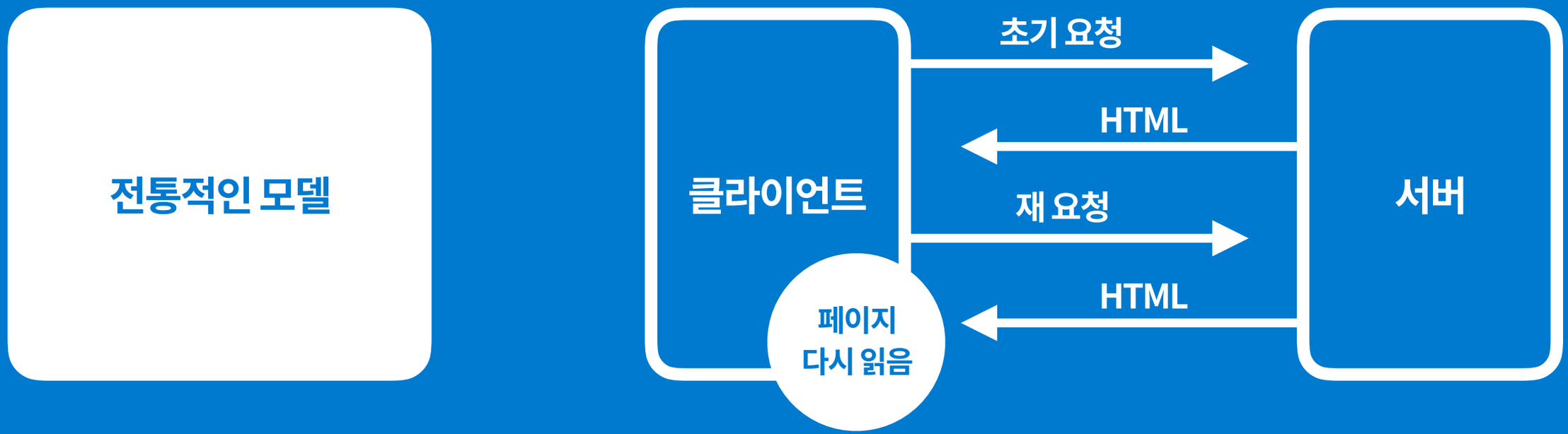
AngularJS

SPA

Single Page Applications

Single Page Applications



Single Page Applications

처음 페이지가 렌더링 된 이후, 페이지 전체를 다시 그리는 것이 아니라
서버로부터 전송 받은 일부분의 데이터를 변경해 현재 페이지에 다른 뷰(View)를 렌더링

Single Page Applications

처음 페이지가 렌더링 된 이후, 페이지 전체를 다시 그리는 것이 아니라
서버로부터 전송 받은 일부분의 데이터를 변경해 현재 페이지에 다른 뷰(View)를 렌더링

Single Page Applications

SPA(Single Page Applications) 구현을 위해서는 다양한 기술이 요구된다.

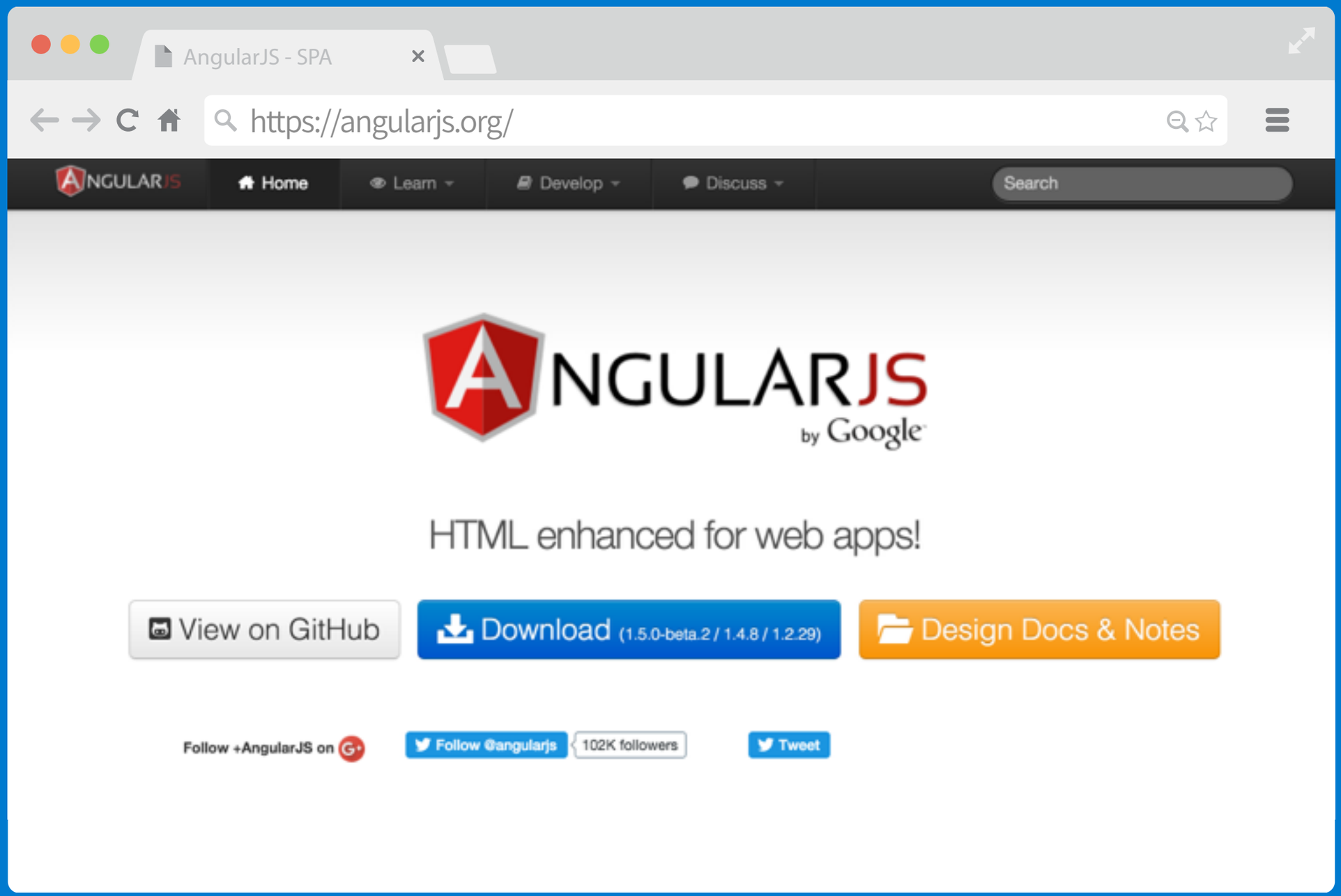
- 문서객체모델 조작(DOM Manipulation)
- 히스토리(기록, History)
- 라우팅(절차, Routing)
- 비동기 통신(Ajax)
- 데이터 연결(Data Binding)
- …

AngularJS

about SPA Framework

AngularJS

AngularJS는 SPA를 구현하기 위한 모든 기능을 제공하는 프레임워크



AngularJS - Full Featured SPA Framework

Model

View

Controller

Directives

Expression

Services

Data Binding

Validation

Testing

Template

History

Depedency Injection

Factories

Routing

Views

jqLite

AngularJS - Zero

Miško Hevery의 개인 프로젝트로 시작된 AngularJS는 초기에 웹 개발자가 아닌, HTML을 사용할 줄 아는 디자이너를 대상으로 개발되었다. (2009년)

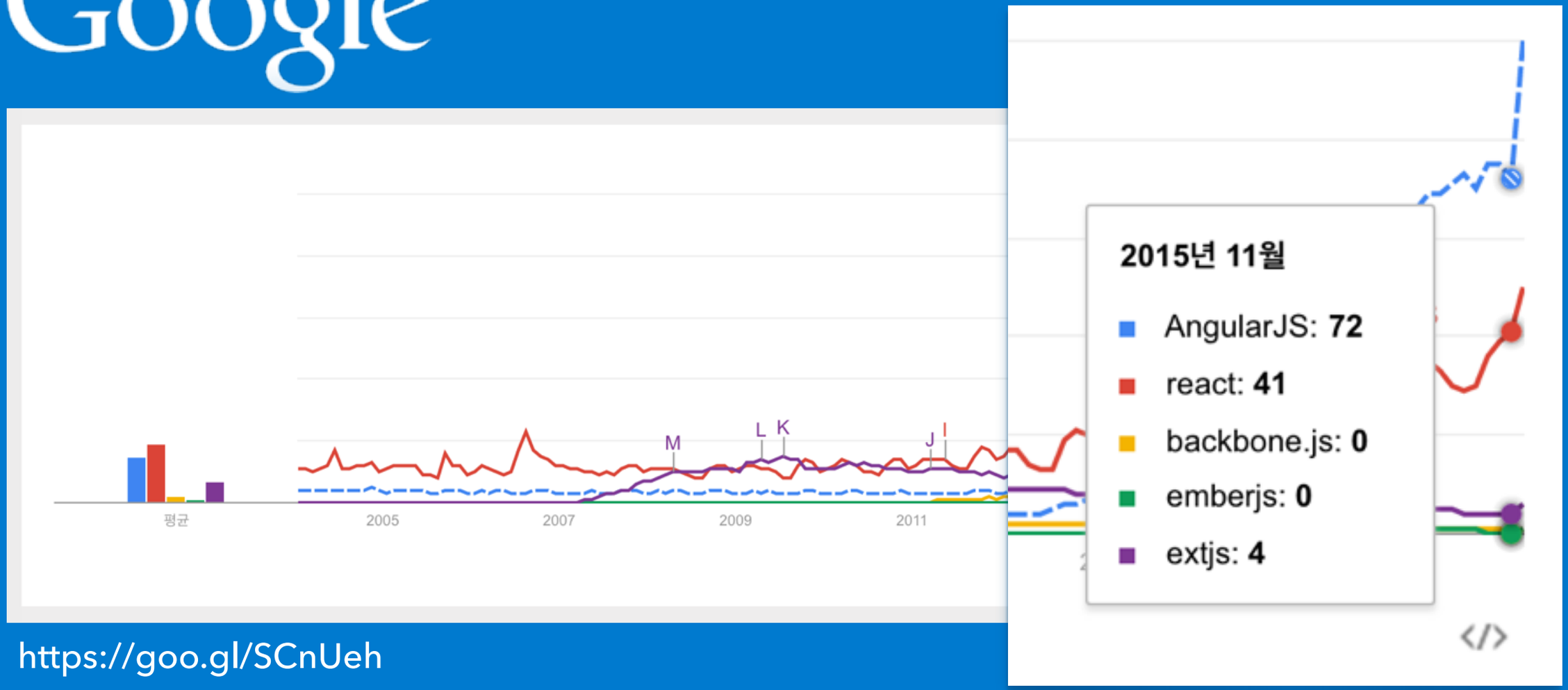
Google Feedback

	기간	코드
GWT	3명 x 6개월	약 17,000 라인
AngularJS	1명 x 3주	약 1,000 라인

AngularJS - Growth

Google의 후광에 힘입은 강력한 마케팅으로 2012년 이후 가파르게 급성장

Google



https://goo.gl/SCnUeh

AngularJS - WHY

프론트-엔드/백-엔드 개발자가 AngularJS 프레임워크를 선호하는 이유

- 코드 일관성 유지 (Maintain Coding Style Consistency)
- MVC 구조 (MVC Structure Pattern)
- 코드양 감소 (Less Code)
- 코드 재사용 (Re-Useable Pattern)
- 양방향 데이터 바인딩 (Two-way Data Binding)
- 다양한 모듈 확장 (Modules Extension)

AngularJS

Getting Started

AngularJS - STEP 1

AngularJS 파일은 로컬/호스팅(CDN) 중 선택하여 호출

Local

```
<script src=“js/vender/angular-1.5.1.min.js”></script>
```

Hosted Version (CDN)

```
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.5.1/angular.min.js"></script>
```

Bower Management 

AngularJS - STEP 2-1

HTML 문서(<html> 요소)에 ngApp 디렉티브(지시자, Directive)를 추가

※ data-* 형식으로 디렉티브 추가 권장(웹표준 준수)

Web Non-Standard

```
<html lang="ko-KR" ng-app>
```

Error Attribute ng-app not allowed on element html at this point.

From line 1, column 16; to line 2, column 26

TYPE html>↩<html lang="ko-KR" ng-app>↩<head

Attributes for element html:

Global attributes

manifest

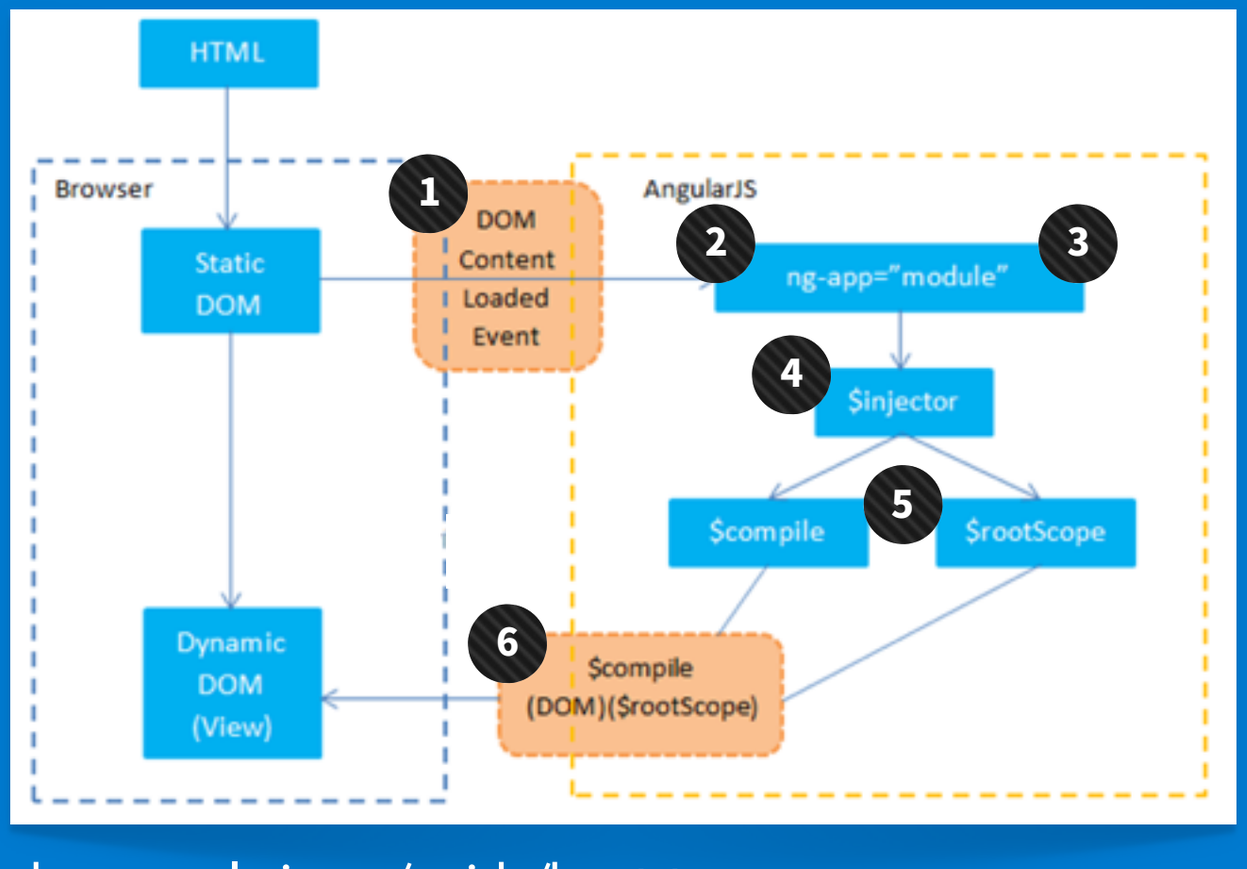
Web Standard

```
<html lang="ko-KR" data-ng-app>
```

AngularJS - STEP 2-2

AngularJS 부트스트랩(Bootstrap) 프로세스

정적인 HTML 문서를 AngularJS 웹 애플리케이션으로 동작시키는 일련의 과정



docs.angularjs.org/guide/bootstrap

1. angularJS 실행
   `DOMContentLoaded Event`
   `document.readystate === "complete"`
2. ngApp 디렉티브 탐색 (애플리케이션 초기화)
   `auto-bootstrap`
3. 모듈(Module)이 있을 경우, 읽어들임 (옵션)
4. $Injector 설정 (하나만 존재)
5. $Compile / $rootScope 생성
6. DOM 렌더링 후, View 생성

AngularJS - STEP 3-1

ngModel, ngBind 디렉티브를 사용하여 데이터를
양방향(Two-Way Data Binding)으로 연결한다.

Directive

```
<input type="text" data-ng-model="two_way_binding">

<p data-ng-bind="two_way_binding"></p>
```

Expression

```
<input type="text" data-ng-model="two_way_binding">

<p>{{two_way_binding}}</p>
```

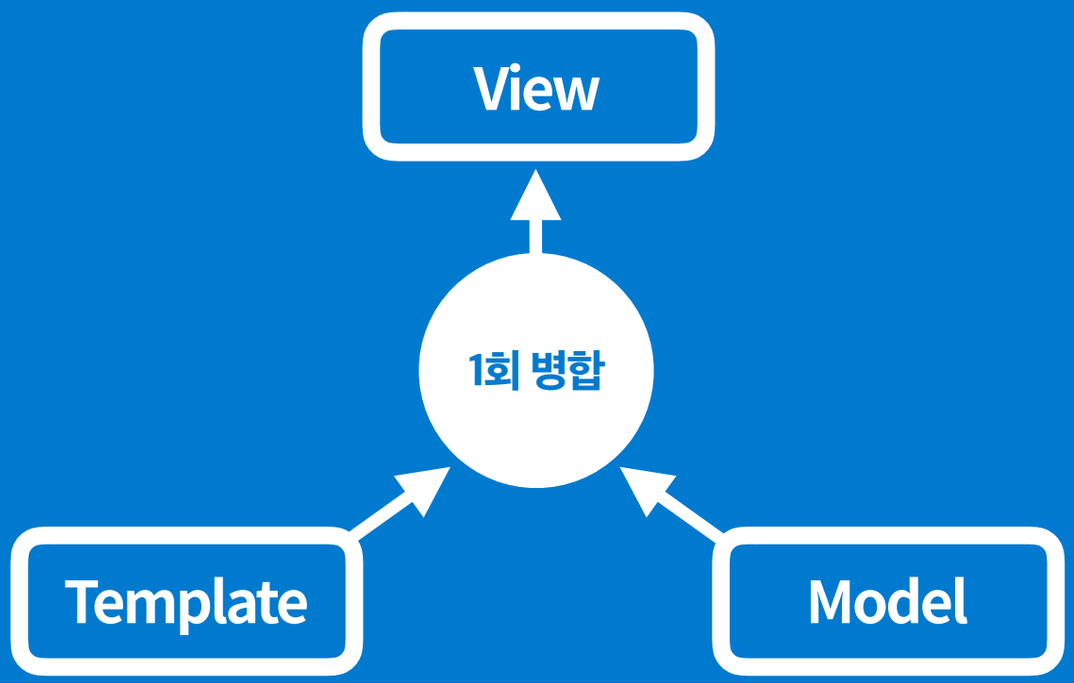
데이터 바인딩 익스프레션
Data Binding Expression

ngBind 디렉티브 대신
익스프레션(표현식, Expression)을
사용할 수도 있다.

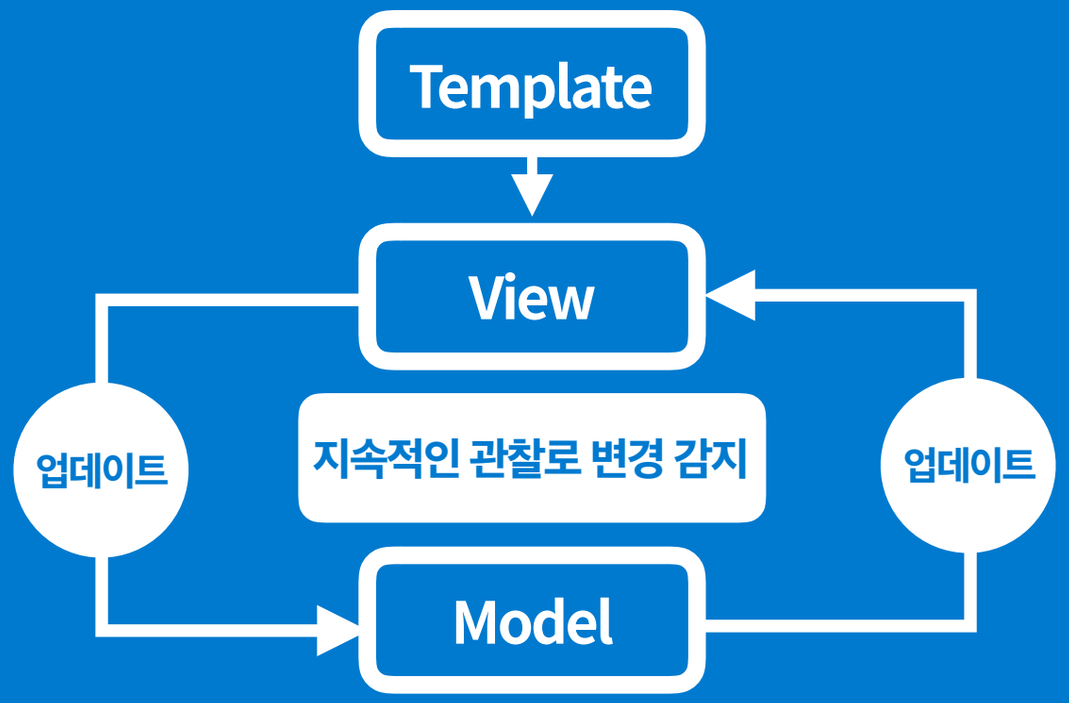
AngularJS - STEP 3-2

AngularJS는 Two-Way Binding 기술을 사용하여 Model ↔ View 업데이트

One-Way Data Binding



Two-Way Data Binding



AngularJS - Start!

DEMO

AngularJS vs jQuery

jQuery DEMO

yamoo9_example.html

yamoo9_example.html

```
1
<!DOCTYPE html>
<html lang="ko-KR">
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <meta charset="UTF-8">
    <title>AngularJS Start</title>
    <script src="//code.jquery.com/jquery.min.js"></script>
    <script>
    function init($) {
        var $twb         = $('#twb'),
            $twb_binding = $('#twb-binding');

        $twb.on('keyup', function(event) {
            $twb_binding.text( event.target.value );
        });
    }
    jQuery.noConflict(true)(init);
    </script>
</head>
<body>

    <input type="text" id="twb">
    <p id="twb-binding"></p>

</body>
</html>
```

AngularJS vs jQuery

AngularJS DEMO

yamoo9_example.html

yamoo9_example.html

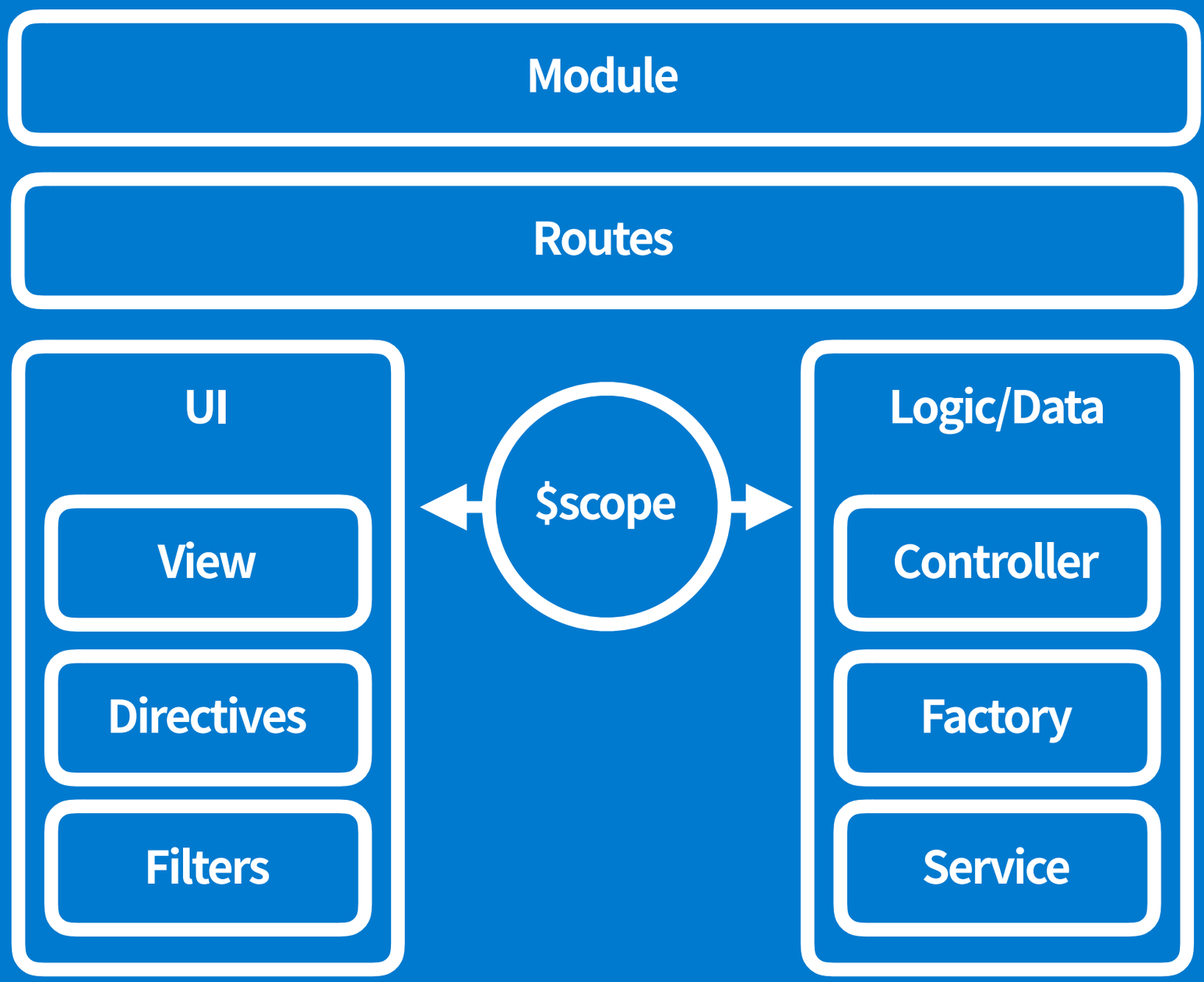
```
<!DOCTYPE html>
<html lang="ko-KR" data-ng-app>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <meta charset="UTF-8">
    <title>AngularJS Start</title>
    <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
</head>
<body>

    <input type="text" data-ng-model="twb">
    <p data-ng-bind="twb"></p>
    <p>{{twb}}</p>

</body>
</html>
```

AngularJS

Big Picture

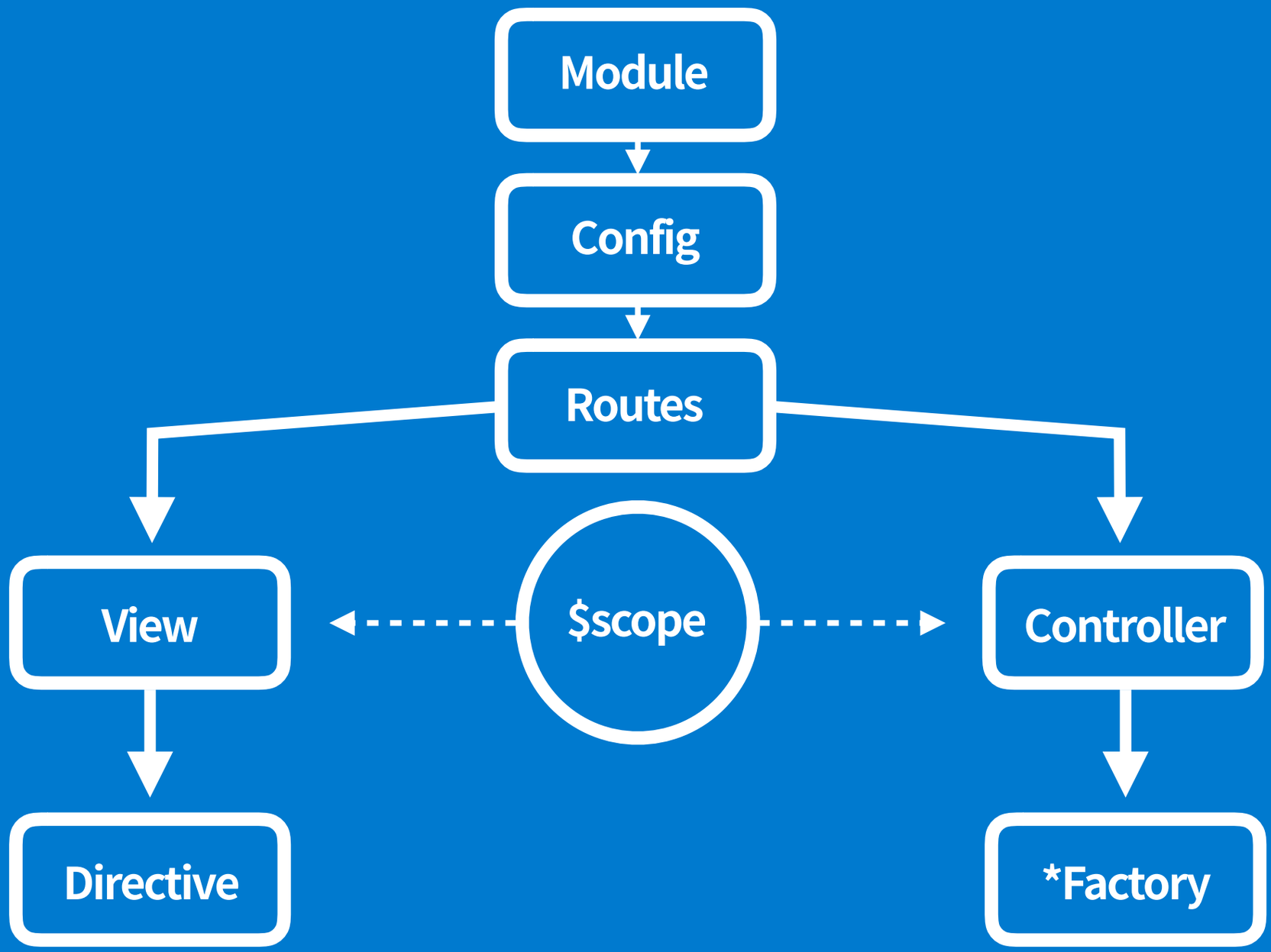
AngularJS - Key Players

AngularJS를 구성하는 애플리케이션 핵심 구조



AngularJS - Big Picture

AngularJS를 구성하는 애플리케이션 핵심 구조



AngularJS - Module

모듈(Module)은 컨테이너(Container)

Module

- 컨트롤러(Controllers)
- 라우트(Routes)
- 팩토리(Factories)/서비스(Services)
- 디렉티브(Directives)
- 필터(Filters)

