

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS


**ROCK YOUR CODE** TOUR 2017 | **CHICAGO**

# Building Powerful Enterprise Apps with Angular and TypeScript

**David Giard**  
Microsoft Technical Evangelist  
blog: [DavidGiard.com](http://DavidGiard.com)  
tv: [TechnologyAndFriends.com](http://TechnologyAndFriends.com)  
twitter: [@DavidGiard](https://twitter.com/DavidGiard)  
Email: [dgiard@microsoft.com](mailto:dgiard@microsoft.com)

# Building Powerful Enterprise Apps

with Angular and TypeScript



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## David Giard

Microsoft Technical Evangelist  
blog: [DavidGiard.com](http://DavidGiard.com)  
tv: [TechnologyAndFriends.com](http://TechnologyAndFriends.com)  
twitter: [@DavidGiard](https://twitter.com/DavidGiard)  
Email: [dgiard@microsoft.com](mailto:dgiard@microsoft.com)



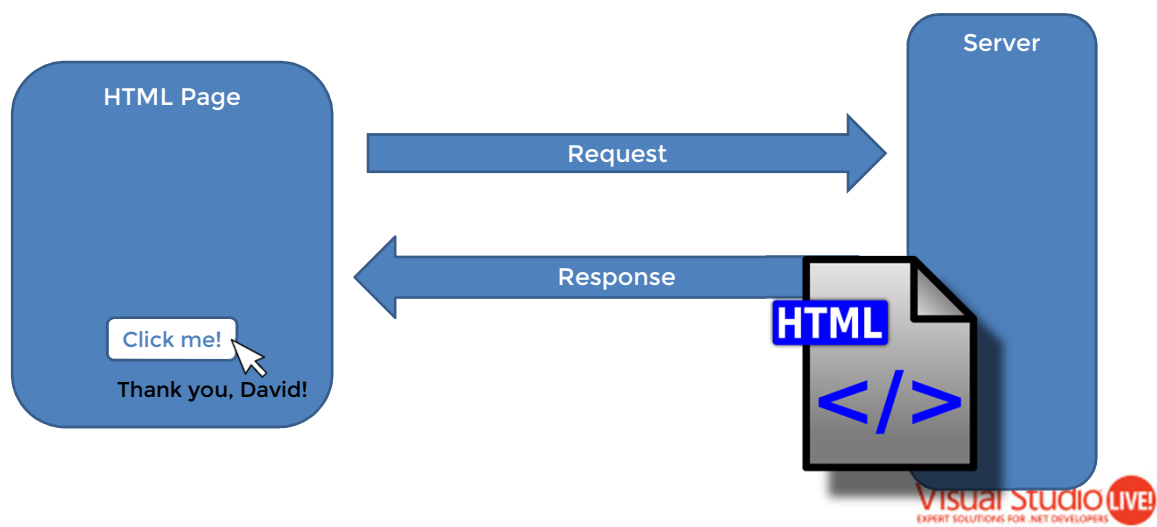
## Versions and Names

- Angular 1.x → AngularJS
- Angular 2.x, 4.x → Angular

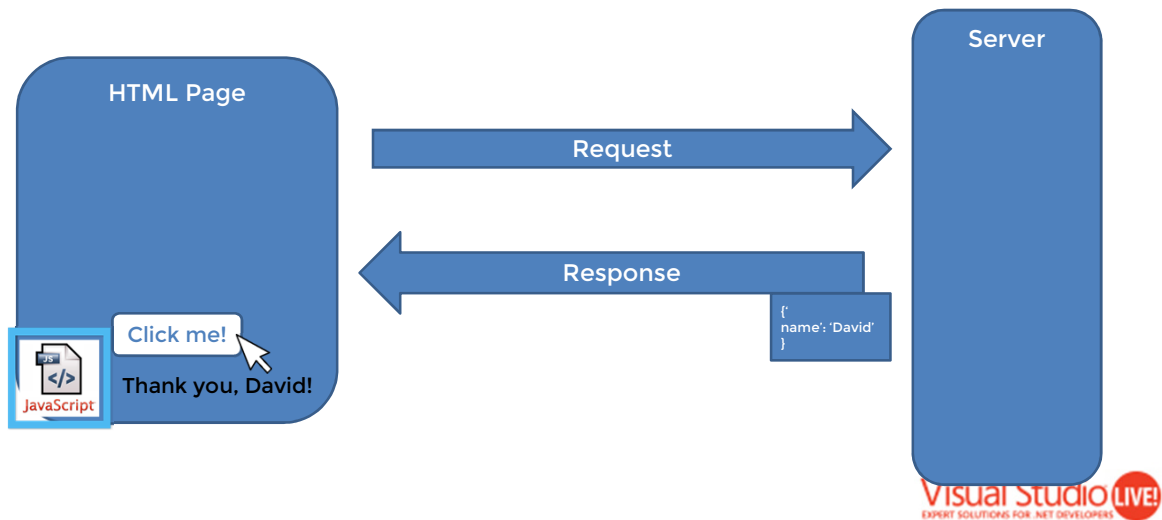
# Single Page Applications

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Traditional Web App



# Single Page App



# Angular

- SPA Framework
- Open Source
- Data Binding
- Components
- Modularize



# TypeScript

- Open Source
- Superset of JavaScript
- Transpiles to JavaScript



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# TypeScript

foo.ts

Transpile

foo.js

foo.map

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

```

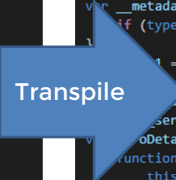
import { Component, EventEmitter, Input, OnInit, Output } from '@angular/core';
import { Hero } from './hero';
import { RouteParams } from '@angular/router-deprecated';
import { HeroService } from './hero.service';

@Component({
  selector: 'my-hero-detail',
  templateUrl: 'app/hero-detail.component.html',
  styleUrls: ['app/hero-detail.component.css']
})
export class HeroDetailComponent implements OnInit {
  constructor(
    private heroService: HeroService,
    private routeParams: RouteParams) {
  }

  @Input() hero: Hero;
  @Output() close = new EventEmitter();
  error: any;
  navigated = false; // true if navigated here

  ngOnInit() {
    if (this.routeParams.get('id') !== null) {
