StormCV

Pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>mipr</groupId>

<artifactId>mipr-full</artifactId>

<version>0.1</version>

</parent>

<artifactId>mipr-core</artifactId>

<version>0.1</version>

<build>

<resources>

<resource>

<directory>src/configuration</directory>

<includes>

<include>main.json</include>

</includes>

</resource>

</resources>

<plugins>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<version>2.4</version>

<configuration>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

</configuration>

<executions>

<execution>

<id>make-assembly</id>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>central</id>

<url>http://repo1.maven.org/maven2/</url>

</repository>

<repository>

<id>cloudera</id>

<url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>

</repository>

</repositories>

<dependencies>

<dependency>

<groupId>org.apache.hadoop</groupId>

<artifactId>hadoop-client</artifactId>

<version>2.0.0-cdh4.6.0</version>

</dependency>

<dependency>

<groupId>com.googlecode.json-simple</groupId>

<artifactId>json-simple</artifactId>

<version>1.1</version>

</dependency>

</dependencies>

</project>

StormCV-deploy

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>nl.tno</groupId>

<artifactId>stormcv</artifactId>

<version>0.7.3</version>

<name>StormCV</name>

<description>A Storm contribution to ease the development of computer vision topologies</description>

<url>https://github.com/sensorstorm/StormCV</url>

<packaging>jar</packaging>

<licenses>

<license>

<name>The Apache License, Version 2.0</name>

<url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>

</license>

</licenses>

<developers>

<developer>

<id>corneversloot</id>

<name>Corne Versloot</name>

<organization>Anchormen</organization>

<organizationUrl>https://www.anchormen.nl/</organizationUrl>

</developer>

<developer>

<name>John Schavemaker</name>

<organization>TNO</organization>

<organizationUrl>http://www.tno.nl</organizationUrl>

</developer>

<developer>

<name>Maarten Kruithof</name>

<organization>TNO</organization>

<organizationUrl>http://www.tno.nl</organizationUrl>

</developer>

</developers>

<scm>

<connection>scm:git:git@github.com:sensorstorm/StormCV.git</connection>

<developerConnection>scm:git:git@github.com:sensorstorm/StormCV.git</developerConnection>

<url>git@github.com:sensorstorm/StormCV.git</url>

</scm>

<repositories>

<repository>

<id>StormCV repo</id>

<url>http://github.com/sensorstorm/maven/raw/master/releases/</url>

</repository>

<repository>

<id>xuggle repo</id>

<url>http://xuggle.googlecode.com/svn/trunk/repo/share/java/</url>

</repository>

</repositories>

<dependencies>

<dependency>

<groupId>org.apache.storm</groupId>

<artifactId>storm-core</artifactId>

<version>0.9.4</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org</groupId>

<artifactId>opencv</artifactId>

<version>2.4.11</version>

</dependency>

<dependency>

<groupId>xuggle</groupId>

<artifactId>xuggle-xuggler</artifactId>

<version>5.4</version>

</dependency>

<dependency>

<groupId>com.amazonaws</groupId>

<artifactId>aws-java-sdk</artifactId>

<version>1.4.7</version> <!-- 1.7.2 is latest version but httpclient version conflicts with storm (4.1 <> 4.2) -->

</dependency>

<dependency>

<groupId>commons-net</groupId>

<artifactId>commons-net</artifactId>

<version>3.3</version>

</dependency>

<dependency>

<groupId>com.sun.jersey</groupId>

<artifactId>jersey-bundle</artifactId>

<version>1.18</version>

</dependency>

<dependency>

<groupId>com.google.guava</groupId>

<artifactId>guava</artifactId>

<version>18.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>1.7</source>

<target>1.7</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

StormCV-examples

Strom-docker

zookeeper:

image: wurstmeister/zookeeper

ports:

- "49181:2181"

- "22"

nimbus:

image: wurstmeister/storm-nimbus

ports:

- "49773:3773"

- "49772:3772"

- "49627:6627"

- "22"

links:

- zookeeper:zk

supervisor:

image: wurstmeister/storm-supervisor

ports:

- "8000"

- "22"

links:

- nimbus:nimbus

- zookeeper:zk

ui:

image: wurstmeister/storm-ui

ports:

- "49080:8080"

- "22"

links:

- nimbus:nimbus

- zookeeper:zk

storm

cluster.xml

<?xml version="1.0" encoding="UTF-8"?>

<configuration scan="true" scanPeriod="60 seconds">

<appender name="A1" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>/var/log/storm/${logfile.name}</file>

<rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">

<fileNamePattern>/var/log/storm/${logfile.name}.%i</fileNamePattern>

<minIndex>1</minIndex>

<maxIndex>9</maxIndex>

</rollingPolicy>

<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">

<maxFileSize>100MB</maxFileSize>

</triggeringPolicy>

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss} %c{1} [%p] %m%n</pattern>

</encoder>

</appender>

<appender name="ACCESS" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>/var/log/storm/access.log</file>

<rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">

<fileNamePattern>/var/log/storm/access.log.%i</fileNamePattern>

<minIndex>1</minIndex>

<maxIndex>9</maxIndex>

</rollingPolicy>

<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">

<maxFileSize>100MB</maxFileSize>

</triggeringPolicy>

<encoder>

<pattern>%d{yyyy-MM-dd HH:mm:ss} %c{1} [%p] %m%n</pattern>

</encoder>

</appender>

<appender name="METRICS" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>/var/log/storm/metrics.log</file>

<rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">

<fileNamePattern>metrics.log.%i</fileNamePattern>

<minIndex>1</minIndex>

<maxIndex>9</maxIndex>

</rollingPolicy>

<triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">

<maxFileSize>2MB</maxFileSize>

</triggeringPolicy>

<encoder>

<pattern>%d %-8r %m%n</pattern>

</encoder>

</appender>

<root level="INFO">

<appender-ref ref="A1"/>

</root>

<logger name="backtype.storm.messaging.netty">

<level value="WARN" />

<appender-ref ref="A1" />

</logger>

<logger name="backtype.storm">

<level value="INFO" />

<appender-ref ref="A1" />

</logger>

<logger name="backtype.storm.security.auth.authorizer" additivity="false">

<level value="INFO" />

<appender-ref ref="ACCESS" />

</logger>

<logger name="backtype.storm.metric.LoggingMetricsConsumer" additivity="false" >

<level value="INFO"/>

<appender-ref ref="METRICS"/>

</logger>

</configuration>

storm-nimbus

storm-supervisor

storm-ui

SpringBootKafkaStorm

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>info.smartkit</groupId>

<artifactId>spring-boot-kafka-storm</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>spring-boot-kafka-storm</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.2.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<start-class>info.smartkit.examples.primenumber.PrimeNumberTopology</start-class>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

<storm.version>1.0.2</storm.version>

<kafka.version>0.8.2.2</kafka.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-stream-kafka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-hateoas</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!-- Storm library -->

<dependency>

<groupId>org.apache.storm</groupId>

<artifactId>storm-core</artifactId>

<version>${storm.version}</version>

</dependency>

<!-- Storm-Kafka integration library -->

<dependency>

<groupId>org.apache.storm</groupId>

<artifactId>storm-kafka</artifactId>

<version>${storm.version}</version>

</dependency>

<!-- Kafka client libraries as Storm-Kafka integration library does not include these -->

<dependency>

<groupId>org.apache.kafka</groupId>

<artifactId>kafka\_2.11</artifactId>

<version>${kafka.version}</version>

<exclusions>

<!-- Excluded to avoid version issues between Kafka zookeeper api and

Storm-kafka zookeeper api -->

<exclusion>

<groupId>org.apache.zookeeper</groupId>

<artifactId>zookeeper</artifactId>

</exclusion>

<!-- Excluded to avoid Pre-emptive StackOverflowException due to version/implementation

issues between Kafka slf4j/log4j api and Storm slf4j/log4js api -->

<exclusion>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

</exclusion>

<exclusion>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

</exclusion>

</exclusions>

</dependency>

<!-- https://mvnrepository.com/artifact/org.alluxio/alluxio-core-client -->

<dependency>

<groupId>org.alluxio</groupId>

<artifactId>alluxio-core-client</artifactId>

<version>1.3.0</version>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>Camden.SR3</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>github-releases</id>

<url>http://oss.sonatype.org/content/repositories/github-releases/</url>

</repository>

<repository>

<id>clojars.org</id>

<url>http://clojars.org/repo</url>

</repository>

</repositories>

</project>

/src

/main/java/info/smartkit

package info.smartkit;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootKafkaStormApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootKafkaStormApplication.class, args);

}

}

examples/alluxio

package info.smartkit.examples.alluxio;

import alluxio.AlluxioURI;

import alluxio.Constants;

import alluxio.client.file.FileInStream;

import alluxio.client.file.FileOutStream;

import alluxio.client.file.FileSystem;

import alluxio.client.file.options.CreateFileOptions;

import alluxio.exception.AlluxioException;

import java.io.IOException;

/\*\*

\* Created by smartkit on 2016/12/22.

\* @see http://www.alluxio.org/docs/master/cn/File-System-API.html

\*/

public class AlluxioClientApplication {

public static void main(String[] args) throws IOException, AlluxioException {

//CRUD file:

createAlluxioFile();

renameAlluxioFile();

deleteAlluxioFile();

}

private static void renameAlluxioFile() throws IOException, AlluxioException {

//Read file:

FileSystem fileSystem = FileSystem.Factory.get();

AlluxioURI alluxioURI = new AlluxioURI("/alluxioFile");

//Open the file for reading and obtains a lock preventing delete.

FileInStream in = fileSystem.openFile(alluxioURI);

// Rename

// in.read();

AlluxioURI alluxioURIR = new AlluxioURI("/alluxioFileR");

if(!fileSystem.exists(alluxioURIR)) {

fileSystem.rename(alluxioURI, alluxioURIR);

}

//Close file relinquishing the lock

// in.close();

}

private static void createAlluxioFile() throws IOException, AlluxioException {

FileSystem fileSystem = FileSystem.Factory.get();

AlluxioURI alluxioURI = new AlluxioURI("/alluxioFile");

//Generate operation to set a custom blocksize of 128MB

CreateFileOptions options = CreateFileOptions.defaults().setBlockSizeBytes(128\* Constants.MB);

if(!fileSystem.exists(alluxioURI)) {

FileOutStream outStream = fileSystem.createFile(alluxioURI, options);

}

}

private static void deleteAlluxioFile() throws IOException, AlluxioException {

//Read file:

FileSystem fileSystem = FileSystem.Factory.get();

AlluxioURI alluxioURI = new AlluxioURI("/alluxioFile");

//Open the file for reading and obtains a lock preventing delete.

// FileSystem in = fileSystem.openFile(alluxioURI);

// Read data

if(fileSystem.exists(alluxioURI)) {

fileSystem.delete(alluxioURI);

//Close file relinquishing the lock

fileSystem.free(alluxioURI);

}

}

}

examples/logger

package info.smartkit.examples.logger;

import org.apache.log4j.Logger;

import org.apache.storm.topology.BasicOutputCollector;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseBasicBolt;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Tuple;

/\*\*

\* Created by smartkit on 2016/12/17.

\*/

public class LoggerBolt extends BaseBasicBolt {

private static final long serialVersionUID = 1L;

private static final Logger LOG = Logger.getLogger(LoggerBolt.class);

@Override

public void execute(Tuple tuple, BasicOutputCollector basicOutputCollector) {

LOG.info(tuple.getInteger(0));

}

@Override

public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

outputFieldsDeclarer.declare(new Fields("message"));

}

}

examples/ primenumber

package info.smartkit.examples.primenumber;

import org.apache.storm.spout.SpoutOutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichSpout;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Values;

import java.util.Map;

/\*\*

\* Created by smartkit on 2016/12/16.

\*/

public class NumberSpout extends BaseRichSpout {

private SpoutOutputCollector collector;

private static int currentNumber = 1;

@Override

public void open(Map conf, TopologyContext context, SpoutOutputCollector collector){

this.collector = collector;

}

@Override

public void nextTuple(){

//Emit next collector.

collector.emit(new Values(new Integer(currentNumber++)));

}

@Override

public void ack(Object id){

}

@Override

public void fail(Object id){

}

@Override

public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

outputFieldsDeclarer.declare(new Fields("number"));

}

}

package info.smartkit.examples.primenumber;

import org.apache.storm.task.OutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichBolt;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Tuple;

import java.util.Map;

/\*\*

\* Created by smartkit on 2016/12/17.

\*/

public class PrimeNumberBolt extends BaseRichBolt {

private OutputCollector collector;

@Override

public void prepare(Map map, TopologyContext topologyContext, OutputCollector outputCollector) {

this.collector = collector;

}

public void execute(Tuple tuple){

int number = tuple.getInteger(0);

if(isPrime(number)){

System.out.println( number );

}

collector.ack(tuple);

}

public void declareOutputFields(OutputFieldsDeclarer declarer){

declarer.declare(new Fields("number"));

}

public boolean isPrime(int n){

if( n == 1 || n == 2 || n == 3 )

{

return true;

}

// Is n an even number?

if( n % 2 == 0 )

{

return false;

}

//if not, then just check the odds

for( int i=3; i\*i<=n; i+=2 )

{

if( n % i == 0)

{

return false;

}

}

return true;

}

}

package info.smartkit.examples.primenumber;

import org.apache.storm.Config;

import org.apache.storm.LocalCluster;

import org.apache.storm.topology.TopologyBuilder;

import org.apache.storm.utils.Utils;

/\*\*

\* Created by smartkit on 2016/12/17.

\*/

public class PrimeNumberTopology {

public static void main(String[] args){

//

TopologyBuilder builder = new TopologyBuilder();

builder.setSpout("spout",new NumberSpout());

builder.setBolt("bolt",new PrimeNumberBolt()).shuffleGrouping("spout");

//

Config config = new Config();

LocalCluster localCluster = new LocalCluster();

localCluster.submitTopology("test",config,builder.createTopology());

Utils.sleep(10000);

localCluster.killTopology("test");

localCluster.shutdown();

}

}

examples/wordcount

package info.smartkit.examples.wordcount;

import org.apache.storm.spout.SpoutOutputCollector;

import org.apache.storm.task.OutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichBolt;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Tuple;

import org.apache.storm.tuple.Values;

import java.util.Map;

/\*\*

\* Created by smartkit on 2016/12/18.

\*/

public class SentenceSplitBolt extends BaseRichBolt {

private static final long serialVersionUID = -2107029392155190729L;

private OutputCollector collector;

@Override

public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

outputFieldsDeclarer.declare(new Fields("word"));

}

@Override

public void prepare(Map map, TopologyContext topologyContext, OutputCollector outputCollector) {

this.collector = outputCollector;

}

@Override

public void execute(Tuple tuple) {

String sentence = tuple.getStringByField("sentence");

String[] words = sentence.split(" ");

for (String word : words){

this.collector.emit(new Values(word));

}

}

}

package info.smartkit.examples.wordcount;

import org.apache.storm.spout.SpoutOutputCollector;

import org.apache.storm.task.OutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichBolt;

import org.apache.storm.topology.base.BaseRichSpout;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Tuple;

import org.apache.storm.tuple.Values;

import org.apache.storm.utils.Utils;

import java.util.Map;

/\*\*

\* Created by smartkit on 2016/12/18.

\*/

public class SentenceSpout extends BaseRichSpout{

private static final long serialVersionUID = 3444934973982660864L;

private SpoutOutputCollector collector;

private String[] sentences = { "my dog has fleas", "i like cold beverages",

"the dog ate my homework", "don't have a cow man",

"i don't think i like fleas" };

private int index = 1;

@Override

public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

outputFieldsDeclarer.declare(new Fields("sentence"));

}

@Override

public void open(Map map, TopologyContext topologyContext, SpoutOutputCollector spoutOutputCollector) {

this.collector = spoutOutputCollector;

}

@Override

public void nextTuple() {

this.collector.emit(new Values(sentences[index]));

index++;

if(index>=sentences.length){

index = 0;

}

Utils.sleep(100);

}

}

package info.smartkit.examples.wordcount;

import org.apache.storm.task.OutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichBolt;

import org.apache.storm.tuple.Fields;

import org.apache.storm.tuple.Tuple;

import org.apache.storm.tuple.Values;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* Created by smartkit on 2016/12/18.

\*/

public class WordCountBlot extends BaseRichBolt{

private OutputCollector collector;

private HashMap<String,Long> counts = null;

@Override

public void prepare(Map map, TopologyContext topologyContext, OutputCollector outputCollector) {

this.collector = outputCollector;

this.counts = new HashMap<String, Long>();

}

@Override

public void execute(Tuple tuple) {

String word = tuple.getStringByField("word");

Long count = this.counts.get(word);

if(count==null)

{

count = 0L;

}

count++;

this.counts.put(word,count);

this.collector.emit(new Values(word,counts));

}

@Override

public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

outputFieldsDeclarer.declare(new Fields("word","count"));

}

}

package info.smartkit.examples.wordcount;

import org.apache.storm.Config;

import org.apache.storm.LocalCluster;

import org.apache.storm.topology.TopologyBuilder;

import org.apache.storm.tuple.Fields;

import org.apache.storm.utils.Utils;

/\*\*

\* Created by smartkit on 2016/12/18.

\*/

public class WordCountTopology {

private static final String SPOUT\_ID = "sentence-spout";

private static final String BOLT\_ID\_SENTENCE\_SPLIT = "sentence-split-bolt";

private static final String BOLT\_ID\_WORD\_COUNT = "word-count-bolt";

private static final String BOLT\_ID\_COUNT\_REPORT = "count--report-bolt";

private static final String TOPOLOGY\_ID = "word-count-topology";

public static void main(String[] args){

TopologyBuilder topologyBuilder = new TopologyBuilder();

topologyBuilder.setSpout(SPOUT\_ID,new SentenceSpout());

topologyBuilder.setBolt(BOLT\_ID\_SENTENCE\_SPLIT,new SentenceSplitBolt()).shuffleGrouping(SPOUT\_ID);

topologyBuilder.setBolt(BOLT\_ID\_WORD\_COUNT,new WordCountBlot()).fieldsGrouping(BOLT\_ID\_SENTENCE\_SPLIT,new Fields("word"));

topologyBuilder.setBolt(BOLT\_ID\_COUNT\_REPORT,new WordsReportBolt()).globalGrouping(BOLT\_ID\_WORD\_COUNT);

Config config = new Config();

LocalCluster localCluster = new LocalCluster();

localCluster.submitTopology(TOPOLOGY\_ID,config,topologyBuilder.createTopology());

//

Utils.sleep(10000);

localCluster.killTopology(TOPOLOGY\_ID);

localCluster.shutdown();

}

}

package info.smartkit.examples.wordcount;

import org.apache.storm.task.OutputCollector;

import org.apache.storm.task.TopologyContext;

import org.apache.storm.topology.OutputFieldsDeclarer;

import org.apache.storm.topology.base.BaseRichBolt;

import org.apache.storm.tuple.Tuple;

import java.util.\*;

/\*\*

\* Created by smartkit on 2016/12/18.

\*/

public class WordsReportBolt extends BaseRichBolt{

private static final long serialVersionUID = 4921144902730095910L;

private HashMap<String,Long> counts = null;

@Override

public void prepare(Map map, TopologyContext topologyContext, OutputCollector outputCollector) {

this.counts = new HashMap<String, Long>();

}

@Override

public void execute(Tuple tuple) {

String word = tuple.getStringByField("word");

Long count = tuple.getLongByField("count");

this.counts.put(word,count);

}

@Override

public void declareOutputFields(OutputFieldsDeclarer outputFieldsDeclarer) {

}

public void clearup(){

System.out.println("-----print count-----");

List<String> keys = new ArrayList<String>();

keys.addAll(counts.keySet());

Collections.sort(keys);

for (String key: keys){

System.out.println("key: "+counts.get(key));

}

System.out.println("-----end of print count---");

}

}

FRONT-END-MyNewApp

// Ionic Starter App

// angular.module is a global place for creating, registering and retrieving Angular modules

// 'starter' is the name of this angular module example (also set in a <body> attribute in index.html)

// the 2nd parameter is an array of 'requires'

// 'starter.controllers' is found in controllers.js

angular.module('starter', ['ionic', 'starter.controllers','starter.services','nvd3','ngCordova'])

////$log configure

.config(['$logProvider', function($logProvider){

$logProvider.debugEnabled(true);

//TODO:https://github.com/ThomasBurleson/angularjs-logDecorator

}])

///ENV\_config

.constant('CONFIG\_ENV', {

'api\_endpoint': 'http://localhost:8080/octo-ninja/constellation/',

'api\_version': '5.16.3',

'stomp\_uri':'ws://www.xyz.com:61614/stomp',

'stomp\_protocol':'v11.stomp',

'debug':false

})

.run(function($ionicPlatform) {

$ionicPlatform.ready(function() {

// Hide the accessory bar by default (remove this to show the accessory bar above the keyboard

// for form inputs)

if(window.cordova && window.cordova.plugins.Keyboard) {

cordova.plugins.Keyboard.hideKeyboardAccessoryBar(true);

}

if(window.StatusBar) {

// org.apache.cordova.statusbar required

StatusBar.styleDefault();

}

});

})

.config(function($stateProvider, $urlRouterProvider) {

$stateProvider

.state('app', {

url: "/app",

abstract: true,

templateUrl: "templates/menu.html",

controller: 'MainCtrl'

})

.state('app.search', {

url: "/search",

views: {

'menuContent' :{

templateUrl: "templates/search.html"

}

}

})

.state('app.browse', {

url: "/browse",

views: {

'menuContent' :{

templateUrl: "templates/browse.html"

}

}

})

.state('app.dashboard', {

url: "/dashboard",

views: {

'menuContent' :{

templateUrl: "templates/dashboard.html",

controller: 'MainCtrl'

}

}

})

.state('app.single', {

url: "/dashboard/:personId",

views: {

'menuContent' :{

templateUrl: "templates/persondetail.html",

controller: 'PersonDetailCtrl'

}

}

});

// if none of the above states are matched, use this as the fallback

$urlRouterProvider.otherwise('/app/dashboard');

});

angular.module('starter.controllers', [])

.controller('MainCtrl', function($scope, $ionicModal, $timeout,$http,$log,CONFIG\_ENV,$ionicLoading) {

// Form data for the login modal

$scope.loginData = {};

// Create the login modal that we will use later

$ionicModal.fromTemplateUrl('templates/modal\_source.html', {

scope: $scope

}).then(function(modal) {

$scope.modal = modal;

});

// Triggered in the login modal to close it

$scope.closeLogin = function() {

$scope.modal.hide();

};

// Open the login modal

$scope.login = function() {

$scope.modal.show();

};

// Perform the login action when the user submits the login form

$scope.doLogin = function() {

console.log('Doing login', $scope.loginData);

// Simulate a login delay. Remove this and replace with your login

// code if using a login system

$timeout(function() {

$scope.closeLogin();

}, 1000);

};

// Function testing.