AngularJS - Factory / Service

팩토리(Factories)/서비스(Services)는 싱글톤 객체

Factories / Services

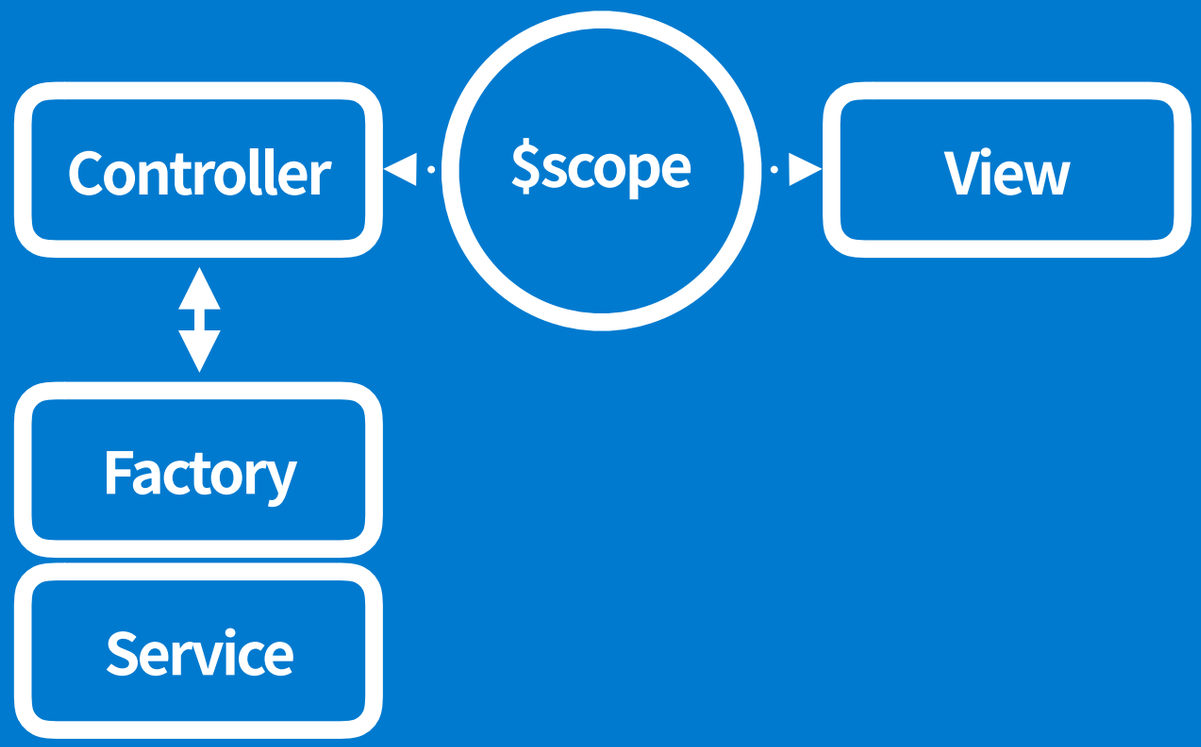
- RESTful 서비스에 사용
- Controller 사이 데이터 공유에 사용
- 사용자 정의 로직에 사용
- 싱글톤 객체(Singleton Object)

AngularJS - Controller

컨트롤러(Controller)는 데이터(Model)/뷰(View) 제어

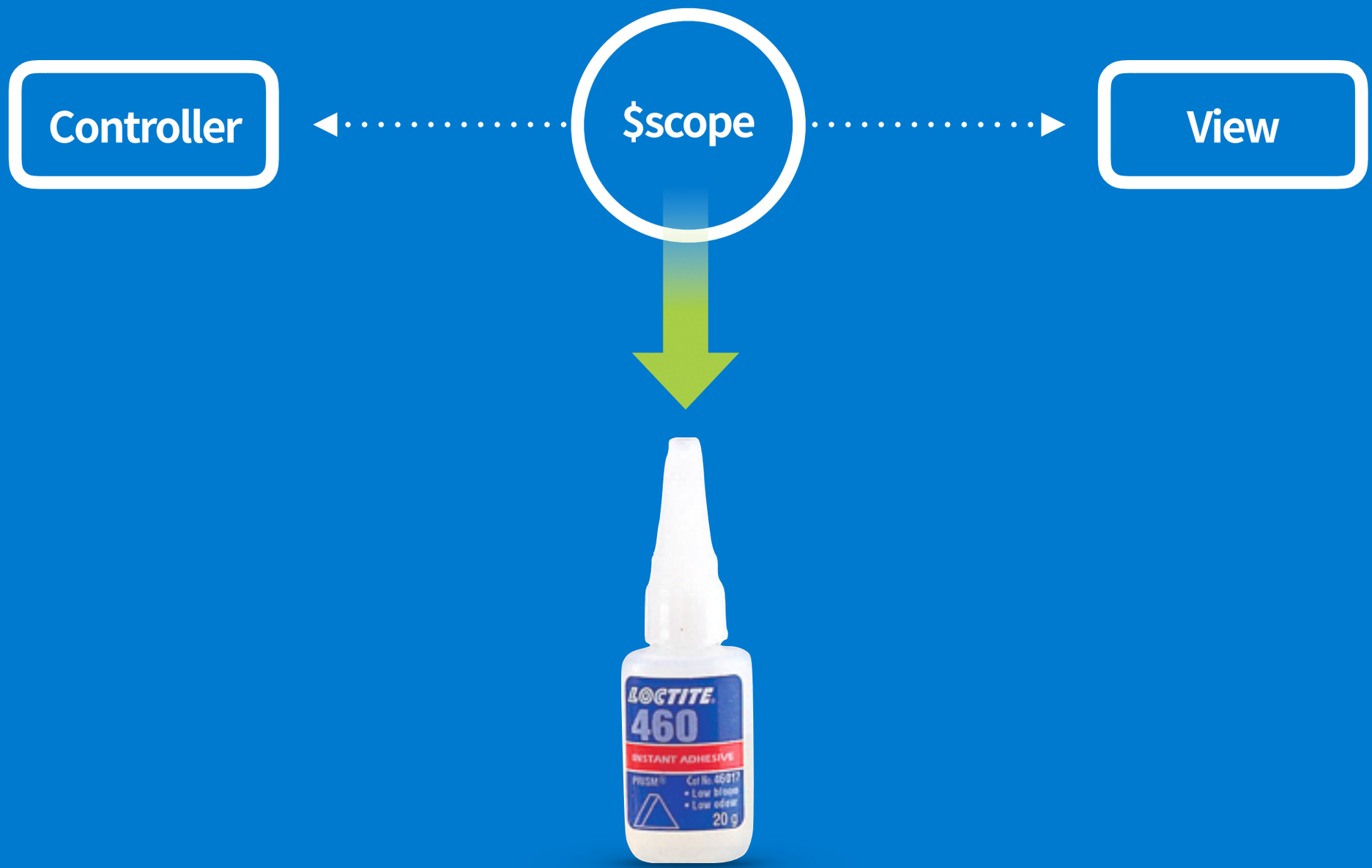
Controller

- View를 제어하는 두뇌(Brain) 역할
- Factory/Service를 이용하여
  데이터 검색 및 보관
- View에 의한 Event 핸들링
- 사용자 정의 로직에 사용
- View와 상호작용하는 $scope와 연결



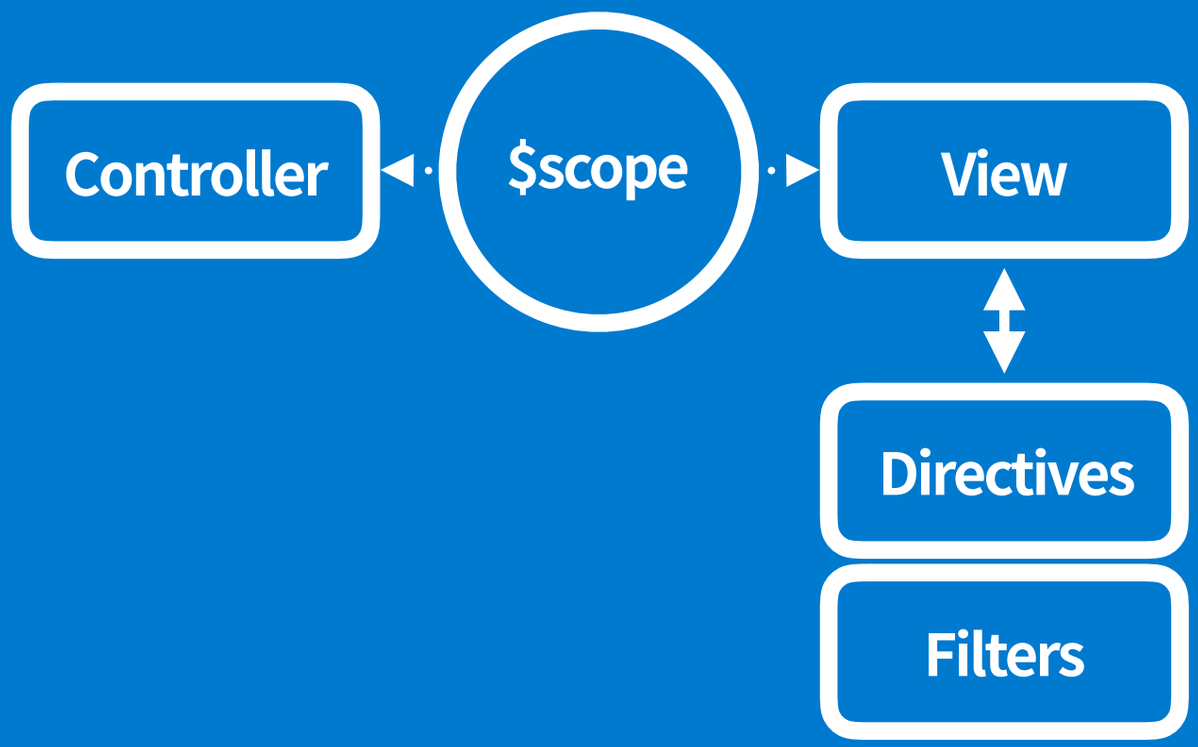
AngularJS - Scope

스코프(Scope)는 컨트롤러(Controller)과 뷰-모델(ViewModel)을 연결하는 `접착제`



AngularJS - View

뷰(View)는 사용자 인터페이스(UI)를 렌더링



View

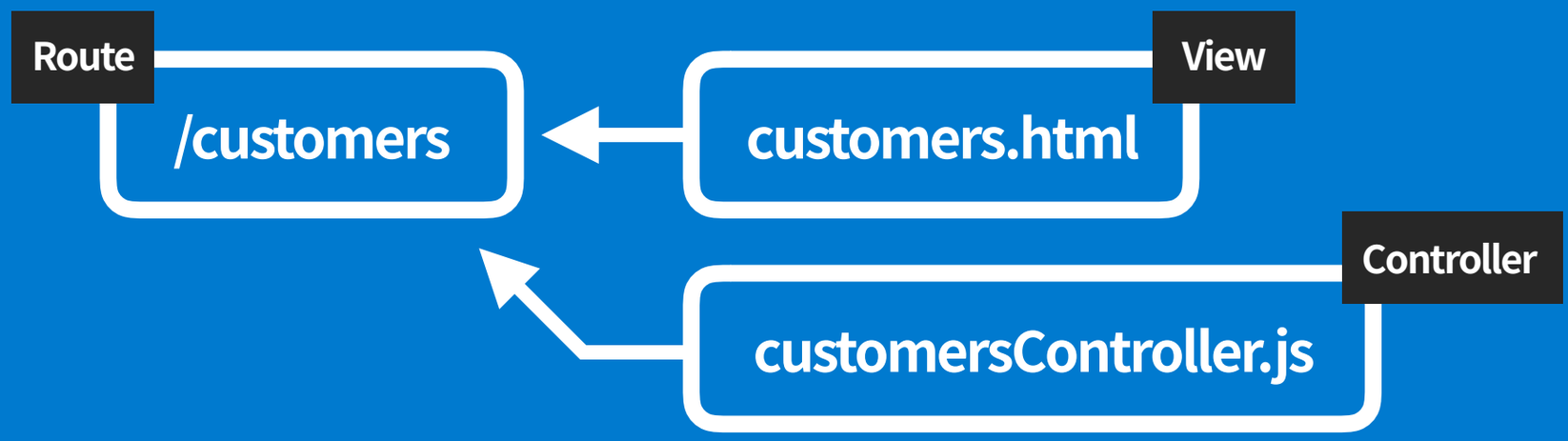
- HTML/CSS를 포함
- 컨트롤러에 연결된 $scope 객체를 통해 연결된 데이터를 처리
- 디렉티브를 사용해 HTML을 확장

AngularJS - Routes

라우트(Routes)는 고유한 연결 주소에 컨트롤러/뷰를 연결

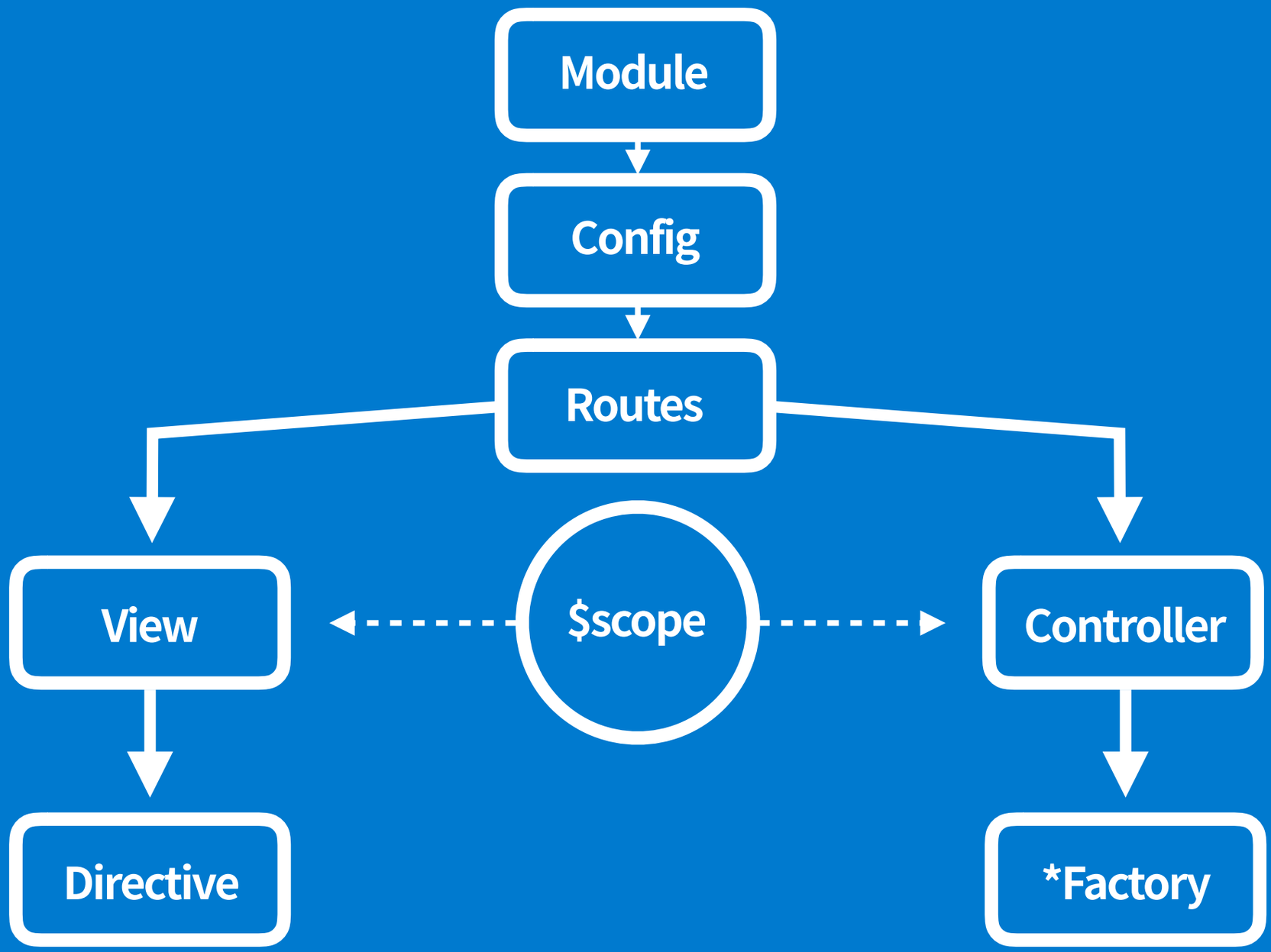
Routes

- 연결 주소에 컨트롤러(Controllers)와 뷰(View)를 각각 연결
- 라우트 파라미터(Route Parameter)를 포함시킬 수 있음
  /customers/:customerId



AngularJS - Big Picture

AngularJS를 구성하는 애플리케이션 핵심 구조

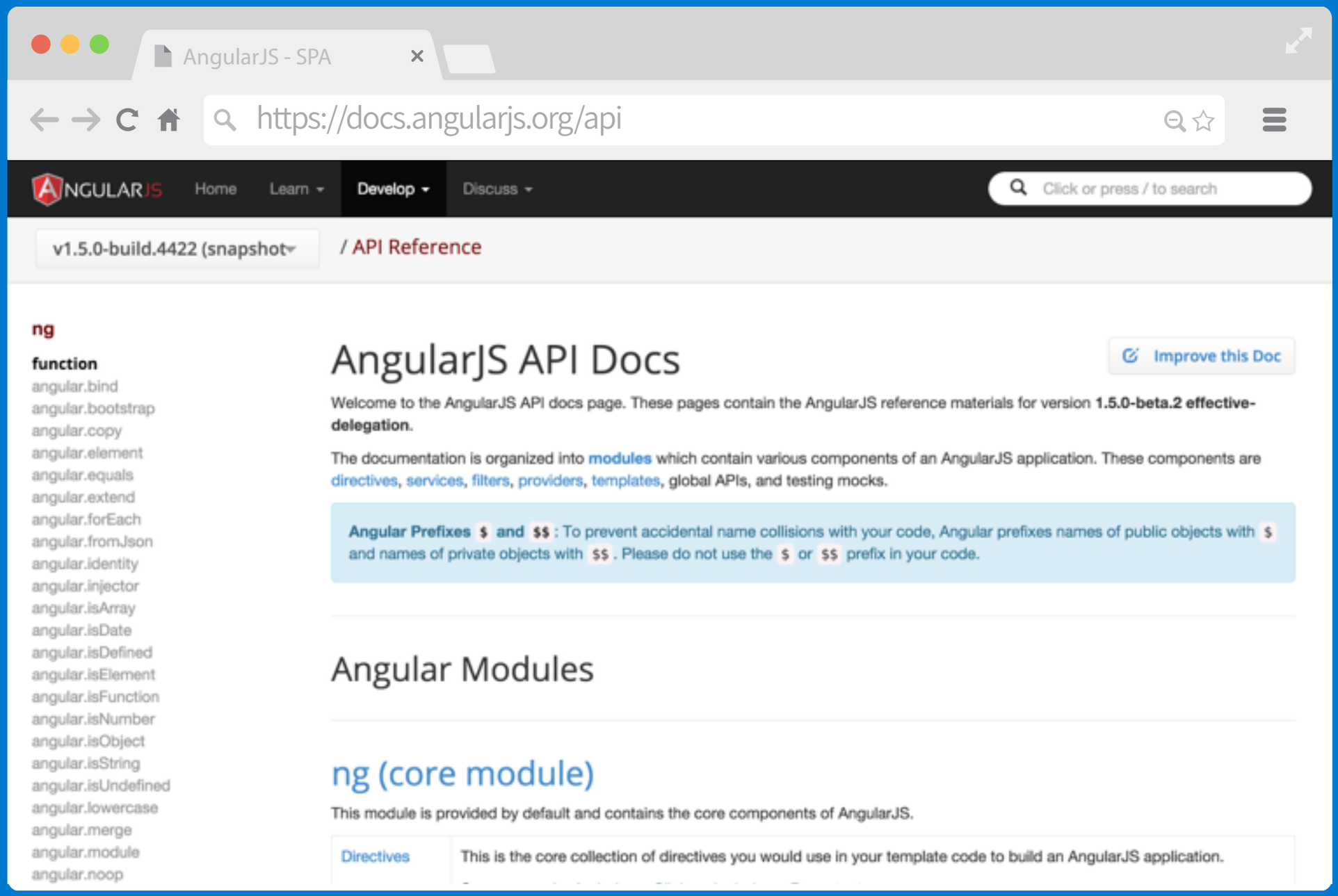


AngularJS

Documentation & Summary

AngularJS - Documentation

AngularJS API Docs: 사용 방법 참고 문서



AngularJS - Summary

AngularJS는 SPA 프레임워크

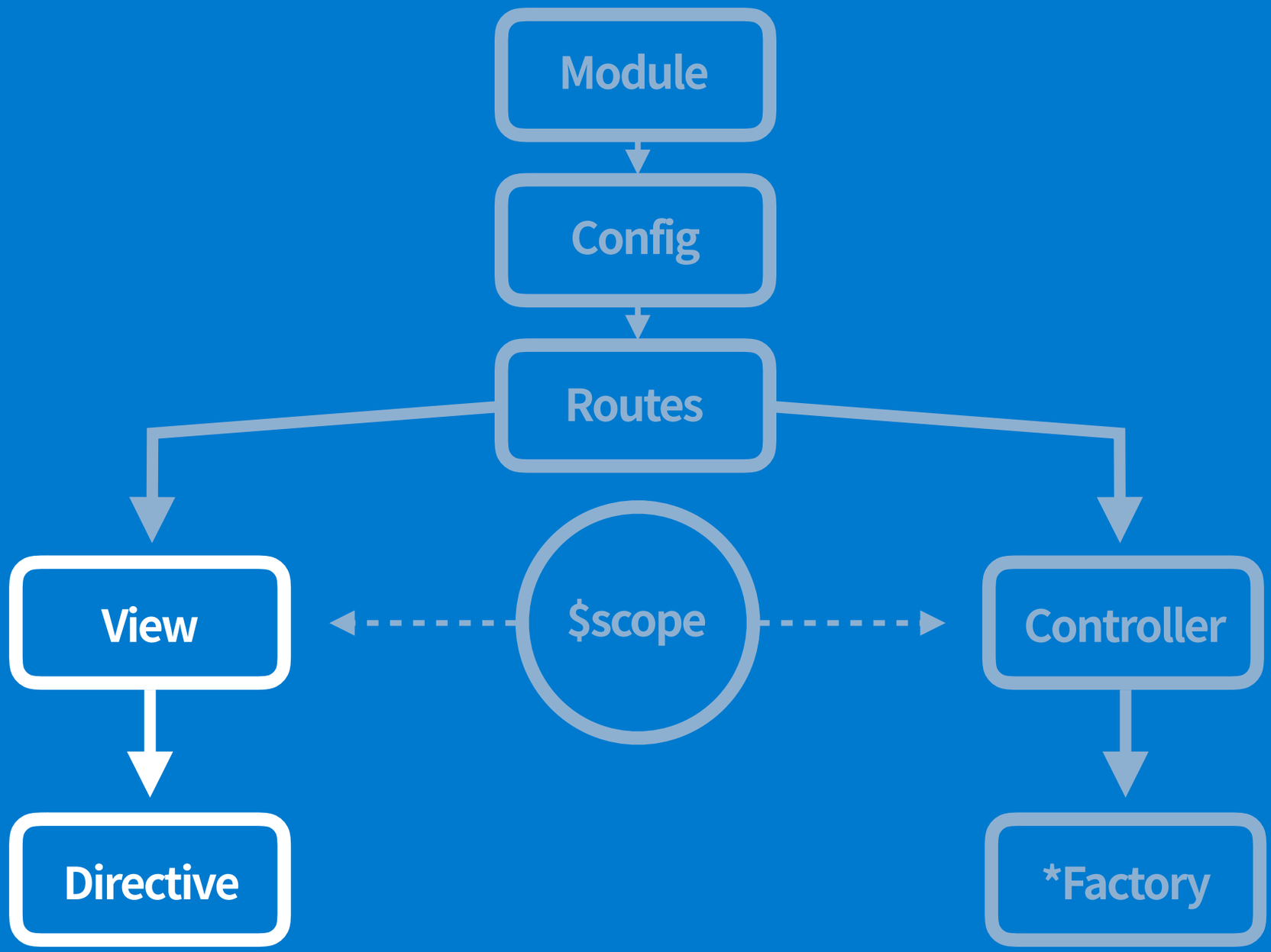
- SPA(Single Page Application) 제작에 최적화된 프레임워크
- 애플리케이션은 한 개 이상의 컨테이너(Modules)로 구성
- 각기 다른 여러 개의 모듈로 구성
  - Controllers
  - Factories/Services
  - View
  - Directives
  - Filters

View

Point

View

Directives / Expressions / Filters



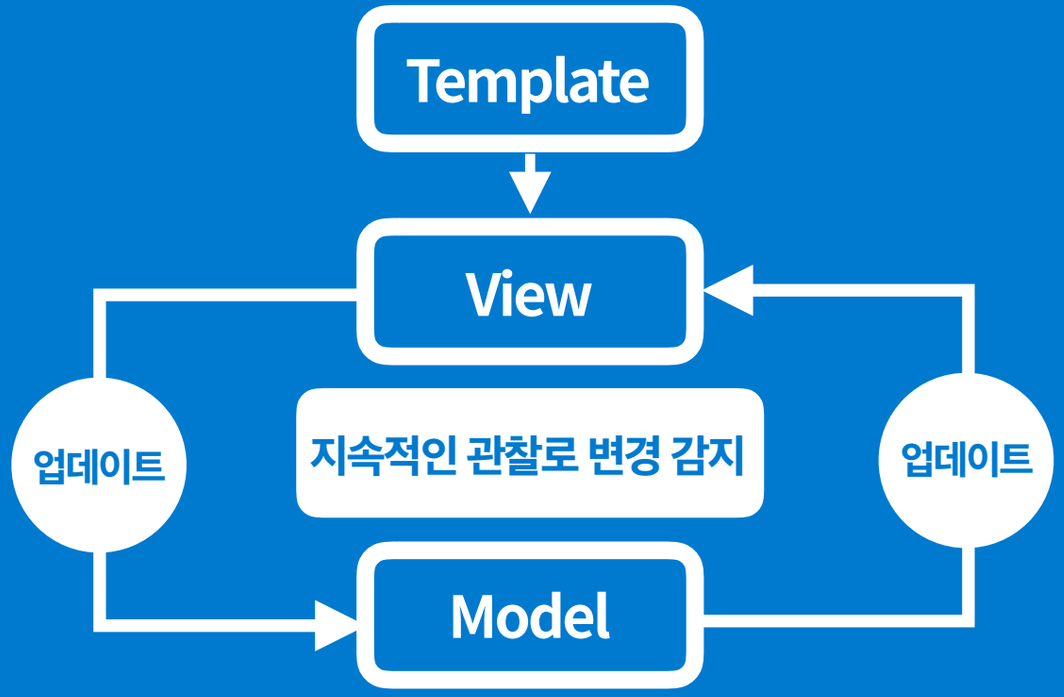
Data Binding

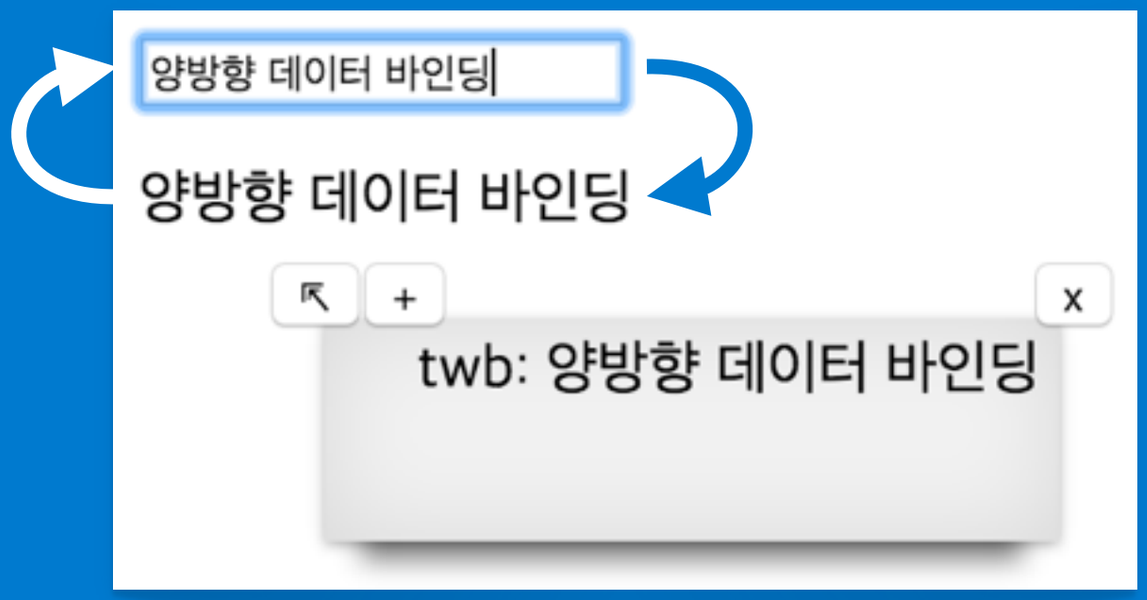
Control-Oriented vs Data-Oriented

Data Binding

Javascript는 데이터 바인딩 기능이 내장(Native Support)되어 있지 않다.

AngularJS의 양방향 데이터 바인딩은 코딩 양을 상당하게 줄여 준다.





Control-Oriented vs Data-Oriented

컨트롤 중심 개발 vs 데이터 중심 개발

ID yamoo9 전송

1. <input> 요소를 선택해서 초기 값 입력

```
document.querySelector('#user-id').value ='yamoo9';
```

2. 사용자가 데이터 입력(수정)
3. 사용자 '전송' 버튼 클릭
4. <input> 요소에 입력된 값을 코드를 통해 가져 옴

```
var u_id = document.querySelector('#user-id').value;
```

1. <input> 요소에 속성 추가

```
<input data-ng-model='u_id'>
```

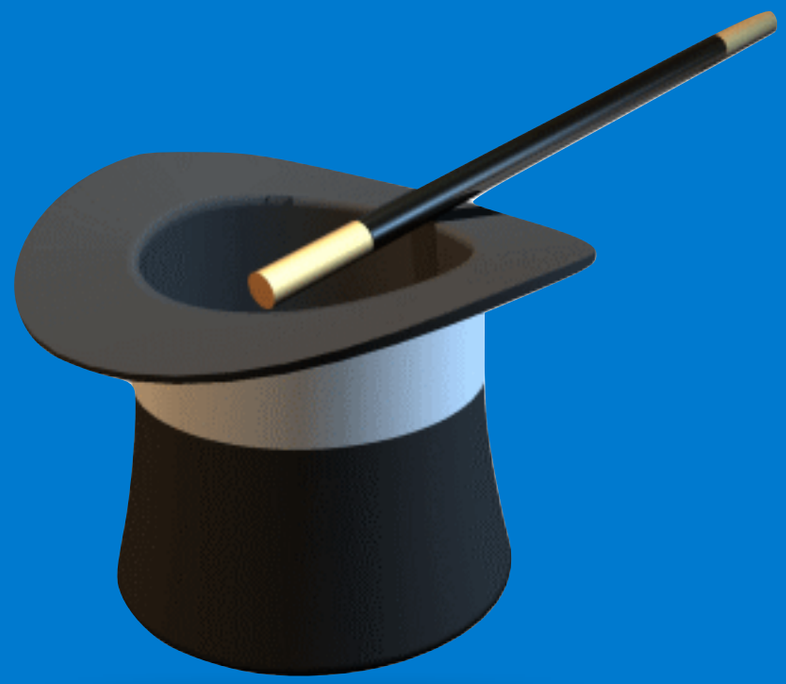
2. 사용자가 데이터 입력(수정)
3. 속성 값 자동 업데이트!

Directives & Expressions

Directives

디렉티브(지시자, Directives)는 HTML을 확장한다.
디렉티브는 다음과 같은 일을 수행할 수 있다.

