AngularJS 입문 - Conceptual Overview

개요

AngularJS에는 몇 가지 개념이 존재하고 처음 응용 프로그램을 만들기 전에 그것을 이해하는 것이 좋습니다. 이 섹션에서는 간단한 예를 사용하여 신속하게 AngularJS의 중요한 부분에 대해 언급하고 있습니다. 그러나 모든 것을 설명 할 수는 없습니다. 더 자세히 알고 싶으시면, 튜토리얼 을 확인하십시오.

개념	설명
Template	HTML과 그에 추가 된 AngularJS 자신의 MarkUp입니다.
Directives	사용자 정의 속성과 요소에 HTML을 확장 한 것입니다.
Model	UI에서 사용자에게 표시되는 데이터.
Scope	지시어와 수식에서 사용할 수 있는 모델이 포함 된 Context.
Expressions	범위에서 접근하는 변수 또는 함수입니다.
Compiler	템플릿 분석 지시어와 식을 인스턴스화.
Filter	사용자에게 표시되는 식의 값을 포맷합니다.
View	사용자에게 표시 할 것 (DOM)입니다.
Data Binding	Model과 View 사이의 데이터 동기화
Controller	View를 지원하는 비즈니스 로직.
Dependency Injection	객체 / 함수의 작성 및 연결.
Injector	의존성 주입 컨테이너.
Module	injector를 구성합니다.
Service	View 독립적 인 재사용 가능한 비즈니스 로직입니다.

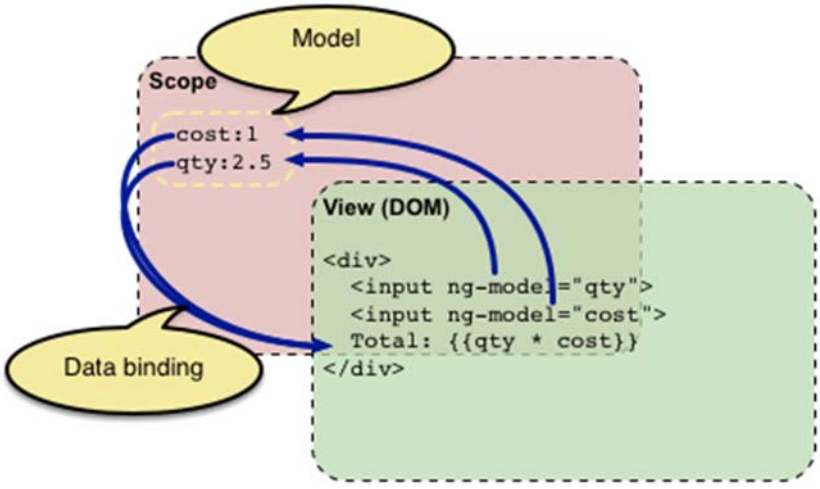
첫 샘플: 데이터 바인딩 (A first example: Data binding)

다음은 청구서 비용을 계산하는 양식 구축의 예입니다. 입력 필드에 수량과 비용을 입력하고 청구서의 금액을 곱해 봅시다.

```
<div ng-app ng-init="qty=1; cost=2">
  <b>Invoice:</b>
  <div>
    Quantity: <input type="number" min="0" ng-model="qty">
  </div>
  <div>
     Costs: <input type="number" min="0" ng-model="cost">
  </div>
  <div>
    <b>Total:</b> {{qty * cost | currency}}
  </div>
</div>
```

위는 새로운 마크 업을 포함한 일반 HTML처럼 보입니다. AngularJS에서 이러한 파일을 "템플릿 (template) "라고 합니다. 응용 프로그램에서 AngularJS의 실행이 시작되면 템플릿에서 새로운 마크 업이 "컴파일러 (compiler)"라는 것에 의해 분석, 처리됩니다. 변형되어 로드 된 DOM을 "뷰 (view)"라고 합니다.