      let id = +this.routeParams.get('id');
      this.navigated = true;
      this.heroService.getHero(id)
        .then(hero => this.hero = hero);
    } else {
      this.navigated = false;
      this.hero = new Hero();
    }
  }
}

```



Transpile

```

"use strict";
var __decorate = (this && this.__decorate) || function (decorators, target, key, desc) {
  var c = arguments.length, r = c < 3 ? target : desc === null ? desc = Object.getPrototypeOf(target) : desc;
  if (typeof Reflect === "object" && typeof Reflect.decorate === "function") r = Reflect.decorate(decorators, target, key, desc);
  else for (var i = decorators.length - 1; i >= 0; i--) if (d = decorators[i]) r = (c < 3 ? target : r) = d(r);
  return c > 3 && r && Object.defineProperty(target, key, r), r;
};
var __metadata = (this && this.__metadata) || function (k, v) {
  if (typeof Reflect === "object" && typeof Reflect.metadata === "function") return Reflect.metadata(k, v);
};
var core_1 = require('@angular/core');
var hero_1 = require('./hero');
var router_deprecated_1 = require('@angular/router-deprecated');
var hero_service_1 = require('./hero.service');
var HeroDetailComponent = (function () {
  function HeroDetailComponent(heroService, routeParams) {
    this.heroService = heroService;
    this.routeParams = routeParams;
    this.close = new core_1.EventEmitter();
    this.navigated = false; // true if navigated here
  }
  HeroDetailComponent.prototype.ngOnInit = function () {
    var _this = this;
    if (this.routeParams.get('id') !== null) {
      var id = +this.routeParams.get('id');
      this.navigated = true;
      this.heroService.getHero(id)
        .then(function (hero) { return _this.hero = hero; });
    }
    else {
      this.navigated = false;
      this.hero = new hero_1.Hero();
    }
  }
  HeroDetailComponent.prototype.save = function () {
}
}());

```

## TypeScript Transpiling – Command Line tsc

-p	Compile a specific project or folder
-w	Compile after each change detected

## TypeScript Transpiling – Continuous Delivery

- Grunt, Gulp, Webpack, etc.
- Visual Studio
- VSTS



## TypeScript Advantages

- Productivity
- Static Type Analysis
- Language Tool Support
- Better management of large codebases