$scope.test = function() {

$ionicLoading.show();

//PersonService testing

$http.get(CONFIG\_ENV.api\_endpoint+'person/create').

success(function(data) {

$log.info("PersonService create result:",data);

$ionicLoading.hide();

})

.error(function(error){

$log.error("PersonService create error:",error);

$ionicLoading.hide();

});

};

})

.controller('SourceCtrl',function($scope,$log,$ionicLoading,$http,CONFIG\_ENV){

$ionicLoading.show();

$scope.sourceFiles=[{name:'文件1',id:000},{name:'文件2',id:001}];

$scope.sourceVideos=[{name:'视频1',id:000},{name:'视频2',id:001}];

$scope.sourceTargets=[

{name:'人物1',image:'icon\_a\_01.png'},

{name:'人物2',image:'icon\_a\_02.png'},

{name:'人物3',image:'icon\_a\_03.png'},

{name:'人物4',image:'icon\_a\_04.png'}

];

//Initialize function call

$scope.init = function(){

//SourceService get files

$http.get(CONFIG\_ENV.api\_endpoint+'source/files').

success(function(data) {

$log.info("SourceService get files result:",data);

$scope.sourceFiles = data;

$ionicLoading.hide();

})

.error(function(error){

$log.error("SourceService ge files error:",error);

$ionicLoading.hide();

});

//SourceService get videos

$http.get(CONFIG\_ENV.api\_endpoint+'source/videos').

success(function(data) {

$log.info("SourceService get videos result:",data);

$scope.sourceVideos = data;

$ionicLoading.hide();

})

.error(function(error){

$log.error("SourceService ge videos error:",error);

$ionicLoading.hide();

});

//SourceService get targets

$http.get(CONFIG\_ENV.api\_endpoint+'source/targets').

success(function(data) {

$log.info("SourceService get targets result:",data);

$scope.sourceTargets = data;

$ionicLoading.hide();

})

.error(function(error){

$log.error("SourceService ge targets error:",error);

$ionicLoading.hide();

});

}

})

.controller('PersonTabCtrl', function($scope,$log,$ionicLoading,$http,CONFIG\_ENV) {

$scope.personlist = [

//{ title: 'Reggae', id: 1 },

//{ title: 'Chill', id: 2 },

{ title: 'Dubstep', id: 3 }

];

//@example:http://krispo.github.io/angular-nvd3/#/

/\* Chart options \*/

$scope.testChartOptions = {

chart: {

type: 'discreteBarChart',

// height:450,

margin : {

top: 20,

right: 20,

bottom: 50,

left: 55

},

x: function(d){ return d.label; },

y: function(d){ return d.value; },

showValues: true,

valueFormat: function(d){

return d3.format(',.4f')(d);

},

transitionDuration: 500,

xAxis: {

axisLabel: 'X Axis'

},

yAxis: {

axisLabel: 'Y Axis',

axisLabelDistance: 30

}

}

};

/\* Chart data \*/

$scope.testChartData = [{

key: "Cumulative Return",

values: [

{ "label" : "A" , "value" : 29.765957771107 },

{ "label" : "B" , "value" : 0 },

{ "label" : "C" , "value" : 32.807804682612 },

{ "label" : "D" , "value" : 196.45946739256 },

{ "label" : "E" , "value" : 0.19434030906893 },

{ "label" : "F" , "value" : 98.079782601442 },

{ "label" : "G" , "value" : 13.925743130903 },

{ "label" : "H" , "value" : 5.1387322875705 }

]

}];

//@example:http://plnkr.co/edit/vtKWU0?p=preview

/\* Chart options \*/

$scope.personNamesChartOptions = {

chart: {

type: 'pieChart',

height: 400,

x: function(d){return d.key;},

y: function(d){return d.y;},

showLabels: true,

transitionDuration: 500,

labelThreshold: 0.01,

legend: {

margin: {

top: 5,

right: 35,

bottom: 5,

left: 0

}

}

}

};

/\* Chart data \*/

$scope.personNamesChartData = [{

key: "One",

y: 5

},

{

key: "Two",

y: 2

},

{

key: "Three",

y: 9

},

{

key: "Four",

y: 7

},

{

key: "Five",

y: 4

},

{

key: "Six",

y: 3

},

{

key: "Seven",

y: .5

}];

//@example:http://plnkr.co/edit/jOoJik?p=preview

/\* Chart options \*/

$scope.personTagsChartOptions = {

chart: {

type: 'pieChart',

height: 350,

donut: true,

x: function(d){return d.key;},

y: function(d){return d.y;},

showLabels: true,

pie: {

startAngle: function(d) { return d.startAngle/2 -Math.PI/2 },

endAngle: function(d) { return d.endAngle/2 -Math.PI/2 }

},

transitionDuration: 500,

legend: {

margin: {

top: 5,

right: 70,

bottom: 5,

left: 0

}

}

}

};

/\* Chart data \*/

$scope.personTagsChartData = [

{

key: "One",

y: 5

},

{

key: "Two",

y: 2

},

{

key: "Three",

y: 9

},

{

key: "Four",

y: 7

},

{

key: "Five",

y: 4

},

{

key: "Six",

y: 3

},

{

key: "Seven",

y: .5

}

];

//@example:http://plnkr.co/edit/U1wWbe?p=preview

/\* Chart options \*/

$scope.personAggregateChartOptions = {

chart: {

type: 'historicalBarChart',

height: 250,

margin : {

top: 20,

right: 20,

bottom: 60,

left: 50

},

x: function(d){return d[0];},

y: function(d){return d[1]/100000;},

showValues: true,

valueFormat: function(d){

return d3.format(',.1f')(d);

},

transitionDuration: 500,

xAxis: {

axisLabel: 'X Axis',

tickFormat: function(d) {

return d3.time.format('%x')(new Date(d))

},

rotateLabels: 50,

showMaxMin: false

},

yAxis: {

axisLabel: 'Y Axis',

axisLabelDistance: 35,

tickFormat: function(d){

return d3.format(',.1f')(d);

}

}

}

};

/\* Chart data \*/

$scope.personAggregateChartData = [{

"key" : "Quantity" ,

"bar": true,

"values" : [ [ 1136005200000 , 1271000.0] , [ 1138683600000 , 1271000.0] , [ 1141102800000 , 1271000.0] , [ 1143781200000 , 0] , [ 1146369600000 , 0] , [ 1149048000000 , 0] , [ 1151640000000 , 0] , [ 1154318400000 , 0] , [ 1156996800000 , 0] , [ 1159588800000 , 3899486.0] , [ 1162270800000 , 3899486.0] , [ 1164862800000 , 3899486.0] , [ 1167541200000 , 3564700.0] , [ 1170219600000 , 3564700.0] , [ 1172638800000 , 3564700.0] , [ 1175313600000 , 2648493.0] , [ 1177905600000 , 2648493.0] , [ 1180584000000 , 2648493.0] , [ 1183176000000 , 2522993.0] , [ 1185854400000 , 2522993.0] , [ 1188532800000 , 2522993.0] , [ 1191124800000 , 2906501.0] , [ 1193803200000 , 2906501.0] , [ 1196398800000 , 2906501.0] , [ 1199077200000 , 2206761.0] , [ 1201755600000 , 2206761.0] , [ 1204261200000 , 2206761.0] , [ 1206936000000 , 2287726.0] , [ 1209528000000 , 2287726.0] , [ 1212206400000 , 2287726.0] , [ 1214798400000 , 2732646.0] , [ 1217476800000 , 2732646.0] , [ 1220155200000 , 2732646.0] , [ 1222747200000 , 2599196.0] , [ 1225425600000 , 2599196.0] , [ 1228021200000 , 2599196.0] , [ 1230699600000 , 1924387.0] , [ 1233378000000 , 1924387.0] , [ 1235797200000 , 1924387.0] , [ 1238472000000 , 1756311.0] , [ 1241064000000 , 1756311.0] , [ 1243742400000 , 1756311.0] , [ 1246334400000 , 1743470.0] , [ 1249012800000 , 1743470.0] , [ 1251691200000 , 1743470.0] , [ 1254283200000 , 1519010.0] , [ 1256961600000 , 1519010.0] , [ 1259557200000 , 1519010.0] , [ 1262235600000 , 1591444.0] , [ 1264914000000 , 1591444.0] , [ 1267333200000 , 1591444.0] , [ 1270008000000 , 1543784.0] , [ 1272600000000 , 1543784.0] , [ 1275278400000 , 1543784.0] , [ 1277870400000 , 1309915.0] , [ 1280548800000 , 1309915.0] , [ 1283227200000 , 1309915.0] , [ 1285819200000 , 1331875.0] , [ 1288497600000 , 1331875.0] , [ 1291093200000 , 1331875.0] , [ 1293771600000 , 1331875.0] , [ 1296450000000 , 1154695.0] , [ 1298869200000 , 1154695.0] , [ 1301544000000 , 1194025.0] , [ 1304136000000 , 1194025.0] , [ 1306814400000 , 1194025.0] , [ 1309406400000 , 1194025.0] , [ 1312084800000 , 1194025.0] , [ 1314763200000 , 1244525.0] , [ 1317355200000 , 475000.0] , [ 1320033600000 , 475000.0] , [ 1322629200000 , 475000.0] , [ 1325307600000 , 690033.0] , [ 1327986000000 , 690033.0] , [ 1330491600000 , 690033.0] , [ 1333166400000 , 514733.0] , [ 1335758400000 , 514733.0]]

}];

//Select option variables for counters:

$scope.personNamesSelOpts=[''];

$scope.personTagsSelOpts=[];

$scope.personAggregateSelOpts=[];

var baseUrl = "http://localhost:9393/metrics";

$http.get(baseUrl + "/" + 'field-value-counters').

success(function(data) {

$log.info("getMenuOptions get result:",data.content);

$scope.personNamesSelOpts= data.content;

$scope.personTagsSelOpts= data.content;

$log.info($scope.personNamesSelOpts,$scope.personTagsSelOpts,$scope.personAggregateSelOpts);

})

.error(function(error){

$log.error("getMenuOptions error:",error);

$ionicLoading.hide();

});

$http.get(baseUrl + "/" + 'aggregate-counters').

success(function(data) {

$log.info("getMenuOptions get result:",data.content);

$scope.personAggregateSelOpts= data.content;

$log.info($scope.personNamesSelOpts,$scope.personTagsSelOpts,$scope.personAggregateSelOpts);

})

.error(function(error){

$log.error("getMenuOptions error:",error);

$ionicLoading.hide();

});

})

.controller('PersonDetailCtrl', function($scope, $stateParams) {

})

.controller('GroupTabCtrl', function($scope,$log,$ionicLoading,$http,CONFIG\_ENV) {

//Slide-box view

$scope.selectedViewIndex=0;

$scope.changeViewIndex = function(index)

{

$log.info("GroupTabCtrl selected view index:",index);

$scope.selectedViewIndex = index;

};

//@example:http://plnkr.co/edit/lBKFld?p=preview

/\* Chart options \*/

$scope.groupAgesChartOptions = {

chart: {

type: 'lineChart',

height: 450,

margin : {

top: 20,

right: 20,

bottom: 40,

left: 55

},

x: function(d){ return d.x; },

y: function(d){ return d.y; },

useInteractiveGuideline: true,

dispatch: {

stateChange: function(e){ console.log("stateChange"); },

changeState: function(e){ console.log("changeState"); },

tooltipShow: function(e){ console.log("tooltipShow"); },

tooltipHide: function(e){ console.log("tooltipHide"); }

},

xAxis: {

axisLabel: 'Time (ms)'

},

yAxis: {

axisLabel: 'Voltage (v)',

tickFormat: function(d){

return d3.format('.02f')(d);

},

axisLabelDistance: 30

},

callback: function(chart){

console.log("!!! lineChart callback !!!");

}

},

title: {

enable: true,

text: ''

},

subtitle: {

enable: true,

text: '',

css: {

'text-align': 'center',

'margin': '10px 13px 0px 7px'

}

},

caption: {

enable: true,

html: '<b>Figure 1.</b> Lorem ipsum dolor sit amet, at eam blandit sadipscing, <span style="text-decoration: underline;">vim adhuc sanctus disputando ex</span>, cu usu affert alienum urbanitas. <i>Cum in purto erat, mea ne nominavi persecuti reformidans.</i> Docendi blandit abhorreant ea has, minim tantas alterum pro eu. <span style="color: darkred;">Exerci graeci ad vix, elit tacimates ea duo</span>. Id mel eruditi fuisset. Stet vidit patrioque in pro, eum ex veri verterem abhorreant, id unum oportere intellegam nec<sup>[1, <a href="https://github.com/krispo/angular-nvd3" target="\_blank">2</a>, 3]</sup>.',

css: {

'text-align': 'justify',

'margin': '10px 13px 0px 7px'

}

}

};

/\* Chart data \*/

$scope.groupAgesChartData = /\*Random Data Generator \*/

function sinAndCos() {

var sin = [],sin2 = [],

cos = [];

//Data is represented as an array of {x,y} pairs.

for (var i = 0; i < 10; i++) {

sin.push({x: i, y: Math.sin(i/10)});

sin2.push({x: i, y: i % 10 == 5 ? null : Math.sin(i/10) \*0.25 + 0.5});

cos.push({x: i, y: .5 \* Math.cos(i/10+ 2) + Math.random() / 10});

}

//Line chart data should be sent as an array of series objects.

return [

{

values: sin, //values - represents the array of {x,y} data points

key: 'Sine Wave', //key - the name of the series.

color: '#ff7f0e' //color - optional: choose your own line color.

},

{

values: cos,

key: 'Cosine Wave',

color: '#2ca02c'

},

{

values: sin2,

key: 'Another sine wave',

color: '#7777ff',

area: true //area - set to true if you want this line to turn into a filled area chart.

}

];

};

//@example:http://plnkr.co/edit/UASCUL?p=preview

/\* Chart options \*/

$scope.groupGendersChartOptions = {

chart: {

type: 'linePlusBarChart',

height: 450,

margin : {

top: 30,

right: 75,

bottom: 50,

left: 75

},

x: function(d, i){return i;},

y: function(d){return d[1];},

color: d3.scale.category10().range(),

transitionDuration: 250,

xAxis: {

axisLabel: 'X Axis',

showMaxMin: false,

tickFormat: function(d) {

var dx = $scope.data[0].values[d] && $scope.data[0].values[d][0] || 0;

return dx ? d3.time.format('%x')(new Date(dx)) : '';

},

staggerLabels: true

},

y1Axis: {

axisLabel: 'Y1 Axis',

tickFormat: function(d){return d3.format(',f')(d)}

},

y2Axis: {

axisLabel: 'Y2 Axis',

tickFormat: function(d) { return '$' + d3.format(',.2f')(d);}

}

}

};

/\* Chart data \*/

$scope.groupGendersChartData = [

{

"key" : "Quantity" ,

"bar": true,

"values" : [ [ 1136005200000 , 1271000.0] , [ 1138683600000 , 1271000.0] , [ 1141102800000 , 1271000.0] , [ 1143781200000 , 0] , [ 1146369600000 , 0] , [ 1149048000000 , 0] , [ 1151640000000 , 0] , [ 1154318400000 , 0] , [ 1156996800000 , 0] , [ 1159588800000 , 3899486.0] , [ 1162270800000 , 3899486.0] , [ 1164862800000 , 3899486.0] , [ 1167541200000 , 3564700.0] , [ 1170219600000 , 3564700.0] , [ 1172638800000 , 3564700.0] , [ 1175313600000 , 2648493.0] , [ 1177905600000 , 2648493.0] , [ 1180584000000 , 2648493.0] , [ 1183176000000 , 2522993.0] , [ 1185854400000 , 2522993.0] , [ 1188532800000 , 2522993.0] , [ 1191124800000 , 2906501.0] , [ 1193803200000 , 2906501.0] , [ 1196398800000 , 2906501.0] , [ 1199077200000 , 2206761.0] , [ 1201755600000 , 2206761.0] , [ 1204261200000 , 2206761.0] , [ 1206936000000 , 2287726.0] , [ 1209528000000 , 2287726.0] , [ 1212206400000 , 2287726.0] , [ 1214798400000 , 2732646.0] , [ 1217476800000 , 2732646.0] , [ 1220155200000 , 2732646.0] , [ 1222747200000 , 2599196.0] , [ 1225425600000 , 2599196.0] , [ 1228021200000 , 2599196.0] , [ 1230699600000 , 1924387.0] , [ 1233378000000 , 1924387.0] , [ 1235797200000 , 1924387.0] , [ 1238472000000 , 1756311.0] , [ 1241064000000 , 1756311.0] , [ 1243742400000 , 1756311.0] , [ 1246334400000 , 1743470.0] , [ 1249012800000 , 1743470.0] , [ 1251691200000 , 1743470.0] , [ 1254283200000 , 1519010.0] , [ 1256961600000 , 1519010.0] , [ 1259557200000 , 1519010.0] , [ 1262235600000 , 1591444.0] , [ 1264914000000 , 1591444.0] , [ 1267333200000 , 1591444.0] , [ 1270008000000 , 1543784.0] , [ 1272600000000 , 1543784.0] , [ 1275278400000 , 1543784.0] , [ 1277870400000 , 1309915.0] , [ 1280548800000 , 1309915.0] , [ 1283227200000 , 1309915.0] , [ 1285819200000 , 1331875.0] , [ 1288497600000 , 1331875.0] , [ 1291093200000 , 1331875.0] , [ 1293771600000 , 1331875.0] , [ 1296450000000 , 1154695.0] , [ 1298869200000 , 1154695.0] , [ 1301544000000 , 1194025.0] , [ 1304136000000 , 1194025.0] , [ 1306814400000 , 1194025.0] , [ 1309406400000 , 1194025.0] , [ 1312084800000 , 1194025.0] , [ 1314763200000 , 1244525.0] , [ 1317355200000 , 475000.0] , [ 1320033600000 , 475000.0] , [ 1322629200000 , 475000.0] , [ 1325307600000 , 690033.0] , [ 1327986000000 , 690033.0] , [ 1330491600000 , 690033.0] , [ 1333166400000 , 514733.0] , [ 1335758400000 , 514733.0]]

},

{

"key" : "Price" ,

"values" : [ [ 1136005200000 , 71.89] , [ 1138683600000 , 75.51] , [ 1141102800000 , 68.49] , [ 1143781200000 , 62.72] , [ 1146369600000 , 70.39] , [ 1149048000000 , 59.77] , [ 1151640000000 , 57.27] , [ 1154318400000 , 67.96] , [ 1156996800000 , 67.85] , [ 1159588800000 , 76.98] , [ 1162270800000 , 81.08] , [ 1164862800000 , 91.66] , [ 1167541200000 , 84.84] , [ 1170219600000 , 85.73] , [ 1172638800000 , 84.61] , [ 1175313600000 , 92.91] , [ 1177905600000 , 99.8] , [ 1180584000000 , 121.191] , [ 1183176000000 , 122.04] , [ 1185854400000 , 131.76] , [ 1188532800000 , 138.48] , [ 1191124800000 , 153.47] , [ 1193803200000 , 189.95] , [ 1196398800000 , 182.22] , [ 1199077200000 , 198.08] , [ 1201755600000 , 135.36] , [ 1204261200000 , 125.02] , [ 1206936000000 , 143.5] , [ 1209528000000 , 173.95] , [ 1212206400000 , 188.75] , [ 1214798400000 , 167.44] , [ 1217476800000 , 158.95] , [ 1220155200000 , 169.53] , [ 1222747200000 , 113.66] , [ 1225425600000 , 107.59] , [ 1228021200000 , 92.67] , [ 1230699600000 , 85.35] , [ 1233378000000 , 90.13] , [ 1235797200000 , 89.31] , [ 1238472000000 , 105.12] , [ 1241064000000 , 125.83] , [ 1243742400000 , 135.81] , [ 1246334400000 , 142.43] , [ 1249012800000 , 163.39] , [ 1251691200000 , 168.21] , [ 1254283200000 , 185.35] , [ 1256961600000 , 188.5] , [ 1259557200000 , 199.91] , [ 1262235600000 , 210.732] , [ 1264914000000 , 192.063] , [ 1267333200000 , 204.62] , [ 1270008000000 , 235.0] , [ 1272600000000 , 261.09] , [ 1275278400000 , 256.88] , [ 1277870400000 , 251.53] , [ 1280548800000 , 257.25] , [ 1283227200000 , 243.1] , [ 1285819200000 , 283.75] , [ 1288497600000 , 300.98] , [ 1291093200000 , 311.15] , [ 1293771600000 , 322.56] , [ 1296450000000 , 339.32] , [ 1298869200000 , 353.21] , [ 1301544000000 , 348.5075] , [ 1304136000000 , 350.13] , [ 1306814400000 , 347.83] , [ 1309406400000 , 335.67] , [ 1312084800000 , 390.48] , [ 1314763200000 , 384.83] , [ 1317355200000 , 381.32] , [ 1320033600000 , 404.78] , [ 1322629200000 , 382.2] , [ 1325307600000 , 405.0] , [ 1327986000000 , 456.48] , [ 1330491600000 , 542.44] , [ 1333166400000 , 599.55] , [ 1335758400000 , 583.98] ]