- 문서객체모델 조작 (DOM Manipulation)
- 데이터 연결 (Data Binding)
- 컨트롤러/모듈 (Controllers/Modules)
- 뷰 로딩 (view Loading)
- 스타일링 (CSS)
- 이벤트 (Events)



Directives

문서객체모델(DOM) 조작

- ngShow
- ngHide
- ngView
- ngRepeat
- ...

모듈(Modules)/컨트롤러(Controllers)

- ngApp
- ngControler
- ...

데이터 바인딩(Data Binding)

- ngInit
- ngModel
- ngBind
- ...

이벤트(Events)

- ngClick
- ngMouseenter
- ngKeydown
- ...

Directives

디렉티브(지시자, Directives)를 정의하는 방법은 다양하다.

하지만.. 웹표준을 준수 하기 위해서는 data-* 접두사를 붙여야 한다.

```
<div ng-hide="is_hidden"></div>
<div data-ng-hide="is_hidden"></div>
<ng-view></ng-view>
```

Expressions

익스프레션(표현식, Expression)으로 연결된 마크업({{}}) 내부에
코드 조각(Code Snippet)이 배치된다.

익스프레션
Expression

{{'yamoo9'+9}}

바인딩 마크업
Binding Markup

Directives & Expressions

DEMO

Directives & Expressions

ngInit / ngModel / ngClick / ngShow / ngHide

{{...}}

모델 연결
input, textarea, select

yamoo9_example.html

yamoo9_example.html

index.html

```
<!DOCTYPE html>
<html lang="ko-KR" data-ng-app>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=Edge">
    <meta charset="UTF-8">
    <title>Directives &amp; Expressions - AngularJS</title>
    <style>...
    </style>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
</head>
<body data-ng-init="bg_color='#5549a6'; is_hidde
    <!--
        Directives & Expressions
            ⊙ ng-init
            ⊙ ng-show
            ⊙ ng-hide
            ⊙ ng-model
            ⊙ ng-click
    -->
...
</body>
</html>
```

https://gist.github.com/yamoo9/a7d9e95e47453696cd89

Directives & Expressions

ngCloak / ngSwitch / ngClass

은폐: {{}} 코드 노출 방지

FOUC(Flash Of Unstyled Content)

yamoo9_example.html

yamoo9_example.html

```
<body data-ng-init="member={name: 'yamoo9', log_in: true, is_visible: true, status: 'failed'}">

    <!--
        Directives & Expressions
            ⊙ ng-cloak [FOUC(Flash Of Unstyled Content)]
            ⊙ ng-switch
            ⊙ ng-class
            ⊙ ng-show
    -->
    <div class="header-bar" data-ng-clock data-ng-sw
        <div class="notice" data-ng-show="member.is_
            공지사항을 알려드립니다.
        </div>
        <div class="log_in_success" data-ng-switch-w
            'success': member.log_in,
            'failed': !member.log_in
        }">
            <p>안녕하세요. {{member.name}}님. 오늘 하루도
        </div>
        <div class="log_in_failed" data-ng-switch-wh
            <p>로그인에 실패하셨습니다. :-( 다시 로그인해보시길
        </div>
        <div class="sign_up" data-ng-switch-default>
            <p>회원 가입을 환영합니다. :-)</p>
        </div>
    </div>

</body>
```

https://gist.github.com/82bc2abf21c6b88b83d1

Data & Repetition

ngRepeat

Repetition - ngRepeat

데이터를 반복/순환하여 각 데이터 아이템에 접근할 경우,
ngRepeat 디렉티브를 사용한다.

yamoo9_example.html

```
<!--
    Repetition Directives
        ⊙ ng-repeat
-->
<div class="container">
    <h2>AngularJS <code>ngRepeat</code> 디렉티브를 활용하여 데이터 반복 순환 처리</h2>
    <ul data-ng-init="categories=[
        {
            'name'        : 'DOM',
            'description' : '문서객체모델'
        },
        {
            'name'        : 'Javascript',
            'description' : '자바스크립트'
        },
        {
            'name'        : 'Node.js',
            'description' : '서버사이드 자바스크립트 환경'
        },
        {
            'name'        : 'AngularJS',
            'description' : '자바스크립트 SPA 프레임워크'
        }
    ]">
        <li data-ng-repeat="category in categories">
            <h4>{{category.name}}</h4>
            <p data-ng-bind="category.description"></p>
        </li>
    </ul>
</div>
```

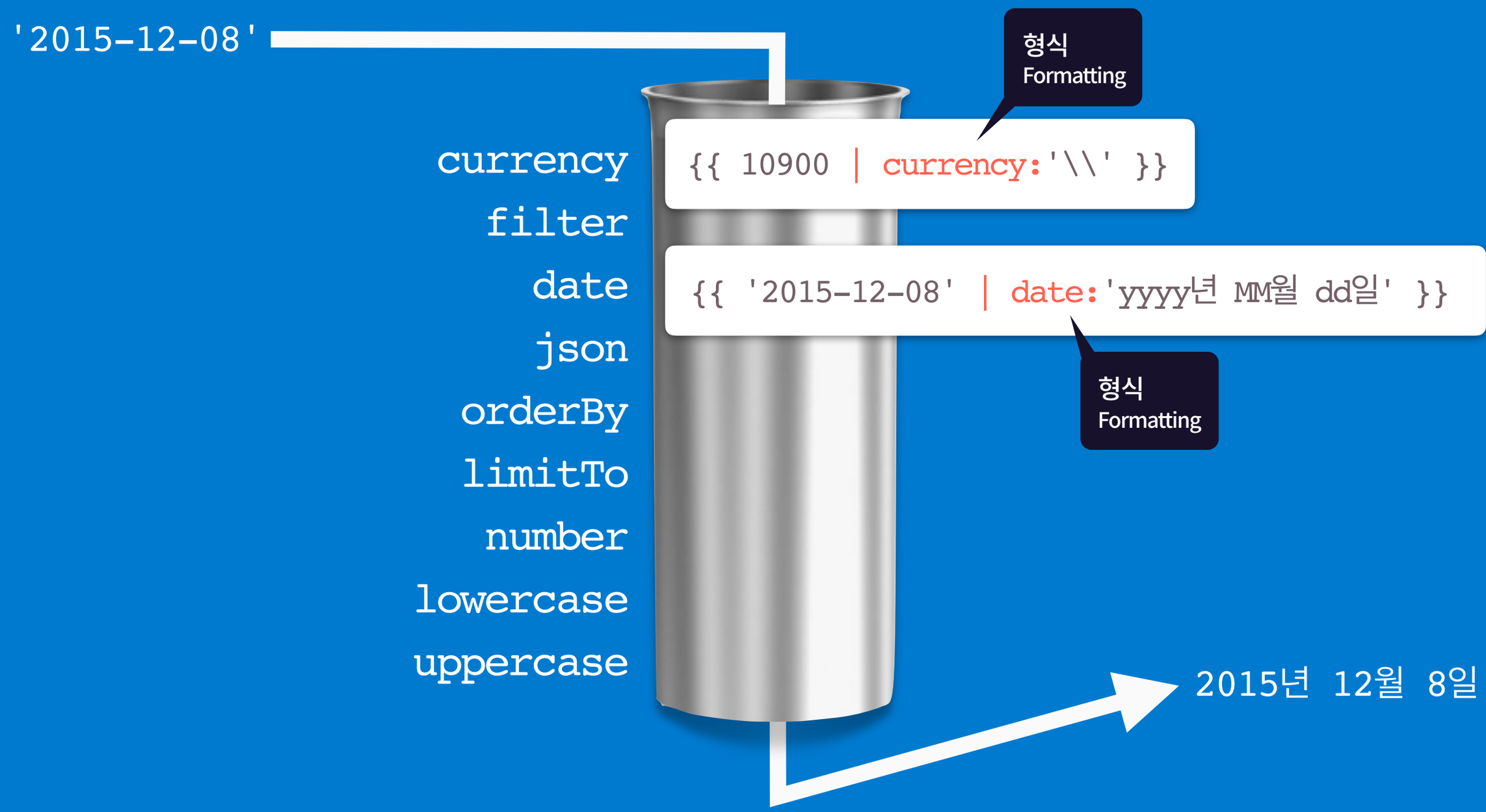
https://gist.github.com/yamoo9/6b6ed02603d4444a139f

Filters

Sorting & Formatting & Filtering Data

Filters & Pipe | Formatting Data

파이프(Pipe, |)를 통해 데이터를 걸러내거나(Filtering),
형식(Formatting)에 맞춰 변경하거나, 정렬(Sorting)하는 함수를 실행한다.



Sorting Data

orderBy `{{ Expression | orderBy:'key':true }}`

json `{{ Expression | json }}`

yamoo9_example.html

yamoo9_example.html

```
<div class="container" data-ng-init="categories=[
            {
                'name'        : 'DOM',
                'description' : '문서객체모델'
            },
            {
                'name'        : 'Javascript',
                'description' : '자바스크립트'
            },
            {
                'name'        : 'Node.js',
                'description' : '서버사이드 자바스크립트 환경'
            },
            {
                'name'        : 'AngularJS',
                'description' : '자바스크립트 SPA 프레임워크'
            }
    ]">
    <h2 class="headline">AngularJS <code>ngRepeat</code> 디렉티브를 활용하여 데이터 반복 순환 처리</h2>
    <ul class="ngRepeat-demo">
        <li data-ng-repeat="category in categories | orderBy:'name'">
            <!-- {{category}} -->
            <h4>{{category.name}}</h4>
            <p data-ng-bind="category.description"></p>
        </li>
    </ul>
```

정렬
Sorting

Filtering Data

limitTo `{{ Expression | limitTo:3 }}`

filter `{{ Expression | filter:model }}`

yamoo9_example.html

search.name

search.description

걸러내기
Filtering

```
<input type="search" data-ng-model="search">
<ul class="ngRepeat-demo">
    <li data-ng-repeat="category in categories | filter:search | limitTo:2 | orderBy:'-name'">
        <h4>{{category.name}}</h4>
        <p data-ng-bind="category.description"></p>
    </li>
</ul>
```

Node.js

서버사이드 자바스크립트 환경

Javascript

자바스크립트

DOM

문서객체모델

AngularJS

자바스크립트 SPA 프레임워크

d

Node.js

서버사이드 자바스크립트 환경

DOM

문서객체모델

Dynamic Sorting with orderBy Filter

orderBy 필터(정렬, Sorting)에 상수가 아닌

sortBy, reverse 변수 값을 설정하여 동적으로 정렬 상태를 변경할 수 있다.

yamoo9_example.html

yamoo9_example.html

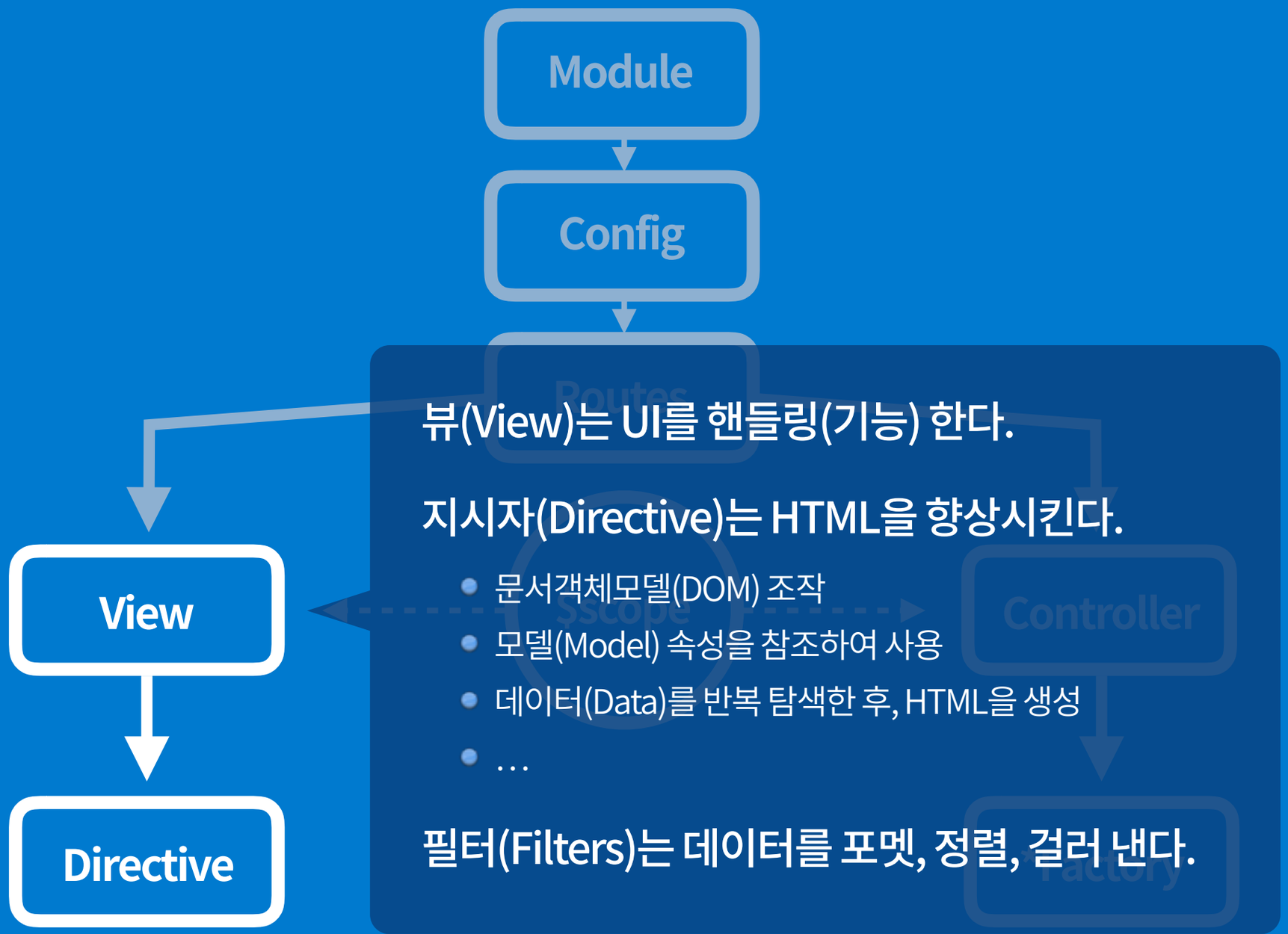
```
1

<table class="ngRepeat-demo" data-ng-init="sortBy='name'">
    <caption class="a11y-hidden">강의 카테고리</caption>
    <tr>
        <th>
            제목
            <button type="button" data-ng-click="sortBy='name'; reverse=!reverse; name_r=!name_r">
                <span data-ng-show="name_r">∧</span>
                <span data-ng-hide="name_r">v</span>
            </button>
        </th>
        <th>
            내용
            <button type="button" data-ng-click="sortBy='description'; reverse=!reverse; desc_r=!desc_r">
                <span data-ng-show="desc_r">∧</span>
                <span data-ng-hide="desc_r">v</span>
            </button>
        </th>
    </tr>
    <tr data-ng-repeat="category in categories | orderBy:sortBy:reverse">
        <td>{{category.name || uppercase}}</td>
        <td>{{category.description}}</td>
    </tr>
</table>
```

제목 v	내용 v
AngularJS	Javascript SPA 프레임워크
DOM	DOM = 문서객체모델
Javascript	자바스크립트
Node.js	서버사이드 Javascript 환경

Summary

View

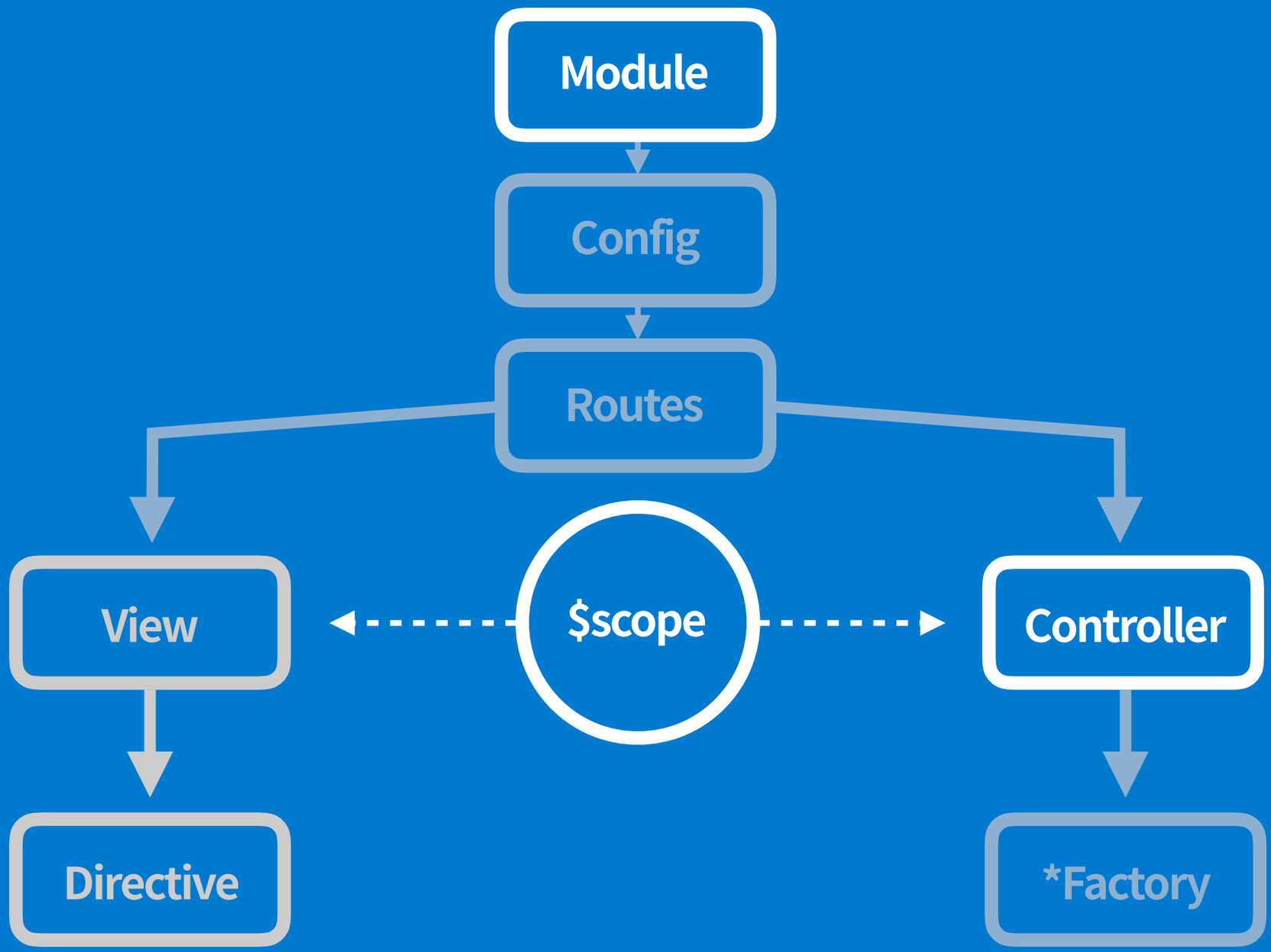
Directives / Expressions / Filters



Controller / Scope / Module

Point

Module / Controller / Scope

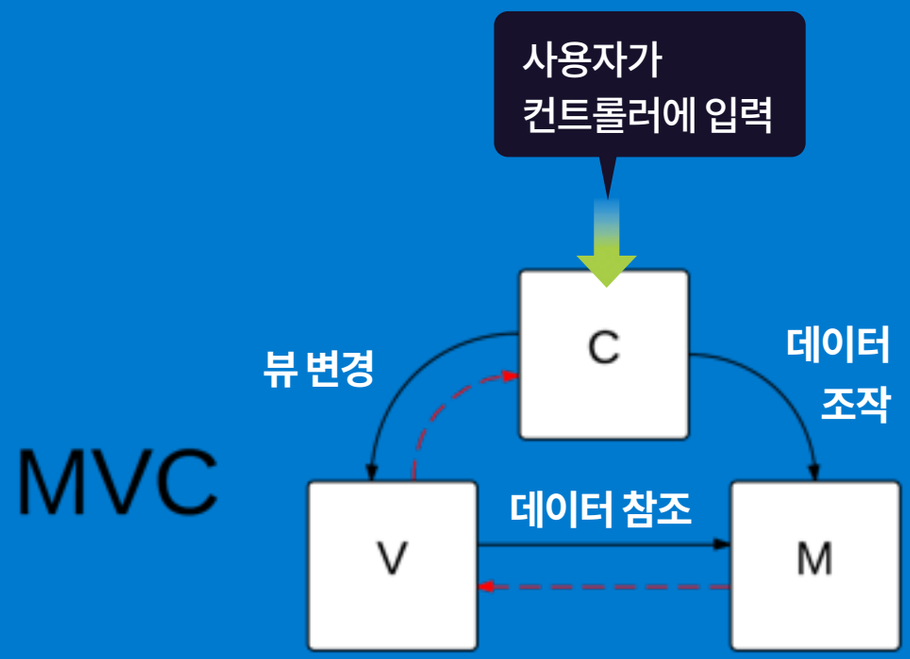


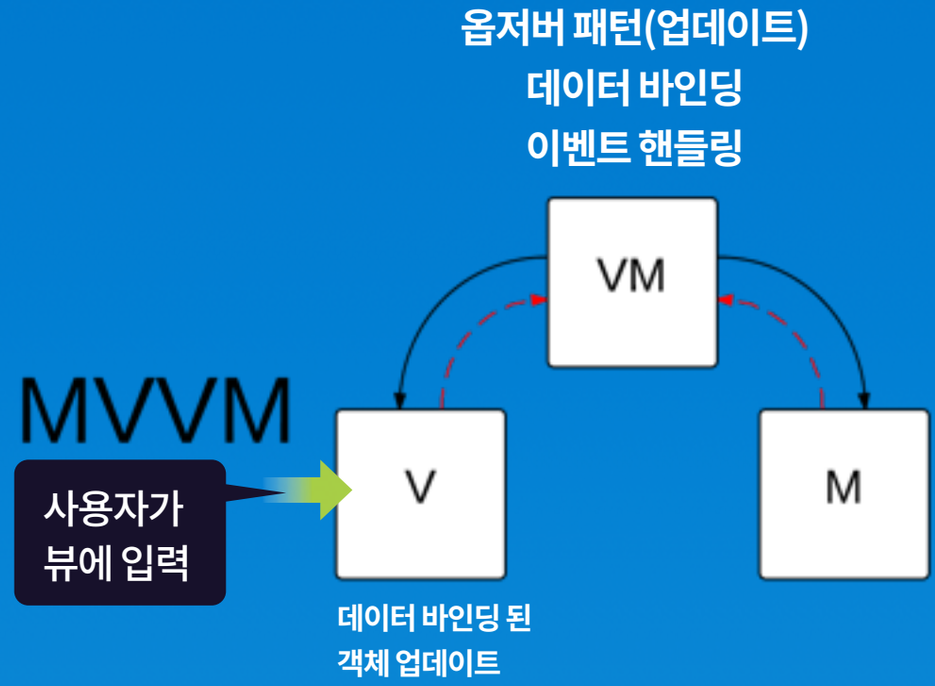
Architecture Patterns

Architecture Patterns

AngularJS는 2가지 디자인 패턴(M-V-C, M-V-VM)을 사용한다.

※ 패턴은 애플리케이션 제작 시 일관된 코딩 스타일을 약속하는 가이드일 뿐, 강제화 된 규칙은 아니지만 패턴을 사용함으로서
모듈화(Modularity), 유연성(Flexibility), 테스트 용이성(Testability), 유지보수(Maintainable) 를 향상시킬 수 있음.





직접 연결
간접 연결

Model 데이터/로직

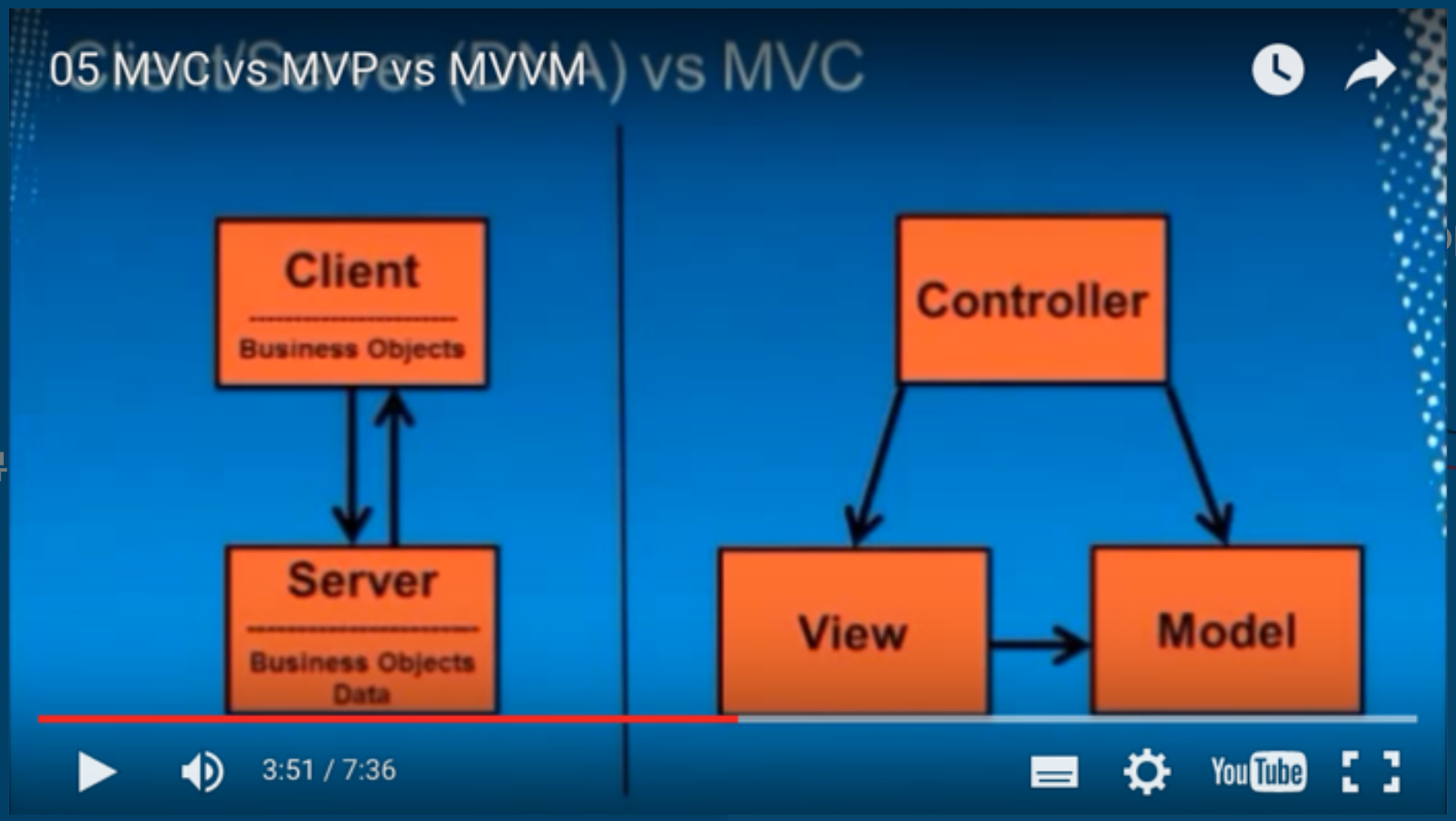
View GUI 레이아웃/컴포넌트

Controller/ViewModel Model/View를 연결하는 접착제

Architecture Patterns

AngularJS는 2가지 디자인 패턴(M-V-C, M-V-VM)을 사용한다.

※ 패턴은 애플리케이션 제작 시 일관된 코딩 스타일을 약속하는 가이드일 뿐, 강제화 된 규칙은 아니지만 패턴을 사용함으로서
모듈화(Modularity), 유연성(Flexibility), 테스트 용이성(Testability), 유지보수(Maintainable) 를 향상시킬 수 있음.



https://www.youtube.com/watch?v=ElHslw5wMWg

MVC

뷰

이트)

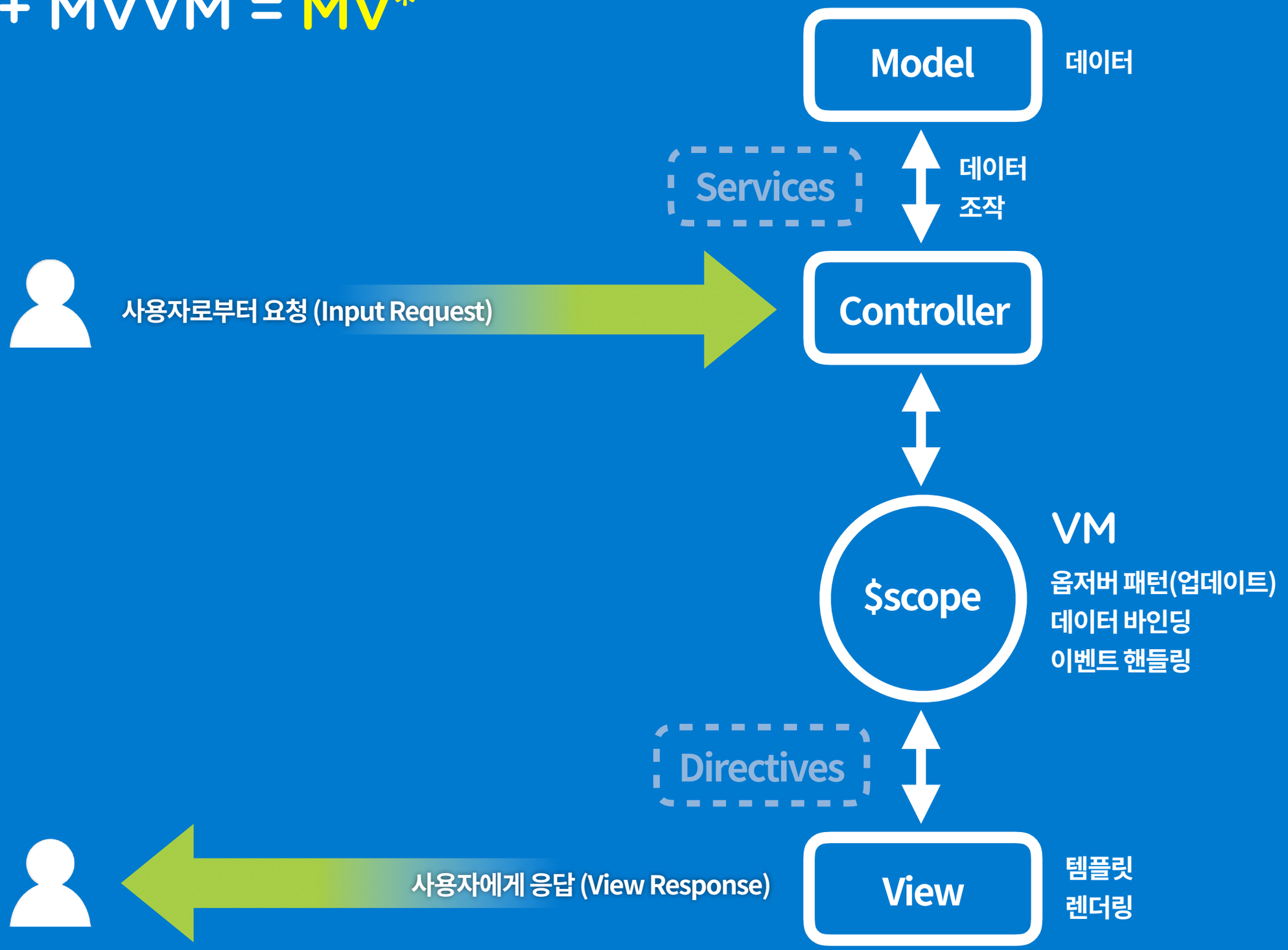
직접 연결

간접 연결

Model 데이터/로직 View GUI 레이아웃/컴포넌트 Controller/ViewModel Model/View를 연결하는 접착제

Architecture Patterns

MVC + MVVM = MV*



Controller

Role of Controllers

Controller

컨트롤러는 뷰(View)를 제어하는 브레인(Brain) 역할

- 속성/메소드 정의
- 뷰(View)에서 데이터/컨트롤을 감춤/보임 조작
- 뷰(View)로부터 이벤트 감지/작동 조작
- 데이터 조작(읽기/쓰기)
- $scope 객체(VM)를 사용하여 뷰(View)와 상호작용(Interaction)

$scope

$scope 객체는 View와 Controller를 연결하는 매개체(Mediator)

- 컨트롤러(Controller) 내부에 포함(주입, Injected)되어 접근 가능
- 뷰모델(ViewModel) 역할
- 뷰(View)는 $scope 객체의 속성/메소드에 연결

Define Controller

컨트롤러(Controller) 내부에 주입된 $scope 객체에 접근하여 속성/메소드를 설정할 수 있다.

