thecodecampus</>

W11K

Advanced AngularJS

Überblick

Einleitung

<> Demo-Anwendung
Client & Server Architekturen </>

Advanced Features

<> Animationen
Routing mit Data-Prefetching </>

Code Qualität

<> AngularJS spezifisches Linting
API Dokumentation generieren
Modul-Systeme </>

Sicherheit

<> Login & Rechteverwaltung
Cross-Site-Request-Forgery
Cross-Site-Scripting </>

Performance analysieren und optimieren

<> Data-Binding & Digest-Lifecycle
Watch-Expressions & One-Time-Bindings
Verwendung von Filtern
Dos & Don'ts </>

Direktiven entwickeln

<> Aufbau der API
Wiederverwendbare Komponenten entwickeln
Transclude
Controller für Direktiven
Isolierte Scopes
Callbacks
Performance Optimieren: Compile-Funktion </>

Interessantes

<> Integration in Legacy Anwendungen
AngularJS 1.x aktualisieren
Ausblick Angular 2 </>

XSS

CROSS-SITE-SCRIPTING

Beispiel

```
1 var source = $('#insecure-input');
2 var text = source.val();
3 var target = $('#insecure-output');
4 target.append(text);
```

Ausprobieren ...

Cross-site-scripting

<> Spezielle Art der HTML Injection

<> HTML-Injection wird ausgenutzt um anderen Benutzer Code unterzuschieben

<> Benutzereingabe wird ohne Prüfung in HTML ausgegeben

<> Ermöglicht Ausführen von Code

<> Angriffe

- Daten auslesen und an Angreifen übermitteln (z.B. Session-Cookie)
- Code ruft URL auf um Aktion mit Rechten des Benutzers auszuführen (ähnlich wie XSRF)

Gegenmaßnahmen

- Benutzereingaben immer escapen
- Daten vom Server escapen
- Sanitizer Bibliothek verwenden
- Kontext beachten in dem Wert verwendet wird

<> Angular escapet alle Data-Bindings automatisch

<> $sanitize Service um sicheres HTML-Subset ausgeben zu können

<> $sce Service um beliebiges HTML aus vertrauenswürdiger Quelle ausgeben zu können

<> Ausführliches Beispiel

AngularJS - Beispiel

```
1 <input type="text" ng-model="text"/>
2 <div ng-bind="text"></div>
3 <div ng-bind-html="text"></div>
```

ng-bind-html

- `ng-bind` und `{{}}` escaped alle HTML Sonderzeichen
- `ng-bind-html` lässt ein sicheres Subset durch

- `ngSanitize`: zusätzliches Modul mit erweitertem Sanitizer für sicheres Subset
- Muss eigebunden werden für `ng-bind-html`, ansonsten Fehler auf Konsole

Strict Contextual Escaping

- `$sce` Service stellt Methoden zum wrappen bereit
- JS, URL, HTML

- `$sce.trustAsHtml` wrapt Text in Objekt
- Objekt markiert Text als sicheren Code
- `ng-bind-html` übernimmt ursprünglichen Text als Code in DOM

Direktiven entwickeln

Controller

<> Nimmt reguläre Controller-Funktion entgegen

<> Arbeitet wie gewohnt mit Dependency Injection

<> Zusätzliche DI-Werte: $element, $attrs, $transclude

<> Array-Notation bei Code-Minimierung

<> **Ausnahme:** Directiven-Controller darf DOM manipulieren

<> Unterschied zu Link-Funktion: API für Inter-Direktiven-Kommunikation

```
angular.module("...").directive('myShoppingCart', function(myService) {
  return {
    controller: function($scope, $element, myOtherService) { }
  };
});
```

Direktiven & Scopes

- Mit `scope: {}` wird ein *isolierter* Scope erzeugt
- Parent-Scope ist nur über $parent erreichbar, nicht per Prototyping
- verhindert Kollisionen mit Variablen anderer Direktiven am gleichen Element (ein Scope pro Direktive statt pro DOM-Element)
- Ideal für wiederverwendbare Komponenten
- Variablen und Werte aus dem Parent-Scope können übergeben werden

Compile

<> Selten benötigt, häufig falsch verwendet

<> Ziel: Template nur ein mal parsen und compilieren

<> Wiederverwendung des compilierten Elements

<> Link-Funktion als Rückgabe (damit Closure auf compiliertes Element möglich)

<> Eingeführt für Performance-Optimierungen in `ng-repeat`

```
<ul>
  <li ng-repeat="user in users">
    <div ng-bind="user.name"></div>
  </li>
</ul>
```

<> Link- und Compile-Funktion Aufrufe bei einfacher Verwendung

<> Link- und Compile-Funktion Aufrufe bei mehrfacher Verwendung (ng-repeat)

info@thecodecampus.de
@thecodecampus

www.w11k.de
www.thecodecampus.de

Copyright