}

];

//@example:http://plnkr.co/edit/CIGW0o?p=preview

/\* Chart options \*/

$scope.groupRacesChartOptions = {

chart: {

type: 'stackedAreaChart',

height: 450,

margin : {

top: 20,

right: 20,

bottom: 60,

left: 40

},

x: function(d){return d[0];},

y: function(d){return d[1];},

useVoronoi: false,

clipEdge: true,

transitionDuration: 500,

useInteractiveGuideline: true,

xAxis: {

showMaxMin: false,

tickFormat: function(d) {

return d3.time.format('%x')(new Date(d))

}

},

yAxis: {

tickFormat: function(d){

return d3.format(',.2f')(d);

}

}

}

};

/\* Chart data \*/

$scope.groupRacesChartData = [

{

"key" : "North America" ,

"values" : [ [ 1025409600000 , 23.041422681023] , [ 1028088000000 , 19.854291255832] , [ 1030766400000 , 21.02286281168] , [ 1033358400000 , 22.093608385173] , [ 1036040400000 , 25.108079299458] , [ 1038632400000 , 26.982389242348] , [ 1041310800000 , 19.828984957662] , [ 1043989200000 , 19.914055036294] , [ 1046408400000 , 19.436150539916] , [ 1049086800000 , 21.558650338602] , [ 1051675200000 , 24.395594061773] , [ 1054353600000 , 24.747089309384] , [ 1056945600000 , 23.491755498807] , [ 1059624000000 , 23.376634878164] , [ 1062302400000 , 24.581223154533] , [ 1064894400000 , 24.922476843538] , [ 1067576400000 , 27.357712939042] , [ 1070168400000 , 26.503020572593] , [ 1072846800000 , 26.658901244878] , [ 1075525200000 , 27.065704156445] , [ 1078030800000 , 28.735320452588] , [ 1080709200000 , 31.572277846319] , [ 1083297600000 , 30.932161503638] , [ 1085976000000 , 31.627029785554] , [ 1088568000000 , 28.728743674232] , [ 1091246400000 , 26.858365172675] , [ 1093924800000 , 27.279922830032] , [ 1096516800000 , 34.408301211324] , [ 1099195200000 , 34.794362930439] , [ 1101790800000 , 35.609978198951] , [ 1104469200000 , 33.574394968037] , [ 1107147600000 , 31.979405070598] , [ 1109566800000 , 31.19009040297] , [ 1112245200000 , 31.083933968994] , [ 1114833600000 , 29.668971113185] , [ 1117512000000 , 31.490638014379] , [ 1120104000000 , 31.818617451128] , [ 1122782400000 , 32.960314008183] , [ 1125460800000 , 31.313383196209] , [ 1128052800000 , 33.125486081852] , [ 1130734800000 , 32.791805509149] , [ 1133326800000 , 33.506038030366] , [ 1136005200000 , 26.96501697216] , [ 1138683600000 , 27.38478809681] , [ 1141102800000 , 27.371377218209] , [ 1143781200000 , 26.309915460827] , [ 1146369600000 , 26.425199957518] , [ 1149048000000 , 26.823411519396] , [ 1151640000000 , 23.850443591587] , [ 1154318400000 , 23.158355444054] , [ 1156996800000 , 22.998689393695] , [ 1159588800000 , 27.9771285113] , [ 1162270800000 , 29.073672469719] , [ 1164862800000 , 28.587640408904] , [ 1167541200000 , 22.788453687637] , [ 1170219600000 , 22.429199073597] , [ 1172638800000 , 22.324103271052] , [ 1175313600000 , 17.558388444187] , [ 1177905600000 , 16.769518096208] , [ 1180584000000 , 16.214738201301] , [ 1183176000000 , 18.729632971229] , [ 1185854400000 , 18.814523318847] , [ 1188532800000 , 19.789986451358] , [ 1191124800000 , 17.070049054933] , [ 1193803200000 , 16.121349575716] , [ 1196398800000 , 15.141659430091] , [ 1199077200000 , 17.175388025297] , [ 1201755600000 , 17.286592443522] , [ 1204261200000 , 16.323141626568] , [ 1206936000000 , 19.231263773952] , [ 1209528000000 , 18.446256391095] , [ 1212206400000 , 17.822632399764] , [ 1214798400000 , 15.53936647598] , [ 1217476800000 , 15.255131790217] , [ 1220155200000 , 15.660963922592] , [ 1222747200000 , 13.254482273698] , [ 1225425600000 , 11.920796202299] , [ 1228021200000 , 12.122809090924] , [ 1230699600000 , 15.691026271393] , [ 1233378000000 , 14.720881635107] , [ 1235797200000 , 15.387939360044] , [ 1238472000000 , 13.765436672228] , [ 1241064000000 , 14.631445864799] , [ 1243742400000 , 14.292446536221] , [ 1246334400000 , 16.170071367017] , [ 1249012800000 , 15.948135554337] , [ 1251691200000 , 16.612872685134] , [ 1254283200000 , 18.778338719091] , [ 1256961600000 , 16.756026065421] , [ 1259557200000 , 19.385804443146] , [ 1262235600000 , 22.950590240168] , [ 1264914000000 , 23.61159018141] , [ 1267333200000 , 25.708586989581] , [ 1270008000000 , 26.883915999885] , [ 1272600000000 , 25.893486687065] , [ 1275278400000 , 24.678914263176] , [ 1277870400000 , 25.937275793024] , [ 1280548800000 , 29.461381693838] , [ 1283227200000 , 27.357322961861] , [ 1285819200000 , 29.057235285673] , [ 1288497600000 , 28.549434189386] , [ 1291093200000 , 28.506352379724] , [ 1293771600000 , 29.449241421598] , [ 1296450000000 , 25.796838168807] , [ 1298869200000 , 28.740145449188] , [ 1301544000000 , 22.091744141872] , [ 1304136000000 , 25.07966254541] , [ 1306814400000 , 23.674906973064] , [ 1309406400000 , 23.418002742929] , [ 1312084800000 , 23.24364413887] , [ 1314763200000 , 31.591854066817] , [ 1317355200000 , 31.497112374114] , [ 1320033600000 , 26.67238082043] , [ 1322629200000 , 27.297080015495] , [ 1325307600000 , 20.174315530051] , [ 1327986000000 , 19.631084213898] , [ 1330491600000 , 20.366462219461] , [ 1333166400000 , 19.284784434185] , [ 1335758400000 , 19.157810257624]]

},

{

"key" : "Africa" ,

"values" : [ [ 1025409600000 , 7.9356392949025] , [ 1028088000000 , 7.4514668527298] , [ 1030766400000 , 7.9085410566608] , [ 1033358400000 , 5.8996782364764] , [ 1036040400000 , 6.0591869346923] , [ 1038632400000 , 5.9667815800451] , [ 1041310800000 , 8.65528925664] , [ 1043989200000 , 8.7690763386254] , [ 1046408400000 , 8.6386160387453] , [ 1049086800000 , 5.9895557449743] , [ 1051675200000 , 6.3840324338159] , [ 1054353600000 , 6.5196511461441] , [ 1056945600000 , 7.0738618553114] , [ 1059624000000 , 6.5745957367133] , [ 1062302400000 , 6.4658359184444] , [ 1064894400000 , 2.7622758754954] , [ 1067576400000 , 2.9794782986241] , [ 1070168400000 , 2.8735432712019] , [ 1072846800000 , 1.6344817513645] , [ 1075525200000 , 1.5869248754883] , [ 1078030800000 , 1.7172279157246] , [ 1080709200000 , 1.9649927409867] , [ 1083297600000 , 2.0261695079196] , [ 1085976000000 , 2.0541261923929] , [ 1088568000000 , 3.9466318927569] , [ 1091246400000 , 3.7826770946089] , [ 1093924800000 , 3.9543021004028] , [ 1096516800000 , 3.8309891064711] , [ 1099195200000 , 3.6340958946166] , [ 1101790800000 , 3.5289755762525] , [ 1104469200000 , 5.702378559857] , [ 1107147600000 , 5.6539569019223] , [ 1109566800000 , 5.5449506370392] , [ 1112245200000 , 4.7579993280677] , [ 1114833600000 , 4.4816139372906] , [ 1117512000000 , 4.5965558568606] , [ 1120104000000 , 4.3747066116976] , [ 1122782400000 , 4.4588822917087] , [ 1125460800000 , 4.4460351848286] , [ 1128052800000 , 3.7989113035136] , [ 1130734800000 , 3.7743883140088] , [ 1133326800000 , 3.7727852823828] , [ 1136005200000 , 7.2968111448895] , [ 1138683600000 , 7.2800122043237] , [ 1141102800000 , 7.1187787503354] , [ 1143781200000 , 8.351887016482] , [ 1146369600000 , 8.4156698763993] , [ 1149048000000 , 8.1673298604231] , [ 1151640000000 , 5.5132447126042] , [ 1154318400000 , 6.1152537710599] , [ 1156996800000 , 6.076765091942] , [ 1159588800000 , 4.6304473798646] , [ 1162270800000 , 4.6301068469402] , [ 1164862800000 , 4.3466656309389] , [ 1167541200000 , 6.830104897003] , [ 1170219600000 , 7.241633040029] , [ 1172638800000 , 7.1432372054153] , [ 1175313600000 , 10.608942063374] , [ 1177905600000 , 10.914964549494] , [ 1180584000000 , 10.933223880565] , [ 1183176000000 , 8.3457524851265] , [ 1185854400000 , 8.1078413081882] , [ 1188532800000 , 8.2697185922474] , [ 1191124800000 , 8.4742436475968] , [ 1193803200000 , 8.4994601179319] , [ 1196398800000 , 8.7387319683243] , [ 1199077200000 , 6.8829183612895] , [ 1201755600000 , 6.984133637885] , [ 1204261200000 , 7.0860136043287] , [ 1206936000000 , 4.3961787956053] , [ 1209528000000 , 3.8699674365231] , [ 1212206400000 , 3.6928925238305] , [ 1214798400000 , 6.7571718894253] , [ 1217476800000 , 6.4367313362344] , [ 1220155200000 , 6.4048441521454] , [ 1222747200000 , 5.4643833239669] , [ 1225425600000 , 5.3150786833374] , [ 1228021200000 , 5.3011272612576] , [ 1230699600000 , 4.1203601430809] , [ 1233378000000 , 4.0881783200525] , [ 1235797200000 , 4.1928665957189] , [ 1238472000000 , 7.0249415663205] , [ 1241064000000 , 7.006530880769] , [ 1243742400000 , 6.994835633224] , [ 1246334400000 , 6.1220222336254] , [ 1249012800000 , 6.1177436137653] , [ 1251691200000 , 6.1413396231981] , [ 1254283200000 , 4.8046006145874] , [ 1256961600000 , 4.6647600660544] , [ 1259557200000 , 4.544865006255] , [ 1262235600000 , 6.0488249316539] , [ 1264914000000 , 6.3188669540206] , [ 1267333200000 , 6.5873958262306] , [ 1270008000000 , 6.2281189839578] , [ 1272600000000 , 5.8948915746059] , [ 1275278400000 , 5.5967320482214] , [ 1277870400000 , 0.99784432084837] , [ 1280548800000 , 1.0950794175359] , [ 1283227200000 , 0.94479734407491] , [ 1285819200000 , 1.222093988688] , [ 1288497600000 , 1.335093106856] , [ 1291093200000 , 1.3302565104985] , [ 1293771600000 , 1.340824670897] , [ 1296450000000 , 0] , [ 1298869200000 , 0] , [ 1301544000000 , 0] , [ 1304136000000 , 0] , [ 1306814400000 , 0] , [ 1309406400000 , 0] , [ 1312084800000 , 0] , [ 1314763200000 , 0] , [ 1317355200000 , 4.4583692315] , [ 1320033600000 , 3.6493043348059] , [ 1322629200000 , 3.8610064091761] , [ 1325307600000 , 5.5144800685202] , [ 1327986000000 , 5.1750695220791] , [ 1330491600000 , 5.6710066952691] , [ 1333166400000 , 5.5611890039181] , [ 1335758400000 , 5.5979368839939]]

},

{

"key" : "South America" ,

"values" : [ [ 1025409600000 , 7.9149900245423] , [ 1028088000000 , 7.0899888751059] , [ 1030766400000 , 7.5996132380614] , [ 1033358400000 , 8.2741174301034] , [ 1036040400000 , 9.3564460833513] , [ 1038632400000 , 9.7066786059904] , [ 1041310800000 , 10.213363052343] , [ 1043989200000 , 10.285809585273] , [ 1046408400000 , 10.222053149228] , [ 1049086800000 , 8.6188592137975] , [ 1051675200000 , 9.3335447543566] , [ 1054353600000 , 8.9312402186628] , [ 1056945600000 , 8.1895089343658] , [ 1059624000000 , 8.260622135079] , [ 1062302400000 , 7.7700786851364] , [ 1064894400000 , 7.9907428771318] , [ 1067576400000 , 8.7769091865606] , [ 1070168400000 , 8.4855077060661] , [ 1072846800000 , 9.6277203033655] , [ 1075525200000 , 9.9685913452624] , [ 1078030800000 , 10.615085181759] , [ 1080709200000 , 9.2902488079646] , [ 1083297600000 , 8.8610439830061] , [ 1085976000000 , 9.1075344931229] , [ 1088568000000 , 9.9156737639203] , [ 1091246400000 , 9.7826003238782] , [ 1093924800000 , 10.55403610555] , [ 1096516800000 , 10.926900264097] , [ 1099195200000 , 10.903144818736] , [ 1101790800000 , 10.862890389067] , [ 1104469200000 , 10.64604998964] , [ 1107147600000 , 10.042790814087] , [ 1109566800000 , 9.7173391591038] , [ 1112245200000 , 9.6122415755443] , [ 1114833600000 , 9.4337921146562] , [ 1117512000000 , 9.814827171183] , [ 1120104000000 , 12.059260396788] , [ 1122782400000 , 12.139649903873] , [ 1125460800000 , 12.281290663822] , [ 1128052800000 , 8.8037085409056] , [ 1130734800000 , 8.6300618239176] , [ 1133326800000 , 9.1225708491432] , [ 1136005200000 , 12.988124170836] , [ 1138683600000 , 13.356778764353] , [ 1141102800000 , 13.611196863271] , [ 1143781200000 , 6.8959030061189] , [ 1146369600000 , 6.9939633271353] , [ 1149048000000 , 6.7241510257676] , [ 1151640000000 , 5.5611293669517] , [ 1154318400000 , 5.6086488714041] , [ 1156996800000 , 5.4962849907033] , [ 1159588800000 , 6.9193153169278] , [ 1162270800000 , 7.0016334389778] , [ 1164862800000 , 6.7865422443273] , [ 1167541200000 , 9.0006454225383] , [ 1170219600000 , 9.2233916171431] , [ 1172638800000 , 8.8929316009479] , [ 1175313600000 , 10.345937520404] , [ 1177905600000 , 10.075914677026] , [ 1180584000000 , 10.089006188111] , [ 1183176000000 , 10.598330295008] , [ 1185854400000 , 9.9689546533009] , [ 1188532800000 , 9.7740580198146] , [ 1191124800000 , 10.558483060626] , [ 1193803200000 , 9.9314651823603] , [ 1196398800000 , 9.3997715873769] , [ 1199077200000 , 8.4086493387262] , [ 1201755600000 , 8.9698309085926] , [ 1204261200000 , 8.2778357995396] , [ 1206936000000 , 8.8585045600123] , [ 1209528000000 , 8.7013756413322] , [ 1212206400000 , 7.7933605469443] , [ 1214798400000 , 7.0236183483064] , [ 1217476800000 , 6.9873088186829] , [ 1220155200000 , 6.8031713070097] , [ 1222747200000 , 6.6869531315723] , [ 1225425600000 , 6.138256993963] , [ 1228021200000 , 5.6434994016354] , [ 1230699600000 , 5.495220262512] , [ 1233378000000 , 4.6885326869846] , [ 1235797200000 , 4.4524349883438] , [ 1238472000000 , 5.6766520778185] , [ 1241064000000 , 5.7675774480752] , [ 1243742400000 , 5.7882863168337] , [ 1246334400000 , 7.2666010034924] , [ 1249012800000 , 7.5191821322261] , [ 1251691200000 , 7.849651451445] , [ 1254283200000 , 10.383992037985] , [ 1256961600000 , 9.0653691861818] , [ 1259557200000 , 9.6705248324159] , [ 1262235600000 , 10.856380561349] , [ 1264914000000 , 11.27452370892] , [ 1267333200000 , 11.754156529088] , [ 1270008000000 , 8.2870811422455] , [ 1272600000000 , 8.0210264360699] , [ 1275278400000 , 7.5375074474865] , [ 1277870400000 , 8.3419527338039] , [ 1280548800000 , 9.4197471818443] , [ 1283227200000 , 8.7321733185797] , [ 1285819200000 , 9.6627062648126] , [ 1288497600000 , 10.187962234548] , [ 1291093200000 , 9.8144201733476] , [ 1293771600000 , 10.275723361712] , [ 1296450000000 , 16.796066079353] , [ 1298869200000 , 17.543254984075] , [ 1301544000000 , 16.673660675083] , [ 1304136000000 , 17.963944353609] , [ 1306814400000 , 16.63774086721] , [ 1309406400000 , 15.84857094609] , [ 1312084800000 , 14.767303362181] , [ 1314763200000 , 24.778452182433] , [ 1317355200000 , 18.370353229999] , [ 1320033600000 , 15.253137429099] , [ 1322629200000 , 14.989600840649] , [ 1325307600000 , 16.052539160125] , [ 1327986000000 , 16.424390322793] , [ 1330491600000 , 17.884020741104] , [ 1333166400000 , 18.372698836036] , [ 1335758400000 , 18.315881576096]]