$scope객체의 속성/메소드는 뷰(View)에서 접근 가능하다.

yamoo9_example.html

yamoo9_example.html

```
// AngularJS - 컨틀로러(Controller)
function myController( $scope ) {

    // $scope 객체 - 속성 설정
    $scope.data = [
        {
            'name'        : 'DOM',
            'description' : 'DOM = 문서객체모델'
        },
        {
            'name'        : 'Javascript',
            'description' : '자바스크립트'
        }
    ];

    // $scope 객체 - 메소드 설정
    $scope.saveData = function( data ) {
        $scope.data.push( data );
    }

}
```

컨트롤러에 동적으로
주입된 $scope 객체

함수를 컨트롤러로 바로 사용할 수 있는 것은
AngularJS 1.2.x 버전까지!

ngController - Directive

yamoo9_example.html

yamoo9_example.html

```
1  <div class="container" data-ng-controller="myController"> ... </div>

   <script>
   // AngularJS - 컨틀로러(Controller)
   function myController( $scope ) {
       // $scope 객체 - 속성 설정
       $scope.sortBy = 'name';
       $scope.reverse = false;

       $scope.categories=[
           {
               'name'        : 'DOM',
               'description' : 'DOM = 문서객체모델'
           },
           {
               'name'        : 'Javascript',
               'description' : '자바스크립트'
           },
           {
               'name'        : 'Node.js',
               'description' : '서버사이드 Javascript 환경'
           },
           {
               'name'        : 'AngularJS',
               'description' : 'Javascript SPA 프레임워크'
           }
       ];
       // $scope 객체 - 메소드 설정
       $scope.doSort = function(prop) {
           $scope.sortBy = prop;
           $scope.reverse = !$scope.reverse;
           if (prop === 'name') {
               $scope.name_reverse = !$scope.name_reverse;
           } else {
               $scope.desc_reverse = !$scope.desc_reverse;
           }
       }
   }
   </script>
```

뷰에 컨트롤러 연결

AngularJS 1.3.x 버전 이후부터는
이 방법을 사용할 수 없어요.

Controller with as

컨트롤러(Controller) 내부에서 $scope 객체를 사용하지 않고도 뷰(view)에 연결 가능한 방법은 this 참조를 사용하는 것이다. 단 컨트롤러 as 문법을 사용하여야 컨트롤러 this 객체에 접근 가능.

yamoo9_example.html

```
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.9/angular.min.js"></script>
</head>
<body>

<div class="container" data-ng-controller="myController as myCtrl">
    <h2 class="headline">AngularJS <code>ngRepeat</code> 디렉티브를 활용하여 데이터 반복 순환 처리</h2>
    <input type="search" data-ng-model="search">
    <ul class="ngRepeat-demo">
        <li data-ng-repeat="category in myCtrl.categories | filter:search | orderBy:'name'">
            <h4>{{category.name}}</h4>
            <p data-ng-bind="category.description"></p>
        </li>
    </ul>

    <hr>

    <table class="ngRepeat-demo">
        <caption class="a11y-hidden">강의 카테고리</caption>
        <tr>
```

```
// AngularJS - 컨틀로러(Controller)
function myController() {
    this.sortBy = 'name';
    this.reverse = false;
    this.name_reverse = false;
    this.desc_reverse = false;
    this.categories=[ ... ];
    this.doSort = function(prop) {
        this.sortBy = prop;
        this.reverse = !this.reverse;
        if (prop === 'name') {
            this.name_reverse = !this.name_reverse;
        } else {
            this.desc_reverse = !this.desc_reverse;
        }
    }
}
```

Module

Role of Modules

Module

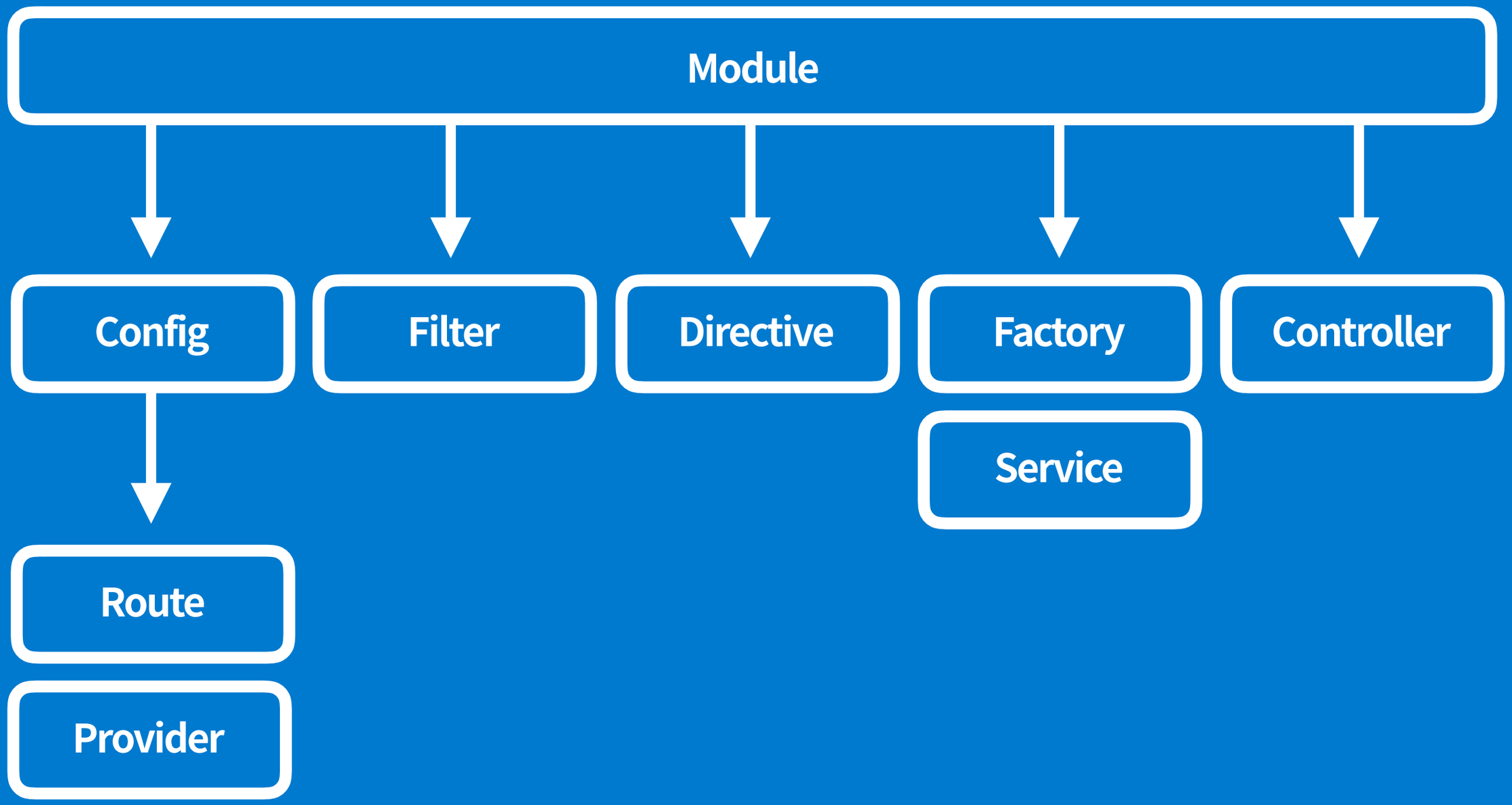
모듈(Module)은 컨테이너(Container)

- 컨트롤러(Controllers)
- 라우트(Routes)
- 팩토리(Factories)/서비스(Services)
- 디렉티브(Directives)
- 필터(Filters)

Module

모듈(Module)은 컨테이너(Container)

```
<html lang="ko-KR" data-ng-app="moduleName">
```



Create a Module

angular.module()을 사용하여 모듈 생성

생성된 모듈을 참조할 변수 설정

의존 모듈이 필요한 경우 [ ] 안에 포함 (Injecting Dependencies)

yamoo9_example.html

yamoo9_example.html

```
// AngularJS - 모듈 생성
var myApp = angular.module('myApp', []);


// AngularJS - 모듈 생성 (의존 모듈 포함)
var myApp = angular.module('myApp', ['modele1', 'module2']);
```

AngularJS 의존 모듈 추가

Adding a Controller - 1

컨트롤러(Contorller)를 모듈(Module)에 추가하는 첫 번째 방법은
정의된 모듈을 참조하는 변수를 통해 컨트롤러를 연결한다.

yamoo9_example.html

생성된 모듈 참조 변수

```
1
var myApp = angular.module('myApp', []);

myApp.controller('myController', function($scope) {...});
```

Adding a Controller - 2

컨트롤러(Contorller)를 모듈(Module)에 추가하는 두 번째 방법은
angular.module()에 모듈 이름을 전달한 모듈 참조에 컨트롤러를 연결한다.

yamoo9_example.html

yamoo9_example.html

```
1

angular.module('myApp', []);

angular.module('myApp')
        .myApp.controller('myController', function($scope) {...});
```

angular.module()을 통한 모듈 참조

Adding a Controller - 3

컨트롤러(Contorller)를 모듈(Module)에 추가하는 세 번째 방법은
정의된 함수를 컨트롤러에 연결한 후, 모듈 참조에 연결한다.

yamoo9_example.html

yamoo9_example.html

```
1

angular.module('myApp', []);

function myController($scope) {...}

angular.module('myApp').controller('myController', myController);
```

함수를 컨트롤러 정의에 연결한 후, 모듈 참조에 추가

Avoiding String Compress on Build Process

빌드(Build) 과정에서 코드를 최적화(압축)하는 과정에 모듈 이름을 줄여쓰게 되면 오류가 발생하게 된다. 이를 미연에 방지하기 위한 방법은 [ ]을 이용하여 모듈 이름을 줄여쓰지 못하도록 설정한다.

yamoo9_example.html

```
var myApp = angular.module('myApp', []);
myApp.controller('myController', ['$scope', function($scope) {...}]);
```

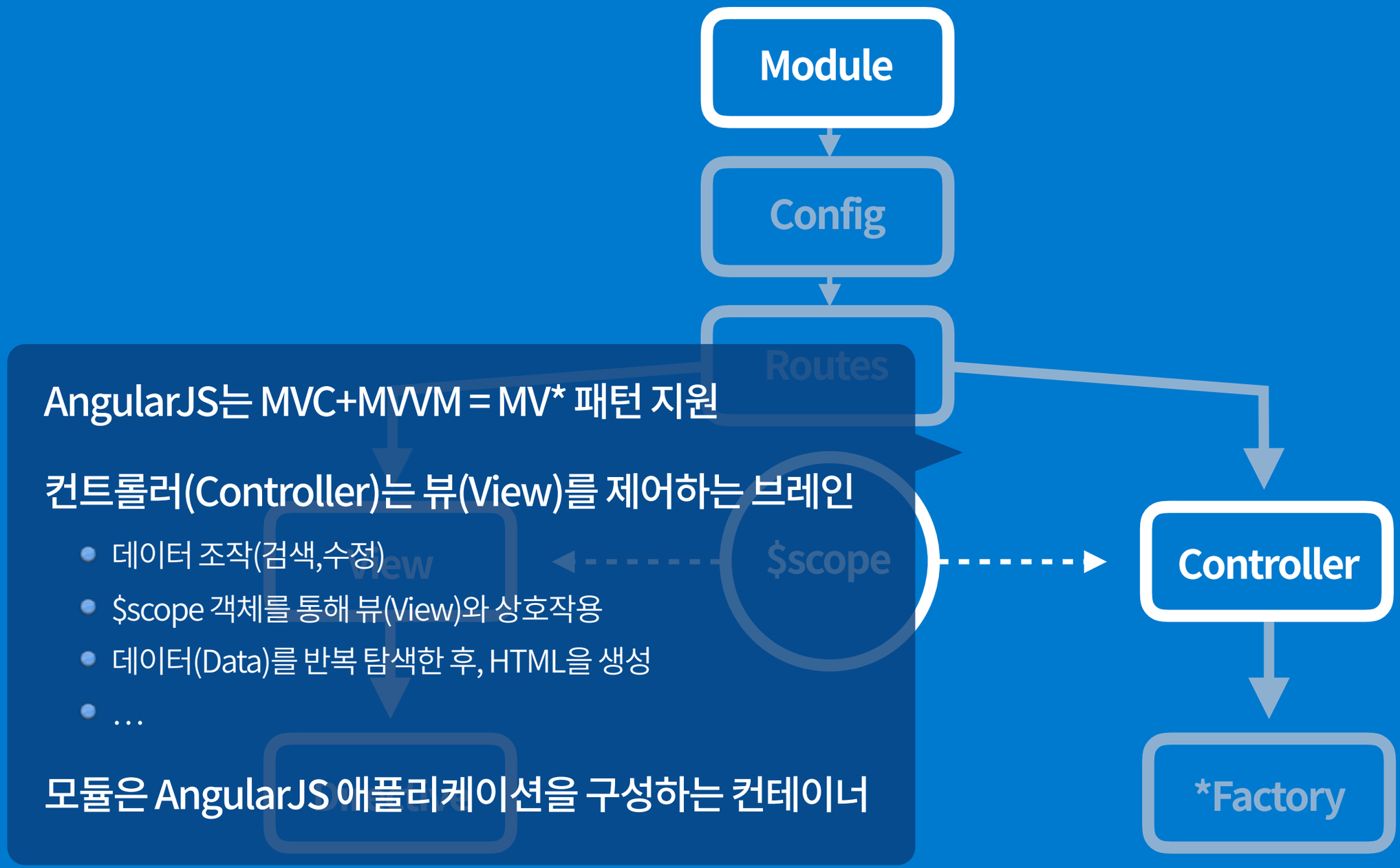
코드 압축 시, [] 안에 명시된 문자는 줄여 쓰지 않는다.

```
var myApp = angular.module('myApp', []);

(function(){
    function myController($scope) {...}
    angular.module('myApp').controller('myController', myController);
    myController.$inject = ['$scope'];
})();
```

코드 압축 시, [] 안에 명시된 문자는 줄여 쓰지 않는다.

Summary

Module / Controller / Scope



AngularJS는 MVC+MVVM = MV* 패턴 지원

컨트롤러(Controller)는 뷰(View)를 제어하는 브레인

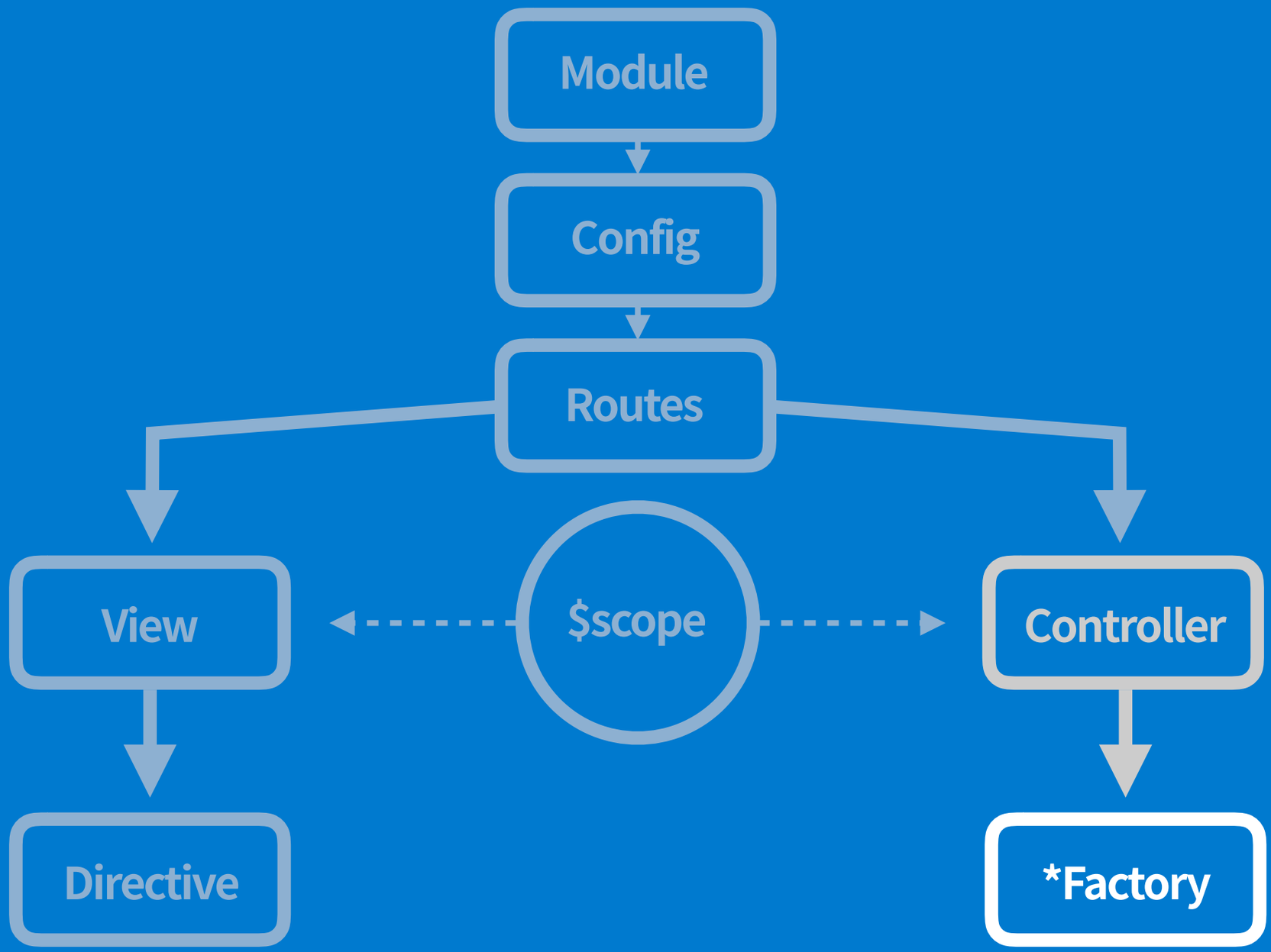
- 데이터 조작(검색,수정)
- $scope 객체를 통해 뷰(View)와 상호작용
- 데이터(Data)를 반복 탐색한 후, HTML을 생성
- …

모듈은 AngularJS 애플리케이션을 구성하는 컨테이너

Factories / Services

Point

Factory / Service



Factory / Service

Overview Factory/Service

Factory / Services

AngularJS는 팩토리(Factory) / 서비스(service)를 지원할 뿐 아니라,
다양한 팩토리/서비스를 내장(Built-in)하고 있다.

팩토리/서비스는 재사용 가능하도록 설계된 객체이다.

- 비동기 호출 (Ajax Calls)
- 비즈니스 규칙 (Business Rules)
- 연산 (Calculations)
- 컨트롤러 간 데이터 공유 (Share Data between Controllers)

Factory vs Services

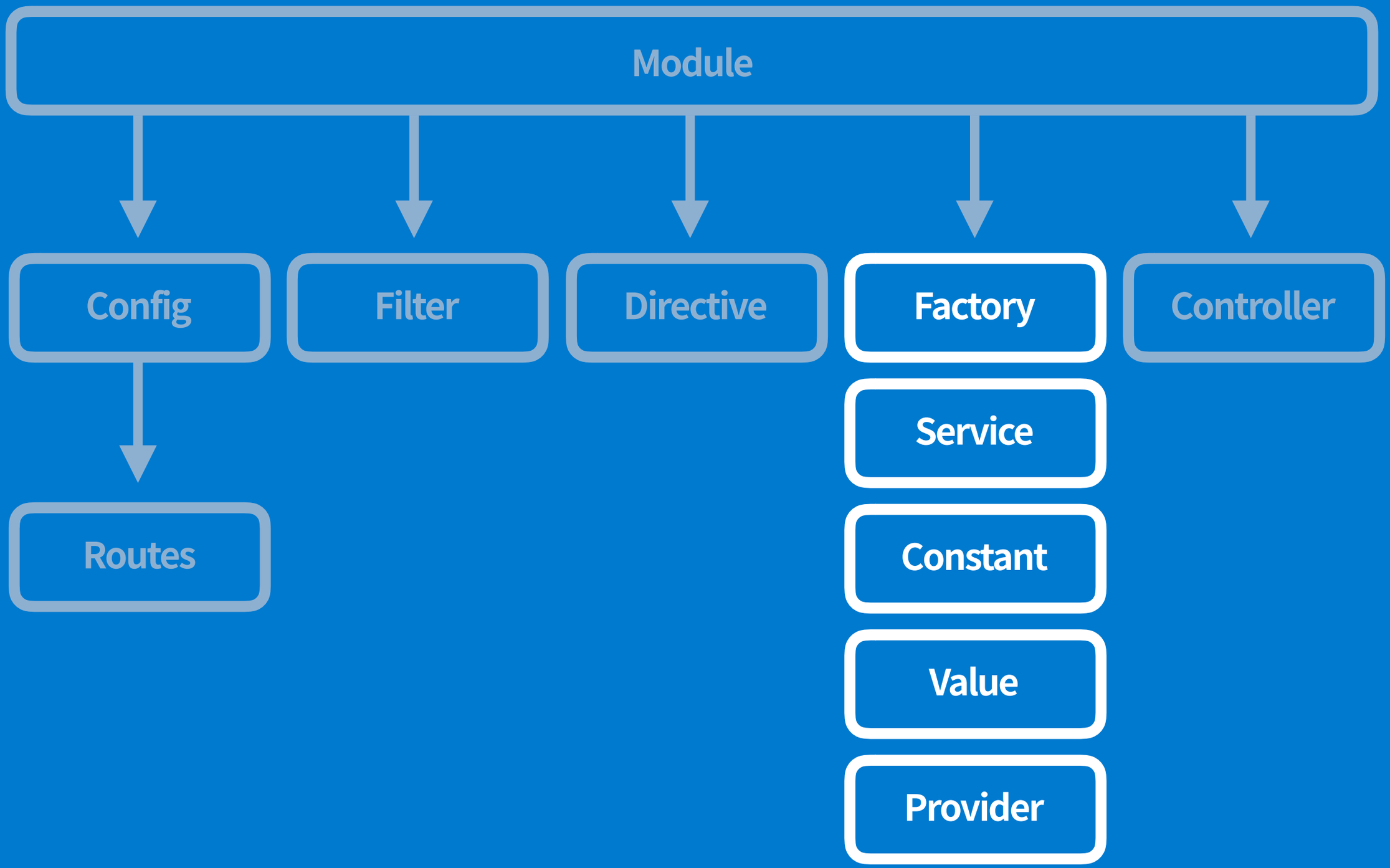
AngularJS는 서비스를 정의하는 5가지 방법을 제공한다.

AngularJS는 서비스를 정의하는 5가지 방법을 제공한다.
[Provider, Factory, Service, Constant, Value]
- 참고 글 -

- **Provider**
  Contant를 제외한 다른 서비스에서 사용
- **Factory**
  객체를 반환(return)하는 함수로 새로운 서비스를 구축할 때 사용
- **Service**
  this 속성을 사용한다는 점을 제외하고는 Factory와 유사 (Prototype 상속 활용)
- **Constant**
  서비스 설정 정보를 제공할 때 사용
- **Value**
  간단한 값이나 객체를 저장할 때 사용

Factory / Services

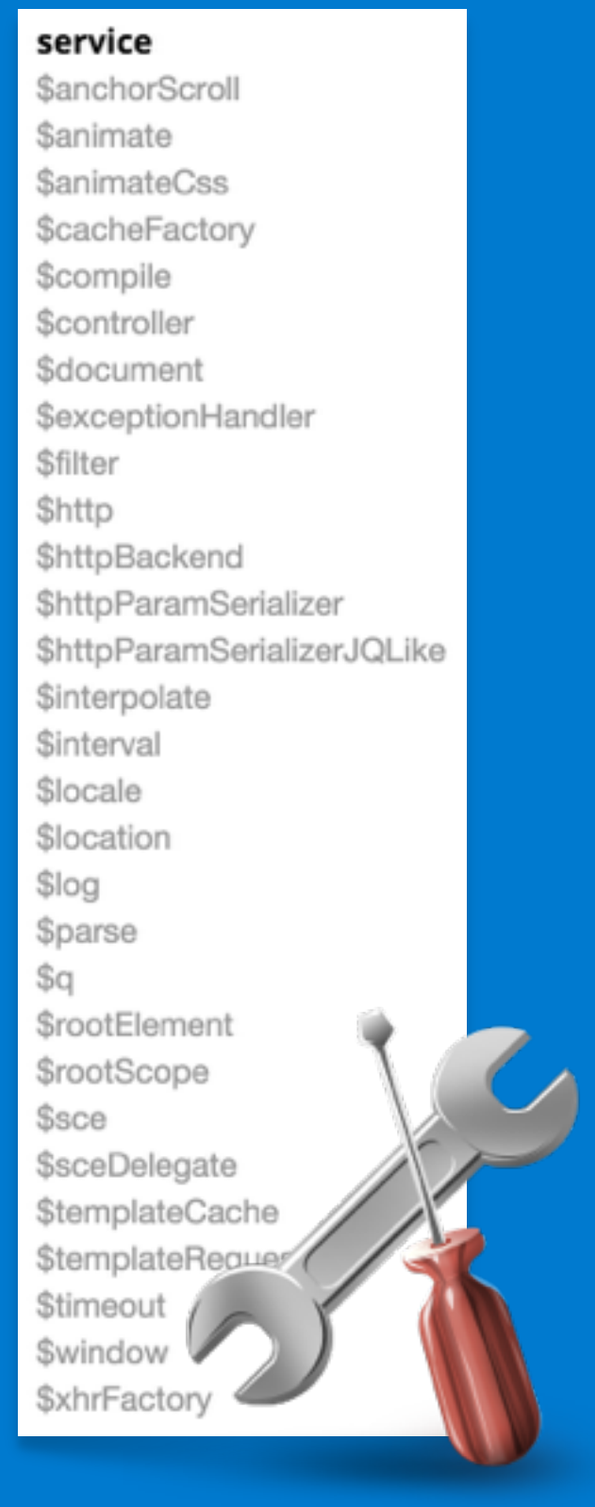
팩토리(Factory)/서비스(Service)



Built-in Services

다양한 AngularJS의 내장 서비스(Built-in Services)

- **$http**
- **$window**
- **$q**
- **$timeout**
- **$location**
- **$filter**
- **$interval**
- **$rootScope**
- **$log**



Factory

Creating & Injecting a Factory

About Factory

팩토리(Factory)란?

- 사용자 정의 싱글톤 객체
- angular.module.factory()를 사용하여 팩토리 생성
- 컴포넌트에 팩토리 삽입 가능
- 의존성 관리 필요

사용자 정의 팩토리(Factory)가 필요한 경우

- 재사용 가능한 업무 정의 (Re-usable Tasks)
- 컨틀롤러 간 데이터 공유 목적 (Share Code & State)

Creating a Factory

팩토리(Factory)를 정의한 후, 모듈에 추가한다.

```
(function(angular){
    'use strict';

    // 사용자 정의 팩토리 함수 정의
    var customFactory = function() {
        // ...
        // 사용자 정의 싱글톤 객체 반환
        return  {
            // ...
        };
    };

    // 사용자 정의 팩토리 함수 등록
    angular
        .module('app')
        .factory('customFactory', customFactory);

})(window.angular);
```

사용자 정의 객체 반환

모듈에 팩토리 함수 추가

Injecting a Factory

컨트롤러에 팩토리(Factory) 모듈을 호출한 후, 사용한다.

```
(function(angular){
    'use strict';

    var customController = function($scope, customFactory) {
        // 사용자 정의 customFactory 모듈 사용
        $scope.customers = customFactory.getCustomers();
        // ...
    };

    customController.$inject(['$scope', 'customFactory']);

    angular.module('app').controller('customController', customController);

})(window.angular);
```

팩토리 모듈 호출

팩토리 모듈 사용

Service

Creating a Service

About Service

서비스(Service)란?