# Type Checking

```
var num1 = 5;  
var num2 = 10;  
...  
num1="6";  
...  
var sum = num1 + num2;
```



# Type Checking

```
var num1: number = 5;  
var num2 : number = 10;  
...  
num1="6";  
...  
var sum: number = num1 + num2;
```





## tsconfig.json

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "system",
    "moduleResolution": "node",
    "sourceMap": true,
    "emitDecoratorMetadata": true,
    "experimentalDecorators": true,
    "removeComments": false,
    "noImplicitAny": false
  },
  "exclude": [
    "node_modules",
    "typings/main",
    "typings/main.d.ts"
  ]
}
```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Angular



## Key Parts of Angular

- Modules
- Components
- Templates
- Directives
- Services
- Routing



## Modules



## Modules

- Built into Angular
- Makes it easier to split code into smaller pieces
- Import one module into another
- Export module to make it available for import



## Modules

Use exported  
module  
In this  
module

```
import {Component} from 'angular2/core';  
@Component({  
  selector: 'my-selector',  
  template: '<h1>Hello World</h1>'  
})
```

Available  
outside this  
module

```
export class DemoComponent { }
```



# Components



# Components

- Class (properties & methods)
- Decorated with @Component
- Template
- Selector
- Imported references

```
import { Component } from '@angular/core';

@Component({
  selector: 'my-app',
  template: '<h1>My First Angular 2 App</h1>'
})
export class AppComponent {}
```



# Templates and Selectors



# Templates and Selectors

```
import {Component} from 'angular2/core';

@Component({
  selector: 'my-selector',
  template: '<h1>Hello World</h1>'
})
export class DemoComponent { }
```



# Selector

```
<my-selector>  
  Loading...  
</my-selector>
```

```
@Component({  
  selector: 'my-selector',  
  template: '<h1>Hello  
World</h1>'  
})  
export class DemoComponent { }
```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Templates

```
<my-selector>  
  Loading...  
</my-selector>
```

```
@Component({  
  selector: 'my-selector',  
  template: '<h1>Hello  
World</h1>'  
})  
export class DemoComponent { }
```

Output

Loading...

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Templates

```
<my-selector>
  Loading...
</my-selector>
```

```
@Component({
  selector: 'my-selector',
  template: '<h1>Hello World</h1>'
})
export class DemoComponent { }
```

Output

Hello World

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Templates: Multi-Line

```
<my-selector>
  Loading...
</my-selector>
```

```
@Component({
  selector: 'my-selector',
  template:
    <h1>Hello World</h1>
    <div>
      Welcome!
    </div>
})
export class DemoComponent { }
```

Output

Hello World  
Welcome

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Templates: External file

```
<my-selector>  
  Loading...  
</my-selector>
```

```
@Component({  
  selector: 'my-selector',  
  templateUrl: 'myPage.html'  
})  
export class DemoComponent {  
}
```

Output

Hello World  
Welcome

```
myPage.html  
<h1>Hello World</h1>  
<div>  
  Welcome!  
</div>
```

## Components: Properties

```
<my-selector>  
  Loading...  
</my-selector>
```

```
@Component({  
  selector: 'my-selector',  
  templateUrl: 'myPage.html'  
})  
export class DemoComponent {  
  customerName: string = "David";  
}
```

Output

Hello World  
Welcome David

```
myPage.html  
<h1>Hello World</h1>  
<div>  
  Welcome  
  {{customerName}}!  
</div>
```



## Data Binding

- 1-Way Binding
  - Interpolation
  - 1-Way Property Binding
- 2-Way Property Binding
- Event Binding



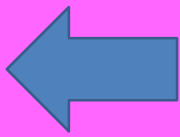
## Interpolation

- Double curly braces around data
- e.g.,  
    `{{customerName}}`



## Interpolation


```
@Component({
  selector: 'my-selector',
  template: '<h1>Hello World</h1>'
})
export class DemoComponent {
  id=1;
  customerFirstName='David';
  customerLastName='Giard';
}
```



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Interpolation

```
@Component({
  selector: 'my-selector',
  template: '<h1>Hello {{customerFirstName}}</h1>'
})
export class DemoComponent {
  id=1;
  customerFirstName='David';
  customerLastName='Giard';
}
```

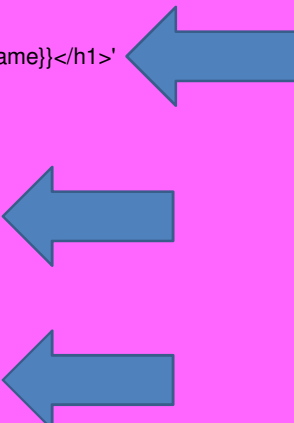


Hello David

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Interpolation