},

{

"key" : "Asia" ,

"values" : [ [ 1025409600000 , 13.153938631352] , [ 1028088000000 , 12.456410521864] , [ 1030766400000 , 12.537048663919] , [ 1033358400000 , 13.947386398309] , [ 1036040400000 , 14.421680682568] , [ 1038632400000 , 14.143238262286] , [ 1041310800000 , 12.229635347478] , [ 1043989200000 , 12.508479916948] , [ 1046408400000 , 12.155368409526] , [ 1049086800000 , 13.335455563994] , [ 1051675200000 , 12.888210138167] , [ 1054353600000 , 12.842092790511] , [ 1056945600000 , 12.513816474199] , [ 1059624000000 , 12.21453674494] , [ 1062302400000 , 11.750848343935] , [ 1064894400000 , 10.526579636787] , [ 1067576400000 , 10.873596086087] , [ 1070168400000 , 11.019967131519] , [ 1072846800000 , 11.235789380602] , [ 1075525200000 , 11.859910850657] , [ 1078030800000 , 12.531031616536] , [ 1080709200000 , 11.360451067019] , [ 1083297600000 , 11.456244780202] , [ 1085976000000 , 11.436991407309] , [ 1088568000000 , 11.638595744327] , [ 1091246400000 , 11.190418301469] , [ 1093924800000 , 11.835608007589] , [ 1096516800000 , 11.540980244475] , [ 1099195200000 , 10.958762325687] , [ 1101790800000 , 10.885791159509] , [ 1104469200000 , 13.605810720109] , [ 1107147600000 , 13.128978067437] , [ 1109566800000 , 13.119012086882] , [ 1112245200000 , 13.003706129783] , [ 1114833600000 , 13.326996807689] , [ 1117512000000 , 13.547947991743] , [ 1120104000000 , 12.807959646616] , [ 1122782400000 , 12.931763821068] , [ 1125460800000 , 12.795359993008] , [ 1128052800000 , 9.6998935538319] , [ 1130734800000 , 9.3473740089131] , [ 1133326800000 , 9.36902067716] , [ 1136005200000 , 14.258619539875] , [ 1138683600000 , 14.21241095603] , [ 1141102800000 , 13.973193618249] , [ 1143781200000 , 15.218233920664] , [ 1146369600000 , 14.382109727451] , [ 1149048000000 , 13.894310878491] , [ 1151640000000 , 15.593086090031] , [ 1154318400000 , 16.244839695189] , [ 1156996800000 , 16.017088850647] , [ 1159588800000 , 14.183951830057] , [ 1162270800000 , 14.148523245696] , [ 1164862800000 , 13.424326059971] , [ 1167541200000 , 12.974450435754] , [ 1170219600000 , 13.232470418021] , [ 1172638800000 , 13.318762655574] , [ 1175313600000 , 15.961407746104] , [ 1177905600000 , 16.287714639805] , [ 1180584000000 , 16.24659058389] , [ 1183176000000 , 17.564505594808] , [ 1185854400000 , 17.872725373164] , [ 1188532800000 , 18.018998508756] , [ 1191124800000 , 15.584518016602] , [ 1193803200000 , 15.480850647182] , [ 1196398800000 , 15.699120036985] , [ 1199077200000 , 19.184281817226] , [ 1201755600000 , 19.691226605205] , [ 1204261200000 , 18.982314051293] , [ 1206936000000 , 18.707820309008] , [ 1209528000000 , 17.459630929759] , [ 1212206400000 , 16.500616076782] , [ 1214798400000 , 18.086324003978] , [ 1217476800000 , 18.929464156259] , [ 1220155200000 , 18.233728682084] , [ 1222747200000 , 16.315776297325] , [ 1225425600000 , 14.632892190251] , [ 1228021200000 , 14.667835024479] , [ 1230699600000 , 13.946993947309] , [ 1233378000000 , 14.394304684398] , [ 1235797200000 , 13.724462792967] , [ 1238472000000 , 10.930879035807] , [ 1241064000000 , 9.8339915513708] , [ 1243742400000 , 10.053858541872] , [ 1246334400000 , 11.786998438286] , [ 1249012800000 , 11.780994901769] , [ 1251691200000 , 11.305889670277] , [ 1254283200000 , 10.918452290083] , [ 1256961600000 , 9.6811395055706] , [ 1259557200000 , 10.971529744038] , [ 1262235600000 , 13.330210480209] , [ 1264914000000 , 14.592637568961] , [ 1267333200000 , 14.605329141157] , [ 1270008000000 , 13.936853794037] , [ 1272600000000 , 12.189480759072] , [ 1275278400000 , 11.676151385046] , [ 1277870400000 , 13.058852800018] , [ 1280548800000 , 13.62891543203] , [ 1283227200000 , 13.811107569918] , [ 1285819200000 , 13.786494560786] , [ 1288497600000 , 14.045162857531] , [ 1291093200000 , 13.697412447286] , [ 1293771600000 , 13.677681376221] , [ 1296450000000 , 19.96151186453] , [ 1298869200000 , 21.049198298156] , [ 1301544000000 , 22.687631094009] , [ 1304136000000 , 25.469010617433] , [ 1306814400000 , 24.88379943712] , [ 1309406400000 , 24.203843814249] , [ 1312084800000 , 22.138760964036] , [ 1314763200000 , 16.034636966228] , [ 1317355200000 , 15.394958944555] , [ 1320033600000 , 12.62564246197] , [ 1322629200000 , 12.973735699739] , [ 1325307600000 , 15.78601833615] , [ 1327986000000 , 15.227368020134] , [ 1330491600000 , 15.899752650733] , [ 1333166400000 , 15.661317319168] , [ 1335758400000 , 15.359891177281]]

} ,

{

"key" : "Europe" ,

"values" : [ [ 1025409600000 , 9.3433263069351] , [ 1028088000000 , 8.4583069475546] , [ 1030766400000 , 8.0342398154196] , [ 1033358400000 , 8.1538966876572] , [ 1036040400000 , 10.743604786849] , [ 1038632400000 , 12.349366155851] , [ 1041310800000 , 10.742682503899] , [ 1043989200000 , 11.360983869935] , [ 1046408400000 , 11.441336039535] , [ 1049086800000 , 10.897508791837] , [ 1051675200000 , 11.469101547709] , [ 1054353600000 , 12.086311476742] , [ 1056945600000 , 8.0697180773504] , [ 1059624000000 , 8.2004392233445] , [ 1062302400000 , 8.4566434900643] , [ 1064894400000 , 7.9565760979059] , [ 1067576400000 , 9.3764619255827] , [ 1070168400000 , 9.0747664160538] , [ 1072846800000 , 10.508939004673] , [ 1075525200000 , 10.69936754483] , [ 1078030800000 , 10.681562399145] , [ 1080709200000 , 13.184786109406] , [ 1083297600000 , 12.668213052351] , [ 1085976000000 , 13.430509403986] , [ 1088568000000 , 12.393086349213] , [ 1091246400000 , 11.942374044842] , [ 1093924800000 , 12.062227685742] , [ 1096516800000 , 11.969974363623] , [ 1099195200000 , 12.14374574055] , [ 1101790800000 , 12.69422821995] , [ 1104469200000 , 9.1235211044692] , [ 1107147600000 , 8.758211757584] , [ 1109566800000 , 8.8072309258443] , [ 1112245200000 , 11.687595946835] , [ 1114833600000 , 11.079723082664] , [ 1117512000000 , 12.049712896076] , [ 1120104000000 , 10.725319428684] , [ 1122782400000 , 10.844849996286] , [ 1125460800000 , 10.833535488461] , [ 1128052800000 , 17.180932407865] , [ 1130734800000 , 15.894764896516] , [ 1133326800000 , 16.412751299498] , [ 1136005200000 , 12.573569093402] , [ 1138683600000 , 13.242301508051] , [ 1141102800000 , 12.863536342041] , [ 1143781200000 , 21.034044171629] , [ 1146369600000 , 21.419084618802] , [ 1149048000000 , 21.142678863692] , [ 1151640000000 , 26.56848967753] , [ 1154318400000 , 24.839144939906] , [ 1156996800000 , 25.456187462166] , [ 1159588800000 , 26.350164502825] , [ 1162270800000 , 26.478333205189] , [ 1164862800000 , 26.425979547846] , [ 1167541200000 , 28.191461582256] , [ 1170219600000 , 28.930307448808] , [ 1172638800000 , 29.521413891117] , [ 1175313600000 , 28.188285966466] , [ 1177905600000 , 27.704619625831] , [ 1180584000000 , 27.49086242483] , [ 1183176000000 , 28.770679721286] , [ 1185854400000 , 29.06048067145] , [ 1188532800000 , 28.240998844973] , [ 1191124800000 , 33.004893194128] , [ 1193803200000 , 34.075180359928] , [ 1196398800000 , 32.548560664834] , [ 1199077200000 , 30.629727432729] , [ 1201755600000 , 28.642858788159] , [ 1204261200000 , 27.973575227843] , [ 1206936000000 , 27.393351882726] , [ 1209528000000 , 28.476095288522] , [ 1212206400000 , 29.29667866426] , [ 1214798400000 , 29.222333802896] , [ 1217476800000 , 28.092966093842] , [ 1220155200000 , 28.107159262922] , [ 1222747200000 , 25.482974832099] , [ 1225425600000 , 21.208115993834] , [ 1228021200000 , 20.295043095268] , [ 1230699600000 , 15.925754618402] , [ 1233378000000 , 17.162864628346] , [ 1235797200000 , 17.084345773174] , [ 1238472000000 , 22.24600710228] , [ 1241064000000 , 24.530543998508] , [ 1243742400000 , 25.084184918241] , [ 1246334400000 , 16.606166527359] , [ 1249012800000 , 17.239620011628] , [ 1251691200000 , 17.336739127379] , [ 1254283200000 , 25.478492475754] , [ 1256961600000 , 23.017152085244] , [ 1259557200000 , 25.617745423684] , [ 1262235600000 , 24.061133998641] , [ 1264914000000 , 23.223933318646] , [ 1267333200000 , 24.425887263936] , [ 1270008000000 , 35.501471156693] , [ 1272600000000 , 33.775013878675] , [ 1275278400000 , 30.417993630285] , [ 1277870400000 , 30.023598978467] , [ 1280548800000 , 33.327519522436] , [ 1283227200000 , 31.963388450372] , [ 1285819200000 , 30.49896723209] , [ 1288497600000 , 32.403696817913] , [ 1291093200000 , 31.47736071922] , [ 1293771600000 , 31.53259666241] , [ 1296450000000 , 41.760282761548] , [ 1298869200000 , 45.605771243237] , [ 1301544000000 , 39.986557966215] , [ 1304136000000 , 43.84633051005] , [ 1306814400000 , 39.857316881858] , [ 1309406400000 , 37.675127768207] , [ 1312084800000 , 35.775077970313] , [ 1314763200000 , 48.631009702578] , [ 1317355200000 , 42.830831754505] , [ 1320033600000 , 35.611502589362] , [ 1322629200000 , 35.320136981738] , [ 1325307600000 , 31.564136901516] , [ 1327986000000 , 32.074407502433] , [ 1330491600000 , 35.053013769977] , [ 1333166400000 , 33.873085184128] , [ 1335758400000 , 32.321039427046]]

} ,

{

"key" : "Australia" ,

"values" : [ [ 1025409600000 , 5.1162447683392] , [ 1028088000000 , 4.2022848306513] , [ 1030766400000 , 4.3543715758736] , [ 1033358400000 , 5.4641223667245] , [ 1036040400000 , 6.0041275884577] , [ 1038632400000 , 6.6050520064486] , [ 1041310800000 , 5.0154059912793] , [ 1043989200000 , 5.1835708554647] , [ 1046408400000 , 5.1142682006164] , [ 1049086800000 , 5.0271381717695] , [ 1051675200000 , 5.3437782653456] , [ 1054353600000 , 5.2105844515767] , [ 1056945600000 , 6.552565997799] , [ 1059624000000 , 6.9873363581831] , [ 1062302400000 , 7.010986789097] , [ 1064894400000 , 4.4254242025515] , [ 1067576400000 , 4.9613848042174] , [ 1070168400000 , 4.8854920484764] , [ 1072846800000 , 4.0441111794228] , [ 1075525200000 , 4.0219596813179] , [ 1078030800000 , 4.3065749225355] , [ 1080709200000 , 3.9148434915404] , [ 1083297600000 , 3.8659430654512] , [ 1085976000000 , 3.9572824600686] , [ 1088568000000 , 4.7372190641522] , [ 1091246400000 , 4.6871476374455] , [ 1093924800000 , 5.0398702564196] , [ 1096516800000 , 5.5221787544964] , [ 1099195200000 , 5.424646299798] , [ 1101790800000 , 5.9240223067349] , [ 1104469200000 , 5.9936860983601] , [ 1107147600000 , 5.8499523215019] , [ 1109566800000 , 6.4149040329325] , [ 1112245200000 , 6.4547895561969] , [ 1114833600000 , 5.9385382611161] , [ 1117512000000 , 6.0486751030592] , [ 1120104000000 , 5.23108613838] , [ 1122782400000 , 5.5857797121029] , [ 1125460800000 , 5.3454665096987] , [ 1128052800000 , 5.0439154120119] , [ 1130734800000 , 5.054634702913] , [ 1133326800000 , 5.3819451380848] , [ 1136005200000 , 5.2638869269803] , [ 1138683600000 , 5.5806167415681] , [ 1141102800000 , 5.4539047069985] , [ 1143781200000 , 7.6728842432362] , [ 1146369600000 , 7.719946716654] , [ 1149048000000 , 8.0144619912942] , [ 1151640000000 , 7.942223133434] , [ 1154318400000 , 8.3998279827444] , [ 1156996800000 , 8.532324572605] , [ 1159588800000 , 4.7324285199763] , [ 1162270800000 , 4.7402397487697] , [ 1164862800000 , 4.9042069355168] , [ 1167541200000 , 5.9583963430882] , [ 1170219600000 , 6.3693899239171] , [ 1172638800000 , 6.261153903813] , [ 1175313600000 , 5.3443942184584] , [ 1177905600000 , 5.4932111235361] , [ 1180584000000 , 5.5747393101109] , [ 1183176000000 , 5.3833633060013] , [ 1185854400000 , 5.5125898831832] , [ 1188532800000 , 5.8116112661327] , [ 1191124800000 , 4.3962296939996] , [ 1193803200000 , 4.6967663605521] , [ 1196398800000 , 4.7963004350914] , [ 1199077200000 , 4.1817985183351] , [ 1201755600000 , 4.3797643870182] , [ 1204261200000 , 4.6966642197965] , [ 1206936000000 , 4.3609995132565] , [ 1209528000000 , 4.4736290996496] , [ 1212206400000 , 4.3749762738128] , [ 1214798400000 , 3.3274661194507] , [ 1217476800000 , 3.0316184691337] , [ 1220155200000 , 2.5718140204728] , [ 1222747200000 , 2.7034994044603] , [ 1225425600000 , 2.2033786591364] , [ 1228021200000 , 1.9850621240805] , [ 1230699600000 , 0] , [ 1233378000000 , 0] , [ 1235797200000 , 0] , [ 1238472000000 , 0] , [ 1241064000000 , 0] , [ 1243742400000 , 0] , [ 1246334400000 , 0] , [ 1249012800000 , 0] , [ 1251691200000 , 0] , [ 1254283200000 , 0.44495950017788] , [ 1256961600000 , 0.33945469262483] , [ 1259557200000 , 0.38348269455195] , [ 1262235600000 , 0] , [ 1264914000000 , 0] , [ 1267333200000 , 0] , [ 1270008000000 , 0] , [ 1272600000000 , 0] , [ 1275278400000 , 0] , [ 1277870400000 , 0] , [ 1280548800000 , 0] , [ 1283227200000 , 0] , [ 1285819200000 , 0] , [ 1288497600000 , 0] , [ 1291093200000 , 0] , [ 1293771600000 , 0] , [ 1296450000000 , 0.52216435716176] , [ 1298869200000 , 0.59275786698454] , [ 1301544000000 , 0] , [ 1304136000000 , 0] , [ 1306814400000 , 0] , [ 1309406400000 , 0] , [ 1312084800000 , 0] , [ 1314763200000 , 0] , [ 1317355200000 , 0] , [ 1320033600000 , 0] , [ 1322629200000 , 0] , [ 1325307600000 , 0] , [ 1327986000000 , 0] , [ 1330491600000 , 0] , [ 1333166400000 , 0] , [ 1335758400000 , 0]]

} ,

{

"key" : "Antarctica" ,

"values" : [ [ 1025409600000 , 1.3503144674343] , [ 1028088000000 , 1.2232741112434] , [ 1030766400000 , 1.3930470790784] , [ 1033358400000 , 1.2631275030593] , [ 1036040400000 , 1.5842699103708] , [ 1038632400000 , 1.9546996043116] , [ 1041310800000 , 0.8504048300986] , [ 1043989200000 , 0.85340686311353] , [ 1046408400000 , 0.843061357391] , [ 1049086800000 , 2.119846992476] , [ 1051675200000 , 2.5285382124858] , [ 1054353600000 , 2.5056570712835] , [ 1056945600000 , 2.5212789901005] , [ 1059624000000 , 2.6192011642534] , [ 1062302400000 , 2.5382187823805] , [ 1064894400000 , 2.3393223047168] , [ 1067576400000 , 2.491219888698] , [ 1070168400000 , 2.497555874906] , [ 1072846800000 , 1.734018115546] , [ 1075525200000 , 1.9307268299646] , [ 1078030800000 , 2.2261679836799] , [ 1080709200000 , 1.7608893704206] , [ 1083297600000 , 1.6242690616808] , [ 1085976000000 , 1.7161663801295] , [ 1088568000000 , 1.7183554537038] , [ 1091246400000 , 1.7179780759145] , [ 1093924800000 , 1.7314274801784] , [ 1096516800000 , 1.2596883356752] , [ 1099195200000 , 1.381177053009] , [ 1101790800000 , 1.4408819615814] , [ 1104469200000 , 3.4743581836444] , [ 1107147600000 , 3.3603749903192] , [ 1109566800000 , 3.5350883257893] , [ 1112245200000 , 3.0949644237828] , [ 1114833600000 , 3.0796455899995] , [ 1117512000000 , 3.3441247640644] , [ 1120104000000 , 4.0947643978168] , [ 1122782400000 , 4.4072631274052] , [ 1125460800000 , 4.4870979780825] , [ 1128052800000 , 4.8404549457934] , [ 1130734800000 , 4.8293016233697] , [ 1133326800000 , 5.2238093263952] , [ 1136005200000 , 3.382306337815] , [ 1138683600000 , 3.7056975170243] , [ 1141102800000 , 3.7561118692318] , [ 1143781200000 , 2.861913700854] , [ 1146369600000 , 2.9933744103381] , [ 1149048000000 , 2.7127537218463] , [ 1151640000000 , 3.1195497076283] , [ 1154318400000 , 3.4066964004508] , [ 1156996800000 , 3.3754571113569] , [ 1159588800000 , 2.2965579982924] , [ 1162270800000 , 2.4486818633018] , [ 1164862800000 , 2.4002308848517] , [ 1167541200000 , 1.9649579750349] , [ 1170219600000 , 1.9385263638056] , [ 1172638800000 , 1.9128975336387] , [ 1175313600000 , 2.3412869836298] , [ 1177905600000 , 2.4337870351445] , [ 1180584000000 , 2.62179703171] , [ 1183176000000 , 3.2642864957929] , [ 1185854400000 , 3.3200396223709] , [ 1188532800000 , 3.3934212707572] , [ 1191124800000 , 4.2822327088179] , [ 1193803200000 , 4.1474964228541] , [ 1196398800000 , 4.1477082879801] , [ 1199077200000 , 5.2947122916128] , [ 1201755600000 , 5.2919843508028] , [ 1204261200000 , 5.198978305031] , [ 1206936000000 , 3.5603057673513] , [ 1209528000000 , 3.3009087690692] , [ 1212206400000 , 3.1784852603792] , [ 1214798400000 , 4.5889503538868] , [ 1217476800000 , 4.401779617494] , [ 1220155200000 , 4.2208301828278] , [ 1222747200000 , 3.89396671475] , [ 1225425600000 , 3.0423832241354] , [ 1228021200000 , 3.135520611578] , [ 1230699600000 , 1.9631418164089] , [ 1233378000000 , 1.8963543874958] , [ 1235797200000 , 1.8266636017025] , [ 1238472000000 , 0.93136635895188] , [ 1241064000000 , 0.92737801918888] , [ 1243742400000 , 0.97591889805002] , [ 1246334400000 , 2.6841193805515] , [ 1249012800000 , 2.5664341140531] , [ 1251691200000 , 2.3887523699873] , [ 1254283200000 , 1.1737801663681] , [ 1256961600000 , 1.0953582317281] , [ 1259557200000 , 1.2495674976653] , [ 1262235600000 , 0.36607452464754] , [ 1264914000000 , 0.3548719047291] , [ 1267333200000 , 0.36769242398939] , [ 1270008000000 , 0] , [ 1272600000000 , 0] , [ 1275278400000 , 0] , [ 1277870400000 , 0] , [ 1280548800000 , 0] , [ 1283227200000 , 0] , [ 1285819200000 , 0.85450741275337] , [ 1288497600000 , 0.91360317921637] , [ 1291093200000 , 0.89647678692269] , [ 1293771600000 , 0.87800687192639] , [ 1296450000000 , 0] , [ 1298869200000 , 0] , [ 1301544000000 , 0.43668720882994] , [ 1304136000000 , 0.4756523602692] , [ 1306814400000 , 0.46947368328469] , [ 1309406400000 , 0.45138896152316] , [ 1312084800000 , 0.43828726648117] , [ 1314763200000 , 2.0820861395316] , [ 1317355200000 , 0.9364411075395] , [ 1320033600000 , 0.60583907839773] , [ 1322629200000 , 0.61096950747437] , [ 1325307600000 , 0] , [ 1327986000000 , 0] , [ 1330491600000 , 0] , [ 1333166400000 , 0] , [ 1335758400000 , 0]]

}

]

})

.controller('GroupDetailCtrl', function($scope, $stateParams) {

})

.controller('SettingCtrl', function($scope, $http, $rootScope, $location,$ionicModal,$ionicLoading,$ionicNavBarDelegate,

CONFIG\_ENV,$log,$cordovaToast) {

$scope.settings = {};

$scope.settings.stompInterval = 5;

//Websocket/Stomp testing:

var client = Stomp.client( CONFIG\_ENV.stomp\_uri, CONFIG\_ENV.stomp\_protocol );

client.connect( "", "",

function() {

client.subscribe("jms.topic.test",

function( message ) {

$log.debug( message );

$cordovaToast.show('Here is a message!', 'long', 'center').then(function(success) {});

},

{ priority: 9 }

);

client.send("jms.topic.test", { priority: 9 }, "Pub/Sub over STOMP!");

}

);

})

;

angular.module('starter.services', [])

//PersonService

.factory('PersonService', function($resource,CONFIG\_ENV) {

var data = $resource(CONFIG\_ENV.api\_endpoint+'person/create');

return data;

})

/\* angular-moment.js / v0.8.2 / (c) 2013, 2014 Uri Shaked / MIT Licence \*/

'format global'; /\* global define \*/

'deps angular';

'deps moment';

(function () {

'use strict';

function angularMoment(angular, moment) {

/\*\*

\* @ngdoc overview

\* @name angularMoment

\*

\* @description

\* angularMoment module provides moment.js functionality for angular.js apps.

