Angular

- Runtime configuration
- Debugging
- Access Parent Component
- Router :: Subscribe to Route Changes
- Structural Directives - The Asterisk (*) Prefix

Runtime configuration

- Runtime environment configuration with Angular

Debugging

Chrome

- inspect element
- select your component and it will be assigned to $0
- run

```
ng.probe($0).componentInstance
```

- change values as you desire or call object class members
- to trigger the changes run

```
ng.probe($0)._debugInfo._view.changeDetectorRef.detectChanges()
```

Access Parent Component

Using `@Input` and a little trick to send the **this** of the parent to the child. I guess you can do it the other way around also, to access child from parent via an `@Output`

Parent Component

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

@Component({
    selector: 'app-root',
    template: '<app-child [parnt]="var2"></app-child>',
    style: ''
})
export class AppComponent {
    title = 'app works!';
    var1 = "val1";
    var2 = this;
}
```

Child Component

```
import { Component, Input, EventEmitter, OnInit } from '@angular/core';

@Component({
  selector: 'app-child',
  template: '{{parnt.var1}}',
  style: ''
})
export class ChildComponent implements OnInit {
  @Input() parnt;
}
```

You should see "val1" printed, the value of the var1 of the parent printed in the child template.

Router :: Subscribe to Route Changes

This example will change the function of a contextual button in a bootstrap/navbar

```
import { Component, OnInit } from "@angular/core";
import { Router, NavigationEnd } from "@angular/router";

import "rxjs/add/operator/filter";

@Component({
    selector: "my-app",
    templateUrl: `
        <nav class="navbar navbar-fixed-top navbar-light bg-faded">
            <a routerLink="/" class="navbar-brand">{{title}}</a>
            <div class="nav navbar-nav pull-xs-left">
                <a routerLink="/contacts" routerLinkActive="active" class="nav-item nav-
link">Contacts</a>
                <a routerLink="/features" routerLinkActive="active" class="nav-item nav-
link">Features</a>
                <a routerLink="/tools" routerLinkActive="active" class="nav-item nav-
link">Tools</a>
            </div>
            <div class="nav navbar-nav pull-xs-right">
                <button routerLink="contextButtonLink" [disabled]="!contextButtonActive"
type="button" class="btn btn-primary">{{contextButtonLabel}}</button>
            </div>
        </nav>
        <router-outlet></router-outlet>
`
})
export class AppComponent implements OnInit {
    title:string = "My App";

    contextButtonLabel:string = "New";
```

```
    contextButtonLink:string = "/";
    contextButtonActive:boolean = false;

    constructor(
        # inject the Router
        private router: Router,
    ) {
    }

    ngOnInit() {
        // subscribe to desired router changes, for example NavigationEnd events
        this.router.events.filter(
            e => e instanceof NavigationEnd
        ).subscribe(
            e => { this.contextButton(e); }
        );
    }

    contextButton(event) {
        if (event.urlAfterRedirects === "/contacts") {
            this.contextButtonLabel = "New";
            this.contextButtonLink = "/contacts/new";
            this.contextButtonActive = true;
        } else {
            // this.contextButtonLabel = "Disabled";
            // this.contextButtonLink = "/";
            this.contextButtonActive = false;
        }
    }
}
```

Structural Directives - The Asterisk (*) Prefix

The (*) is a syntactic sugar. You'll have a ng-template wrapped arround the host element with the *directive.

This snippet

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styles: []
})
export class AppComponent {
  la = [1, 2, 3];
}
```

app.component.html

```
<div *bla="la">
  some content
</div>
```

will be transformed into

```
<ng-template [bla]="la">
<div>
    some content
    </div>
</ng-template>
```

- The `bla` directive moved to `ng-template` element where it is now a property binding
  - meaning the BlaDirective should have an `@Input() bla` which will get the value of `la`.
- The div element who hosted the *directive moved inside the `ng-template`.

and you need the directive like this

```
import { Directive, Input, TemplateRef, ViewContainerRef } from '@angular/core';

@Directive({
  selector: '[bla]'
})
export class BlaDirective {

  @Input() set bla(value) {
    console.log(value);
  }

  constructor(
    private templateRef: TemplateRef<any>,
    private viewContainer: ViewContainerRef) { }
}
```

Structural directives need TemplateRef and ViewContainer.

TODO display the content of the template