```
@Component({
  selector: 'my-selector',
  template: '<h1>Hello {{customer.FirstName}}</h1>'
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName='David';
    LastName='Giard';
  }
}
export class Customer{
  FirstName: string;
  LastName: string;
}
```



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Interpolation

```
@Component({
  selector: 'my-selector',
  template: `
    <h1>{{customer.FirstName}} Details</h1>
    <div>First: {{customer.FirstName}}</div>
    <div>Last: {{customer.LastName}}
  `
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName='David';
    LastName='Giard';
  }
}
```

## David Details

First: David  
Last: Giard

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

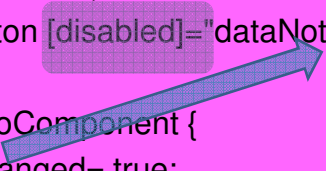
# 1-Way Data Binding

- Square brackets around property
- []



# 1-Way Data Binding

```
@Component({
  selector: 'my-selector',
  template: '<button [disabled]="dataNotChanged">Save</button>'
})
export class DemoComponent {
  dataNotChanged= true;
}
```

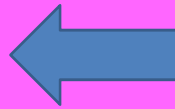


Save



# 1-Way Data Binding

```
@Component({  
  selector: 'my-selector',  
  template: '<button [disabled]=" dataNotChanged">Save</button>'  
})  
export class DemoComponent {  
  dataNotChanged = true;  
}
```

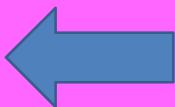


Save

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# 1-Way Data Binding

```
@Component({  
  selector: 'my-selector',  
  template: '<button [disabled]=" dataNotChanged">Save</button>'  
})  
export class DemoComponent {  
  dataNotChanged = false;  
}
```



Save

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## 2-Way Data Binding

- Requires FormsModule
- [(property\_to\_bind)]

## 2-Way Data Binding

```
@Component({
  selector: 'my-selector',
  template: `
    <h1>{{customer.FirstName}} </h1>
    <div>First: <input [(ngModel)]="customer.FirstName" /> </div>
    <div>Last: <input [(ngModel)]="customer.LastName" /> </div>
  `
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName:'David';
    LastName:'Giard';
  }
}
```

### David Details

First: Last:

## 2-Way Data Binding

```
@Component({
  selector: 'my-selector',
  template: `
<h1>{{customer.FirstName}} Details</h1>
<div>First: <input [(ngModel)]="customer.LastName" /> </div>
<div>Last: <input [(ngModel)]="customer.FirstName" /> </div>
`
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName='David';
    LastName='Giard';
  }
}
```

### D Details

First: Last: **Visual Studio LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## 2-Way Data Binding

```
@Component({
  selector: 'my-selector',
  template: `
<h1>{{customer.FirstName}} Details</h1>
<div>First: <input [(ngModel)]="customer.LastName" /> </div>
<div>Last: <input [(ngModel)]="customer.FirstName" /> </div>
`
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName='David';
    LastName='Giard';
  }
}
```

### Da Details

First: Last: **Visual Studio LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## 2-Way Data Binding

```
@Component({
  selector: 'my-selector',
  template: `
<h1>{{customer.FirstName}} Details</h1>
<div>First: <input [(ngModel)]="customer.LastName" /> </div>
<div>Last: <input [(ngModel)]="customer.FirstName" /> </div>
`
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName='David';
    LastName='Giard';
  }
}
```

### Dan Details

First: Last: Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## app.module.ts (required for ngModel)

```
@NgModule({
  imports: [
    FormsModule
  ]
})
```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS



# Events binding

`<control (eventname)="methodname(parameters)">`

## click event


`<control (click)="methodtocall(parameters)">`

e.g.,


`<div (click)="onClick(customer)">`

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## click



```
@Component({
  selector: 'my-selector',
  template: `
    <h1 (click)="onClick (customer)">{{customer.FirstName}} Details</h1>
    <div>First: <input [(ngModel)]="customer.LastName" </div>
    <div>Last: <input [(ngModel)]="customer.FirstName" </div>
  `
})
export class DemoComponent {
  id=1;
  customer: Customer = {
    FirstName:'David';
    LastName:'Giard';
  }
  onClick(cust: Customer) { alert ("You Selected " + customer.FirstName); }
}
```



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Bootstrapping Your Angular app

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## Bootstrapping