먼저 배우는 새로운 마크 업은 "지시어 (directive)"라는 것입니다. 지시어는 HTML의 특성/요소에 특별한 동작을 적용합니다. 위의 예제는 어플리케이션이 초기화 될 때 자동으로 지시어와 연결하는 ng-app 속성이 사용되고 있습니다. AngularJS는 또한 input 요소에 대해 지시어를 정의하여 이 요소의 동작을 확장합니다. (예: 자동으로 입력 된 텍스트의 검증이 가능하며, `required` 속성으로 평가하여 텍스트 미 입력을 방지합니다.) ng-model 지시어는 입력 필드 값을 변수로 저장 (변경) 하고 CSS 클래스를 추가하여 입력 필드에 검증 상태를 표시합니다. 위의 예제에서는 이 기능을 이용하여 입력 필드를 비워두면 테두리를 빨간색으로 표시하도록 되어 있습니다.



DOM에 액세스하기 위한 사용자 정의 지시어:
AngularJS는 응용 프로그램이 DOM에 접할 수 있는 유일한 장소는 directive 내부 입니다. DOM에 액세스는 테스트를 어렵게 할 수 있기 때문에 이것은 좋은 습관이라 할 수 있습니다. 만약 DOM에 직접 액세스 할 필요가 있는 경우, 이를 위해 사용자 정의 지시어를 작성 할 필요가 있습니다. 이는 지시어 가이드에서 참조 할 수 있습니다.
두 번째 새로운 마크 업은 이중 중괄호 `{{식 | 필터}}` 입니다. 컴파일러가 이 마크 업을 찾아, 이 태그의 값을 평가하고 대체합니다. 템플릿의 "식 (expression)"은 JavaScript와 같은 코드 조각으로 변수의 읽기, 쓰기가 가능합니다. 그러나 이러한 변수는 글로벌 공간의 변수가 아닌 것에 주의하십시오. AngularJS는 JavaScript 함수의 변수로 식에 액세스 할 수 있도록 "범위 (scope)"을 제공합니다. scope 변수에 저장된 값은 "모델 (model)"로 참조됩니다. (모델에 대해서는 후술합니다) 이 두 번째 새로운 마크 업은 위의 예제에서는 AngularJS에 대해 "입력란에서 데이터를 검색하고 그 값을 서로 곱하세요"라고 지시하고 있습니다.

또한 이 예제는 "필터 (filter)"도 포함되어 있습니다. 식의 값을 필터를 통해 사용자에 대한 표시 값으로 포맷합니다. 이 예제에서는 숫자를 currency 형식으로 금액 형태로 보이도록 출력합니다.

이 예제에서 중요한 것은 AngularJS가 실시간으로 데이터를 라이브 바인딩 하고 있는 것입니다. 입력 값이 변경 될 때마다, 식의 값은 자동으로 다시 계산되어 DOM이 그 값으로 업데이트됩니다. 이 개념이 "양방향 데이터 바인딩"입니다.

UI로직 추가: 컨트롤러 (Adding UI logic: Controllers)
샘플에 로직을 추가하여 다른 통화로 입력 및 청구에 대한 지불을 할 수 있도록 합시다.

index.html

```
<div ng-app="invoice1" ng-controller="InvoiceController as invoice">
  <b>Invoice:</b>
  <div>
    Quantity: <input type="number" min="0" ng-model="invoice.qty" required >
  </div>
  <div>
    Costs: <input type="number" min="0" ng-model="invoice.cost" required >
    <select ng-model="invoice.inCurr">
      <option ng-repeat="c in invoice.currencies">{{c}}</option>
    </select>
  </div>
  <div>
```

```
    <b>Total:</b>
    <span ng-repeat="c in invoice.currencies">
      {{invoice.total(c) | currency:c}}
    </span>
    <button class="btn" ng-click="invoice.pay()">Pay</button>
  </div>
</div>
```