\*/

return angular.module('angularMoment', [])

/\*\*

\* @ngdoc object

\* @name angularMoment.config:angularMomentConfig

\*

\* @description

\* Common configuration of the angularMoment module

\*/

.constant('angularMomentConfig', {

/\*\*

\* @ngdoc property

\* @name angularMoment.config.angularMomentConfig#preprocess

\* @propertyOf angularMoment.config:angularMomentConfig

\* @returns {string} The default preprocessor to apply

\*

\* @description

\* Defines a default preprocessor to apply (e.g. 'unix', 'etc', ...). The default value is null,

\* i.e. no preprocessor will be applied.

\*/

preprocess: null, // e.g. 'unix', 'utc', ...

/\*\*

\* @ngdoc property

\* @name angularMoment.config.angularMomentConfig#timezone

\* @propertyOf angularMoment.config:angularMomentConfig

\* @returns {string} The default timezone

\*

\* @description

\* The default timezone (e.g. 'Europe/London'). Empty string by default (does not apply

\* any timezone shift).

\*/

timezone: '',

/\*\*

\* @ngdoc property

\* @name angularMoment.config.angularMomentConfig#format

\* @propertyOf angularMoment.config:angularMomentConfig

\* @returns {string} The pre-conversion format of the date

\*

\* @description

\* Specify the format of the input date. Essentially it's a

\* default and saves you from specifying a format in every

\* element. Overridden by element attr. Null by default.

\*/

format: null

})

/\*\*

\* @ngdoc object

\* @name angularMoment.object:moment

\*

\* @description

\* moment global (as provided by the moment.js library)

\*/

.constant('moment', moment)

/\*\*

\* @ngdoc object

\* @name angularMoment.config:amTimeAgoConfig

\* @module angularMoment

\*

\* @description

\* configuration specific to the amTimeAgo directive

\*/

.constant('amTimeAgoConfig', {

/\*\*

\* @ngdoc property

\* @name angularMoment.config.amTimeAgoConfig#withoutSuffix

\* @propertyOf angularMoment.config:amTimeAgoConfig

\* @returns {boolean} Whether to include a suffix in am-time-ago directive

\*

\* @description

\* Defaults to false.

\*/

withoutSuffix: false,

/\*\*

\* @ngdoc property

\* @name angularMoment.config.amTimeAgoConfig#serverTime

\* @propertyOf angularMoment.config:amTimeAgoConfig

\* @returns {number} Server time in milliseconds since the epoch

\*

\* @description

\* If set, time ago will be calculated relative to the given value.

\* If null, local time will be used. Defaults to null.

\*/

serverTime: null,

/\*\*

\* @ngdoc property

\* @name angularMoment.config.amTimeAgoConfig#format

\* @propertyOf angularMoment.config:amTimeAgoConfig

\* @returns {string} The format of the date to be displayed in the title of the element. If null,

\* the directive set the title of the element.

\*

\* @description

\* Specify the format of the date when displayed. null by default.

\*/

titleFormat: null

})

/\*\*

\* @ngdoc directive

\* @name angularMoment.directive:amTimeAgo

\* @module angularMoment

\*

\* @restrict A

\*/

.directive('amTimeAgo', ['$window', 'moment', 'amMoment', 'amTimeAgoConfig', 'angularMomentConfig', function ($window, moment, amMoment, amTimeAgoConfig, angularMomentConfig) {

return function (scope, element, attr) {

var activeTimeout = null;

var currentValue;

var currentFormat = angularMomentConfig.format;

var withoutSuffix = amTimeAgoConfig.withoutSuffix;

var titleFormat = amTimeAgoConfig.titleFormat;

var localDate = new Date().getTime();

var preprocess = angularMomentConfig.preprocess;

var modelName = attr.amTimeAgo.replace(/^::/, '');

var isBindOnce = (attr.amTimeAgo.indexOf('::') === 0);

var isTimeElement = ('TIME' === element[0].nodeName.toUpperCase());

var unwatchChanges;

function getNow() {

var now;

if (amTimeAgoConfig.serverTime) {

var localNow = new Date().getTime();

var nowMillis = localNow - localDate + amTimeAgoConfig.serverTime;

now = moment(nowMillis);

}

else {

now = moment();

}

return now;

}

function cancelTimer() {

if (activeTimeout) {

$window.clearTimeout(activeTimeout);

activeTimeout = null;

}

}

function updateTime(momentInstance) {

element.text(momentInstance.from(getNow(), withoutSuffix));

if (titleFormat && !element.attr('title')) {

element.attr('title', momentInstance.local().format(titleFormat));

}

if (!isBindOnce) {

var howOld = Math.abs(getNow().diff(momentInstance, 'minute'));

var secondsUntilUpdate = 3600;

if (howOld < 1) {

secondsUntilUpdate = 1;

} else if (howOld < 60) {

secondsUntilUpdate = 30;

} else if (howOld < 180) {

secondsUntilUpdate = 300;

}

activeTimeout = $window.setTimeout(function () {

updateTime(momentInstance);

}, secondsUntilUpdate \* 1000);

}

}

function updateDateTimeAttr(value) {

if (isTimeElement) {

element.attr('datetime', value);

}

}

function updateMoment() {

cancelTimer();

if (currentValue) {

var momentValue = amMoment.preprocessDate(currentValue, preprocess, currentFormat);

updateTime(momentValue);

updateDateTimeAttr(momentValue.toISOString());

}

}

unwatchChanges = scope.$watch(modelName, function (value) {

if ((typeof value === 'undefined') || (value === null) || (value === '')) {

cancelTimer();

if (currentValue) {

element.text('');

updateDateTimeAttr('');

currentValue = null;

}

return;

}

currentValue = value;

updateMoment();

if (value !== undefined && isBindOnce) {

unwatchChanges();

}

});

if (angular.isDefined(attr.amWithoutSuffix)) {

scope.$watch(attr.amWithoutSuffix, function (value) {

if (typeof value === 'boolean') {

withoutSuffix = value;

updateMoment();

} else {

withoutSuffix = amTimeAgoConfig.withoutSuffix;

}

});

}

attr.$observe('amFormat', function (format) {

if (typeof format !== 'undefined') {

currentFormat = format;

updateMoment();

}

});

attr.$observe('amPreprocess', function (newValue) {

preprocess = newValue;

updateMoment();

});

scope.$on('$destroy', function () {

cancelTimer();

});

scope.$on('amMoment:localeChanged', function () {

updateMoment();

});

};

}])

/\*\*

\* @ngdoc service

\* @name angularMoment.service.amMoment

\* @module angularMoment

\*/

.service('amMoment', ['moment', '$rootScope', '$log', 'angularMomentConfig', function (moment, $rootScope, $log, angularMomentConfig) {

var that = this;

/\*\*

\* @ngdoc property

\* @name angularMoment:amMoment#preprocessors

\* @module angularMoment

\*

\* @description

\* Defines the preprocessors for the preprocessDate method. By default, the following preprocessors

\* are defined: utc, unix.

\*/

this.preprocessors = {

utc: moment.utc,

unix: moment.unix

};

/\*\*

\* @ngdoc function

\* @name angularMoment.service.amMoment#changeLocale

\* @methodOf angularMoment.service.amMoment

\*

\* @description

\* Changes the locale for moment.js and updates all the am-time-ago directive instances

\* with the new locale. Also broadcasts a `amMoment:localeChanged` event on $rootScope.

\*

\* @param {string} locale 2-letter language code (e.g. en, es, ru, etc.)

\*/

this.changeLocale = function (locale) {

var result = (moment.locale||moment.lang)(locale);

if (angular.isDefined(locale)) {

$rootScope.$broadcast('amMoment:localeChanged');

// The following event is deprecated and will be removed in an upcoming

// major release.

$rootScope.$broadcast('amMoment:languageChange');

}

return result;

};

/\*\*

\* @ngdoc function

\* @name angularMoment.service.amMoment#changeLanguage

\* @methodOf angularMoment.service.amMoment

\* @deprecated Please use changeLocale() instead.

\*

\* @description

\* Deprecated. Please use changeLocale() instead.

\*/

this.changeLanguage = function (lang) {

$log.warn('angular-moment: Usage of amMoment.changeLanguage() is deprecated. Please use changeLocale()');

return that.changeLocale(lang);

};

/\*\*

\* @ngdoc function

\* @name angularMoment.service.amMoment#preprocessDate

\* @methodOf angularMoment.service.amMoment

\*

\* @description

\* Preprocess a given value and convert it into a Moment instance appropriate for use in the

\* am-time-ago directive and the filters.

\*

\* @param {\*} value The value to be preprocessed

\* @param {string} preprocess The name of the preprocessor the apply (e.g. utc, unix)

\* @param {string=} format Specifies how to parse the value (see {@link http://momentjs.com/docs/#/parsing/string-format/})

\* @return {Moment} A value that can be parsed by the moment library

\*/

this.preprocessDate = function (value, preprocess, format) {

if (angular.isUndefined(preprocess)) {

preprocess = angularMomentConfig.preprocess;

}

if (this.preprocessors[preprocess]) {

return this.preprocessors[preprocess](value, format);

}

if (preprocess) {

$log.warn('angular-moment: Ignoring unsupported value for preprocess: ' + preprocess);

}

if (!isNaN(parseFloat(value)) && isFinite(value)) {

// Milliseconds since the epoch

return moment(parseInt(value, 10));

}

// else just returns the value as-is.

return moment(value, format);

};

/\*\*

\* @ngdoc function

\* @name angularMoment.service.amMoment#applyTimezone

\* @methodOf angularMoment.service.amMoment

\*

\* @description

\* Apply a timezone onto a given moment object - if moment-timezone.js is included

\* Otherwise, it'll not apply any timezone shift.

\*

\* @param {Moment} aMoment a moment() instance to apply the timezone shift to

\* @returns {Moment} The given moment with the timezone shift applied

\*/

this.applyTimezone = function (aMoment) {

var timezone = angularMomentConfig.timezone;

if (aMoment && timezone) {

if (aMoment.tz) {

aMoment = aMoment.tz(timezone);

} else {

$log.warn('angular-moment: timezone specified but moment.tz() is undefined. Did you forget to include moment-timezone.js?');

}

}

return aMoment;

};

}])

/\*\*

\* @ngdoc filter

\* @name angularMoment.filter:amCalendar

\* @module angularMoment

\*/

.filter('amCalendar', ['moment', 'amMoment', function (moment, amMoment) {

return function (value, preprocess) {

if (typeof value === 'undefined' || value === null) {

return '';

}

value = amMoment.preprocessDate(value, preprocess);

var date = moment(value);

if (!date.isValid()) {

return '';

}

return amMoment.applyTimezone(date).calendar();

};

}])

/\*\*

\* @ngdoc filter

\* @name angularMoment.filter:amDateFormat

\* @module angularMoment

\* @function

\*/

.filter('amDateFormat', ['moment', 'amMoment', function (moment, amMoment) {

return function (value, format, preprocess) {

if (typeof value === 'undefined' || value === null) {

return '';

}

value = amMoment.preprocessDate(value, preprocess);

var date = moment(value);

if (!date.isValid()) {

return '';

}

return amMoment.applyTimezone(date).format(format);

};

}])

/\*\*

\* @ngdoc filter

\* @name angularMoment.filter:amDurationFormat

\* @module angularMoment

\* @function

\*/

.filter('amDurationFormat', ['moment', function (moment) {

return function (value, format, suffix) {

if (typeof value === 'undefined' || value === null) {

return '';

}

return moment.duration(value, format).humanize(suffix);

};

}]);

}

if (typeof define === 'function' && define.amd) {

define('angular-moment', ['angular', 'moment'], angularMoment);

} else {

angularMoment(angular, window.moment);

}

})();

/\*\*

\* @license AngularJS v1.3.0-rc.5

\* (c) 2010-2014 Google, Inc. http://angularjs.org

\* License: MIT

\*/

(function(window, angular, undefined) {'use strict';

var $resourceMinErr = angular.$$minErr('$resource');

// Helper functions and regex to lookup a dotted path on an object

// stopping at undefined/null. The path must be composed of ASCII

// identifiers (just like $parse)

var MEMBER\_NAME\_REGEX = /^(\.[a-zA-Z\_$][0-9a-zA-Z\_$]\*)+$/;

function isValidDottedPath(path) {

return (path != null && path !== '' && path !== 'hasOwnProperty' &&

MEMBER\_NAME\_REGEX.test('.' + path));

}

function lookupDottedPath(obj, path) {

if (!isValidDottedPath(path)) {

throw $resourceMinErr('badmember', 'Dotted member path "@{0}" is invalid.', path);

}

var keys = path.split('.');

for (var i = 0, ii = keys.length; i < ii && obj !== undefined; i++) {

var key = keys[i];

obj = (obj !== null) ? obj[key] : undefined;

}

return obj;

}

/\*\*

\* Create a shallow copy of an object and clear other fields from the destination

\*/

function shallowClearAndCopy(src, dst) {

dst = dst || {};

angular.forEach(dst, function(value, key){

delete dst[key];

});

for (var key in src) {

if (src.hasOwnProperty(key) && !(key.charAt(0) === '$' && key.charAt(1) === '$')) {

dst[key] = src[key];

}

}

return dst;

}

/\*\*

\* @ngdoc module

\* @name ngResource

\* @description

\*

\* # ngResource

\*

\* The `ngResource` module provides interaction support with RESTful services

\* via the $resource service.

\*

\*

\* <div doc-module-components="ngResource"></div>

\*

\* See {@link ngResource.$resource `$resource`} for usage.

\*/

/\*\*

\* @ngdoc service

\* @name $resource

\* @requires $http

\*

\* @description

\* A factory which creates a resource object that lets you interact with

\* [RESTful](http://en.wikipedia.org/wiki/Representational\_State\_Transfer) server-side data sources.

\*

\* The returned resource object has action methods which provide high-level behaviors without

\* the need to interact with the low level {@link ng.$http $http} service.

\*

\* Requires the {@link ngResource `ngResource`} module to be installed.

\*

\* By default, trailing slashes will be stripped from the calculated URLs,

\* which can pose problems with server backends that do not expect that

\* behavior. This can be disabled by configuring the `$resourceProvider` like

\* this:

\*

\* ```js

app.config(['$resourceProvider', function ($resourceProvider) {

// Don't strip trailing slashes from calculated URLs

$resourceProvider.defaults.stripTrailingSlashes = false;

}]);

\* ```

\*

\* @param {string} url A parametrized URL template with parameters prefixed by `:` as in

\* `/user/:username`. If you are using a URL with a port number (e.g.

\* `http://example.com:8080/api`), it will be respected.

\*

\* If you are using a url with a suffix, just add the suffix, like this:

\* `$resource('http://example.com/resource.json')` or `$resource('http://example.com/:id.json')`

\* or even `$resource('http://example.com/resource/:resource\_id.:format')`

\* If the parameter before the suffix is empty, :resource\_id in this case, then the `/.` will be

\* collapsed down to a single `.`. If you need this sequence to appear and not collapse then you

\* can escape it with `/\.`.

\*

\* @param {Object=} paramDefaults Default values for `url` parameters. These can be overridden in

\* `actions` methods. If any of the parameter value is a function, it will be executed every time

\* when a param value needs to be obtained for a request (unless the param was overridden).

\*

\* Each key value in the parameter object is first bound to url template if present and then any

\* excess keys are appended to the url search query after the `?`.

\*

\* Given a template `/path/:verb` and parameter `{verb:'greet', salutation:'Hello'}` results in

\* URL `/path/greet?salutation=Hello`.

\*

\* If the parameter value is prefixed with `@` then the value for that parameter will be extracted

\* from the corresponding property on the `data` object (provided when calling an action method). For

\* example, if the `defaultParam` object is `{someParam: '@someProp'}` then the value of `someParam`

\* will be `data.someProp`.

\*

\* @param {Object.<Object>=} actions Hash with declaration of custom action that should extend

\* the default set of resource actions. The declaration should be created in the format of {@link

\* ng.$http#usage\_parameters $http.config}:

\*

\* {action1: {method:?, params:?, isArray:?, headers:?, ...},

\* action2: {method:?, params:?, isArray:?, headers:?, ...},

\* ...}

\*

\* Where:

\*

\* - \*\*`action`\*\* 鈥� {string} 鈥� The name of action. This name becomes the name of the method on

\* your resource object.

\* - \*\*`method`\*\* 鈥� {string} 鈥� Case insensitive HTTP method (e.g. `GET`, `POST`, `PUT`,

\* `DELETE`, `JSONP`, etc).

\* - \*\*`params`\*\* 鈥� {Object=} 鈥� Optional set of pre-bound parameters for this action. If any of

\* the parameter value is a function, it will be executed every time when a param value needs to

\* be obtained for a request (unless the param was overridden).

\* - \*\*`url`\*\* 鈥� {string} 鈥� action specific `url` override. The url templating is supported just

\* like for the resource-level urls.

\* - \*\*`isArray`\*\* 鈥� {boolean=} 鈥� If true then the returned object for this action is an array,

\* see `returns` section.

\* - \*\*`transformRequest`\*\* 鈥�

\* `{function(data, headersGetter)|Array.<function(data, headersGetter)>}` 鈥�

\* transform function or an array of such functions. The transform function takes the http

\* request body and headers and returns its transformed (typically serialized) version.

\* By default, transformRequest will contain one function that checks if the request data is

\* an object and serializes to using `angular.toJson`. To prevent this behavior, set

\* `transformRequest` to an empty array: `transformRequest: []`

\* - \*\*`transformResponse`\*\* 鈥�

\* `{function(data, headersGetter)|Array.<function(data, headersGetter)>}` 鈥�

\* transform function or an array of such functions. The transform function takes the http

\* response body and headers and returns its transformed (typically deserialized) version.

\* By default, transformResponse will contain one function that checks if the response looks like

\* a JSON string and deserializes it using `angular.fromJson`. To prevent this behavior, set

\* `transformResponse` to an empty array: `transformResponse: []`

\* - \*\*`cache`\*\* 鈥� `{boolean|Cache}` 鈥� If true, a default $http cache will be used to cache the

\* GET request, otherwise if a cache instance built with

\* {@link ng.$cacheFactory $cacheFactory}, this cache will be used for

\* caching.

\* - \*\*`timeout`\*\* 鈥� `{number|Promise}` 鈥� timeout in milliseconds, or {@link ng.$q promise} that

\* should abort the request when resolved.

\* - \*\*`withCredentials`\*\* - `{boolean}` - whether to set the `withCredentials` flag on the

\* XHR object. See

\* [requests with credentials](https://developer.mozilla.org/en/http\_access\_control#section\_5)

\* for more information.

\* - \*\*`responseType`\*\* - `{string}` - see

\* [requestType](https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest#responseType).

\* - \*\*`interceptor`\*\* - `{Object=}` - The interceptor object has two optional methods -

\* `response` and `responseError`. Both `response` and `responseError` interceptors get called

\* with `http response` object. See {@link ng.$http $http interceptors}.

\*

\* @param {Object} options Hash with custom settings that should extend the

\* default `$resourceProvider` behavior. The only supported option is

\*

\* Where:

\*

\* - \*\*`stripTrailingSlashes`\*\* 鈥� {boolean} 鈥� If true then the trailing

\* slashes from any calculated URL will be stripped. (Defaults to true.)

\*

\* @returns {Object} A resource "class" object with methods for the default set of resource actions

\* optionally extended with custom `actions`. The default set contains these actions:

\* ```js

\* { 'get': {method:'GET'},

\* 'save': {method:'POST'},

\* 'query': {method:'GET', isArray:true},

\* 'remove': {method:'DELETE'},

\* 'delete': {method:'DELETE'} };

\* ```

\*

\* Calling these methods invoke an {@link ng.$http} with the specified http method,

\* destination and parameters. When the data is returned from the server then the object is an

\* instance of the resource class. The actions `save`, `remove` and `delete` are available on it

\* as methods with the `$` prefix. This allows you to easily perform CRUD operations (create,

\* read, update, delete) on server-side data like this:

\* ```js

\* var User = $resource('/user/:userId', {userId:'@id'});

\* var user = User.get({userId:123}, function() {

\* user.abc = true;

\* user.$save();

\* });

\* ```

\*

\* It is important to realize that invoking a $resource object method immediately returns an

\* empty reference (object or array depending on `isArray`). Once the data is returned from the

\* server the existing reference is populated with the actual data. This is a useful trick since

\* usually the resource is assigned to a model which is then rendered by the view. Having an empty

\* object results in no rendering, once the data arrives from the server then the object is

\* populated with the data and the view automatically re-renders itself showing the new data. This

\* means that in most cases one never has to write a callback function for the action methods.

\*

\* The action methods on the class object or instance object can be invoked with the following

\* parameters:

\*

\* - HTTP GET "class" actions: `Resource.action([parameters], [success], [error])`

\* - non-GET "class" actions: `Resource.action([parameters], postData, [success], [error])`

\* - non-GET instance actions: `instance.$action([parameters], [success], [error])`

\*

\* Success callback is called with (value, responseHeaders) arguments. Error callback is called

\* with (httpResponse) argument.

\*

\* Class actions return empty instance (with additional properties below).

\* Instance actions return promise of the action.

\*

\* The Resource instances and collection have these additional properties:

\*

\* - `$promise`: the {@link ng.$q promise} of the original server interaction that created this

\* instance or collection.