```
<script>
```

```
...
```

```
System.import('main.js')
```

```
</script>
```

```
import {AppComponent}  
  from './app.component';  
platformBrowserDynamic().bootstrapModule(AppModule);
```

main.ts / main.js  
= bootstrap file

```
@NgModule({  
  bootstrap: [AppComponent]  
})  
export class AppModule {  
}
```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Directives



# Directives

- Allow you to attach behavior to DOM elements



## Directives

- \*ngFor
- \*ngIf
- ngSwitch
- ngClass
- Custom Directives



## \*ngfor

```
var customers: Customer[] = [  
  { "id": 1, "firstName": " Satya", "lastName": " Nadella" },  
  { "id": 2, "firstName": "Bill", "lastName": "Gates" },  
  { "id": 3, "firstName": "Steve", "lastName": "Ballmer" },  
  { "id": 4, "firstName": " David ", "lastName": " Giard " }  
];
```

```
<div *ngFor="#cust of customers">  
  {{cust.lastName}}, {{cust.firstName}}  
</div>
```

```
Nadella, Satya  
Gates, Bill  
Ballmer, Steve  
Giard, David
```



## \*ngIf

- Syntax: `*ngIf="condition"`
- Removes element from DOM if condition is not “truthy”

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## \*ngIf

```
<h1>People I hate:</div>  
<div *ngIf="true">  
  David Giard  
</div>
```

**People I hate:**  
David Giard

```
<h1>People I hate:</div>  
<div *ngIf="false">  
  David Giard  
</div>
```

**People I hate:**

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## \*ngIf

```
<div>
  <button (click)="clicked()">Toggle</button>
  <div *ngIf="show">
    Can you see me?
  </div>
</div>
```

```
export class DemoComponent {
  show: boolean = true;
  clicked() {this.show = !this.show; }
}
```



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## \*ngIf

```
<div>
  <button (click)="clicked()">Toggle</button>
  <div *ngIf="show">
    Can you see me?
  </div>
</div>
```

```
export class DemoComponent {
  show: boolean = true;
  clicked() {this.show = !this.show; }
}
```



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

## LifeCycle Hooks

- OnInit
- OnChanges
- OnDestroy



## OnInit

```
export class foo implements
OnInit {
    ...
    ngOnInit(){
    ...
    }
}
```



# Services



# Services

- Class containing logic
- Shared Code: Used by components or other services
- Substitutable Objects
- Dependency Injection





# Services

CustomerService.ts

```
import { Injectable } from '@angular/core';

@Injectable()
export class CustService {
  getCustomers() {
    return customers;
  }
}

var customers: Customer[] = [
  { "id": 1, "firstname": "David", "lastname": "Giard" },
  { "id": 2, "firstname": "Bill", "lastname": "Gates" },
  { "id": 3, "firstname": "Steve", "lastname": "Ballmer" },
  { "id": 4, "firstname": "Satya", "lastname": "Nadella" }
];
```

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Services

CustomerService.ts

```
import { Injectable } from '@angular/core';
```

```
@Injectable()
export class CustService {
  getCustomers() {
    return customers;
  }
}
...
```

```
import { OnInit } from '@angular/core';
import { CustService } from CustomerService
```

```
@Component({
  selector: 'xxx',
  template: 'yyy',
  ...
  providers: [CustService]
})
export class DemoComponent implements OnInit {