invoice1.js

```
angular.module('invoice1', [])
  .controller('InvoiceController', function() {
    this.qty = 1;
    this.cost = 2;
    this.inCurr = 'EUR';
    this.currencies = ['USD', 'EUR', 'CNY'];
    this.usdToForeignRates = {
      USD: 1,
      EUR: 0.74,
      CNY: 6.09
    };

    this.total = function total(outCurr) {
      return this.convertCurrency(this.qty * this.cost, this.inCurr, outCurr);
    };
    this.convertCurrency = function convertCurrency(amount, inCurr, outCurr) {
      return amount * this.usdToForeignRates[outCurr] /
this.usdToForeignRates[inCurr];
    };
    this.pay = function pay() {
      window.alert("Thanks!");
    };
});
```

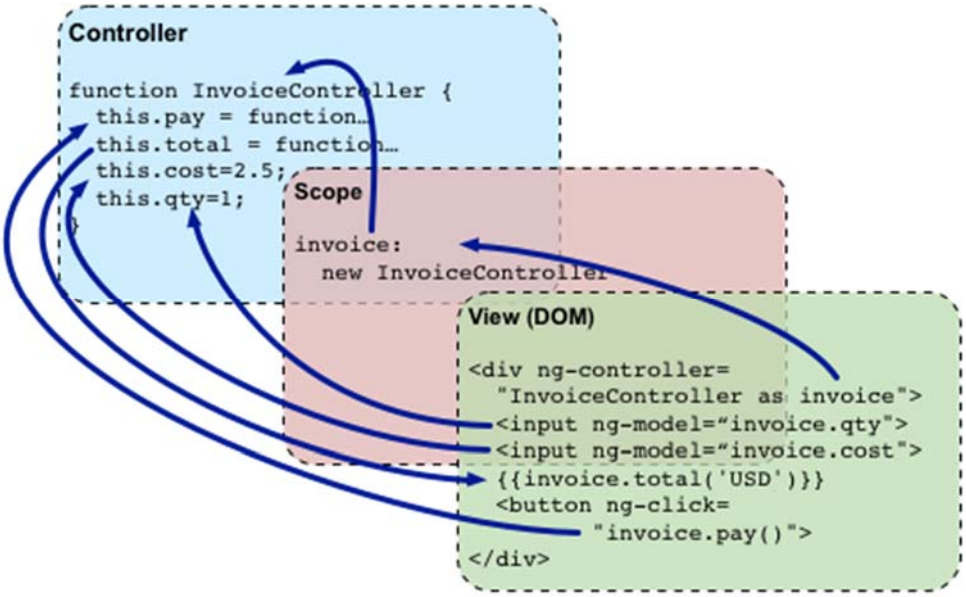
무엇이 바뀐 것일까요?
첫째, "컨트롤러 (controller)"라는 것을 포함한 새로운 JavaScript 파일이 추가되었습니다. 좀 더 정확하게 말하면, 이 파일은 실제 컨트롤러의 인스턴스를 만드는 생성자 함수를 포함합니다. 컨트롤러의 목적은 변수와 식의 함수, directive를 사용할 수 있도록 하는 것입니다.

컨트롤러 코드를 포함하는 새 파일을 추가, 더욱 HTML에 ng-controller directive가 추가 되어 있습니다. 이 지시어는 AngularJS에 InvoiceController이 요소와 모든 하위 요소에 대해 작업을 할당하라는 것을 알려줍니다. `InvoiceController as invoice` 문장은 Angular에 컨트롤러를 인스턴스화 하고 이를 현재 범위의 `invoice` 변수에 저장하도록 전하고 있습니다.

또한 페이지의 모든 식에 `invoice.` 접두사를 붙여서 컨트롤러의 인스턴스에 변수 읽기, 쓰기가 가능하도록 변경되었습니다. 선택 가능한 통화는 컨트롤러에 정의되어 ng-repeat 를 사용하여 템플릿에 추가되어 있습니다. 컨트롤러에는 total 함수가 포함되어 있으며, `{invoice.total(...)}}` 로 DOM에 그 함수의 결과를 연결하여 값을 넣을 수 있습니다.
또한, 이 샘플의 실시간 데이터 바인딩을 확인 해 봅시다. 함수의 결과가 변경 될 때마다 DOM이 자동으로 업데이트됩니다. 청구의 "결제"버튼은 ngClick 지시어를 사용하고 있습니다. 버튼이 클릭 될 때마다 지정된 표현식이 평가됩니다.