- 팩토리와 기능적으로 매우 유사 (return 대신 this 사용)
- angular.module.service()를 사용하여 서비스 생성
- 컴포넌트에 팩토리 삽입 가능
- 의존성 관리 필요

Creating a Service

서비스(service)를 정의한 후, 모듈에 추가한다.

```
(function(angular){
    'use strict';

    // 사용자 정의 서비스 함수 정의
    var customService = function() {
        var customers = [];
        // this 참조 사용
        this.getCustomers = function() {
            // ...
        };
    };

    // 사용자 정의 팩토리 함수 등록
    angular
        .module('app')
        .service('customService', customService);

})(window.angular);
```

this 사용대신 Factory처럼
사용자 정의 객체를 리턴 해도 됨.

this 속성 사용

모듈에 서비스 함수 추가

Injecting a Service

컨트롤러에 서비스(Service) 모듈을 호출한 후, 사용한다.

```
(function(angular){
    'use strict';

    var customController = function($scope, customService) {
        // 사용자 정의 customService 모듈 사용
        $scope.customers = customService.getCustomers();
        // ...
    };

    customController.$inject(['$scope', 'customService']);

    angular.module('app').controller('customController', customController);
})(window.angular);
```

서비스 모듈 호출

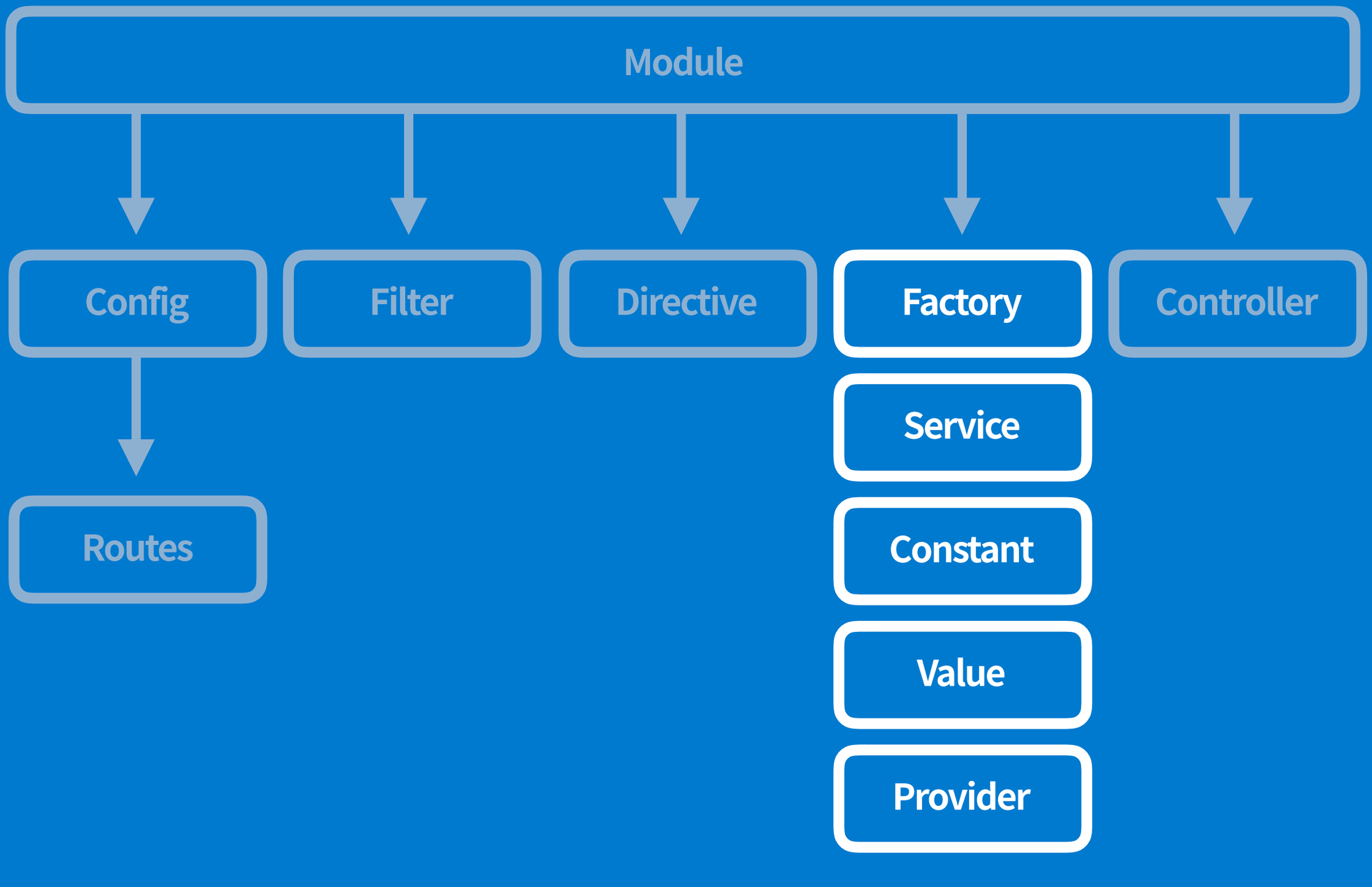
서비스 모듈 사용

Values

Application Values

Values

값(Values)



Value vs Constant

value와 cosntant는 비슷한 방법으로 정의하나, 설정(Config)에 포함(Inject)되는 여부가 다르다.

```
(function(angular){
    'use strict';

    // angular.module
    var app = angular.module('app', []);

    // angular.module.value(key, value)
    app.value('app_settings', {...});

    // angular.module.contant(key, value)
    app.constant('version', '1.1.2');

})(window.angular);
```

angular.module.config()에 동적으로 포함(Inject)되지 않음.
하지만 컨트롤러 내부에는 삽입하여 사용 가능.

angular.module.config()에 동적으로 포함(Inject)됨.

Ajax Calls

Ajax calls from a Factory/Service

Ajax & AngularJS Service

$http 서비스는 비동기 통신 기술(Ajax)을 사용한다.

$http

- $http 서비스는 비동기 통신(Ajax)을 수행하기 위한 함수를 제공한다.
- XMLHttpRequest 브라우저 내장 객체를 사용한다.
- 비동기 요청(Asynchronous Request)을 수행한다.
- 다양한 HTTP 단축 메소드(Shortcut methods)를 제공한다.
  - $http.head
  - $http.get
  - $http.post
  - $http.put
  - $http.delete
  - $http.jsonp

using $http Service

팩토리 함수 내부에 $http 서비스를 포함(주입, Inject)한 후,
$http 서비스를 사용할 수 있다.

```
(function(angular){
    'use strict';

    // 사용자 정의 팩토리 함수 정의
    var customFactory = function( $http ) {
        var factory = {};
        factory.getCustomers = function() {
            return $http.get('api/customers');
        };

        return factory;
    };

    angular
        .module('app')
        .factory('customFactory', customFactory);

})(window.angular);
```

$http 서비스 포함(주입)시킨다.

$http 서비스를 사용하여 비동기 호출

Accessing $http Response Data

$http 서비스는 비동기 통신으로 데이터를 호출한다.

- $q 서비스는 deferred/promise APIs 패턴을 사용하는 서비스를 제공한다.
- 비동기 통신으로 전달받은 데이터는 .then() 또는 .success()/.error() 함수를 사용하여 콜백 처리할 수 있다.

```
(function(angular){
    'use strict';

    var customController = function($scope, customFactory) {
        customFactory.getCustomers()
            .success(function( customers ){
                $scope.customers = customers;
            })
            .error(function( data, status, headers, config ){
                // error
            })
    };

    customController.$inject(['$scope', 'customFactory']);

    angular.module('app').controller('customController', customController);

})(window.angular);
```

$http 서비스에서 반환된 데이터

Summary

Factory / Service

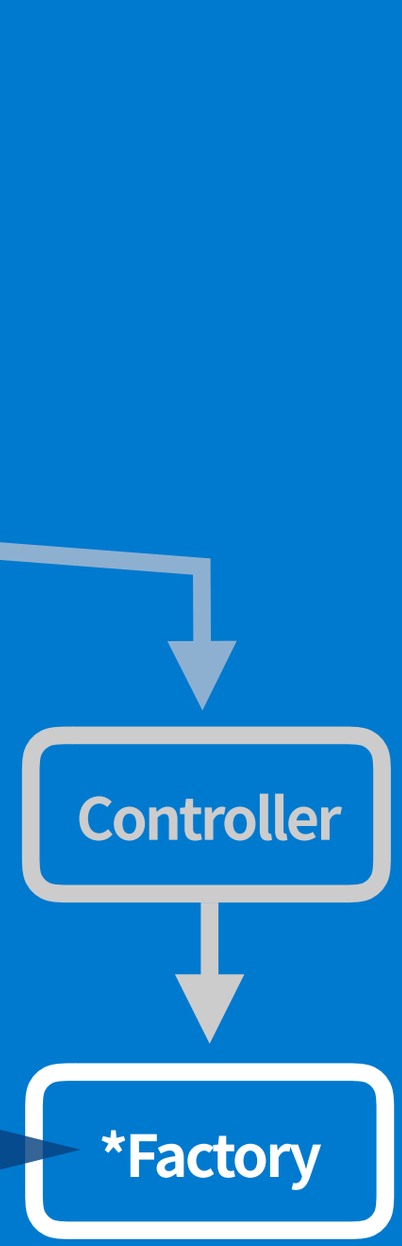
팩토리(Factory), 서비스(Service)는 싱글톤 객체.

재사용 가능한 업무 등록 시 사용한다.

- 비동기 호출 (Ajax Calls)
- 비즈니스 규칙 (Business Rules)
- 연산 (Calculations)
- 컨트롤러 간 데이터 공유 (Share Data between Controllers)

angular.module의 factory(), service()를 사용해 정의한다.

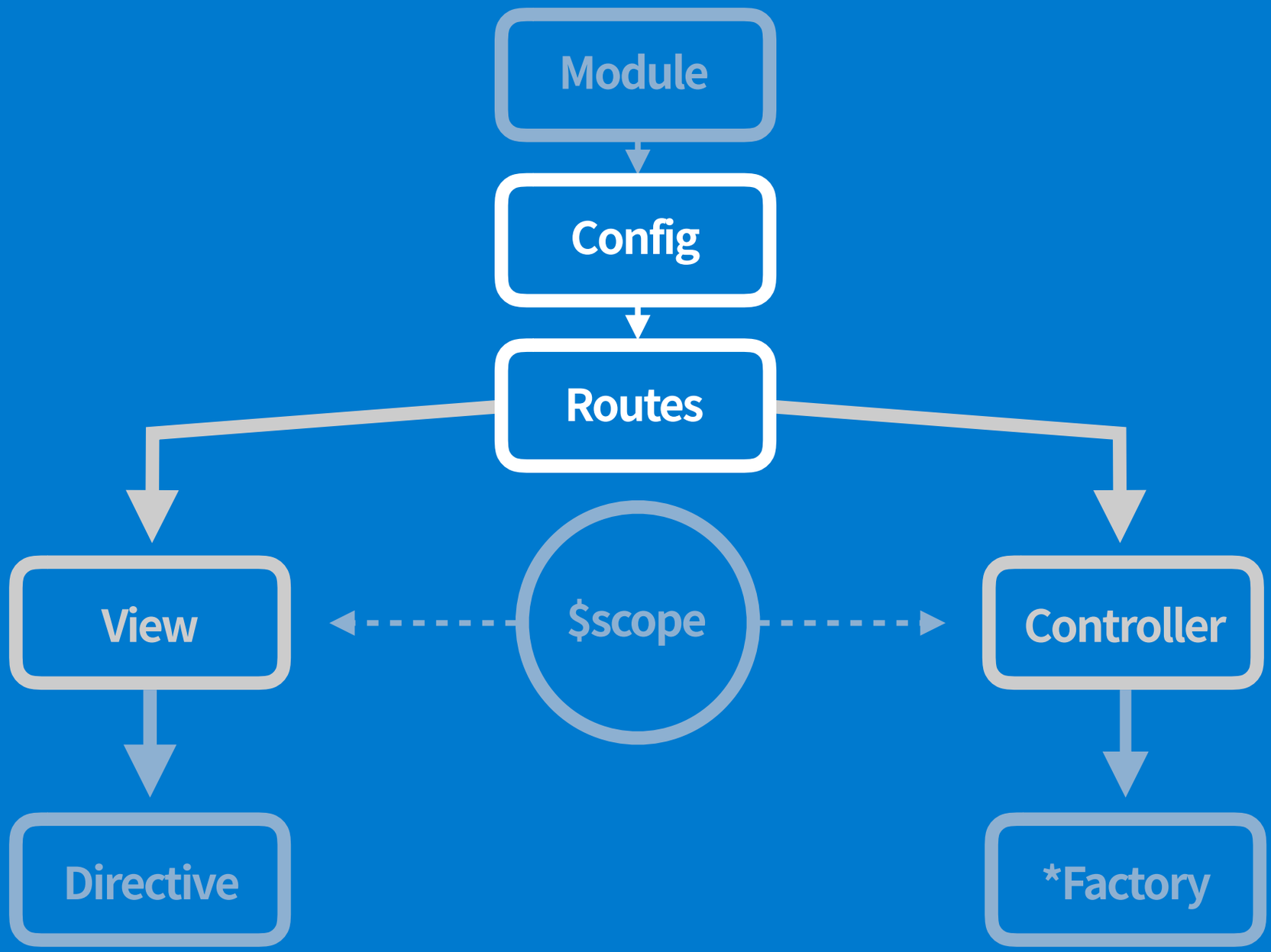
컨트롤러 내부에 팩토리/서비스 등을 주입(Inject)하여 사용한다.



Routing

Point

Routes / Configuration

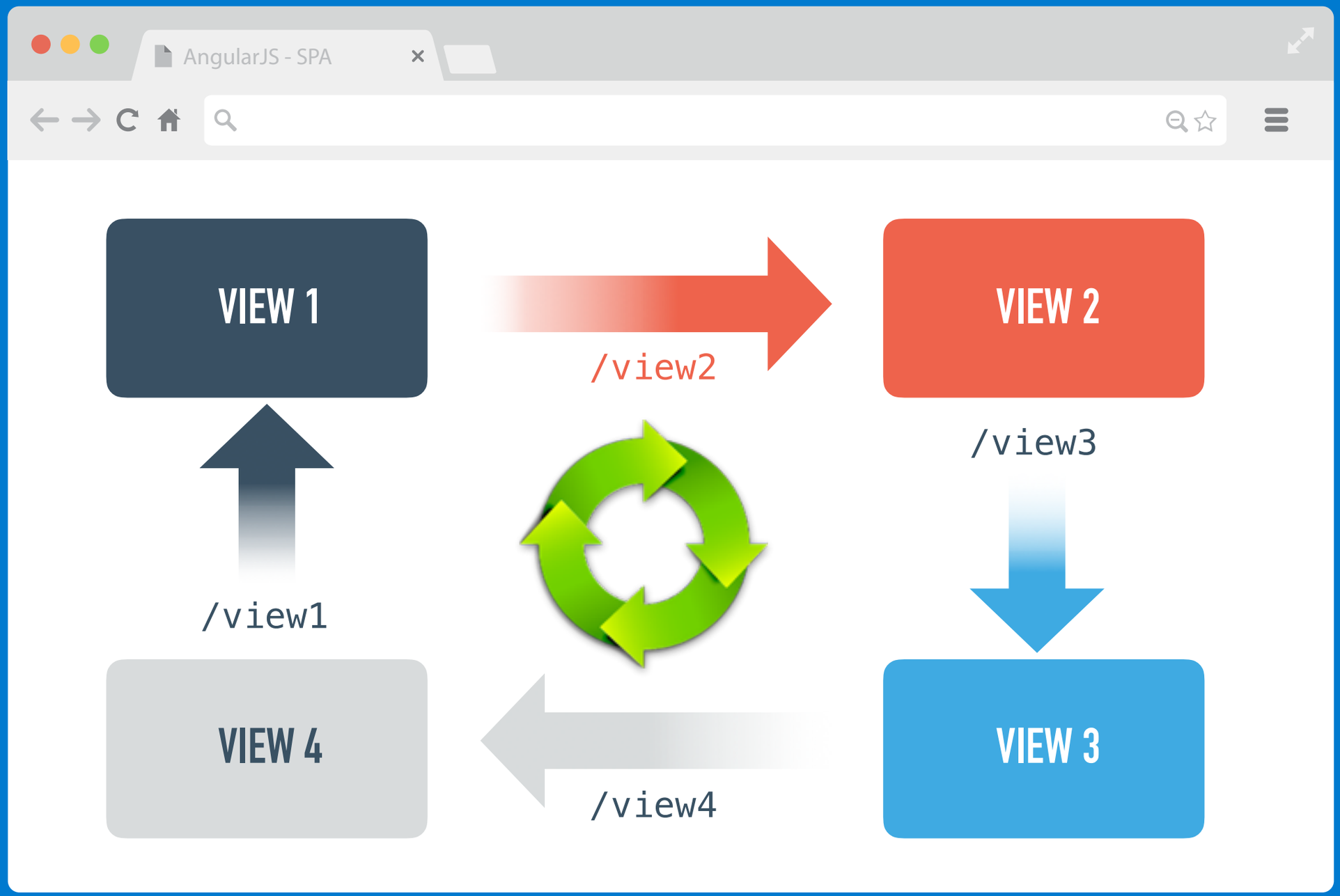


Routes

Navigation Each Page

Routes

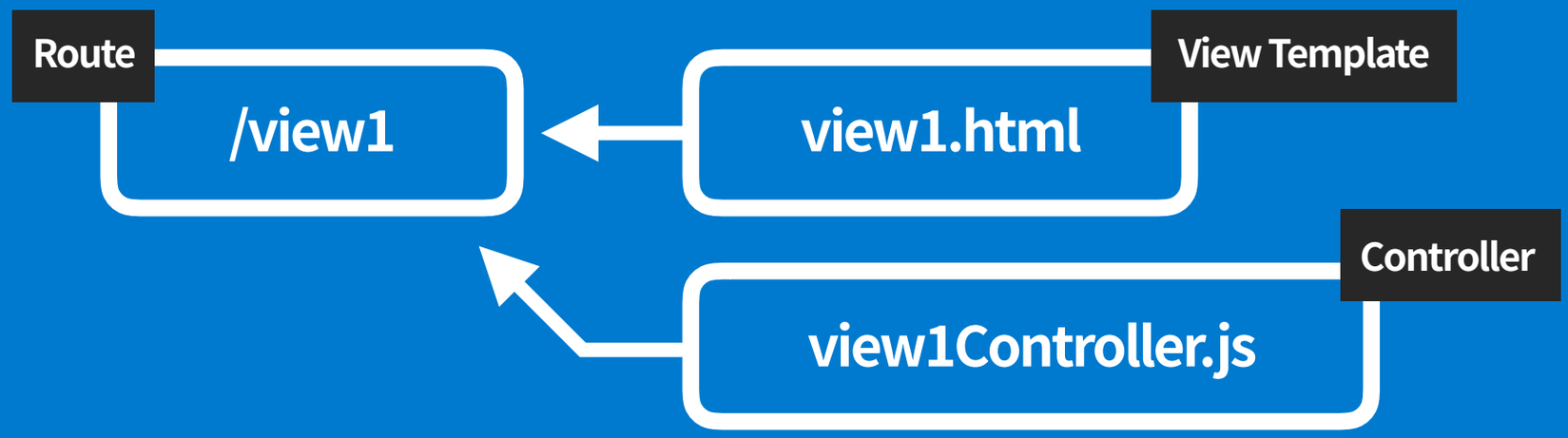
Routes는 항로를 설정하는 역할을 수행한다.



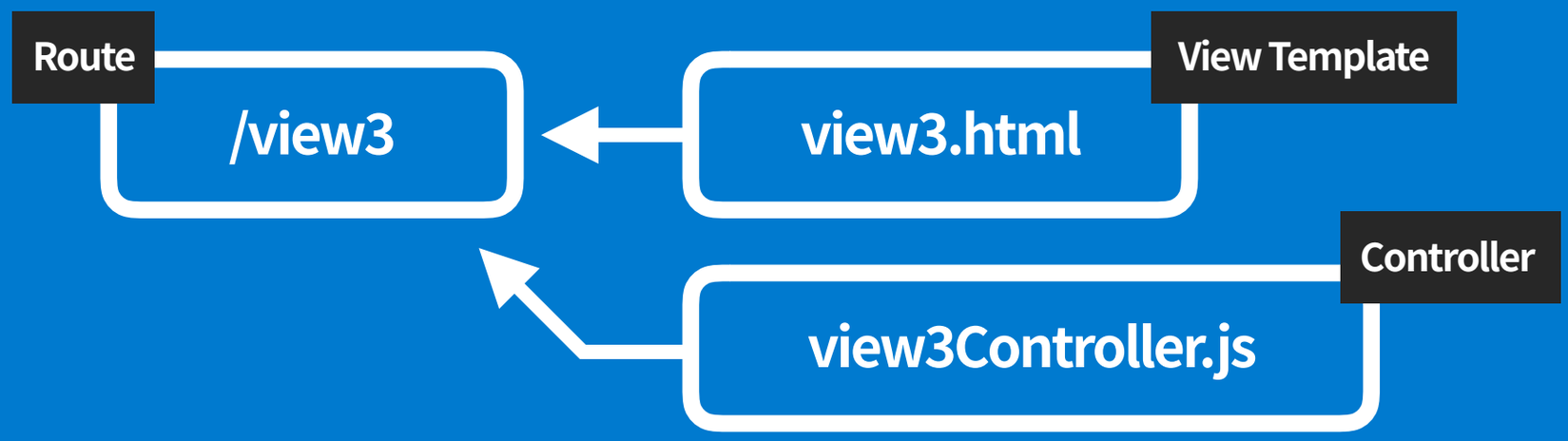
Routes

Routes는 해당 항로에 Controller와 View Template을 연결시켜주는 역할을 수행한다.

/VIEW 1



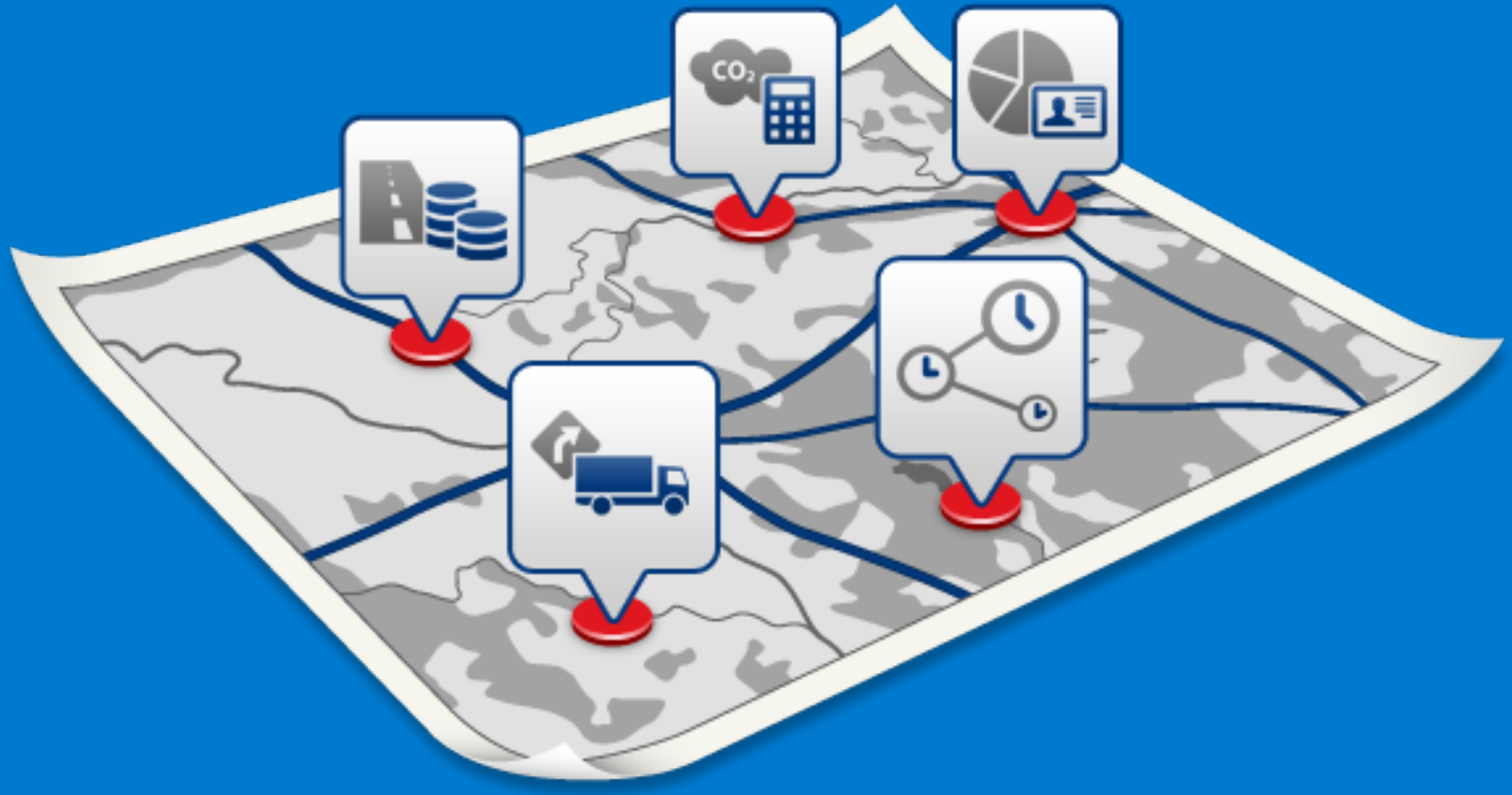
/VIEW 3



Routes

Routes를 사용하기 위한 조건과 환경 설정

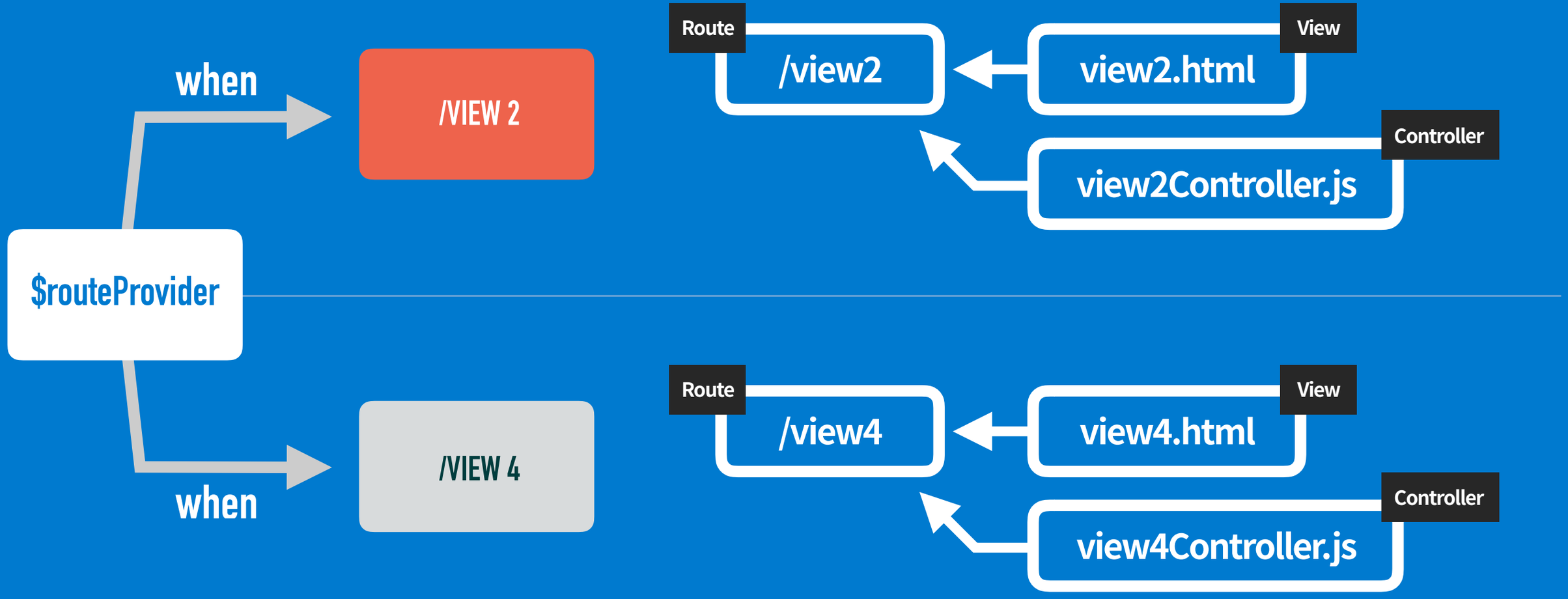
- ngRoute 모듈(module)이 필요
- Routes 설정(Configuration)은 $routeProvider를 사용



Routes & Module

Routes 설정은 angular.module.config()를 통해 수행하며
$routeProvider는 동적으로 내부에 추가된다.

```
angular.module.config()
```

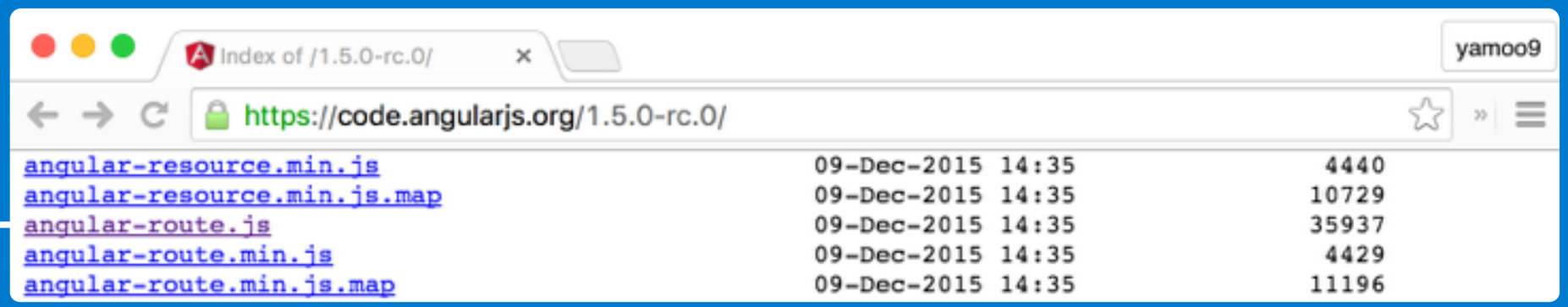


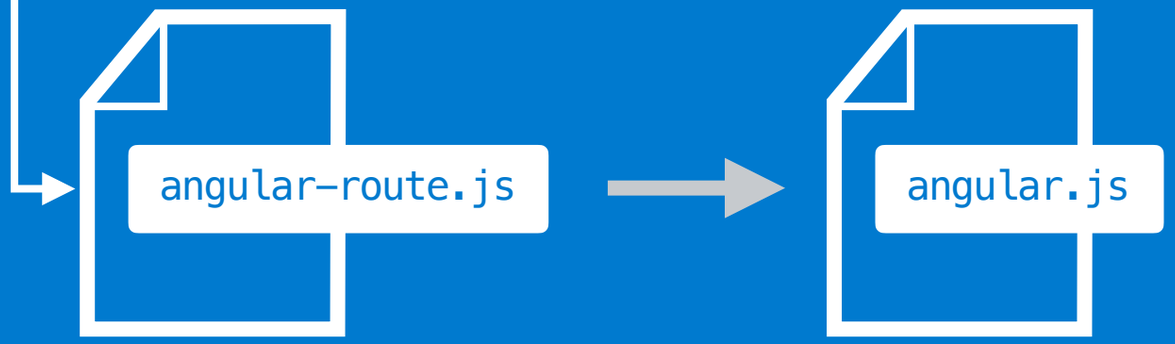
ngRoute

angular Route Module

ngRoute Module

ngRoute 모듈은 angular-route.js 모듈 파일을 호출해야 사용 가능하다.