  constructor(private customerService:CustService) { }

  ngOnInit() {
    this.customers = this.customerService.getCustomer
  }
}
```



# Routing



# Routing

- Load components dynamically into page
- Link via URL
- Client-side
- Step 1: Bootstrap array of routes

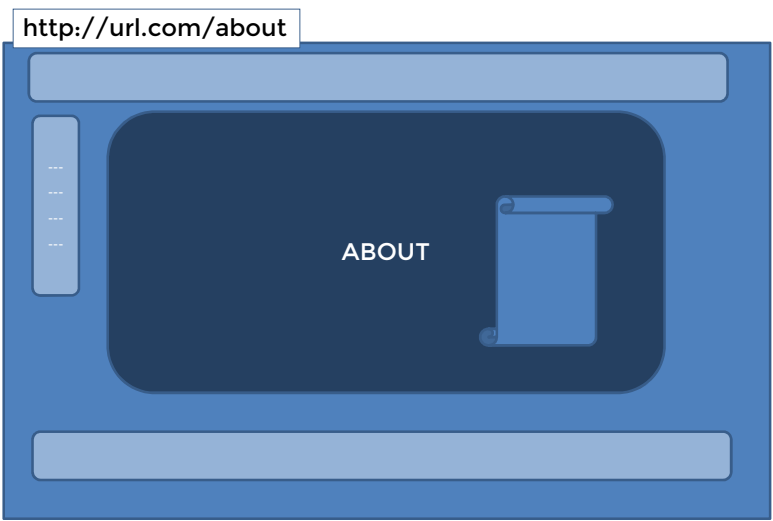


# Routing



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Routing

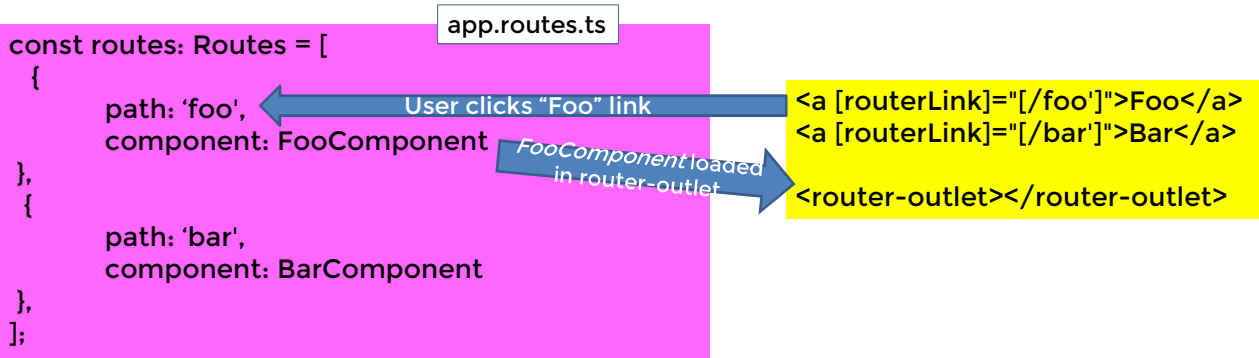


Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Routing



# Routing



# HTTP

```
bootstrap(DemoComponent, [  
  HTTP_PROVIDERS,  
]);
```

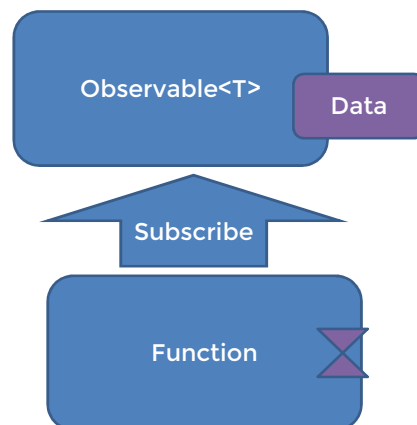
main.ts

```
import {Http } from '@angular/http';  
...  
this.http.get(webServiceUrl);
```

Component

Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Observables (via RxJs)



Visual Studio **LIVE!**  
EXPERT SOLUTIONS FOR .NET DEVELOPERS

# Observables

```
getCustomers(): Observable<customer[]> {  
    return Observable.create((observer: Observer<any>) => {  
        ...  
        observer.next(this.customers);  
    })  
}
```

```
this.episodeService.getCustomers().subscribe((data) => {  
    ...  
})
```



# DEMO



## Links

- [angular.io](http://angular.io)
- [typescriptlang.org](http://typescriptlang.org)
- [github.com/Microsoft/TypeScript](https://github.com/Microsoft/TypeScript)
- [nodejs.org/en/download](https://nodejs.org/en/download)
- [code.visualstudio.com](https://code.visualstudio.com)
- [tinyurl.com/angular2cheatsheet](https://tinyurl.com/angular2cheatsheet)
  
- [slideshare.net/dgiard/angular2-and-typescript](https://slideshare.net/dgiard/angular2-and-typescript)
- [github.com/DavidGiard/dgtv](https://github.com/DavidGiard/dgtv)
- [davidgiard.com](http://davidgiard.com)