또한 새로운 JavaScript 파일에서 컨트롤러를 등록하는 새 모듈을 만들고 있습니다. 모듈내용은 다음 섹션에서 설명합니다.

아래 그림은 컨트롤러의 구현 후 scope와 View가 각각 서로 어떻게 작동하고 있는지를 설명 한 것입니다.



View의 비즈니스 로직으로부터 독립: Service (View independent business logic: Services)

현재 이 샘플의 모든 로직은 InvoiceController에 포함되어 있습니다. 응용 프로그램이 확장해가는 과정에서 컨트롤러에서 View를 "서비스 (service)"라고 불리는 다른 응용 프로그램의 일부로 재사용 가능한 것으로 독립, 분리시키는 것은 좋은 습관이라 할 수 있습니다. 나중에 컨트롤러를 변경하지 않고 WEB에서 환율 (예: Yahoo 금융 API)를 서비스로 읽도록 응용 프로그램을 변경하려고 합니다.

그럼 샘플을 리팩토링하여 `currency` 기능을 다른 파일에 서비스로 이동 해 봅시다.

finance2.js

```
angular.module('finance2', [])
  .factory('currencyConverter', function() {
    var currencies = ['USD', 'EUR', 'CNY'];
    var usdToForeignRates = {
      USD: 1,
      EUR: 0.74,
      CNY: 6.09
    };
    var convert = function (amount, inCurr, outCurr) {
      return amount * usdToForeignRates[outCurr] / usdToForeignRates[inCurr];
    };

    return {
      currencies: currencies,
      convert: convert
    };
  });
```

invoice2.js

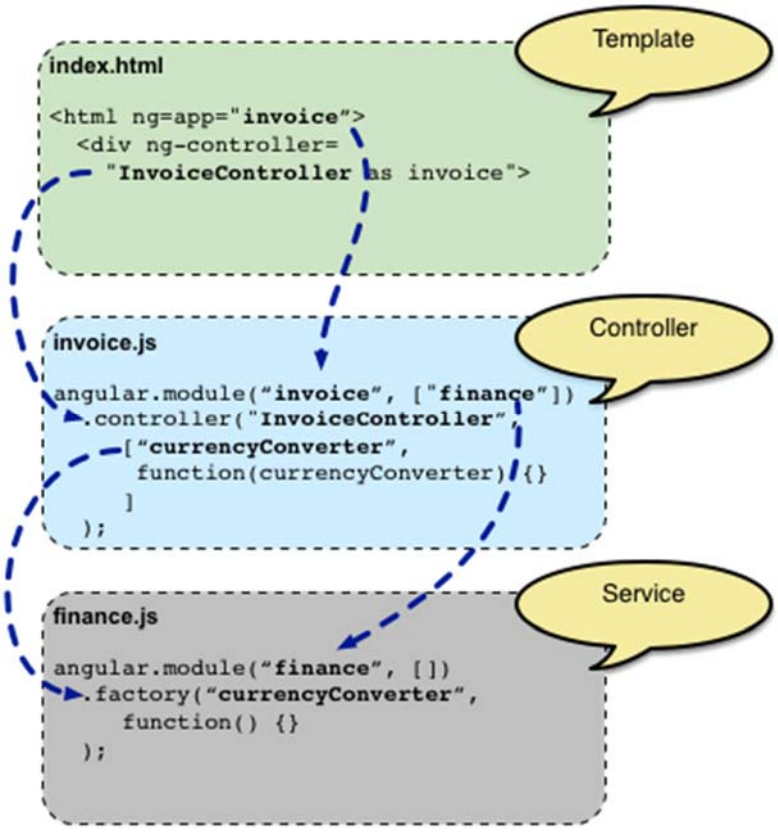
```
angular.module('invoice2', ['finance2'])
  .controller('InvoiceController', ['currencyConverter',
function(currencyConverter) {
    this.qty = 1;
    this.cost = 2;
    this.inCurr = 'EUR';
```

```
    this.currencies = currencyConverter.currencies;


    this.total = function total(outCurr) {
      return currencyConverter.convert(this.qty * this.cost, this.inCurr,
outCurr);
    };
    this.pay = function pay() {
      window.alert("Thanks!");
    };
  }]);
```