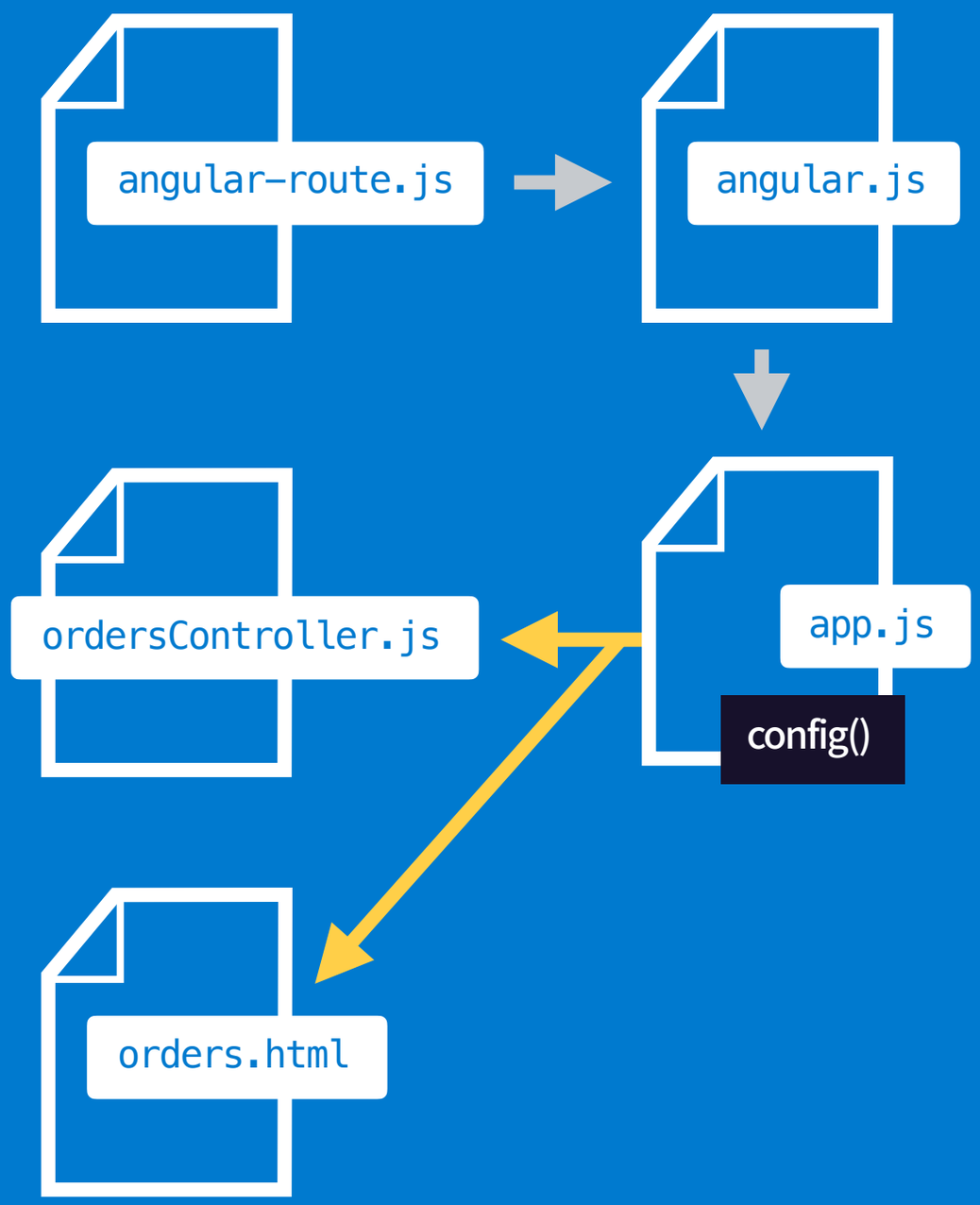




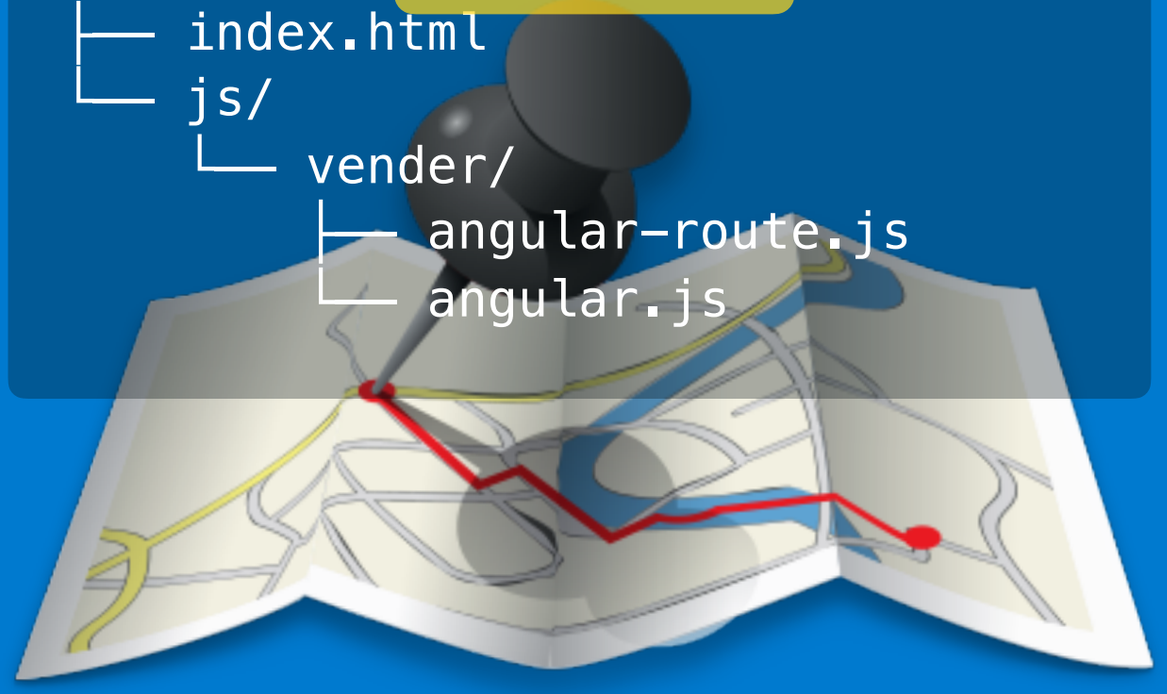

```
var myApp = angular.module('myApp', ['ngRoute']);
```

Route = View + Controller

라우트(항로) 설정을 통해 각 View, Controller를 라우트 페이지에 연결한다.




```
.
├── app/
│   ├── app.js
│   ├── controllers/
│   │   ├── customersController.js
│   │   └── ordersController.js
│   └── views/
│       ├── customers.html
│       └── orders.html
├── index.html
└── js/
    └── vender/
        ├── angular-route.js
        └── angular.js
```



Routes Config

Configuration Routes

Configuring Routes

라우트 설정은 angular.module.config()를 사용한다.

```
var myApp = angular.module('myApp', [‘ngRoute']);


myApp.config(function( $routeProvider ) {
    // Routes 설정
});
```

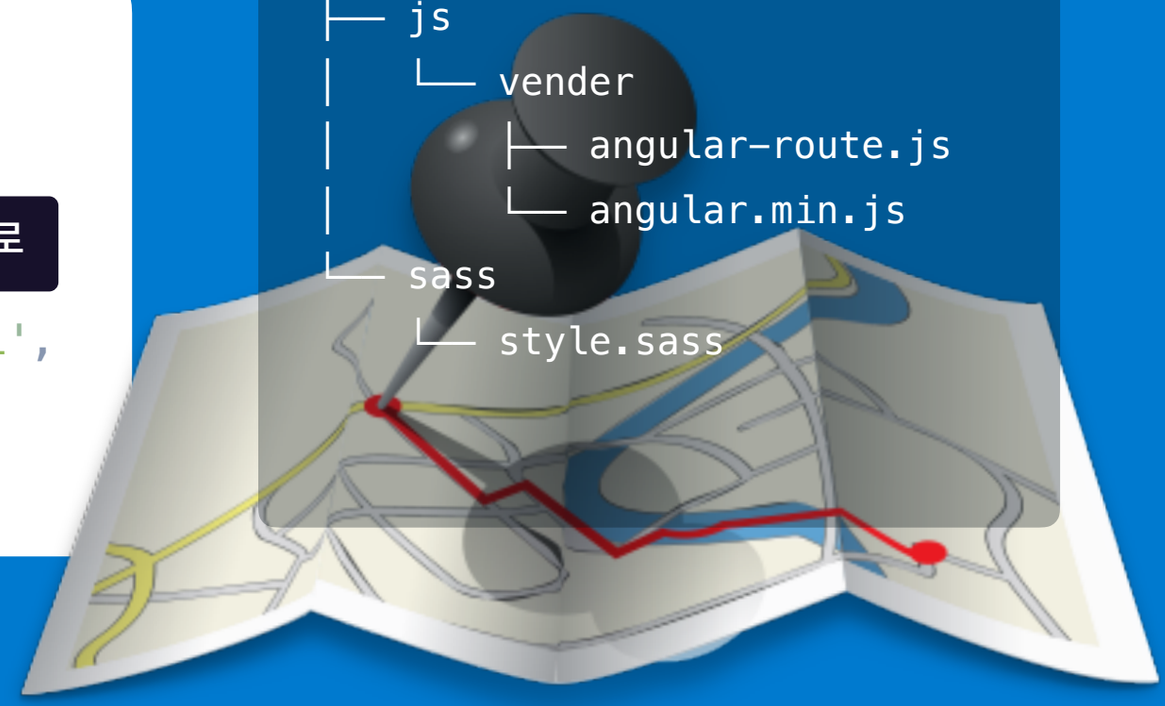
$routeProvider 동적으로 인자 설정됨.

```
var myApp = angular.module('myApp', [‘ngRoute']);


myApp.config(function( $routeProvider ) {
    $routeProvider
        .when('/', {
            'templateUrl' : 'app/views/view1.html',
            'controller'  : 'view1Controller'
        });
});
```

파일 경로

myApp.controller()에 등록된 이름

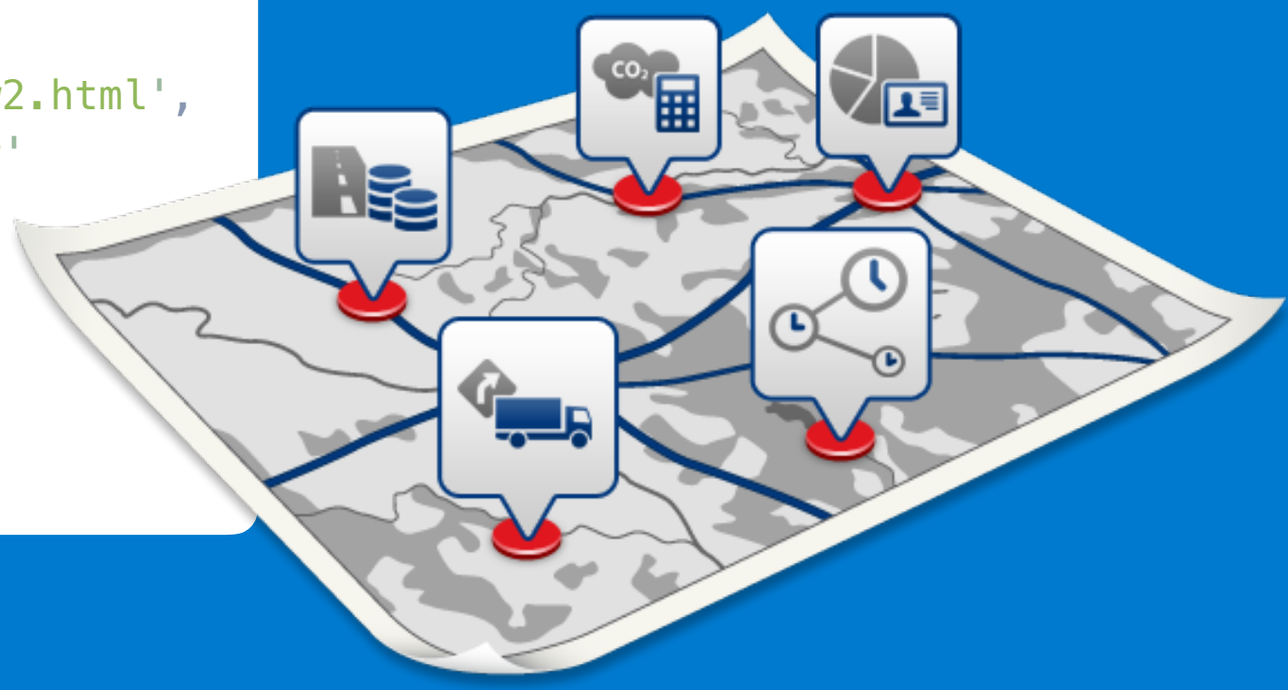
```
.
├── app
│   ├── app.js
│   ├── controllers
│   │   └── view1Controller.js
│   └── views
│       └── view1.html
├── css
│   └── app.css
├── index.html
├── js
│   └── vender
│       ├── angular-route.js
│       └── angular.min.js
└── sass
    └── style.sass
```



Configuring Routes

각 라우트(항로)는 when() 내부에 경로, 템플릿, 컨트롤러를 지정한다.
지정되지 않은 항로일 경우, otherwise()를 사용하여 항로를 재설정(redirectTo) 한다.

```
var myApp = angular.module('myApp', ['ngRoute']);

// $routeProvider 동적으로 인자 설정됨.
myApp.config(function( $routeProvider ) {
    // Routes 설정
    $routeProvider
        .when('/', {
            'templateUrl' : '/app/views/view1.html',
            'controller'  : 'view1Controller'
        })
        .when('/view2', {
            'templateUrl' : '/app/views/view2.html',
            'controller'  : 'view2Controller'
        })
        .otherwise({
            redirectTo: '/'
        });
});
```



Route Parameters

라우트 매개변수(Parameter)는 : 문자열을 사용하여 설정한다.

라우트 매개변수
Route Parameter

```
.when('/editView/:layoutId', {
    'templateUrl' : '/app/views/edit-view-layout.html',
    'controller'  : 'viewEditController'
})
```

```
/editView/94
```

라우트 예시

컨트롤러에서 라우트 매개변수(Parameter)를 사용할 경우, $routeParams를 사용한다.

```
var myApp = angular.module('myApp');
myApp.controller('view1Controller', function($scope, $routeParams) {
    $scope.layoutId = $routeParams.layoutId;
});
```

ngView

angular View Directive

ngView Directive

동적으로 읽어들인 템플릿 뷰 페이지(HTML Template)는
ngView 디렉티브가 설정된 곳에 삽입된다. (속성 설정 권장! 표준 준수)

Location Config

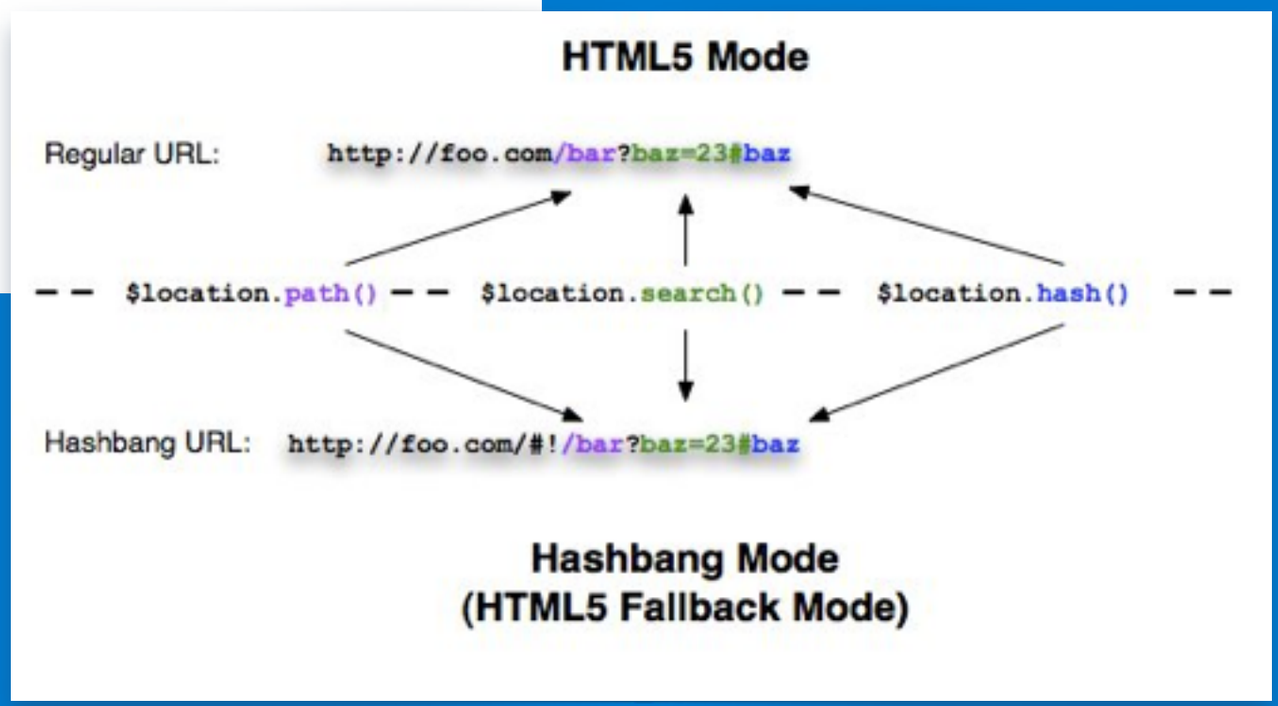
Configuration Location

Configuring Location

해시 뱅(Hash Bang, #!)을 URL에 사용하지 않고자 할 경우
$locationProvider를 사용하여 html5Mode를 활성화(true) 설정한다.

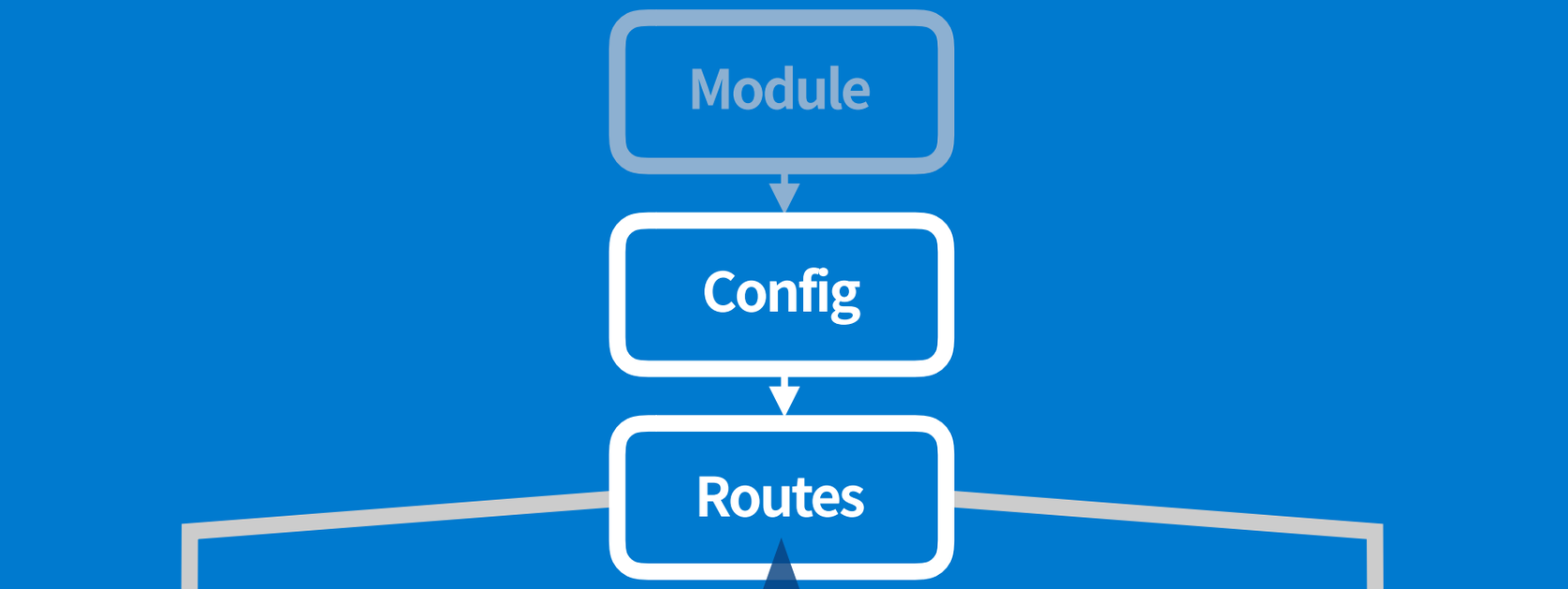
```
var myApp = angular.module('myApp', [‘ngRoute']);

myApp.config(function( $routeProvider, $locationProvider ) {

    // Routes 설정
    $routeProvider.when( ... );

    // Location 설정
    $locationProvider.html5Mode(true);
});
```

#!



Summary

Routes / Configuration



AngularJS는 라우트(Route)/히스토리(History) 기능을 제공.

라우트는 뷰(View) + 컨트로러(Controller)

ngRoute 모듈에 의존 (Dependency Module)

$routeProvider 인자는 angular.module.config()
내부 함수에 동적으로 자동 삽입.

$routeParams 인자는 컨트롤러에 등록된 함수에 인자로 호출 가능.

Animation

Animation

Animation in AngularJS

Animation in AngularJS

ngAnimate 모듈 의존(주입)

애니메이션 훅(Hook) 내장
사용자 정의 디렉티브(Directive)

다양한 시나리오 상황에서
애니메이션 실행

Directives & Animations

ngAnimate 모듈을 통해 애니메이션 설정이 가능한 디렉티브(Directive)는 다음과 같다.

내장 디렉티브 (Common Directive)	사용자 정의 디렉티브 (Custom Directive)
ngClass	$animate 서비스(Service)
ngShow / ngHide	
ngInclude	
ngRepeat	
ngSwitch	
ngView	

Directives & Supported Animations

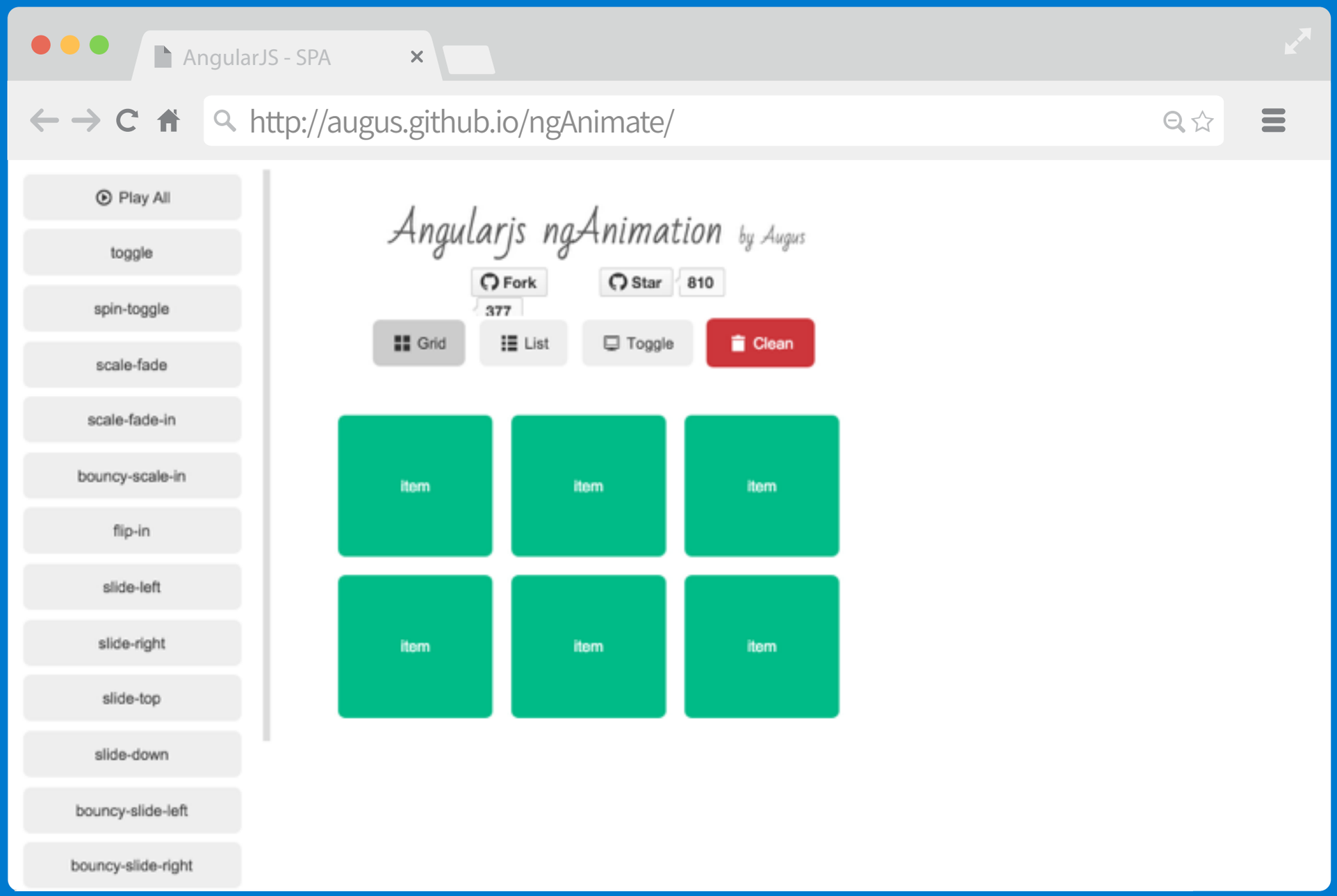
디렉티브에 따라 지원하는 애니메이션 유형은 다음과 같다.

내장 디렉티브 (Common Directive)	지원하는 애니메이션 종류 (Supported Animations)
ngClass	Add / Remove
ngShow / ngHide	Add / Remove
ngInclude	Enter / Leave
ngRepeat	Enter / Leave / Move
ngSwitch	Enter / Leave
ngView	Enter / Leave

DEMO &
Resources

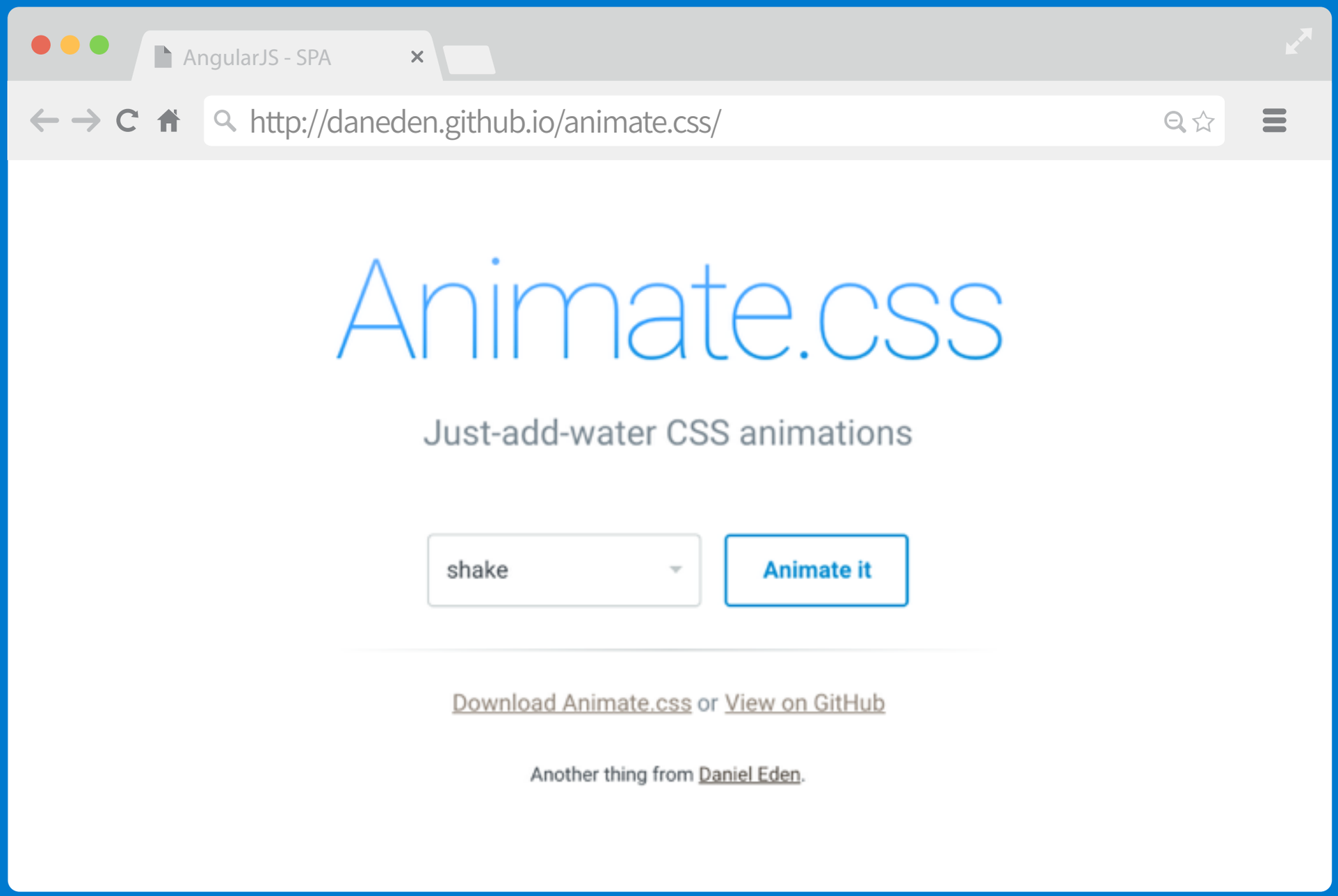
Animation Example

AngularJS 애니메이션 데모 - "AngularJS ngAnimation by Augus"



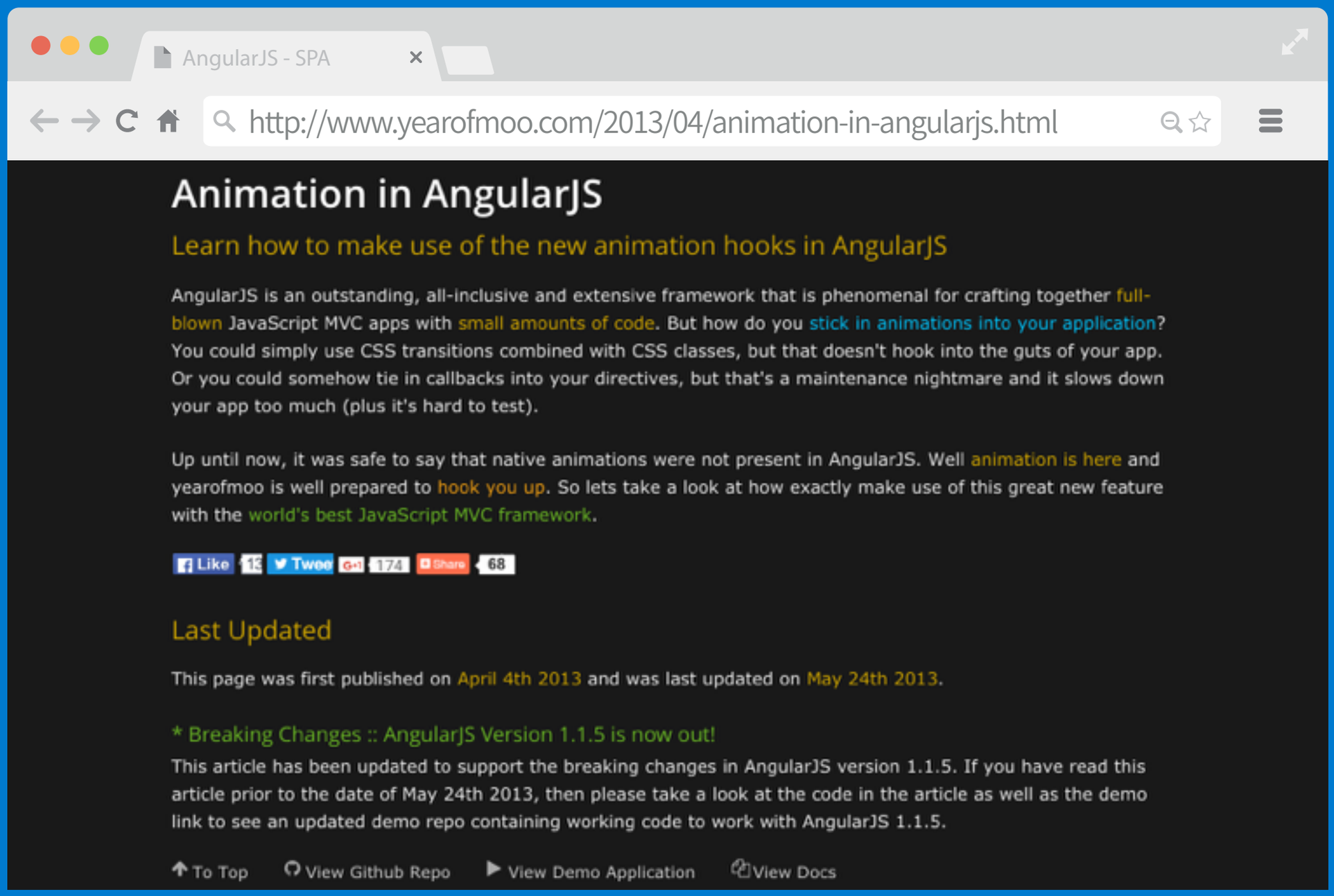
Animation Example

AngularJS 애니메이션 리소스 - "Animate.css"



Animation Example

AngularJS 애니메이션 튜토리얼 - "Animation in AngularJS"

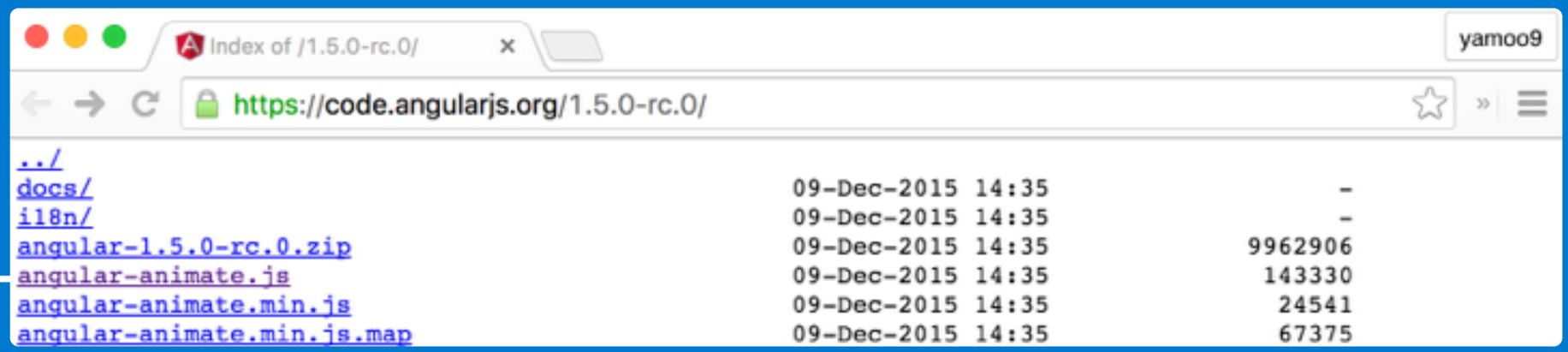


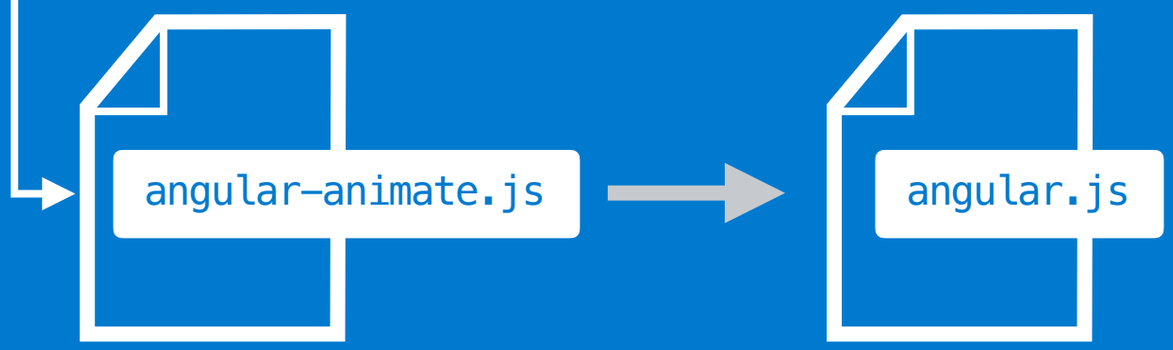
ngAnimate

angular Animation Module

ngAnimate Module

ngAnimate 모듈은 angular-animate.js 모듈 파일을 호출해야 사용 가능하다.





