.ts (style)
-----
styles : [`
.todos{
margin: 0 0 2em 0;
list-style-type: none;
padding: 0;
width: 15em;
}
`]
```

Put this code just below the end of template ending backtick. Make sure that inside the [``] we have placed backtick, so that multiline code can be written.

Now with this change, save and check your browser.

This will remove the bullets from each li and put some style for each todo

Put some more style for todos

src/app/app.component.ts (style)

```
-----  
styles : [`  
-----  
.todos li {  
  cursor: pointer;  
  position: relative;  
  left: 0;  
  background-color: #EEE;  
  margin: .5em;  
  padding: .3em 0;  
  height: 1.6em;  
  border-radius: 4px;  
}  
  
.todos li.selected:hover {  
  background-color: #BBD8DC !important;  
  color: white;  
}  
`]
```

Put some more style for todos

```
src/app/app.component.ts (style)
```

```
-----  
styles : [`  
-----  
  .todos li:hover {  
    color: #607D8B;  
    background-color: #DDD;  
    left: .1em;  
  }  
  
  .todos .text {  
    position: relative;  
    top: -3px;  
  }  
  
`]
```

Put some more style for todos

```
src/app/app.component.ts (style)
```

```
-----  
styles : [`  
-----  
.todos .badge {  
display: inline-block;  
font-size: small;  
color: white;  
padding: 0.8em 0.7em 0 0.7em;  
background-color: #607D8B;  
line-height: 1em;  
position: relative;  
left: -1px;  
top: -4px;  
height: 1.8em;  
margin-right: .8em;  
border-radius: 4px 0 0 4px;  
}  
`]
```

Now save the changes and check your browser.

The app now displays a list of todos as well as a single todo in the details view. But the list and the details view are not connected. When users select a todo from the list, the selected todo should appear in the details view. This UI pattern is known as "*master/detail*." In this case, the **master** is the todos list and the **detail** is the selected todo.

Now we will connect the master to the detail through a `selectedTodo` component property, which is bound to a click event.

Handle click Event

Add a click event binding to the `` like this:

```
src/app/app.component.ts (click event)
```

```
-----  
<li *ngFor="let todo of todos" (click)="onSelect(todo)">  
  <span class="badge">{{todo.id}}</span>{{todo.name}}  
</li>
```

The parentheses identify the `` element's click event as the target.

The `onSelect(todo)` expression calls the AppComponent method, `onSelect()`, passing the template input variable `todo`, as an argument. That's the same `todo` variable you defined previously in the `ngFor` directive.

Add a click handler to expose the selected todo

You no longer need the `todo` property because you're no longer displaying a single todo; you're displaying a list of todos. But the user will be able to select one of the todos by clicking on it. So replace the `todo` property with this simple `selectedTodo` property:

```
src/app/app.component.ts (selectedTodo)
-----
export class AppComponent {
  title='Todo - App';
  todos=TODOs;
  selectedTodo:Todo;
  todo: Todo = {
    id:1,
    name:'Create Angular 2 Todo app'
  }
}
```

You no longer need the `todo` property because you're no longer displaying a single todo; you're displaying a list of todos. But the user will be able to select one of the todos by clicking on it. So replace the `todo` property with this simple `selectedTodo` property:

```
src/app/app.component.ts (selectedTodo)
```

```
-----
```

```
export class AppComponent {  
  title='Todo - App';  
  todos=TODOs;  
  selectedTodo:Todo;  
  onSelect(todo: Todo):void {  
    this.selectedTodo=todo;  
  }  
}
```

The template still refers to the old todo property. Bind to the new selectedTodo property instead as follows:

```
src/app/app.component.ts (template)
-----
<h2>{{selectedTodo.name}} - details!</h2>
<div><label>id: </label>{{selectedTodo.id}}</div>
<div>
<label>name: </label>
<input [(ngModel)]="selectedTodo.name" placeholder="add task">
</div>
```

Save changes and see browser, list will not be displayed, and detail view will be present but empty, now let us hide the detail view and show the list back.

Hide the empty detail with ngIf

When the app loads, `selectedTodo` is undefined. The selected todo is initialized when the user clicks a todo's name. Angular can't display properties of the undefined `selectedTodo` and throws the following error, visible in the browser's console:

ERROR TypeError: Cannot read property 'name' of undefined

Although `selectedTodo.name` is displayed in the template, you must keep the todo detail out of the DOM until there is a selected todo.

Hide the empty detail with ngIf

Wrap the HTML todo detail content of the template with a `<div>`. Then add the `ngIf` built-in directive and set it to the `selectedTodo` property of the component.

```
src/app/app.component.ts (template)
-----
<div *ngIf="selectedTodo">
<h2>{{selectedTodo.name}} - details!</h2>
<div><label>id: </label>{{selectedTodo.id}}</div>
<div>
<label>name: </label>
<input [(ngModel)]="selectedTodo.name" placeholder="add task">
</div>
</div>
```

The app no longer fails and the list of names displays again in the browser.

When there is no selected todo, the `ngIf` directive removes the todo detail HTML from the DOM. There are no todo detail elements or bindings to worry about.

When the user picks a todo, `selectedTodo` becomes defined and `ngIf` puts the todo detail content into the DOM and evaluates the nested bindings.

Styling the Selected todo

When we select a particular todo, it is difficult to identify the selected todo from the list of todos.

In styles metadata add below code.

```
src/app/app.component.ts (styles)
-----
.selected {
background-color: #CFD8DC !important;
color: white;
}
```

Our objective is to highlight the selectedTodo:

My Todos

- 1 Create Angular2 todo app
- 2 Take Rest WS Test
- 3 Create PPT for Angular todo

Styling the Selected todo

In the template, add the following [class.selected] binding to the :

```
src/app/app.component.ts (template)
-----
<li *ngFor="let todo of todos"
  [class.selected]="todo === selectedTodo"
  (click)="onSelect(todo)">
  <span class="badge">{{todo.id}}</span>{{todo.name}}
</li>
```

When the expression (todo=== selectedTodo) is true, Angular adds the selected CSS class. When the expression is false, Angular removes the selected class.

Thank You!

Email: info@yash.com

Web: www.yash.com

© YASH Technologies, 1996-2013. All rights reserved.

The information in this document is based on certain assumptions and as such is subject to change. No part of this document may be reproduced, stored or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of YASH Technologies Inc. This document makes reference to trademarks that may be owned by others. The use of such trademarks herein is not as assertion of ownership of such trademarks by YASH and is not intended to represent or imply the existence of an association between YASH and the lawful owners of such trademarks.