index.html

```
<div ng-app="invoice2" ng-controller="InvoiceController as invoice">
  <b>Invoice:</b>
  <div>
    Quantity: <input type="number" min="0" ng-model="invoice.qty" required >
  </div>
  <div>
    Costs: <input type="number" min="0" ng-model="invoice.cost" required >
    <select ng-model="invoice.inCurr">
      <option ng-repeat="c in invoice.currencies">{{c}}</option>
    </select>
  </div>
  <div>
    <b>Total:</b>
    <span ng-repeat="c in invoice.currencies">
      {{invoice.total(c) | currency:c}}
    </span>
    <button class="btn" ng-click="invoice.pay()">Pay</button>
  </div>
</div>
```



무엇이 바뀐 것일까요? convertCurrency 함수와 기존 currencies의 정의를 새로운 finance2.js 파일로 이동했지만, 그 분리 된 함수에 컨트롤러는 어떻게 접근하면 좋은 것일까요?

여기에서 "의존성 주입 (DI)"의 차례가 됩니다. 의존성 주입 (DI)는 객체와 함수의 생성과 그 의존성의 유지를 취급하는 소프트웨어 디자인 패턴입니다. AngularJS의 모든 개념 (지시어, 필터, 컨트롤러, 서비스, ...)은 의존성 주입을 사용하여 작성 및 구성이 이루어집니다. AngularJS는 DI 컨테이너를 "인젝터 (injector)"라고 합니다.

DI를 사용하려면 함께 작동해야 모든 것이 등록되는 장소가 필요합니다. Angular는 "모듈 (modules)"라는 것이 이 역할을 합니다. Angular가 시작되면 ng-app 지시어에 이름이 정의 된 모듈, 이 모듈에 의존하는 모듈 모든 것을 포함하는 구성이 사용됩니다.

위의 예제에서는 템플릿 `ng-app = "invoice"` 지시어를 포함합니다. 이것은 Angular에 invoice 모듈을 응용 프로그램의 주요 모듈로 사용하도록 지시하고 있습니다. 코드 `angular.module ('invoice', 'finance')` 부분은 invoice 모듈이 finance 모듈에 의존하도록 지정하고 있습니다. 이렇게 하면 Angular는 InvoiceController와 마찬가지로 currencyConverter 서비스도 사용합니다.
이제 Angular는 응용 프로그램의 구축에 필요한 모든 상태를 파악했습니다. 이전 섹션에서 컨트롤러는 팩토리 함수를 사용하여 생성되는 것을 설명했습니다. 서비스를 위한 팩토리의 정의에는 여러 가지 방법이 존재합니다. (가이드 서비스)을 확인하십시오. 위의 예제에서는 currencyConverter 함수를 서비스 팩토리로 반환하는 함수를 사용하고 있습니다.

첫 번째 의문으로 돌아갑니다만, InvoiceController은 어떻게 해서 currencyConverter 함수의 참조를 얻을 수 있을까요? Angular에서는 이를 단순히 생성자 함수의 인수로 정의하여 수행됩니다. 이렇게 하면 Injector가 차례로 객체를 생성하고 전에 만든 개체를 그들에게 의존하는 객체의 Factory에 전달할 수 있게 합니다. 샘플은 InvoiceController는 currencyConverter라고 명명 된 인수를 갖습니다. 따라서 Angular는 컨트롤러와 서비스의 종속성을 알 수 인수로 서비스 인스턴스의 컨트롤러를 호출합니다.