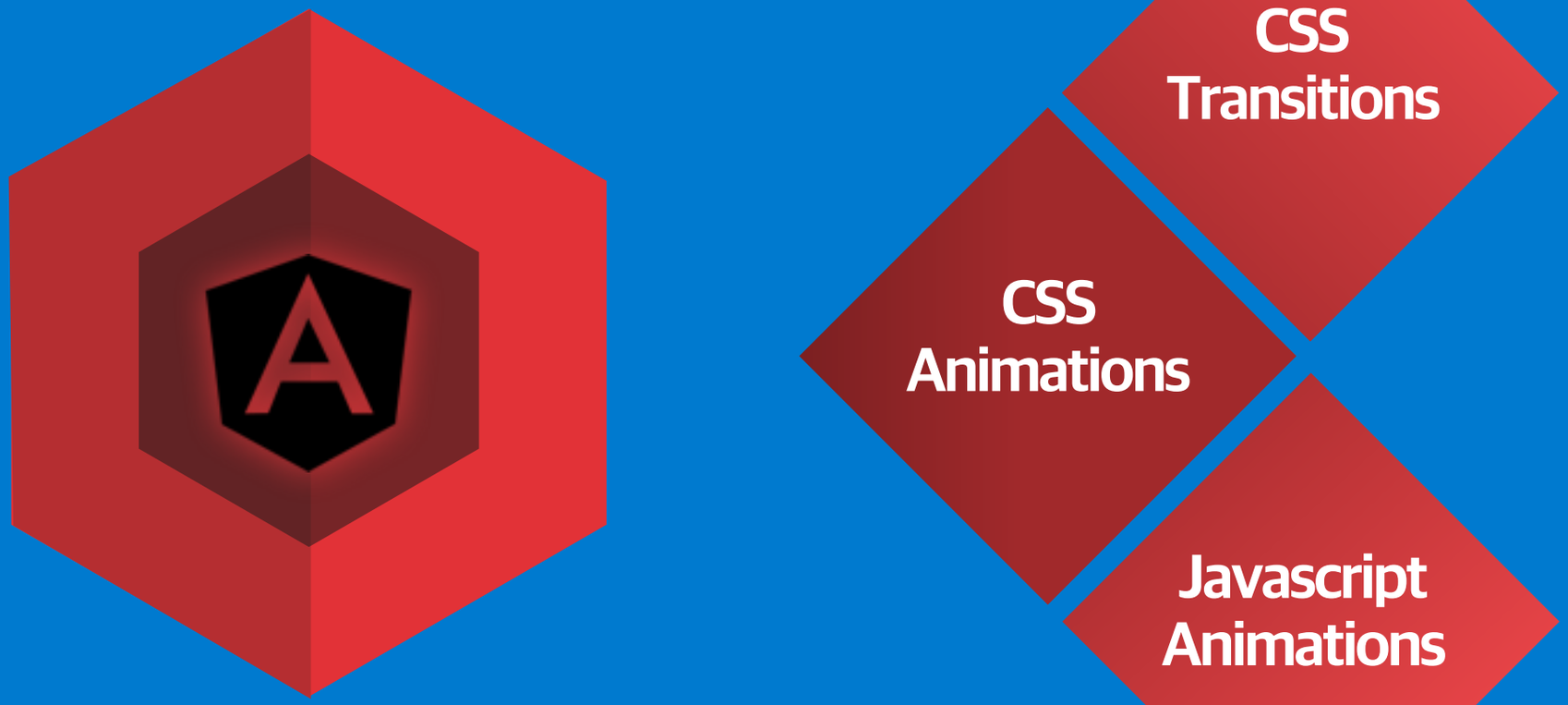
```
var myApp = angular.module('myApp', ['ngAnimate']);
```

Define Anim

Define Animation in CSS

Animation Techniques

AngularJS 애니메이션 테크닉은 CSS, Javascript를 이용하여 처리된다.



Defining Animations with CSS

CSS는 다양한 방법(Transition, Animation)으로
애니메이션을 정의하는 방법을 제공한다.

CSS Transitions

```
.trans {
    transition: color 0.345s cubic-bezier(0.550, 0.085, 0.680, 0.530);
}
```

CSS Animations

```
.anim {
    animation: slide-up 0.29s cubic-bezier(0.175, 0.885, 0.320, 1.275);
}
```

CSS Transition Example

CSS Transition은 “장면전환” 기술로 Before, After 사이에
진행되는 간단한 애니메이션 기술이다.

CSS Transitions

```
.fade-in.ng-enter,
.fade-in.ng-leave {
    transition: all 0.35s ease;
    position: relative;
    left: 0;
    top: 0;
    height: 400px;
}

.fade-in.ng-enter {
    z-index: 100;
    top: 0;
    opacity: 1;
}

.fade-in.ng-leave {
    z-index: 101;
    top: -400px;
    opacity: 0;
}
```

CSS Animation Example

CSS Animation은 단계별(0%~100%)로 키프레임을 설정하여
정밀한 애니메이션 설정이 가능한 기술이다.

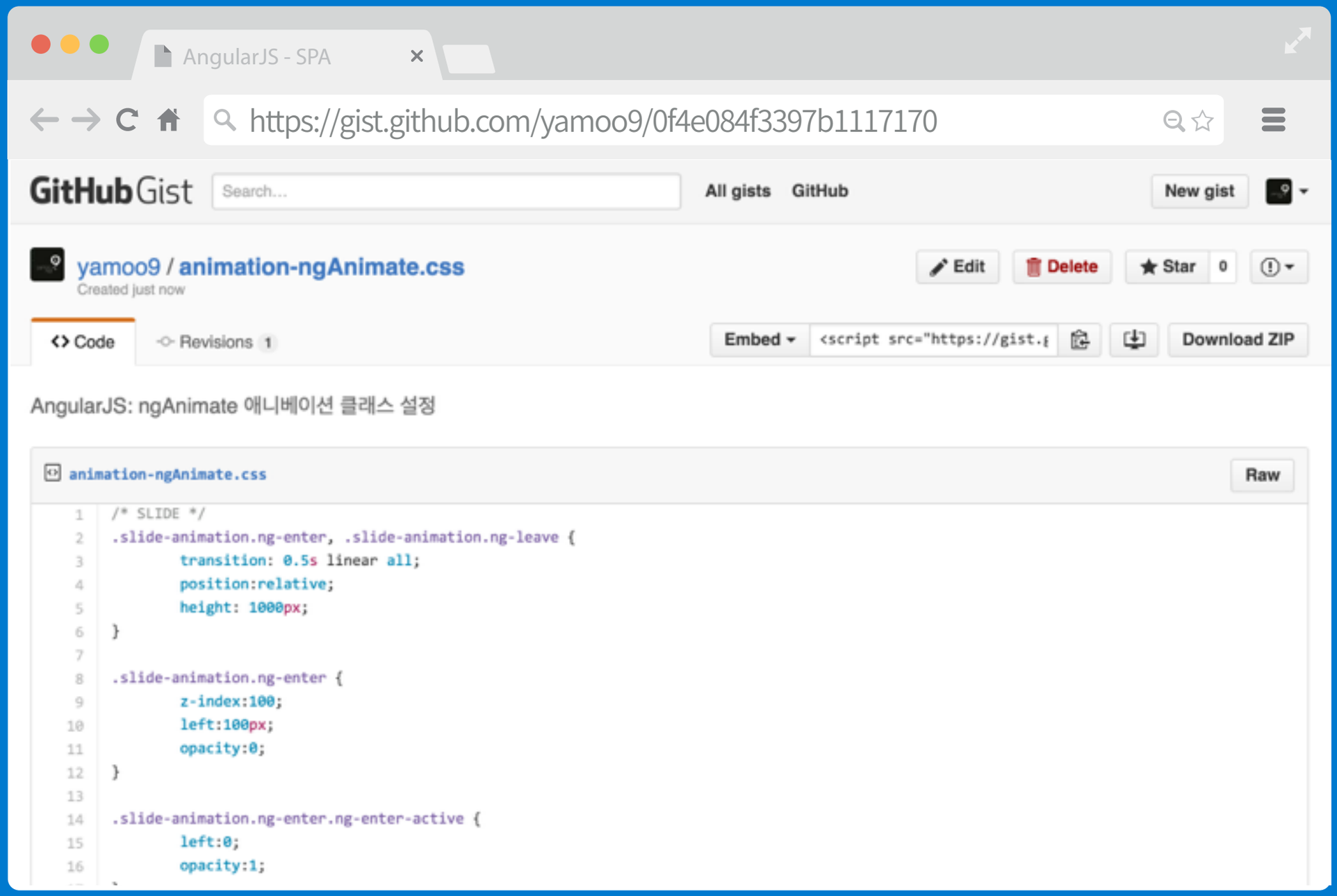
CSS Animations

```
.fade-in.ng-enter {
    animation: fade-in 0.45s ease;
}

@keyframes fade-in {
    from {
        z-index: 100;
        top: -400px;
        opacity: 0;
    }
    to {
        top: 0px;
        opacity: 1;
    }
}
```

```
.fade-in.ng-leave {
    animation: fade-out 0.45s ease;
}

@keyframes fade-out {
    from {
        z-index: 101;
        top: 0px;
        opacity: 1;
    }
    to {
        top: -400px;
        opacity: 0;
    }
}
```

Define Animation class style module

AngularJS 용: ngAnimate 애니메이션 class 스타일 모듈



class Attribute

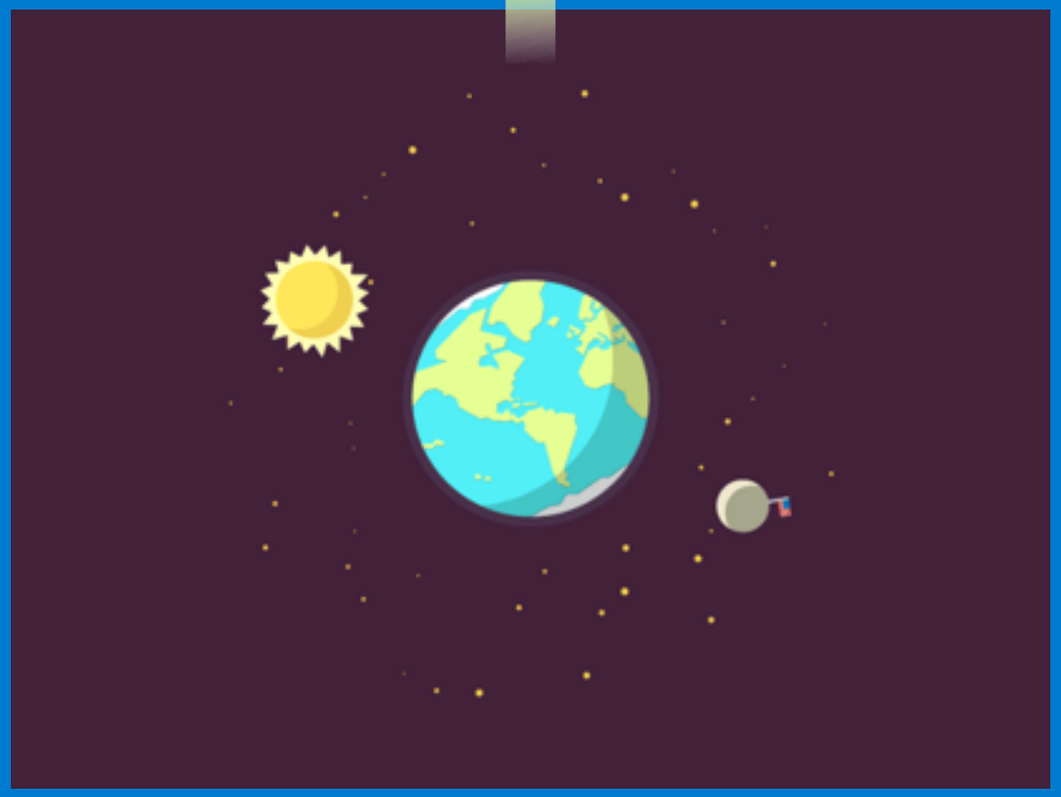
Referencing by Class Attribute

Animations Referenced by class Attribute

디렉티브에 설정된 class 속성 값을 참조하여 애니메이션이 설정된다.

```
<div data-ng-view class="fade-in"></div>

<div data-ng-show="list_view_enabled" class="toggle-anim"></div>

<ul data-ng-repeat="item in list" class="repeat-anim"></ul>
```

CSS 애니메이션 설정



Summary

AngularJS는 CSS, Javascript 애니메이션 기술을 통해
애니메이션하는 기능을 지원 한다.

ngAnimate 모듈에 의존(주입) 한다.

ngView, ngRepeat, ngInclude 등 다양한 디렉티브가
애니메이션을 지원한다.

CSS Transition, Animation 기술을 사용해 사용자 정의
애니메이션을 등록/사용할 수 있다.

Advanced Directives

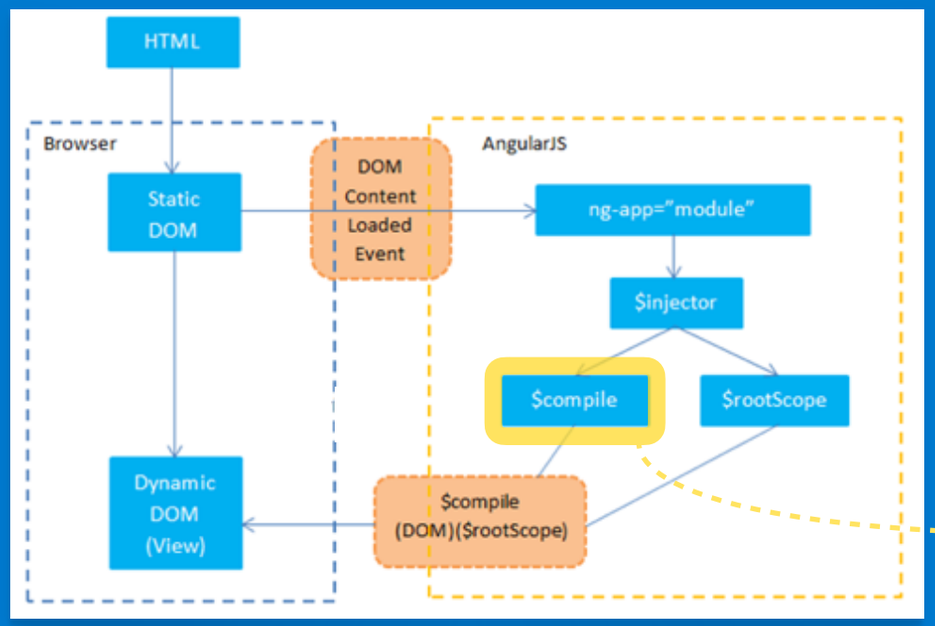
Directives

Built-in & 3rd Party

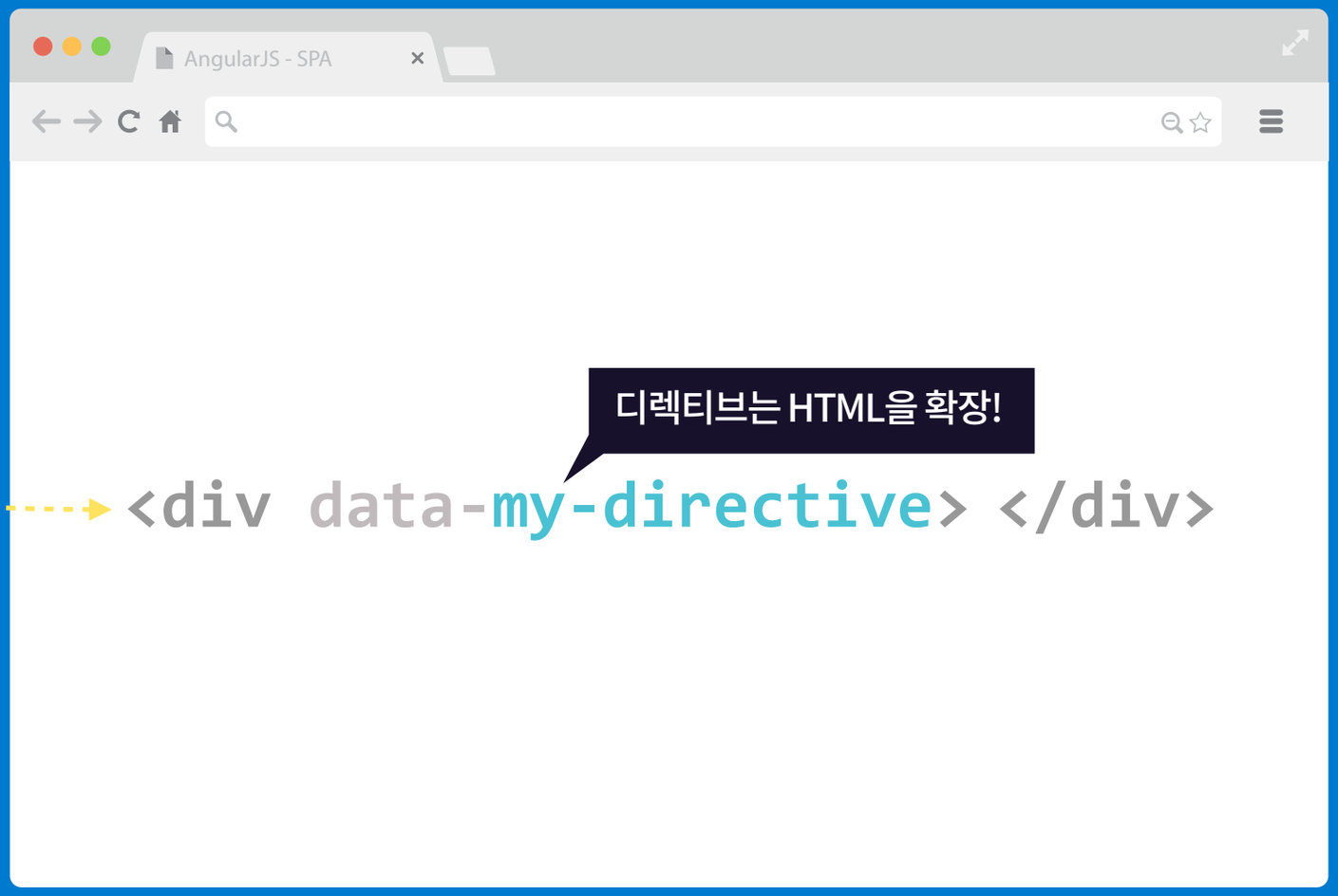
Role of Directives

디렉티브(Directives)란?

DOM 요소 마커(Marker)로, DOM 요소에 명시된 동작을 DOM Element에게 붙여주는 일을
AngularJS HTML 컴파일러($compiler)에게 수행(Attach Specified Behavior)하게 한다.



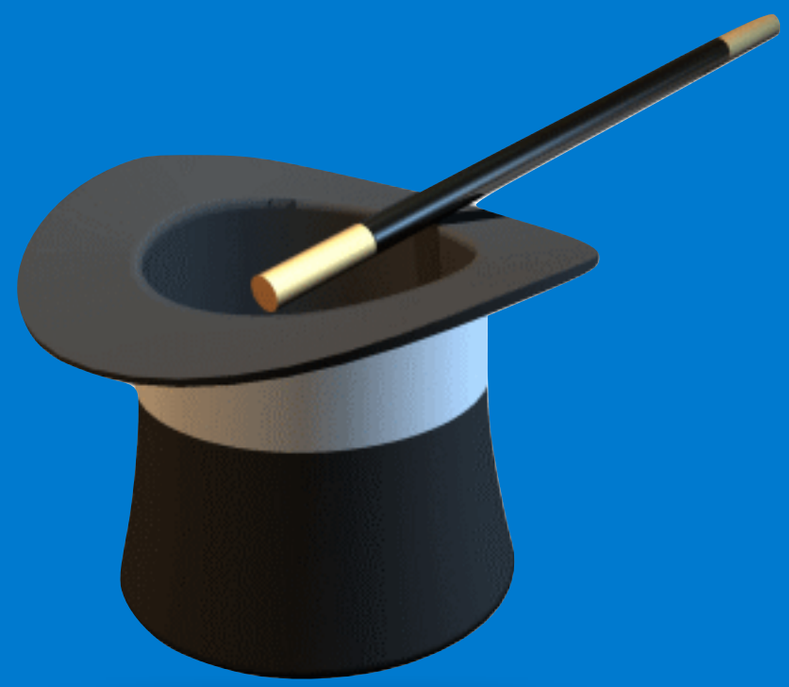
docs.angularjs.org/guide/bootstrap



Role of Directives

디렉티브(Directives)가 수행하는 일들

- 문서객체모델 조작 (DOM Manipulation)
- 이벤트 핸들링 (Events Handling)
- CSS 수정 (Modify CSS)
- 데이터를 본석하여 끝까지 반복 (DOM Iterate through Data)
- 데이터 유효성검사 (Validate Data)
- 데이터 바인딩 (Binding Data)



Role of Directives

AngularJS 내장 디렉티브(Native Directives)

Application

- ngApp
- ngController

Forms

- ngMaxlength
- ngMinlength
- ngPattern
- ngRequired
- ngSubmit
- ...

Data Binding

- ngInit
- ngModel
- ngBind
- ngHref
- ngSrc
- ngStyle
- ...

DOM

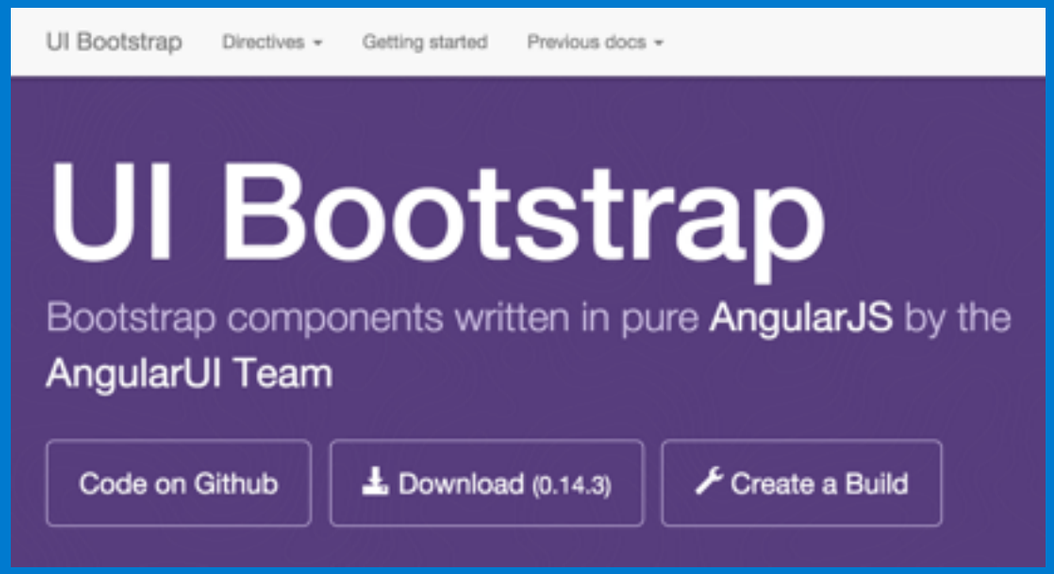
- ngCloak
- ngRepeat
- ngView
- ngIf
- ngSwitch
- ngShow
- ngHide
- ngDisabled
- ...

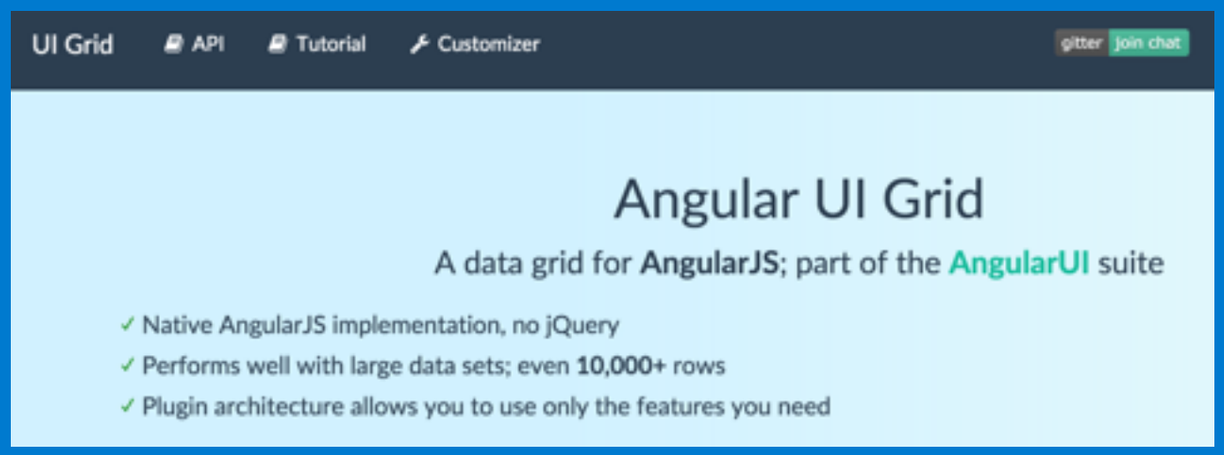
Behavior

- ngClick
- ngMouse*
- ngKey*
- ngChecked
- ngChange
- ngBlur
- ...