\*

\* On success, the promise is resolved with the same resource instance or collection object,

\* updated with data from server. This makes it easy to use in

\* {@link ngRoute.$routeProvider resolve section of $routeProvider.when()} to defer view

\* rendering until the resource(s) are loaded.

\*

\* On failure, the promise is resolved with the {@link ng.$http http response} object, without

\* the `resource` property.

\*

\* If an interceptor object was provided, the promise will instead be resolved with the value

\* returned by the interceptor.

\*

\* - `$resolved`: `true` after first server interaction is completed (either with success or

\* rejection), `false` before that. Knowing if the Resource has been resolved is useful in

\* data-binding.

\*

\* @example

\*

\* # Credit card resource

\*

\* ```js

// Define CreditCard class

var CreditCard = $resource('/user/:userId/card/:cardId',

{userId:123, cardId:'@id'}, {

charge: {method:'POST', params:{charge:true}}

});

// We can retrieve a collection from the server

var cards = CreditCard.query(function() {

// GET: /user/123/card

// server returns: [ {id:456, number:'1234', name:'Smith'} ];

var card = cards[0];

// each item is an instance of CreditCard

expect(card instanceof CreditCard).toEqual(true);

card.name = "J. Smith";

// non GET methods are mapped onto the instances

card.$save();

// POST: /user/123/card/456 {id:456, number:'1234', name:'J. Smith'}

// server returns: {id:456, number:'1234', name: 'J. Smith'};

// our custom method is mapped as well.

card.$charge({amount:9.99});

// POST: /user/123/card/456?amount=9.99&charge=true {id:456, number:'1234', name:'J. Smith'}

});

// we can create an instance as well

var newCard = new CreditCard({number:'0123'});

newCard.name = "Mike Smith";

newCard.$save();

// POST: /user/123/card {number:'0123', name:'Mike Smith'}

// server returns: {id:789, number:'0123', name: 'Mike Smith'};

expect(newCard.id).toEqual(789);

\* ```

\*

\* The object returned from this function execution is a resource "class" which has "static" method

\* for each action in the definition.

\*

\* Calling these methods invoke `$http` on the `url` template with the given `method`, `params` and

\* `headers`.

\* When the data is returned from the server then the object is an instance of the resource type and

\* all of the non-GET methods are available with `$` prefix. This allows you to easily support CRUD

\* operations (create, read, update, delete) on server-side data.

```js

var User = $resource('/user/:userId', {userId:'@id'});

User.get({userId:123}, function(user) {

user.abc = true;

user.$save();

});

```

\*

\* It's worth noting that the success callback for `get`, `query` and other methods gets passed

\* in the response that came from the server as well as $http header getter function, so one

\* could rewrite the above example and get access to http headers as:

\*

```js

var User = $resource('/user/:userId', {userId:'@id'});

User.get({userId:123}, function(u, getResponseHeaders){

u.abc = true;

u.$save(function(u, putResponseHeaders) {

//u => saved user object

//putResponseHeaders => $http header getter

});

});

```

\*

\* You can also access the raw `$http` promise via the `$promise` property on the object returned

\*

```

var User = $resource('/user/:userId', {userId:'@id'});

User.get({userId:123})

.$promise.then(function(user) {

$scope.user = user;

});

```

\* # Creating a custom 'PUT' request

\* In this example we create a custom method on our resource to make a PUT request

\* ```js

\* var app = angular.module('app', ['ngResource', 'ngRoute']);

\*

\* // Some APIs expect a PUT request in the format URL/object/ID

\* // Here we are creating an 'update' method

\* app.factory('Notes', ['$resource', function($resource) {

\* return $resource('/notes/:id', null,

\* {

\* 'update': { method:'PUT' }

\* });

\* }]);

\*

\* // In our controller we get the ID from the URL using ngRoute and $routeParams

\* // We pass in $routeParams and our Notes factory along with $scope

\* app.controller('NotesCtrl', ['$scope', '$routeParams', 'Notes',

function($scope, $routeParams, Notes) {

\* // First get a note object from the factory

\* var note = Notes.get({ id:$routeParams.id });

\* $id = note.id;

\*

\* // Now call update passing in the ID first then the object you are updating

\* Notes.update({ id:$id }, note);

\*

\* // This will PUT /notes/ID with the note object in the request payload

\* }]);

\* ```

\*/

angular.module('ngResource', ['ng']).

provider('$resource', function () {

var provider = this;

this.defaults = {

// Strip slashes by default

stripTrailingSlashes: true,

// Default actions configuration

actions: {

'get': {method: 'GET'},

'save': {method: 'POST'},

'query': {method: 'GET', isArray: true},

'remove': {method: 'DELETE'},

'delete': {method: 'DELETE'}

}

};

this.$get = ['$http', '$q', function ($http, $q) {

var noop = angular.noop,

forEach = angular.forEach,

extend = angular.extend,

copy = angular.copy,

isFunction = angular.isFunction;

/\*\*

\* We need our custom method because encodeURIComponent is too aggressive and doesn't follow

\* http://www.ietf.org/rfc/rfc3986.txt with regards to the character set

\* (pchar) allowed in path segments:

\* segment = \*pchar

\* pchar = unreserved / pct-encoded / sub-delims / ":" / "@"

\* pct-encoded = "%" HEXDIG HEXDIG

\* unreserved = ALPHA / DIGIT / "-" / "." / "\_" / "~"

\* sub-delims = "!" / "$" / "&" / "'" / "(" / ")"

\* / "\*" / "+" / "," / ";" / "="

\*/

function encodeUriSegment(val) {

return encodeUriQuery(val, true).

replace(/%26/gi, '&').

replace(/%3D/gi, '=').

replace(/%2B/gi, '+');

}

/\*\*

\* This method is intended for encoding \*key\* or \*value\* parts of query component. We need a

\* custom method because encodeURIComponent is too aggressive and encodes stuff that doesn't

\* have to be encoded per http://tools.ietf.org/html/rfc3986:

\* query = \*( pchar / "/" / "?" )

\* pchar = unreserved / pct-encoded / sub-delims / ":" / "@"

\* unreserved = ALPHA / DIGIT / "-" / "." / "\_" / "~"

\* pct-encoded = "%" HEXDIG HEXDIG

\* sub-delims = "!" / "$" / "&" / "'" / "(" / ")"

\* / "\*" / "+" / "," / ";" / "="

\*/

function encodeUriQuery(val, pctEncodeSpaces) {

return encodeURIComponent(val).

replace(/%40/gi, '@').

replace(/%3A/gi, ':').

replace(/%24/g, '$').

replace(/%2C/gi, ',').

replace(/%20/g, (pctEncodeSpaces ? '%20' : '+'));

}

function Route(template, defaults) {

this.template = template;

this.defaults = extend({}, provider.defaults, defaults);

this.urlParams = {};

}

Route.prototype = {

setUrlParams: function (config, params, actionUrl) {

var self = this,

url = actionUrl || self.template,

val,

encodedVal;

var urlParams = self.urlParams = {};

forEach(url.split(/\W/), function (param) {

if (param === 'hasOwnProperty') {

throw $resourceMinErr('badname', "hasOwnProperty is not a valid parameter name.");

}

if (!(new RegExp("^\\d+$").test(param)) && param &&

(new RegExp("(^|[^\\\\]):" + param + "(\\W|$)").test(url))) {

urlParams[param] = true;

}

});

url = url.replace(/\\:/g, ':');

params = params || {};

forEach(self.urlParams, function (\_, urlParam) {

val = params.hasOwnProperty(urlParam) ? params[urlParam] : self.defaults[urlParam];

if (angular.isDefined(val) && val !== null) {

encodedVal = encodeUriSegment(val);

url = url.replace(new RegExp(":" + urlParam + "(\\W|$)", "g"), function (match, p1) {

return encodedVal + p1;

});

} else {

url = url.replace(new RegExp("(\/?):" + urlParam + "(\\W|$)", "g"), function (match,

leadingSlashes, tail) {

if (tail.charAt(0) == '/') {

return tail;

} else {

return leadingSlashes + tail;

}

});

}

});

// strip trailing slashes and set the url (unless this behavior is specifically disabled)

if (self.defaults.stripTrailingSlashes) {

url = url.replace(/\/+$/, '') || '/';

}

// then replace collapse `/.` if found in the last URL path segment before the query

// E.g. `http://url.com/id./format?q=x` becomes `http://url.com/id.format?q=x`

url = url.replace(/\/\.(?=\w+($|\?))/, '.');

// replace escaped `/\.` with `/.`

config.url = url.replace(/\/\\\./, '/.');

// set params - delegate param encoding to $http

forEach(params, function (value, key) {

if (!self.urlParams[key]) {

config.params = config.params || {};

config.params[key] = value;

}

});

}

};

function resourceFactory(url, paramDefaults, actions, options) {

var route = new Route(url, options);

actions = extend({}, provider.defaults.actions, actions);

function extractParams(data, actionParams) {

var ids = {};

actionParams = extend({}, paramDefaults, actionParams);

forEach(actionParams, function (value, key) {

if (isFunction(value)) { value = value(); }

ids[key] = value && value.charAt && value.charAt(0) == '@' ?

lookupDottedPath(data, value.substr(1)) : value;

});

return ids;

}

function defaultResponseInterceptor(response) {

return response.resource;

}

function Resource(value) {

shallowClearAndCopy(value || {}, this);

}

Resource.prototype.toJSON = function () {

var data = extend({}, this);

delete data.$promise;

delete data.$resolved;

return data;

};

forEach(actions, function (action, name) {

var hasBody = /^(POST|PUT|PATCH)$/i.test(action.method);

Resource[name] = function (a1, a2, a3, a4) {

var params = {}, data, success, error;

/\* jshint -W086 \*/ /\* (purposefully fall through case statements) \*/

switch (arguments.length) {

case 4:

error = a4;

success = a3;

//fallthrough

case 3:

case 2:

if (isFunction(a2)) {

if (isFunction(a1)) {

success = a1;

error = a2;

break;

}

success = a2;

error = a3;

//fallthrough

} else {

params = a1;

data = a2;

success = a3;

break;

}

case 1:

if (isFunction(a1)) success = a1;

else if (hasBody) data = a1;

else params = a1;

break;

case 0: break;

default:

throw $resourceMinErr('badargs',

"Expected up to 4 arguments [params, data, success, error], got {0} arguments",

arguments.length);

}

/\* jshint +W086 \*/ /\* (purposefully fall through case statements) \*/

var isInstanceCall = this instanceof Resource;

var value = isInstanceCall ? data : (action.isArray ? [] : new Resource(data));

var httpConfig = {};

var responseInterceptor = action.interceptor && action.interceptor.response ||

defaultResponseInterceptor;

var responseErrorInterceptor = action.interceptor && action.interceptor.responseError ||

undefined;

forEach(action, function (value, key) {

if (key != 'params' && key != 'isArray' && key != 'interceptor') {

httpConfig[key] = copy(value);

}

});

if (hasBody) httpConfig.data = data;

route.setUrlParams(httpConfig,

extend({}, extractParams(data, action.params || {}), params),

action.url);

var promise = $http(httpConfig).then(function (response) {

var data = response.data,

promise = value.$promise;

if (data) {

// Need to convert action.isArray to boolean in case it is undefined

// jshint -W018

if (angular.isArray(data) !== (!!action.isArray)) {

throw $resourceMinErr('badcfg',

'Error in resource configuration for action `{0}`. Expected response to ' +

'contain an {1} but got an {2}', name, action.isArray ? 'array' : 'object',

angular.isArray(data) ? 'array' : 'object');

}

// jshint +W018

if (action.isArray) {

value.length = 0;

forEach(data, function (item) {

if (typeof item === "object") {

value.push(new Resource(item));

} else {

// Valid JSON values may be string literals, and these should not be converted

// into objects. These items will not have access to the Resource prototype

// methods, but unfortunately there

value.push(item);

}

});

} else {

shallowClearAndCopy(data, value);

value.$promise = promise;

}

}

value.$resolved = true;

response.resource = value;

return response;

}, function (response) {

value.$resolved = true;

(error || noop)(response);

return $q.reject(response);

});

promise = promise.then(

function (response) {

var value = responseInterceptor(response);

(success || noop)(value, response.headers);

return value;

},

responseErrorInterceptor);

if (!isInstanceCall) {

// we are creating instance / collection

// - set the initial promise

// - return the instance / collection

value.$promise = promise;

value.$resolved = false;

return value;

}

// instance call

return promise;

};

Resource.prototype['$' + name] = function (params, success, error) {

if (isFunction(params)) {

error = success; success = params; params = {};

}

var result = Resource[name].call(this, params, this, success, error);

return result.$promise || result;

};

});

Resource.bind = function (additionalParamDefaults) {

return resourceFactory(url, extend({}, paramDefaults, additionalParamDefaults), actions);

};

return Resource;

}

return resourceFactory;

}];

});

})(window, window.angular);

/\*\*

\* @license AngularJS v1.2.16

\* (c) 2010-2014 Google, Inc. http://angularjs.org

\* License: MIT

\*/

(function(window, angular, undefined) {'use strict';

/\*\*

\* @ngdoc module

\* @name ngRoute

\* @description

\*

\* # ngRoute

\*

\* The `ngRoute` module provides routing and deeplinking services and directives for angular apps.

\*

\* ## Example

\* See {@link ngRoute.$route#example $route} for an example of configuring and using `ngRoute`.

\*

\*

\* <div doc-module-components="ngRoute"></div>

\*/

/\* global -ngRouteModule \*/

var ngRouteModule = angular.module('ngRoute', ['ng']).

provider('$route', $RouteProvider);

/\*\*

\* @ngdoc provider

\* @name $routeProvider

\* @function

\*

\* @description

\*

\* Used for configuring routes.

\*

\* ## Example

\* See {@link ngRoute.$route#example $route} for an example of configuring and using `ngRoute`.

\*

\* ## Dependencies

\* Requires the {@link ngRoute `ngRoute`} module to be installed.

\*/

function $RouteProvider(){

function inherit(parent, extra) {

return angular.extend(new (angular.extend(function() {}, {prototype:parent}))(), extra);

}

var routes = {};

/\*\*

\* @ngdoc method

\* @name $routeProvider#when

\*

\* @param {string} path Route path (matched against `$location.path`). If `$location.path`

\* contains redundant trailing slash or is missing one, the route will still match and the

\* `$location.path` will be updated to add or drop the trailing slash to exactly match the

\* route definition.

\*

\* \* `path` can contain named groups starting with a colon: e.g. `:name`. All characters up

\* to the next slash are matched and stored in `$routeParams` under the given `name`

\* when the route matches.

\* \* `path` can contain named groups starting with a colon and ending with a star:

\* e.g.`:name\*`. All characters are eagerly stored in `$routeParams` under the given `name`

\* when the route matches.

\* \* `path` can contain optional named groups with a question mark: e.g.`:name?`.

\*

\* For example, routes like `/color/:color/largecode/:largecode\*\/edit` will match

\* `/color/brown/largecode/code/with/slashes/edit` and extract:

\*

\* \* `color: brown`

\* \* `largecode: code/with/slashes`.

\*

\*

\* @param {Object} route Mapping information to be assigned to `$route.current` on route

\* match.

\*

\* Object properties:

\*

\* - `controller` – `{(string|function()=}` – Controller fn that should be associated with

\* newly created scope or the name of a {@link angular.Module#controller registered

\* controller} if passed as a string.

\* - `controllerAs` – `{string=}` – A controller alias name. If present the controller will be

\* published to scope under the `controllerAs` name.

\* - `template` – `{string=|function()=}` – html template as a string or a function that

\* returns an html template as a string which should be used by {@link

\* ngRoute.directive:ngView ngView} or {@link ng.directive:ngInclude ngInclude} directives.

\* This property takes precedence over `templateUrl`.

\*

\* If `template` is a function, it will be called with the following parameters:

\*

\* - `{Array.<Object>}` - route parameters extracted from the current

\* `$location.path()` by applying the current route

\*

\* - `templateUrl` – `{string=|function()=}` – path or function that returns a path to an html

\* template that should be used by {@link ngRoute.directive:ngView ngView}.

\*

\* If `templateUrl` is a function, it will be called with the following parameters:

\*

\* - `{Array.<Object>}` - route parameters extracted from the current

\* `$location.path()` by applying the current route

\*

\* - `resolve` - `{Object.<string, function>=}` - An optional map of dependencies which should

\* be injected into the controller. If any of these dependencies are promises, the router

\* will wait for them all to be resolved or one to be rejected before the controller is

\* instantiated.

\* If all the promises are resolved successfully, the values of the resolved promises are

\* injected and {@link ngRoute.$route#$routeChangeSuccess $routeChangeSuccess} event is

\* fired. If any of the promises are rejected the

\* {@link ngRoute.$route#$routeChangeError $routeChangeError} event is fired. The map object

\* is:

\*

\* - `key` – `{string}`: a name of a dependency to be injected into the controller.

\* - `factory` - `{string|function}`: If `string` then it is an alias for a service.

\* Otherwise if function, then it is {@link auto.$injector#invoke injected}

\* and the return value is treated as the dependency. If the result is a promise, it is

\* resolved before its value is injected into the controller. Be aware that

\* `ngRoute.$routeParams` will still refer to the previous route within these resolve

\* functions. Use `$route.current.params` to access the new route parameters, instead.

\*

\* - `redirectTo` – {(string|function())=} – value to update

\* {@link ng.$location $location} path with and trigger route redirection.

\*

\* If `redirectTo` is a function, it will be called with the following parameters:

\*

\* - `{Object.<string>}` - route parameters extracted from the current

\* `$location.path()` by applying the current route templateUrl.

\* - `{string}` - current `$location.path()`

\* - `{Object}` - current `$location.search()`

\*

\* The custom `redirectTo` function is expected to return a string which will be used

\* to update `$location.path()` and `$location.search()`.

\*

\* - `[reloadOnSearch=true]` - {boolean=} - reload route when only `$location.search()`

\* or `$location.hash()` changes.

\*

\* If the option is set to `false` and url in the browser changes, then

\* `$routeUpdate` event is broadcasted on the root scope.

\*

\* - `[caseInsensitiveMatch=false]` - {boolean=} - match routes without being case sensitive

\*

\* If the option is set to `true`, then the particular route can be matched without being

\* case sensitive

\*

\* @returns {Object} self

\*

\* @description

\* Adds a new route definition to the `$route` service.

\*/

this.when = function(path, route) {

routes[path] = angular.extend(

{reloadOnSearch: true},

route,

path && pathRegExp(path, route)

);

// create redirection for trailing slashes

if (path) {

var redirectPath = (path[path.length-1] == '/')

? path.substr(0, path.length-1)

: path +'/';

routes[redirectPath] = angular.extend(

{redirectTo: path},

pathRegExp(redirectPath, route)

);

}

return this;

};

/\*\*

\* @param path {string} path

\* @param opts {Object} options

\* @return {?Object}

\*

\* @description

\* Normalizes the given path, returning a regular expression

\* and the original path.

\*

\* Inspired by pathRexp in visionmedia/express/lib/utils.js.

\*/

function pathRegExp(path, opts) {

var insensitive = opts.caseInsensitiveMatch,

ret = {

originalPath: path,

regexp: path

},

keys = ret.keys = [];

path = path

.replace(/([().])/g, '\\$1')

.replace(/(\/)?:(\w+)([\?\\*])?/g, function(\_, slash, key, option){

var optional = option === '?' ? option : null;

var star = option === '\*' ? option : null;

keys.push({ name: key, optional: !!optional });

slash = slash || '';

return ''

+ (optional ? '' : slash)

+ '(?:'

+ (optional ? slash : '')

+ (star && '(.+?)' || '([^/]+)')

+ (optional || '')

+ ')'

+ (optional || '');

})

.replace(/([\/$\\*])/g, '\\$1');

ret.regexp = new RegExp('^' + path + '$', insensitive ? 'i' : '');

return ret;

}

/\*\*

\* @ngdoc method

\* @name $routeProvider#otherwise

\*

\* @description

\* Sets route definition that will be used on route change when no other route definition

\* is matched.

\*

\* @param {Object} params Mapping information to be assigned to `$route.current`.

\* @returns {Object} self

\*/

this.otherwise = function(params) {

this.when(null, params);

return this;

};

this.$get = ['$rootScope',

'$location',

'$routeParams',

'$q',

'$injector',

'$http',

'$templateCache',

'$sce',

function($rootScope, $location, $routeParams, $q, $injector, $http, $templateCache, $sce) {

/\*\*

\* @ngdoc service

\* @name $route

\* @requires $location

\* @requires $routeParams

\*

\* @property {Object} current Reference to the current route definition.

\* The route definition contains:

\*

\* - `controller`: The controller constructor as define in route definition.

\* - `locals`: A map of locals which is used by {@link ng.$controller $controller} service for

\* controller instantiation. The `locals` contain

\* the resolved values of the `resolve` map. Additionally the `locals` also contain:

\*

\* - `$scope` - The current route scope.

\* - `$template` - The current route template HTML.

\*

\* @property {Object} routes Object with all route configuration Objects as its properties.

\*

\* @description

\* `$route` is used for deep-linking URLs to controllers and views (HTML partials).

\* It watches `$location.url()` and tries to map the path to an existing route definition.

\*

\* Requires the {@link ngRoute `ngRoute`} module to be installed.

\*

\* You can define routes through {@link ngRoute.$routeProvider $routeProvider}'s API.