마지막으로, 이전 섹션의 예제와, 이 섹션의 샘플에서의 차이점입니다만, 여기에서는 module.Controller 함수에 일반 함수 대신 배열을 전달합니다. 배열의 첫 번째는 컨트롤러가 필요로 하는 종속 서비스의 이름을 포함하며, 배열의 마지막에는 컨트롤러의 생성자 함수를 포함합니다. Angular는 배열의 문법을 사용하여 의존성을 정의합니다. 이것은 DI 코드의 압축 된 후에도 동작하기 위함으로, 많은 경우 코드의 압축화에 의하여 컨트롤러의 생성자 함수의 인수가 a와 같은 짧은 형태로 변경하여 사용하기 위함입니다.

Accessing the backend

Yahoo 금융 API 환율을 사용하여, 이 샘플을 완성합시다. 다음 예제에서 이 방법을 확인하십시오.

invoice3.js

```
angular.module('invoice3', ['finance3'])
  .controller('InvoiceController', ['currencyConverter',
function(currencyConverter) {
    this.qty = 1;
    this.cost = 2;
    this.inCurr = 'EUR';
    this.currencies = currencyConverter.currencies;

    this.total = function total(outCurr) {
      return currencyConverter.convert(this.qty * this.cost, this.inCurr,
outCurr);
    };
    this.pay = function pay() {
      window.alert("Thanks!");
    };
  }]);
```

finance3.js

```
angular.module('finance3', [])
  .factory('currencyConverter', ['$http', function($http) {
    var YAHOO_FINANCE_URL_PATTERN =
          '//query.yahooapis.com/v1/public/yql?q=select * from '+
          'yahoo.finance.xchange where pair in ("PAIRS")&format=json&'+
          'env=store://datatables.org/alltableswithkeys&callback=JSON_CALLBACK';
```

```
  var currencies = ['USD', 'EUR', 'CNY'];
  var usdToForeignRates = {};
  var convert = function (amount, inCurr, outCurr) {
    return amount * usdToForeignRates[outCurr] / usdToForeignRates[inCurr];
  };

  var refresh = function() {
    var url = YAHOO_FINANCE_URL_PATTERN.
              replace('PAIRS', 'USD' + currencies.join('',"USD'));
    return $http.jsonp(url).success(function(data) {
      var newUsdToForeignRates = {};
      angular.forEach(data.query.results.rate, function(rate) {
        var currency = rate.id.substring(3,6);
        newUsdToForeignRates[currency] = window.parseFloat(rate.Rate);
      });
      usdToForeignRates = newUsdToForeignRates;
    });
  };

  refresh();

  return {
    currencies: currencies,
    convert: convert,
    refresh: refresh
  };
}]);
```

index.html

```
<div ng-app="invoice3" ng-controller="InvoiceController as invoice">
  <b>Invoice:</b>
  <div>
    Quantity: <input type="number" min="0" ng-model="invoice.qty" required >
  </div>
  <div>
    Costs: <input type="number" min="0" ng-model="invoice.cost" required >
    <select ng-model="invoice.inCurr">
      <option ng-repeat="c in invoice.currencies">{{c}}</option>
    </select>
  </div>
  <div>
    <b>Total:</b>
    <span ng-repeat="c in invoice.currencies">
      {{invoice.total(c) | currency:c}}
    </span>
    <button class="btn" ng-click="invoice.pay()">Pay</button>
  </div>
</div>
```

무엇이 바뀐 것일까요?

finance 모듈 currencyConverter 서비스가 백 엔드 (서버)에 액세스하기 위해 AngularJS에서 제공되는 내장 $http 서비스를 사용하고 있습니다. 이것은 XMLHttpRequest 와 JSONP 전송 주위를 래핑합니다. 자세한 내용은 API 문서를 확인하십시오.