3rd Party Directives

3rd 파티 디렉티브(3rd Party Directives)





Custom Directive

User Define Directive

Directive Categories

디렉티브 카테고리

DOM-Driven-Directives

DOM 조작에 관한 디렉티브

Data-Driven-Directives

다른 디렉티브/컨트롤러에서 사용되는 데이터에 관한 디렉티브

Behavior-Driven-Directives

DOM 이벤트에 관한 디렉티브

Custom Directives

사용자 정의 디렉티브(User Define Directive)

모듈의 directive() 메소드를 사용하여 디렉티브를 정의한 후, 디렉티브 정의 객체(DDO)를 반환한다.
디렉티브로 정의된 이름은 카멜케이스 표기법을 사용하되, HTML에서는 하이픈(-) 표기법을 사용한다.

yamoo9_example.html

```
var app = angular.module('app');

// 사용자 정의 디렉티브(Custom Directive)
app.directive('customDirective', function() {
    return {
        'template': '<p>Angular JS Custom Directive</p>'
    };
});
```

DDO

```
<!-- 표준을 준수하지 않은 형태의 디렉티브 표기법 -->
<custom-directive></custom-directive>
<div custom-directive></div>

<!-- 표준을 준수한 형태의 디렉티브 표기법 -->
<div data-custom-directive></div>
```

Directive Types

디렉티브 유형

요소 (E)lement / 속성 (A)ttribute / 클래스 (C)lass / 주석 co(M)ment

yamoo9_example.html

yamoo9_example.html

```
1

  <!-- 요소 유형 디렉티브 (Element Type)
  ================================================== -->
  <custom-directive></custom-directive>
  <div custom-directive></div>
  <div data-custom-directive></div>

  <!-- 속성 유형 디렉티브 (Attribute Type)
  ================================================== -->
  <div custom-directive="expression"></div>
  <div data-custom-directive="expression"></div>

  <!-- CSS 클래스 유형 디렉티브 (CSS Class Type)
  ================================================== -->
  <div class="custom-directive: expression;"></div>

  <!-- 주석 유형 디렉티브 (Comment Type)
  ================================================== -->
  <!-- custom-directive: expression -->
```

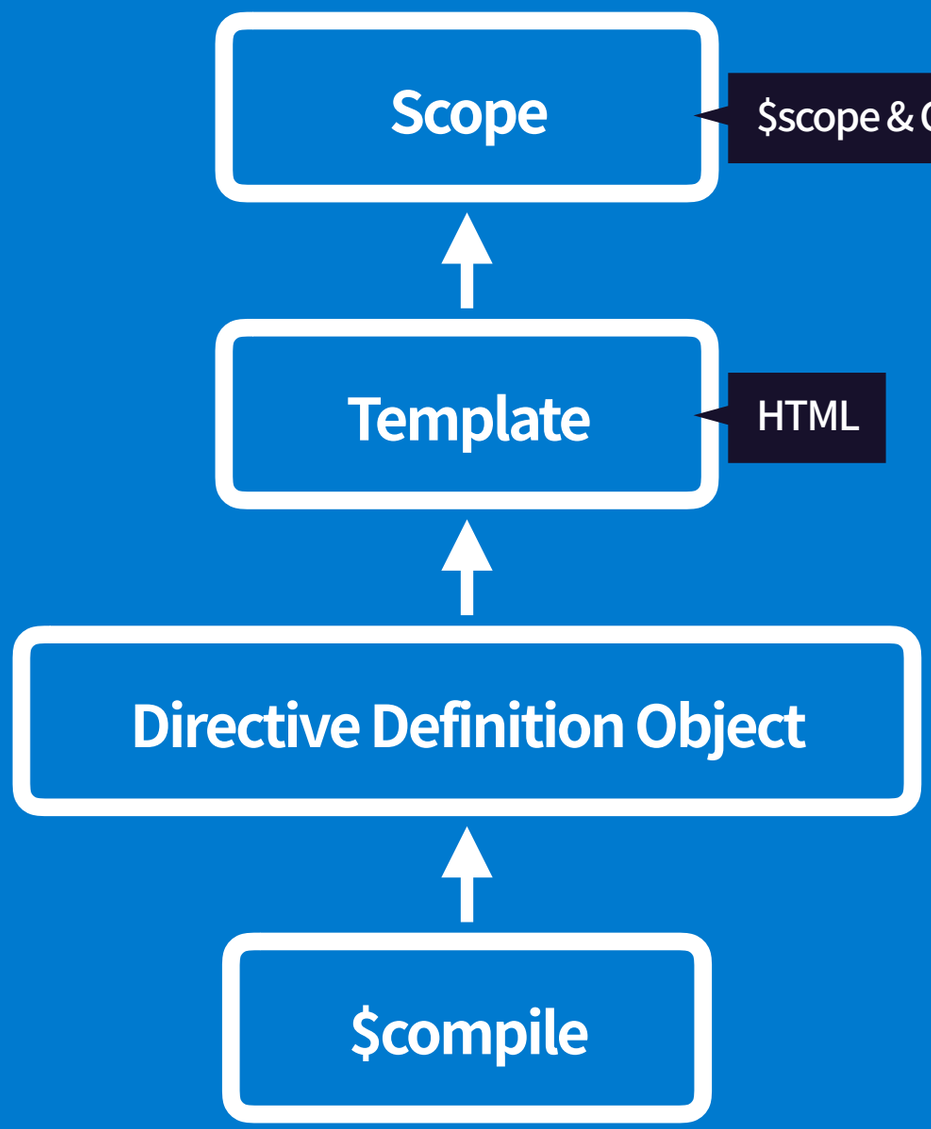
Template — Scope

템플릿 — 스코프 관계

Directive Building Blocks

디렉티브 생성/처리 과정

$compile > DDO > Template > Scope



$compile

$compile 제공자(Provider)

HTML 문자열 또는 DOM 안의 템플릿을 해석하고
스코프와 템플릿을 함께 연결하는 템플릿 함수를 생성한다.

https://docs.angularjs.org/api/ng/service/$compile

$compile

- $compileProvider
  - service in module ng

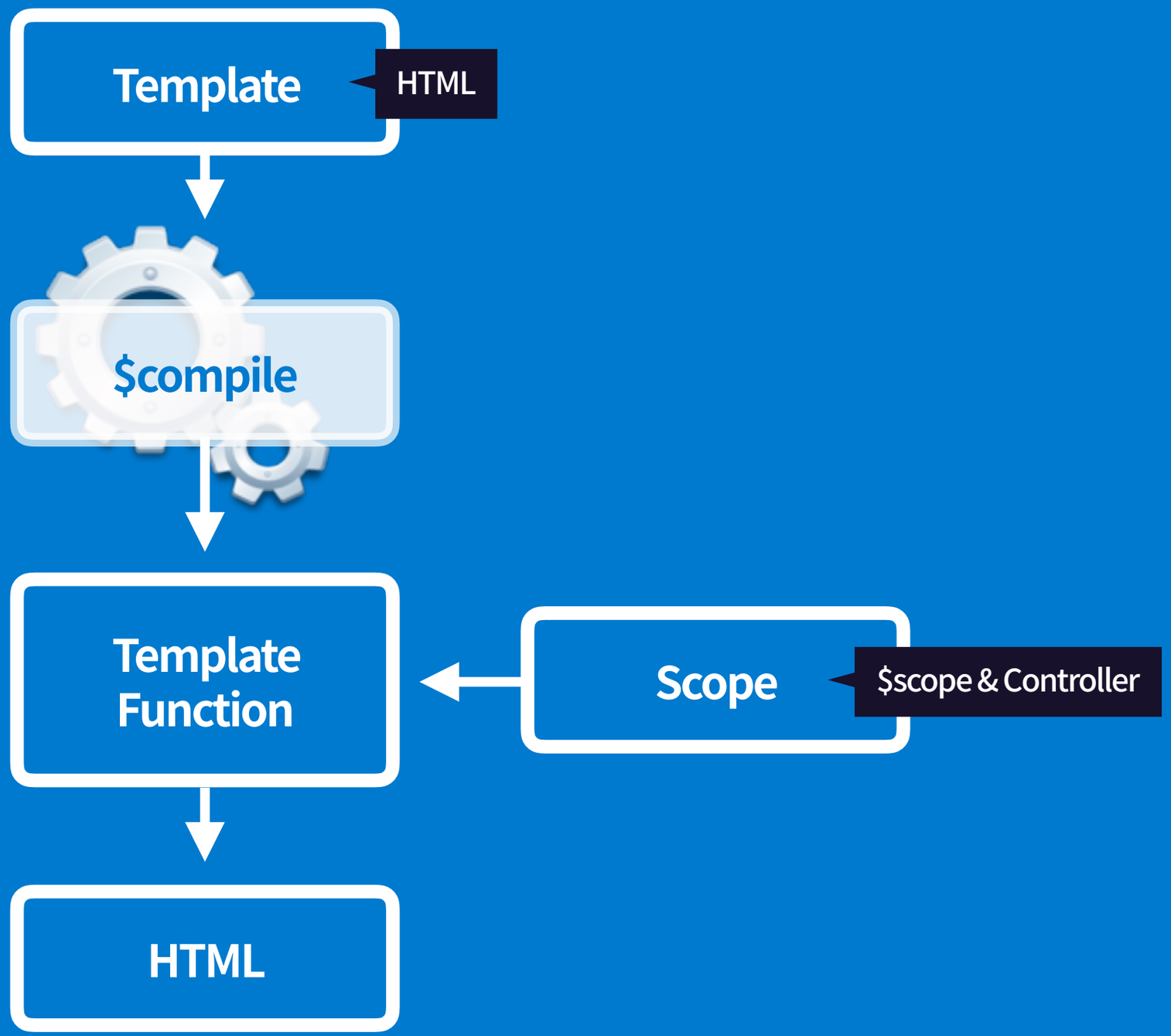
View Source Improve this Doc

Compiles an HTML string or DOM into a template and produces a template function, which can then be used to link `scope` and the template together.

The compilation is a process of walking the DOM tree and matching DOM elements to directives.

Note: This document is an in-depth reference of all directive options. For a gentle introduction to directives with examples of common use cases, see the directive guide.

$compile 프로세스(Process)



$compile — DDO

$compile 과 디렉티브 정의 객체 DDO(Directive Definition Object)

- $compile은 디렉티브 정의 객체(DDO)를 제공한다.
- DDO가 수행하는 기능
  - 디렉티브 템플릿 정의
  - DOM 조작코드 포함
  - 디렉티브 컨트롤러 정의
  - 디렉티브 스코프 제어
  - 디렉티브 사용법 정의
  - …

DDO Properties

디렉티브 정의 객체 속성

- restrict (제한)
- template
- templateUrl
- scope
- controller
- link

yamoo9_example.html

```
aunglar.module('app')
    .directive('customDirective', function() {
        return {
            'restrict':   'EA',
            'scope':      {},
            'template':   '<div> 사용자 정의 디렉티브 </div>'
            'controller': controller,
            'link':       function(scope, element, attrs) {}
        };
    });
```

요소(E), 속성(A) 형태로만 제한

Summary

커스텀 디렉티브

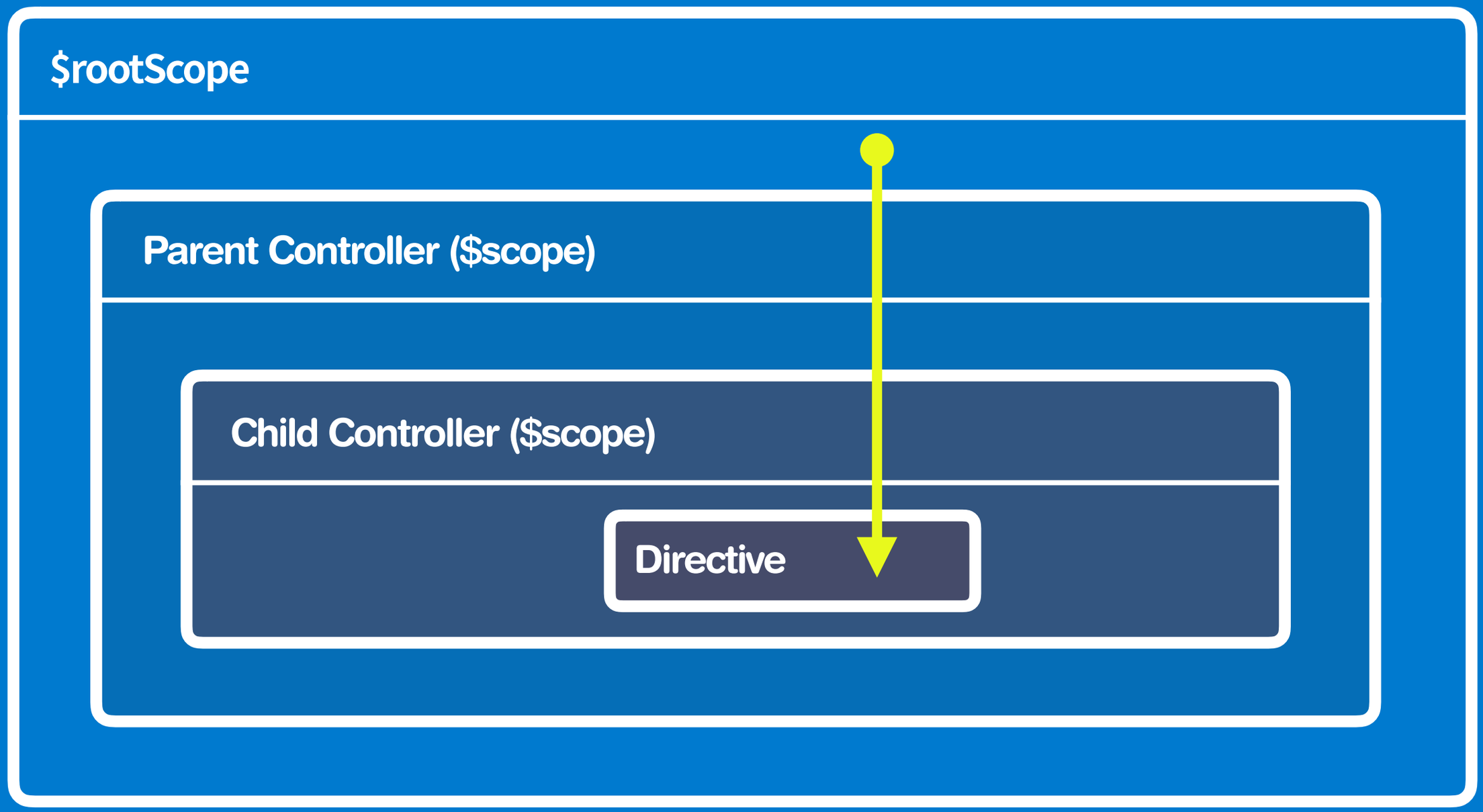
- 디렉티브는 AngularJS 애플리케이션에서 매우 중요한 역할을 수행한다.
  - HTML 렌더링
  - DOM 조작
  - 이벤트 핸들링
  - …
- 디렉티브는 다양한 방법을 사용하여 정의 가능하다.
- 디렉티브 정의 객체(DDO)는 커스텀 디렉티브의 빌딩 블록(제작 덩어리)이다.

Isolate Scope

Share Scope vs Isolate Scope

Scope Inheritance

스코프 상속 (Scope Inheritance)



Scope Type

공유되는 스코프(Shared Scope) vs 격리된 스코프(Isolate Scope)

공유되는 스코프 (Shared Scope)

Controller

```
$scope.customers = [];
```

공유

Directive

```
$scope.customers = [];
```

부모 컨트롤러(스코프)를 상속함.

VS

격리된 스코프 (Isolate Scope)

Controller

```
$scope.customers = [];
```

공유 X

Directive

```
$scope.isolated = true;
```

부모 컨트롤러(스코프)에서 격리됨.

Shared Scope

컨트롤러 스코프를 상속받는 디렉티브 (Shared Scope)

yamoo9_example.html

yamoo9_example.html

```
1

var app = angular.module('ScopeApp', []);

app.controller('Controller', ['$scope', function($scope){
    $scope.customer = [
        'name': 'yamoo9',
        'job': 'Instructor'
    ];
}]);

// 공유되는 스코프(상속됨)
app.directive('sharedScope', function() {
    return {
        'replace': true,
        'template': '<div> 닉네임: {{customer.name}}, 직업: {{customer.job}} </div>'
    };
});
```

스코프 상속

```
<div data-shared-scope> 닉네임: yamoo9, 직업: Instructor </div>
```

Isolate Scope

컨트롤러 스코프를 상속받는 디렉티브 (Shared Scope)

yamoo9_example.html

```
var app = angular.module('ScopeApp', []);

app.controller('Controller', ['$scope', function($scope){
    $scope.customer = [
        'name': 'yamoo9',
        'job': 'Instructor'
    ];
}]);

// 격리된 스코프(상속되지 않음)
app.directive('isolateScope', function() {
    return {
        'replace': true,
        'scope': {},
        'template': '<div> 닉네임: {{customer.name}}, 직업: {{customer.job}} </div>'
    };
});
```

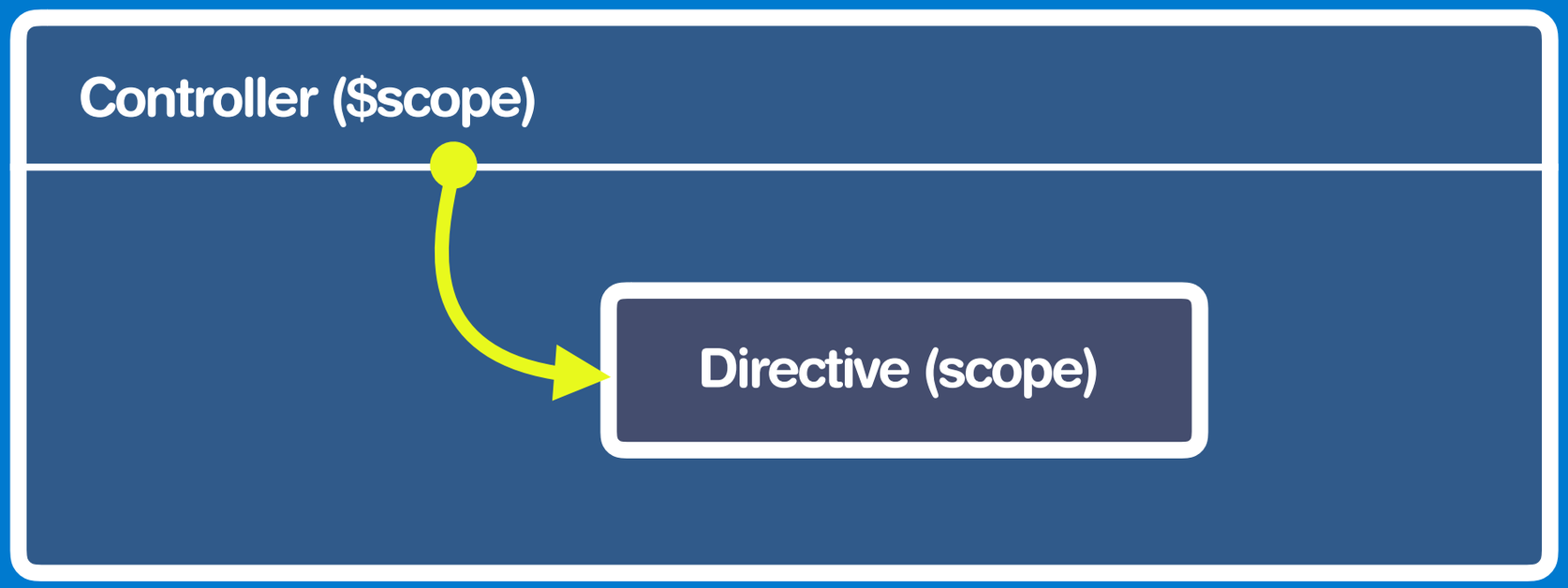
스코프 상속 X

```
<div data-isolate-scope> 닉네임: , 직업: </div>
```

값이 출력되지 않음.

Local Scope

Local Scope 속성 @
단방향 바인딩 (1 Way Binding)



Local Scope

단방향 데이터 바인딩(1 Way Data Binding) 생성

※ 문자열만 가능.

@ Local Scope 속성

Controller

```
$scope.name = 'yamoo9';
```

```
<div data-custom-directive name="{{name}}"></div>
```

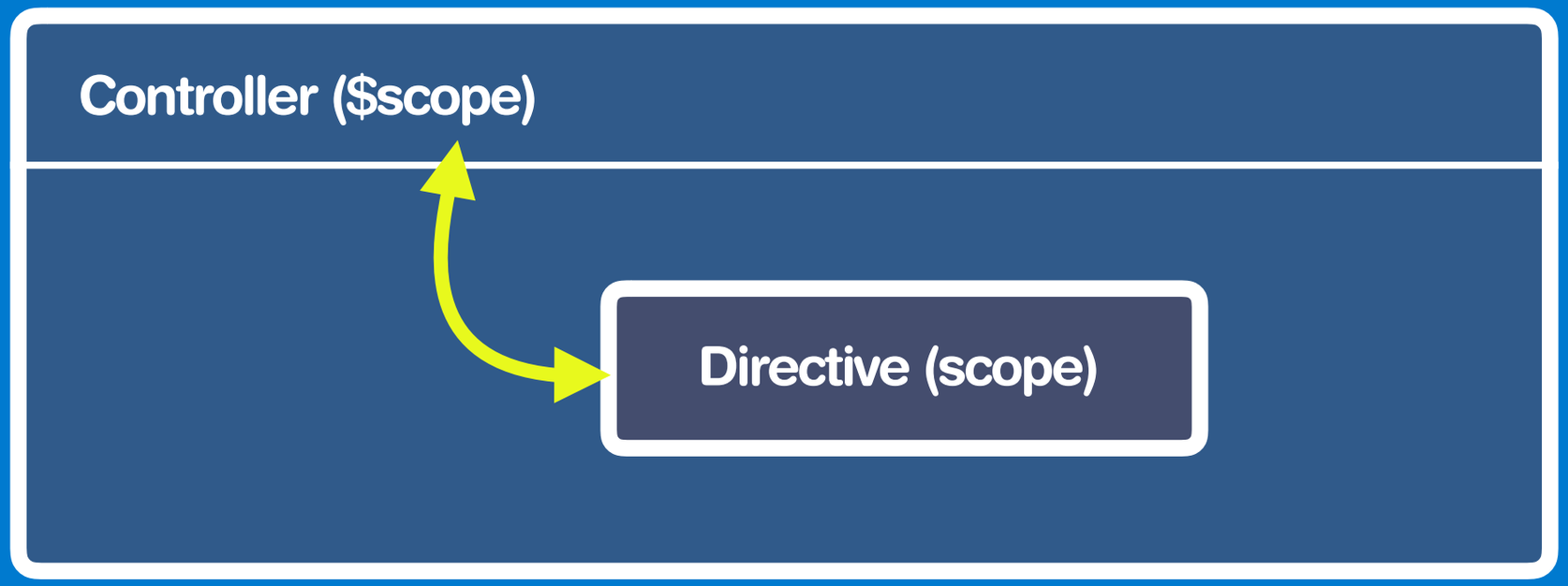
Directive

```
scope:{ name: '@' }
```

1 Way Binding Created.

Local Scope

= Local Scope 속성 =
양방향 바인딩 (2 Way Binding)



Local Scope

양방향 데이터 바인딩(2 Way Data Binding) 생성

= Local Scope 속성

Controller

```
$scope.person = {...};
```

```
<div data-custom-directive customer="person"></div>
```

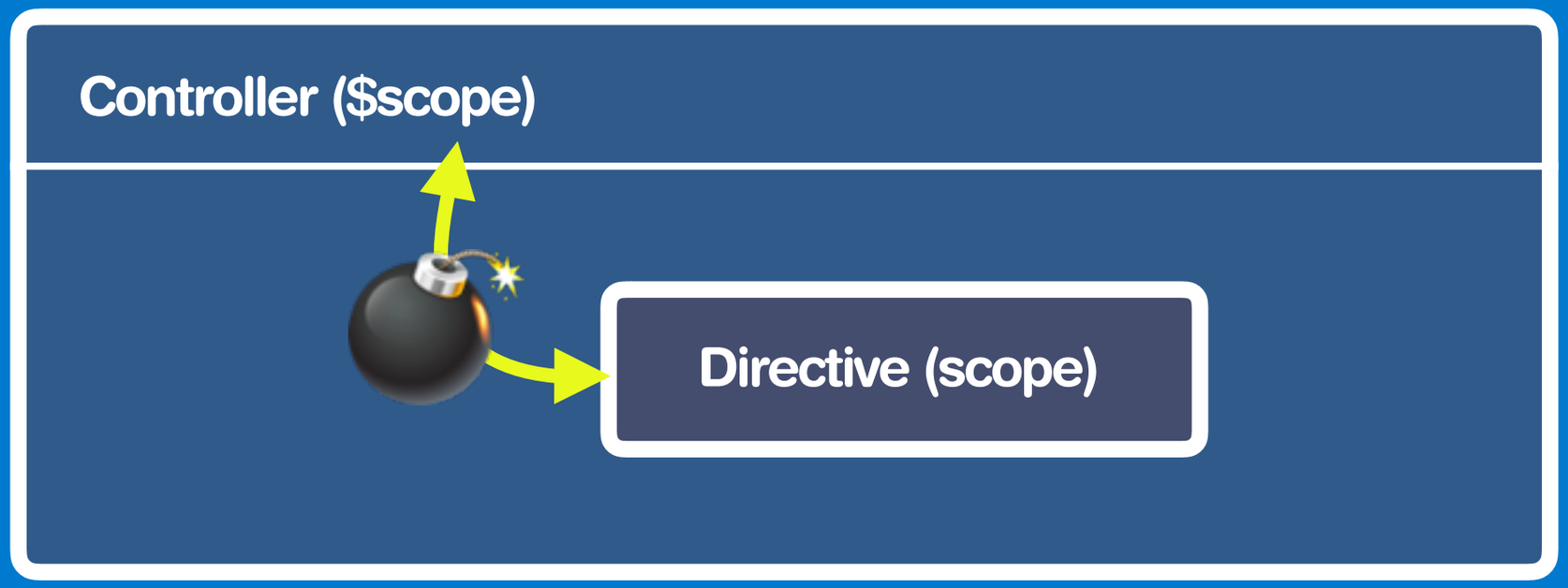
Directive

```
scope:{ customer: '=' }
```

2 Way Binding Created.

Local Scope

& Local Scope 속성 &
표현식/함수 실행(콜백)



Local Scope

표현식/함수 실행(콜백)

= Local Scope 속성

Controller

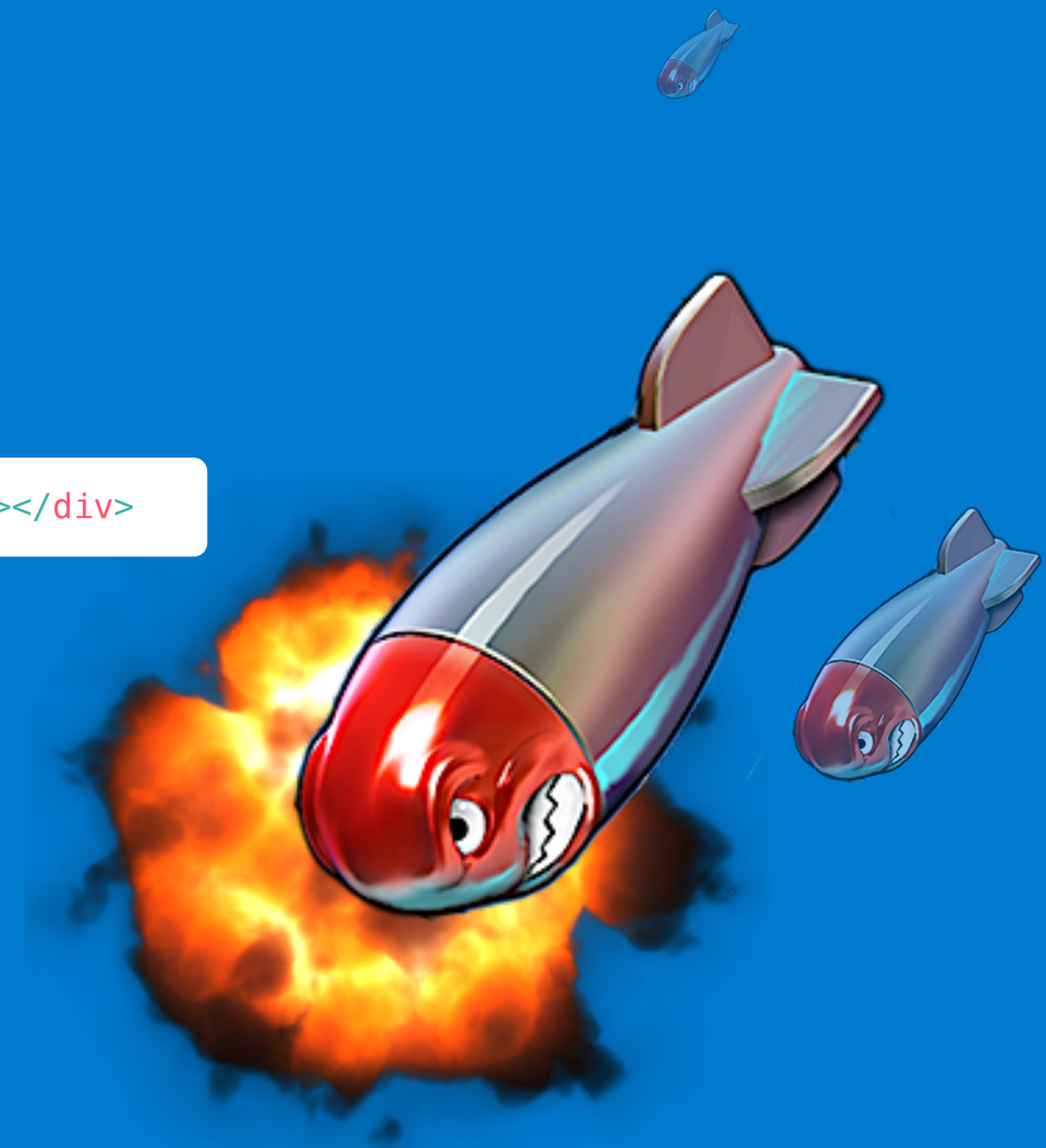
```
$scope.click = function(){};
```

```
<div data-custom-directive action="click()"></div>
```

Directive

```
scope:{ action: '&' }
```

click() 외부 함수 실행!



Summary

디렉티브: 격리된 스코프 (Isolate Scope)

- @는 로컬 스코프 속성과 DOM 속성 값을 데이터 바인딩(단방향) 함. 결과는 항상 문자.

```
scope: {name: '@'} | <div data-custom-directive name="{{name}}"></div>
```

- =는 로컬 스코프 속성과 부모 스코프 속성 사이에 양방향 데이터 바인딩을 생성한다.

```
scope: {customer: '='} | <div data-custom-directive costumer="person"></div>
```

- &는 부모 스코프 컨텍스트의 표현식/함수를 실행한다.

```
scope: {action: '&'} | <div data-custom-directive action="click()"></div>
```

link Function

AngularJS Directive

Directive Categories

디렉티브 카테고리

DOM-Driven-Directives

DOM 조작에 관한 모든 디렉티브

Data-Driven-Directives

다른 디렉티브/컨트롤러에서 사용
되는 데이터에 관한 모든 디렉티브

Behavior-Driven-Directives

DOM 이벤트에 관한 모든 디렉티브

DDO Properties

디렉티브 정의 객체 속성

restrict	template	templateUrl
scope	controller	**link**

yamoo9_example.html

```
aunglar.module('app')
    .directive('customDirective', function() {
        return {
            'restrict':   'EA',
            'scope':      {},
            'template':   '<div> {{ custom_var }} </div>'
            'controller': controller,
            'link':       function(scope, element, attrs, controller) {}
        };
    });
```

Link Function Parameters

디렉티브 정의 객체 - link 함수 매개변수
link 함수를 통해 DOM 요소를 제어할 수 있다.

yamoo9_example.html

yamoo9_example.html

```
angular.module('app')
.directive('linkFunctionDirective', function() {

    // link 함수
    var linkFunction = function(scope, element, attritubes, controller) {
        // element는 디렉티브가 적용된 요소를 가리킴.
        // attributes는 디렉티브가 적용된 요소에 설정된 속성을 가리킴.
    };

    // DDO 디렉티브 정의 객체
    return {
        'link': linkFunction
    };

});
```

스코프 (공유/격리)

요소 (jqLite 사용)

설정된 속성

연결된 컨트롤러

jqLite

Angular's jqLite

jqLite provides only the following jQuery methods:

- `addClass()`
- `after()`
- `append()`
- `attr()` - Does not support functions as parameters
- `bind()` - Does not support namespaces, selectors or eventData
- `children()` - Does not support selectors
- `clone()`
- `contents()`
- `css()` - Only retrieves inline-styles, does not call `getComputedStyle()`. As a setter, does not convert numbers to strings or append 'px', and also does not have automatic property prefixing.
- `data()`
- `detach()`
- `empty()`
- `eq()`
- `find()` - Limited to lookups by tag name
- `hasClass()`
- `html()`
- `next()` - Does not support selectors
- `on()` - Does not support namespaces, selectors or eventData
- `off()` - Does not support namespaces, selectors or event object as parameter
- `one()` - Does not support namespaces or selectors
- `parent()` - Does not support selectors
- `prepend()`
- `prop()`
- `ready()`
- `remove()`
- `removeAttr()`
- `removeClass()`
- `removeData()`
- `replaceWith()`
- `text()`
- `toggleClass()`
- `triggerHandler()` - Passes a dummy event object to handlers.
- `unbind()` - Does not support namespaces or event object as parameter
- `val()`
- `wrap()`

함수를 매개변수로 사용할 수 없다.

네임스페이스를 지원하지 않는다.

선택자를 지원하지 않는다.

인라인 스타일 속성만 가져온다. 숫자 값을 문자열 또는 'px' 단위가 붙은 문자열로 변경하지 않는다. 접두사가 붙는 속성을 자동 적용하지 않는다.

요소 이름으로만 찾을 수 있다.

선택자를 지원하지 않는다.

이벤트 네임스페이스, 선택자,
이벤트 객체의 속성을 지원하지 않는다.

선택자를 지원하지 않는다.

핸들러에 더미 이벤트 객체를 통과시킨다.

이벤트 네임스페이스,
이벤트 객체의 속성을 지원하지 않는다.

jqLite or jQuery

jQuery와 angularJS를 함께 사용할 경우, jqLite가 아닌 jQuery를 사용한다.

그럴 경우 angular.element는 jQuery로 설정. 즉, element는 jQuery 인스턴스 객체이며 angular.element()는 jQuery() 팩토리 함수와 같다.

참고 URL: https://github.com/angular/angular.js/blob/master/src/Angular.js#L1717