\*

\* The `$route` service is typically used in conjunction with the

\* {@link ngRoute.directive:ngView `ngView`} directive and the

\* {@link ngRoute.$routeParams `$routeParams`} service.

\*

\* @example

\* This example shows how changing the URL hash causes the `$route` to match a route against the

\* URL, and the `ngView` pulls in the partial.

\*

\* Note that this example is using {@link ng.directive:script inlined templates}

\* to get it working on jsfiddle as well.

\*

\* <example name="$route-service" module="ngRouteExample"

\* deps="angular-route.js" fixBase="true">

\* <file name="index.html">

\* <div ng-controller="MainController">

\* Choose:

\* <a href="Book/Moby">Moby</a> |

\* <a href="Book/Moby/ch/1">Moby: Ch1</a> |

\* <a href="Book/Gatsby">Gatsby</a> |

\* <a href="Book/Gatsby/ch/4?key=value">Gatsby: Ch4</a> |

\* <a href="Book/Scarlet">Scarlet Letter</a><br/>

\*

\* <div ng-view></div>

\*

\* <hr />

\*

\* <pre>$location.path() = {{$location.path()}}</pre>

\* <pre>$route.current.templateUrl = {{$route.current.templateUrl}}</pre>

\* <pre>$route.current.params = {{$route.current.params}}</pre>

\* <pre>$route.current.scope.name = {{$route.current.scope.name}}</pre>

\* <pre>$routeParams = {{$routeParams}}</pre>

\* </div>

\* </file>

\*

\* <file name="book.html">

\* controller: {{name}}<br />

\* Book Id: {{params.bookId}}<br />

\* </file>

\*

\* <file name="chapter.html">

\* controller: {{name}}<br />

\* Book Id: {{params.bookId}}<br />

\* Chapter Id: {{params.chapterId}}

\* </file>

\*

\* <file name="script.js">

\* angular.module('ngRouteExample', ['ngRoute'])

\*

\* .controller('MainController', function($scope, $route, $routeParams, $location) {

\* $scope.$route = $route;

\* $scope.$location = $location;

\* $scope.$routeParams = $routeParams;

\* })

\*

\* .controller('BookController', function($scope, $routeParams) {

\* $scope.name = "BookController";

\* $scope.params = $routeParams;

\* })

\*

\* .controller('ChapterController', function($scope, $routeParams) {

\* $scope.name = "ChapterController";

\* $scope.params = $routeParams;

\* })

\*

\* .config(function($routeProvider, $locationProvider) {

\* $routeProvider

\* .when('/Book/:bookId', {

\* templateUrl: 'book.html',

\* controller: 'BookController',

\* resolve: {

\* // I will cause a 1 second delay

\* delay: function($q, $timeout) {

\* var delay = $q.defer();

\* $timeout(delay.resolve, 1000);

\* return delay.promise;

\* }

\* }

\* })

\* .when('/Book/:bookId/ch/:chapterId', {

\* templateUrl: 'chapter.html',

\* controller: 'ChapterController'

\* });

\*

\* // configure html5 to get links working on jsfiddle

\* $locationProvider.html5Mode(true);

\* });

\*

\* </file>

\*

\* <file name="protractor.js" type="protractor">

\* it('should load and compile correct template', function() {

\* element(by.linkText('Moby: Ch1')).click();

\* var content = element(by.css('[ng-view]')).getText();

\* expect(content).toMatch(/controller\: ChapterController/);

\* expect(content).toMatch(/Book Id\: Moby/);

\* expect(content).toMatch(/Chapter Id\: 1/);

\*

\* element(by.partialLinkText('Scarlet')).click();

\*

\* content = element(by.css('[ng-view]')).getText();

\* expect(content).toMatch(/controller\: BookController/);

\* expect(content).toMatch(/Book Id\: Scarlet/);

\* });

\* </file>

\* </example>

\*/

/\*\*

\* @ngdoc event

\* @name $route#$routeChangeStart

\* @eventType broadcast on root scope

\* @description

\* Broadcasted before a route change. At this point the route services starts

\* resolving all of the dependencies needed for the route change to occur.

\* Typically this involves fetching the view template as well as any dependencies

\* defined in `resolve` route property. Once all of the dependencies are resolved

\* `$routeChangeSuccess` is fired.

\*

\* @param {Object} angularEvent Synthetic event object.

\* @param {Route} next Future route information.

\* @param {Route} current Current route information.

\*/

/\*\*

\* @ngdoc event

\* @name $route#$routeChangeSuccess

\* @eventType broadcast on root scope

\* @description

\* Broadcasted after a route dependencies are resolved.

\* {@link ngRoute.directive:ngView ngView} listens for the directive

\* to instantiate the controller and render the view.

\*

\* @param {Object} angularEvent Synthetic event object.

\* @param {Route} current Current route information.

\* @param {Route|Undefined} previous Previous route information, or undefined if current is

\* first route entered.

\*/

/\*\*

\* @ngdoc event

\* @name $route#$routeChangeError

\* @eventType broadcast on root scope

\* @description

\* Broadcasted if any of the resolve promises are rejected.

\*

\* @param {Object} angularEvent Synthetic event object

\* @param {Route} current Current route information.

\* @param {Route} previous Previous route information.

\* @param {Route} rejection Rejection of the promise. Usually the error of the failed promise.

\*/

/\*\*

\* @ngdoc event

\* @name $route#$routeUpdate

\* @eventType broadcast on root scope

\* @description

\*

\* The `reloadOnSearch` property has been set to false, and we are reusing the same

\* instance of the Controller.

\*/

var forceReload = false,

$route = {

routes: routes,

/\*\*

\* @ngdoc method

\* @name $route#reload

\*

\* @description

\* Causes `$route` service to reload the current route even if

\* {@link ng.$location $location} hasn't changed.

\*

\* As a result of that, {@link ngRoute.directive:ngView ngView}

\* creates new scope, reinstantiates the controller.

\*/

reload: function() {

forceReload = true;

$rootScope.$evalAsync(updateRoute);

}

};

$rootScope.$on('$locationChangeSuccess', updateRoute);

return $route;

/////////////////////////////////////////////////////

/\*\*

\* @param on {string} current url

\* @param route {Object} route regexp to match the url against

\* @return {?Object}

\*

\* @description

\* Check if the route matches the current url.

\*

\* Inspired by match in

\* visionmedia/express/lib/router/router.js.

\*/

function switchRouteMatcher(on, route) {

var keys = route.keys,

params = {};

if (!route.regexp) return null;

var m = route.regexp.exec(on);

if (!m) return null;

for (var i = 1, len = m.length; i < len; ++i) {

var key = keys[i - 1];

var val = 'string' == typeof m[i]

? decodeURIComponent(m[i])

: m[i];

if (key && val) {

params[key.name] = val;

}

}

return params;

}

function updateRoute() {

var next = parseRoute(),

last = $route.current;

if (next && last && next.$$route === last.$$route

&& angular.equals(next.pathParams, last.pathParams)

&& !next.reloadOnSearch && !forceReload) {

last.params = next.params;

angular.copy(last.params, $routeParams);

$rootScope.$broadcast('$routeUpdate', last);

} else if (next || last) {

forceReload = false;

$rootScope.$broadcast('$routeChangeStart', next, last);

$route.current = next;

if (next) {

if (next.redirectTo) {

if (angular.isString(next.redirectTo)) {

$location.path(interpolate(next.redirectTo, next.params)).search(next.params)

.replace();

} else {

$location.url(next.redirectTo(next.pathParams, $location.path(), $location.search()))

.replace();

}

}

}

$q.when(next).

then(function() {

if (next) {

var locals = angular.extend({}, next.resolve),

template, templateUrl;

angular.forEach(locals, function(value, key) {

locals[key] = angular.isString(value) ?

$injector.get(value) : $injector.invoke(value);

});

if (angular.isDefined(template = next.template)) {

if (angular.isFunction(template)) {

template = template(next.params);

}

} else if (angular.isDefined(templateUrl = next.templateUrl)) {

if (angular.isFunction(templateUrl)) {

templateUrl = templateUrl(next.params);

}

templateUrl = $sce.getTrustedResourceUrl(templateUrl);

if (angular.isDefined(templateUrl)) {

next.loadedTemplateUrl = templateUrl;

template = $http.get(templateUrl, {cache: $templateCache}).

then(function(response) { return response.data; });

}

}

if (angular.isDefined(template)) {

locals['$template'] = template;

}

return $q.all(locals);

}

}).

// after route change

then(function(locals) {

if (next == $route.current) {

if (next) {

next.locals = locals;

angular.copy(next.params, $routeParams);

}

$rootScope.$broadcast('$routeChangeSuccess', next, last);

}

}, function(error) {

if (next == $route.current) {

$rootScope.$broadcast('$routeChangeError', next, last, error);

}

});

}

}

/\*\*

\* @returns {Object} the current active route, by matching it against the URL

\*/

function parseRoute() {

// Match a route

var params, match;

angular.forEach(routes, function(route, path) {

if (!match && (params = switchRouteMatcher($location.path(), route))) {

match = inherit(route, {

params: angular.extend({}, $location.search(), params),

pathParams: params});

match.$$route = route;

}

});

// No route matched; fallback to "otherwise" route

return match || routes[null] && inherit(routes[null], {params: {}, pathParams:{}});

}

/\*\*

\* @returns {string} interpolation of the redirect path with the parameters

\*/

function interpolate(string, params) {

var result = [];

angular.forEach((string||'').split(':'), function(segment, i) {

if (i === 0) {

result.push(segment);

} else {

var segmentMatch = segment.match(/(\w+)(.\*)/);

var key = segmentMatch[1];

result.push(params[key]);

result.push(segmentMatch[2] || '');

delete params[key];

}

});

return result.join('');

}

}];

}

ngRouteModule.provider('$routeParams', $RouteParamsProvider);

/\*\*

\* @ngdoc service

\* @name $routeParams

\* @requires $route

\*

\* @description

\* The `$routeParams` service allows you to retrieve the current set of route parameters.

\*

\* Requires the {@link ngRoute `ngRoute`} module to be installed.

\*

\* The route parameters are a combination of {@link ng.$location `$location`}'s

\* {@link ng.$location#search `search()`} and {@link ng.$location#path `path()`}.

\* The `path` parameters are extracted when the {@link ngRoute.$route `$route`} path is matched.

\*

\* In case of parameter name collision, `path` params take precedence over `search` params.

\*

\* The service guarantees that the identity of the `$routeParams` object will remain unchanged

\* (but its properties will likely change) even when a route change occurs.

\*

\* Note that the `$routeParams` are only updated \*after\* a route change completes successfully.

\* This means that you cannot rely on `$routeParams` being correct in route resolve functions.

\* Instead you can use `$route.current.params` to access the new route's parameters.

\*

\* @example

\* ```js

\* // Given:

\* // URL: http://server.com/index.html#/Chapter/1/Section/2?search=moby

\* // Route: /Chapter/:chapterId/Section/:sectionId

\* //

\* // Then

\* $routeParams ==> {chapterId:1, sectionId:2, search:'moby'}

\* ```

\*/

function $RouteParamsProvider() {

this.$get = function() { return {}; };

}

ngRouteModule.directive('ngView', ngViewFactory);

ngRouteModule.directive('ngView', ngViewFillContentFactory);

/\*\*

\* @ngdoc directive

\* @name ngView

\* @restrict ECA

\*

\* @description

\* # Overview

\* `ngView` is a directive that complements the {@link ngRoute.$route $route} service by

\* including the rendered template of the current route into the main layout (`index.html`) file.

\* Every time the current route changes, the included view changes with it according to the

\* configuration of the `$route` service.

\*

\* Requires the {@link ngRoute `ngRoute`} module to be installed.

\*

\* @animations

\* enter - animation is used to bring new content into the browser.

\* leave - animation is used to animate existing content away.

\*

\* The enter and leave animation occur concurrently.

\*

\* @scope

\* @priority 400

\* @param {string=} onload Expression to evaluate whenever the view updates.

\*

\* @param {string=} autoscroll Whether `ngView` should call {@link ng.$anchorScroll

\* $anchorScroll} to scroll the viewport after the view is updated.

\*

\* - If the attribute is not set, disable scrolling.

\* - If the attribute is set without value, enable scrolling.

\* - Otherwise enable scrolling only if the `autoscroll` attribute value evaluated

\* as an expression yields a truthy value.

\* @example

<example name="ngView-directive" module="ngViewExample"

deps="angular-route.js;angular-animate.js"

animations="true" fixBase="true">

<file name="index.html">

<div ng-controller="MainCtrl as main">

Choose:

<a href="Book/Moby">Moby</a> |

<a href="Book/Moby/ch/1">Moby: Ch1</a> |

<a href="Book/Gatsby">Gatsby</a> |

<a href="Book/Gatsby/ch/4?key=value">Gatsby: Ch4</a> |

<a href="Book/Scarlet">Scarlet Letter</a><br/>

<div class="view-animate-container">

<div ng-view class="view-animate"></div>

</div>

<hr />

<pre>$location.path() = {{main.$location.path()}}</pre>

<pre>$route.current.templateUrl = {{main.$route.current.templateUrl}}</pre>

<pre>$route.current.params = {{main.$route.current.params}}</pre>

<pre>$route.current.scope.name = {{main.$route.current.scope.name}}</pre>

<pre>$routeParams = {{main.$routeParams}}</pre>

</div>

</file>

<file name="book.html">

<div>

controller: {{book.name}}<br />

Book Id: {{book.params.bookId}}<br />

</div>

</file>

<file name="chapter.html">

<div>

controller: {{chapter.name}}<br />

Book Id: {{chapter.params.bookId}}<br />

Chapter Id: {{chapter.params.chapterId}}

</div>

</file>

<file name="animations.css">

.view-animate-container {

position:relative;

height:100px!important;

position:relative;

background:white;

border:1px solid black;

height:40px;

overflow:hidden;

}

.view-animate {

padding:10px;

}

.view-animate.ng-enter, .view-animate.ng-leave {

-webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 1.5s;

transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 1.5s;

display:block;

width:100%;

border-left:1px solid black;

position:absolute;

top:0;

left:0;

right:0;

bottom:0;

padding:10px;

}

.view-animate.ng-enter {

left:100%;

}

.view-animate.ng-enter.ng-enter-active {

left:0;

}

.view-animate.ng-leave.ng-leave-active {

left:-100%;

}

</file>

<file name="script.js">

angular.module('ngViewExample', ['ngRoute', 'ngAnimate'])

.config(['$routeProvider', '$locationProvider',

function($routeProvider, $locationProvider) {

$routeProvider

.when('/Book/:bookId', {

templateUrl: 'book.html',

controller: 'BookCtrl',

controllerAs: 'book'

})

.when('/Book/:bookId/ch/:chapterId', {

templateUrl: 'chapter.html',

controller: 'ChapterCtrl',

controllerAs: 'chapter'

});

// configure html5 to get links working on jsfiddle

$locationProvider.html5Mode(true);

}])

.controller('MainCtrl', ['$route', '$routeParams', '$location',

function($route, $routeParams, $location) {

this.$route = $route;

this.$location = $location;

this.$routeParams = $routeParams;

}])

.controller('BookCtrl', ['$routeParams', function($routeParams) {

this.name = "BookCtrl";

this.params = $routeParams;

}])

.controller('ChapterCtrl', ['$routeParams', function($routeParams) {

this.name = "ChapterCtrl";

this.params = $routeParams;

}]);

</file>

<file name="protractor.js" type="protractor">

it('should load and compile correct template', function() {

element(by.linkText('Moby: Ch1')).click();

var content = element(by.css('[ng-view]')).getText();

expect(content).toMatch(/controller\: ChapterCtrl/);

expect(content).toMatch(/Book Id\: Moby/);

expect(content).toMatch(/Chapter Id\: 1/);

element(by.partialLinkText('Scarlet')).click();

content = element(by.css('[ng-view]')).getText();

expect(content).toMatch(/controller\: BookCtrl/);

expect(content).toMatch(/Book Id\: Scarlet/);

});

</file>

</example>

\*/

/\*\*

\* @ngdoc event

\* @name ngView#$viewContentLoaded

\* @eventType emit on the current ngView scope

\* @description

\* Emitted every time the ngView content is reloaded.

\*/

ngViewFactory.$inject = ['$route', '$anchorScroll', '$animate'];

function ngViewFactory( $route, $anchorScroll, $animate) {

return {

restrict: 'ECA',

terminal: true,

priority: 400,

transclude: 'element',

link: function(scope, $element, attr, ctrl, $transclude) {

var currentScope,

currentElement,

previousElement,

autoScrollExp = attr.autoscroll,

onloadExp = attr.onload || '';

scope.$on('$routeChangeSuccess', update);

update();

function cleanupLastView() {

if(previousElement) {

previousElement.remove();

previousElement = null;

}

if(currentScope) {

currentScope.$destroy();

currentScope = null;

}

if(currentElement) {

$animate.leave(currentElement, function() {

previousElement = null;

});

previousElement = currentElement;

currentElement = null;

}

}

function update() {

var locals = $route.current && $route.current.locals,

template = locals && locals.$template;

if (angular.isDefined(template)) {

var newScope = scope.$new();

var current = $route.current;

// Note: This will also link all children of ng-view that were contained in the original

// html. If that content contains controllers, ... they could pollute/change the scope.

// However, using ng-view on an element with additional content does not make sense...

// Note: We can't remove them in the cloneAttchFn of $transclude as that

// function is called before linking the content, which would apply child

// directives to non existing elements.

var clone = $transclude(newScope, function(clone) {

$animate.enter(clone, null, currentElement || $element, function onNgViewEnter () {

if (angular.isDefined(autoScrollExp)

&& (!autoScrollExp || scope.$eval(autoScrollExp))) {

$anchorScroll();

}

});

cleanupLastView();

});

currentElement = clone;

currentScope = current.scope = newScope;

currentScope.$emit('$viewContentLoaded');

currentScope.$eval(onloadExp);

} else {

cleanupLastView();

}

}

}

};

}

// This directive is called during the $transclude call of the first `ngView` directive.

// It will replace and compile the content of the element with the loaded template.

// We need this directive so that the element content is already filled when

// the link function of another directive on the same element as ngView

// is called.

ngViewFillContentFactory.$inject = ['$compile', '$controller', '$route'];

function ngViewFillContentFactory($compile, $controller, $route) {

return {

restrict: 'ECA',

priority: -400,

link: function(scope, $element) {

var current = $route.current,

locals = current.locals;

$element.html(locals.$template);

var link = $compile($element.contents());

if (current.controller) {

locals.$scope = scope;

var controller = $controller(current.controller, locals);

if (current.controllerAs) {

scope[current.controllerAs] = controller;

}

$element.data('$ngControllerController', controller);

$element.children().data('$ngControllerController', controller);

}

link(scope);

}

};

}

})(window, window.angular);

/\*\*

\* @license AngularJS v1.2.16

\* (c) 2010-2014 Google, Inc. http://angularjs.org

\* License: MIT

\*/

(function(window, document, undefined) {'use strict';

/\*\*

\* @description

\*

\* This object provides a utility for producing rich Error messages within

\* Angular. It can be called as follows:

\*

\* var exampleMinErr = minErr('example');

\* throw exampleMinErr('one', 'This {0} is {1}', foo, bar);

\*

\* The above creates an instance of minErr in the example namespace. The

\* resulting error will have a namespaced error code of example.one. The

\* resulting error will replace {0} with the value of foo, and {1} with the

\* value of bar. The object is not restricted in the number of arguments it can

\* take.

\*

\* If fewer arguments are specified than necessary for interpolation, the extra

\* interpolation markers will be preserved in the final string.

\*

\* Since data will be parsed statically during a build step, some restrictions

\* are applied with respect to how minErr instances are created and called.

\* Instances should have names of the form namespaceMinErr for a minErr created

\* using minErr('namespace') . Error codes, namespaces and template strings

\* should all be static strings, not variables or general expressions.

\*

\* @param {string} module The namespace to use for the new minErr instance.

\* @returns {function(code:string, template:string, ...templateArgs): Error} minErr instance

\*/

function minErr(module) {

return function () {

var code = arguments[0],

prefix = '[' + (module ? module + ':' : '') + code + '] ',

template = arguments[1],

templateArgs = arguments,

stringify = function (obj) {

if (typeof obj === 'function') {

return obj.toString().replace(/ \{[\s\S]\*$/, '');

} else if (typeof obj === 'undefined') {

return 'undefined';

} else if (typeof obj !== 'string') {

return JSON.stringify(obj);

}

return obj;

},

message, i;

message = prefix + template.replace(/\{\d+\}/g, function (match) {

var index = +match.slice(1, -1), arg;

if (index + 2 < templateArgs.length) {

arg = templateArgs[index + 2];

if (typeof arg === 'function') {

return arg.toString().replace(/ ?\{[\s\S]\*$/, '');

} else if (typeof arg === 'undefined') {

return 'undefined';

} else if (typeof arg !== 'string') {

return toJson(arg);

}

return arg;

}

return match;

});

message = message + '\nhttp://errors.angularjs.org/1.2.16/' +

(module ? module + '/' : '') + code;

for (i = 2; i < arguments.length; i++) {

message = message + (i == 2 ? '?' : '&') + 'p' + (i-2) + '=' +

encodeURIComponent(stringify(arguments[i]));

}

return new Error(message);

};

}

/\* We need to tell jshint what variables are being exported \*/

/\* global

-angular,

-msie,

-jqLite,

-jQuery,

-slice,

-push,

-toString,

-ngMinErr,

-\_angular,

-angularModule,

-nodeName\_,

-uid,

-lowercase,

-uppercase,

-manualLowercase,

-manualUppercase,

-nodeName\_,

-isArrayLike,

-forEach,

-sortedKeys,

-forEachSorted,

-reverseParams,

-nextUid,

-setHashKey,

-extend,

-int,

-inherit,

-noop,

-identity,

-valueFn,

-isUndefined,

-isDefined,

-isObject,

-isString,

-isNumber,

-isDate,

-isArray,

-isFunction,

-isRegExp,

-isWindow,

-isScope,

-isFile,

-isBlob,

-isBoolean,

-trim,

-isElement,

-makeMap,

-map,

-size,

-includes,

-indexOf,

-arrayRemove,

-isLeafNode,

-copy,

-shallowCopy,

-equals,

-csp,

-concat,

-sliceArgs,

-bind,

-toJsonReplacer,

-toJson,

-fromJson,

-toBoolean,

-startingTag,

-tryDecodeURIComponent,

-parseKeyValue,

-toKeyValue,

-encodeUriSegment,

-encodeUriQuery,

-angularInit,

-bootstrap,

-snake\_case,

-bindJQuery,

-assertArg,

-assertArgFn,

-assertNotHasOwnProperty,

-getter,

-getBlockElements,

-hasOwnProperty,

\*/

////////////////////////////////////

/\*\*

\* @ngdoc module

\* @name ng

\* @module ng

\* @description

\*

\* # ng (core module)

\* The ng module is loaded by default when an AngularJS application is started. The module itself

\* contains the essential components for an AngularJS application to function. The table below

\* lists a high level breakdown of each of the services/factories, filters, directives and testing

\* components available within this core module.

\*

\* <div doc-module-components="ng"></div>

\*/

/\*\*

\* @ngdoc function

\* @name angular.lowercase

\* @module ng

\* @function

\*

\* @description Converts the specified string to lowercase.

\* @param {string} string String to be converted to lowercase.

\* @returns {string} Lowercased string.

\*/

var lowercase = function(string){return isString(string) ? string.toLowerCase() : string;};

var hasOwnProperty = Object.prototype.hasOwnProperty;

/\*\*

\* @ngdoc function

\* @name angular.uppercase

\* @module ng

\* @function

\*

\* @description Converts the specified string to uppercase.

\* @param {string} string String to be converted to uppercase.

\* @returns {string} Uppercased string.

\*/

var uppercase = function(string){return isString(string) ? string.toUpperCase() : string;};

var manualLowercase = function(s) {

/\* jshint bitwise: false \*/

return isString(s)

? s.replace(/[A-Z]/g, function(ch) {return String.fromCharCode(ch.charCodeAt(0) | 32);})

: s;

};

var manualUppercase = function(s) {

/\* jshint bitwise: false \*/

return isString(s)

? s.replace(/[a-z]/g, function(ch) {return String.fromCharCode(ch.charCodeAt(0) & ~32);})

: s;

};

// String#toLowerCase and String#toUpperCase don't produce correct results in browsers with Turkish

// locale, for this reason we need to detect this case and redefine lowercase/uppercase methods

// with correct but slower alternatives.

if ('i' !== 'I'.toLowerCase()) {

lowercase = manualLowercase;

uppercase = manualUppercase;

}

var /\*\* holds major version number for IE or NaN for real browsers \*/

msie,

jqLite, // delay binding since jQuery could be loaded after us.

jQuery, // delay binding

slice = [].slice,

push = [].push,

toString = Object.prototype.toString,

ngMinErr = minErr('ng'),

\_angular = window.angular,

/\*\* @name angular \*/

angular = window.angular || (window.angular = {}),

angularModule,

nodeName\_,

uid = ['0', '0', '0'];

/\*\*

\* IE 11 changed the format of the UserAgent string.

\* See http://msdn.microsoft.com/en-us/library/ms537503.aspx

\*/

msie = int((/msie (\d+)/.exec(lowercase(navigator.userAgent)) || [])[1]);

if (isNaN(msie)) {

msie = int((/trident\/.\*; rv:(\d+)/.exec(lowercase(navigator.userAgent)) || [])[1]);

}

/\*\*

\* @private

\* @param {\*} obj

\* @return {boolean} Returns true if `obj` is an array or array-like object (NodeList, Arguments,

\* String ...)

\*/

function isArrayLike(obj) {

if (obj == null || isWindow(obj)) {

return false;

}

var length = obj.length;

if (obj.nodeType === 1 && length) {

return true;

}

return isString(obj) || isArray(obj) || length === 0 ||

typeof length === 'number' && length > 0 && (length - 1) in obj;

}

/\*\*

\* @ngdoc function

\* @name angular.forEach

\* @module ng

\* @function

\*

\* @description

\* Invokes the `iterator` function once for each item in `obj` collection, which can be either an

\* object or an array. The `iterator` function is invoked with `iterator(value, key)`, where `value`

\* is the value of an object property or an array element and `key` is the object property key or

\* array element index. Specifying a `context` for the function is optional.

\*

\* It is worth noting that `.forEach` does not iterate over inherited properties because it filters

\* using the `hasOwnProperty` method.

\*

```js

var values = {name: 'misko', gender: 'male'};

var log = [];

angular.forEach(values, function(value, key){

this.push(key + ': ' + value);

}, log);

expect(log).toEqual(['name: misko', 'gender: male']);

```

\*

\* @param {Object|Array} obj Object to iterate over.

\* @param {Function} iterator Iterator function.

\* @param {Object=} context Object to become context (`this`) for the iterator function.

\* @returns {Object|Array} Reference to `obj`.

\*/

function forEach(obj, iterator, context) {

var key;

if (obj) {

if (isFunction(obj)){

for (key in obj) {

// Need to check if hasOwnProperty exists,

// as on IE8 the result of querySelectorAll is an object without a hasOwnProperty function

if (key != 'prototype' && key != 'length' && key != 'name' && (!obj.hasOwnProperty || obj.hasOwnProperty(key))) {

iterator.call(context, obj[key], key);

}

}

} else if (obj.forEach && obj.forEach !== forEach) {

obj.forEach(iterator, context);

} else if (isArrayLike(obj)) {

for (key = 0; key < obj.length; key++)

iterator.call(context, obj[key], key);

} else {

for (key in obj) {

if (obj.hasOwnProperty(key)) {

iterator.call(context, obj[key], key);

}

}

}

}

return obj;

}

function sortedKeys(obj) {

var keys = [];

for (var key in obj) {

if (obj.hasOwnProperty(key)) {

keys.push(key);

}

}

return keys.sort();

}

function forEachSorted(obj, iterator, context) {

var keys = sortedKeys(obj);

for ( var i = 0; i < keys.length; i++) {

iterator.call(context, obj[keys[i]], keys[i]);

}

return keys;

}

/\*\*

\* when using forEach the params are value, key, but it is often useful to have key, value.

\* @param {function(string, \*)} iteratorFn

\* @returns {function(\*, string)}

\*/

function reverseParams(iteratorFn) {

return function(value, key) { iteratorFn(key, value); };

}

/\*\*

\* A consistent way of creating unique IDs in angular. The ID is a sequence of alpha numeric

\* characters such as '012ABC'. The reason why we are not using simply a number counter is that

\* the number string gets longer over time, and it can also overflow, where as the nextId

\* will grow much slower, it is a string, and it will never overflow.

\*

\* @returns {string} an unique alpha-numeric string

\*/

function nextUid() {

var index = uid.length;

var digit;

while(index) {

index--;

digit = uid[index].charCodeAt(0);

if (digit == 57 /\*'9'\*/) {

uid[index] = 'A';

return uid.join('');

}

if (digit == 90 /\*'Z'\*/) {

uid[index] = '0';

} else {

uid[index] = String.fromCharCode(digit + 1);

return uid.join('');

}

}

uid.unshift('0');

return uid.join('');

}

/\*\*

\* Set or clear the hashkey for an object.

\* @param obj object

\* @param h the hashkey (!truthy to delete the hashkey)

\*/

function setHashKey(obj, h) {

if (h) {

obj.$$hashKey = h;

}

else {

delete obj.$$hashKey;

}

}

/\*\*

\* @ngdoc function

\* @name angular.extend

\* @module ng

\* @function

\*

\* @description

\* Extends the destination object `dst` by copying all of the properties from the `src` object(s)

\* to `dst`. You can specify multiple `src` objects.

\*

\* @param {Object} dst Destination object.

\* @param {...Object} src Source object(s).

\* @returns {Object} Reference to `dst`.

\*/

function extend(dst) {

var h = dst.$$hashKey;

forEach(arguments, function(obj){

if (obj !== dst) {

forEach(obj, function(value, key){

dst[key] = value;

});

}

});

setHashKey(dst,h);

return dst;

}

function int(str) {

return parseInt(str, 10);

}

function inherit(parent, extra) {

return extend(new (extend(function() {}, {prototype:parent}))(), extra);

}

/\*\*

\* @ngdoc function

\* @name angular.noop

\* @module ng

\* @function

\*

\* @description

\* A function that performs no operations. This function can be useful when writing code in the

\* functional style.

```js

function foo(callback) {

var result = calculateResult();

(callback || angular.noop)(result);

}

```

\*/

function noop() {}

noop.$inject = [];

/\*\*

\* @ngdoc function

\* @name angular.identity

\* @module ng

\* @function

\*

\* @description

\* A function that returns its first argument. This function is useful when writing code in the

\* functional style.

\*

```js

function transformer(transformationFn, value) {

return (transformationFn || angular.identity)(value);

};

```

\*/

function identity($) {return $;}

identity.$inject = [];

function valueFn(value) {return function() {return value;};}

/\*\*

\* @ngdoc function

\* @name angular.isUndefined

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is undefined.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is undefined.

\*/

function isUndefined(value){return typeof value === 'undefined';}

/\*\*

\* @ngdoc function

\* @name angular.isDefined

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is defined.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is defined.

\*/

function isDefined(value){return typeof value !== 'undefined';}

/\*\*

\* @ngdoc function

\* @name angular.isObject

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is an `Object`. Unlike `typeof` in JavaScript, `null`s are not

\* considered to be objects. Note that JavaScript arrays are objects.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is an `Object` but not `null`.

\*/

function isObject(value){return value != null && typeof value === 'object';}

/\*\*

\* @ngdoc function

\* @name angular.isString

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is a `String`.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is a `String`.

\*/

function isString(value){return typeof value === 'string';}

/\*\*

\* @ngdoc function

\* @name angular.isNumber

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is a `Number`.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is a `Number`.

\*/

function isNumber(value){return typeof value === 'number';}

/\*\*

\* @ngdoc function

\* @name angular.isDate

\* @module ng

\* @function

\*

\* @description

\* Determines if a value is a date.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is a `Date`.

\*/

function isDate(value){

return toString.call(value) === '[object Date]';

}

/\*\*

\* @ngdoc function

\* @name angular.isArray

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is an `Array`.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is an `Array`.

\*/

function isArray(value) {

return toString.call(value) === '[object Array]';

}

/\*\*

\* @ngdoc function

\* @name angular.isFunction

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is a `Function`.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is a `Function`.

\*/

function isFunction(value){return typeof value === 'function';}

/\*\*

\* Determines if a value is a regular expression object.

\*

\* @private

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is a `RegExp`.

\*/

function isRegExp(value) {

return toString.call(value) === '[object RegExp]';

}

/\*\*

\* Checks if `obj` is a window object.

\*

\* @private

\* @param {\*} obj Object to check

\* @returns {boolean} True if `obj` is a window obj.

\*/

function isWindow(obj) {

return obj && obj.document && obj.location && obj.alert && obj.setInterval;

}

function isScope(obj) {

return obj && obj.$evalAsync && obj.$watch;

}

function isFile(obj) {

return toString.call(obj) === '[object File]';

}

function isBlob(obj) {

return toString.call(obj) === '[object Blob]';

}

function isBoolean(value) {

return typeof value === 'boolean';

}

var trim = (function() {

// native trim is way faster: http://jsperf.com/angular-trim-test

// but IE doesn't have it... :-(

// TODO: we should move this into IE/ES5 polyfill

if (!String.prototype.trim) {

return function(value) {

return isString(value) ? value.replace(/^\s\s\*/, '').replace(/\s\s\*$/, '') : value;

};

}

return function(value) {

return isString(value) ? value.trim() : value;

};

})();

/\*\*

\* @ngdoc function

\* @name angular.isElement

\* @module ng

\* @function

\*

\* @description

\* Determines if a reference is a DOM element (or wrapped jQuery element).

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is a DOM element (or wrapped jQuery element).

\*/

function isElement(node) {

return !!(node &&

(node.nodeName // we are a direct element

|| (node.prop && node.attr && node.find))); // we have an on and find method part of jQuery API

}

/\*\*

\* @param str 'key1,key2,...'

\* @returns {object} in the form of {key1:true, key2:true, ...}

\*/

function makeMap(str){

var obj = {}, items = str.split(","), i;

for ( i = 0; i < items.length; i++ )

obj[ items[i] ] = true;

return obj;

}

if (msie < 9) {

nodeName\_ = function(element) {

element = element.nodeName ? element : element[0];

return (element.scopeName && element.scopeName != 'HTML')

? uppercase(element.scopeName + ':' + element.nodeName) : element.nodeName;

};

} else {

nodeName\_ = function(element) {

return element.nodeName ? element.nodeName : element[0].nodeName;

};

}

function map(obj, iterator, context) {

var results = [];

forEach(obj, function(value, index, list) {

results.push(iterator.call(context, value, index, list));

});

return results;

}

/\*\*

\* @description

\* Determines the number of elements in an array, the number of properties an object has, or

\* the length of a string.

\*

\* Note: This function is used to augment the Object type in Angular expressions. See

\* {@link angular.Object} for more information about Angular arrays.

\*

\* @param {Object|Array|string} obj Object, array, or string to inspect.

\* @param {boolean} [ownPropsOnly=false] Count only "own" properties in an object

\* @returns {number} The size of `obj` or `0` if `obj` is neither an object nor an array.

\*/

function size(obj, ownPropsOnly) {

var count = 0, key;

if (isArray(obj) || isString(obj)) {

return obj.length;

} else if (isObject(obj)){

for (key in obj)

if (!ownPropsOnly || obj.hasOwnProperty(key))

count++;

}

return count;

}

function includes(array, obj) {

return indexOf(array, obj) != -1;

}

function indexOf(array, obj) {

if (array.indexOf) return array.indexOf(obj);

for (var i = 0; i < array.length; i++) {

if (obj === array[i]) return i;

}

return -1;

}

function arrayRemove(array, value) {

var index = indexOf(array, value);

if (index >=0)

array.splice(index, 1);

return value;

}

function isLeafNode (node) {

if (node) {

switch (node.nodeName) {

case "OPTION":

case "PRE":

case "TITLE":

return true;

}

}

return false;

}

/\*\*

\* @ngdoc function

\* @name angular.copy

\* @module ng

\* @function

\*

\* @description

\* Creates a deep copy of `source`, which should be an object or an array.

\*

\* \* If no destination is supplied, a copy of the object or array is created.

\* \* If a destination is provided, all of its elements (for array) or properties (for objects)

\* are deleted and then all elements/properties from the source are copied to it.

\* \* If `source` is not an object or array (inc. `null` and `undefined`), `source` is returned.

\* \* If `source` is identical to 'destination' an exception will be thrown.

\*

\* @param {\*} source The source that will be used to make a copy.

\* Can be any type, including primitives, `null`, and `undefined`.

\* @param {(Object|Array)=} destination Destination into which the source is copied. If

\* provided, must be of the same type as `source`.

\* @returns {\*} The copy or updated `destination`, if `destination` was specified.

\*

\* @example

<example>

<file name="index.html">

<div ng-controller="Controller">

<form novalidate class="simple-form">

Name: <input type="text" ng-model="user.name" /><br />

E-mail: <input type="email" ng-model="user.email" /><br />

Gender: <input type="radio" ng-model="user.gender" value="male" />male

<input type="radio" ng-model="user.gender" value="female" />female<br />

<button ng-click="reset()">RESET</button>

<button ng-click="update(user)">SAVE</button>

</form>

<pre>form = {{user | json}}</pre>

<pre>master = {{master | json}}</pre>

</div>

<script>

function Controller($scope) {

$scope.master= {};

$scope.update = function(user) {

// Example with 1 argument

$scope.master= angular.copy(user);

};

$scope.reset = function() {

// Example with 2 arguments

angular.copy($scope.master, $scope.user);

};

$scope.reset();

}

</script>

</file>

</example>

\*/

function copy(source, destination){

if (isWindow(source) || isScope(source)) {

throw ngMinErr('cpws',

"Can't copy! Making copies of Window or Scope instances is not supported.");

}

if (!destination) {

destination = source;

if (source) {

if (isArray(source)) {

destination = copy(source, []);

} else if (isDate(source)) {

destination = new Date(source.getTime());

} else if (isRegExp(source)) {

destination = new RegExp(source.source);

} else if (isObject(source)) {

destination = copy(source, {});

}

}

} else {

if (source === destination) throw ngMinErr('cpi',

"Can't copy! Source and destination are identical.");

if (isArray(source)) {

destination.length = 0;

for ( var i = 0; i < source.length; i++) {

destination.push(copy(source[i]));

}

} else {

var h = destination.$$hashKey;

forEach(destination, function(value, key){

delete destination[key];

});

for ( var key in source) {

destination[key] = copy(source[key]);

}

setHashKey(destination,h);

}

}

return destination;

}

/\*\*

\* Create a shallow copy of an object

\*/

function shallowCopy(src, dst) {

dst = dst || {};

for(var key in src) {

// shallowCopy is only ever called by $compile nodeLinkFn, which has control over src

// so we don't need to worry about using our custom hasOwnProperty here

if (src.hasOwnProperty(key) && !(key.charAt(0) === '$' && key.charAt(1) === '$')) {

dst[key] = src[key];

}

}

return dst;

}

/\*\*

\* @ngdoc function

\* @name angular.equals

\* @module ng

\* @function

\*

\* @description

\* Determines if two objects or two values are equivalent. Supports value types, regular

\* expressions, arrays and objects.

\*

\* Two objects or values are considered equivalent if at least one of the following is true:

\*

\* \* Both objects or values pass `===` comparison.

\* \* Both objects or values are of the same type and all of their properties are equal by

\* comparing them with `angular.equals`.

\* \* Both values are NaN. (In JavaScript, NaN == NaN => false. But we consider two NaN as equal)

\* \* Both values represent the same regular expression (In JavasScript,

\* /abc/ == /abc/ => false. But we consider two regular expressions as equal when their textual

\* representation matches).

\*

\* During a property comparison, properties of `function` type and properties with names

\* that begin with `$` are ignored.

\*

\* Scope and DOMWindow objects are being compared only by identify (`===`).

\*

\* @param {\*} o1 Object or value to compare.

\* @param {\*} o2 Object or value to compare.

\* @returns {boolean} True if arguments are equal.

\*/

function equals(o1, o2) {

if (o1 === o2) return true;

if (o1 === null || o2 === null) return false;

if (o1 !== o1 && o2 !== o2) return true; // NaN === NaN

var t1 = typeof o1, t2 = typeof o2, length, key, keySet;

if (t1 == t2) {

if (t1 == 'object') {

if (isArray(o1)) {

if (!isArray(o2)) return false;

if ((length = o1.length) == o2.length) {

for(key=0; key<length; key++) {

if (!equals(o1[key], o2[key])) return false;

}

return true;

}

} else if (isDate(o1)) {

return isDate(o2) && o1.getTime() == o2.getTime();

} else if (isRegExp(o1) && isRegExp(o2)) {

return o1.toString() == o2.toString();

} else {

if (isScope(o1) || isScope(o2) || isWindow(o1) || isWindow(o2) || isArray(o2)) return false;

keySet = {};

for(key in o1) {

if (key.charAt(0) === '$' || isFunction(o1[key])) continue;

if (!equals(o1[key], o2[key])) return false;

keySet[key] = true;

}

for(key in o2) {

if (!keySet.hasOwnProperty(key) &&

key.charAt(0) !== '$' &&

o2[key] !== undefined &&

!isFunction(o2[key])) return false;

}

return true;

}

}

}

return false;

}

function csp() {

return (document.securityPolicy && document.securityPolicy.isActive) ||

(document.querySelector &&

!!(document.querySelector('[ng-csp]') || document.querySelector('[data-ng-csp]')));

}

function concat(array1, array2, index) {

return array1.concat(slice.call(array2, index));

}

function sliceArgs(args, startIndex) {

return slice.call(args, startIndex || 0);

}

/\* jshint -W101 \*/

/\*\*

\* @ngdoc function

\* @name angular.bind

\* @module ng

\* @function

\*

\* @description

\* Returns a function which calls function `fn` bound to `self` (`self` becomes the `this` for

\* `fn`). You can supply optional `args` that are prebound to the function. This feature is also

\* known as [partial application](http://en.wikipedia.org/wiki/Partial\_application), as

\* distinguished from [function currying](http://en.wikipedia.org/wiki/Currying#Contrast\_with\_partial\_function\_application).

\*

\* @param {Object} self Context which `fn` should be evaluated in.

\* @param {function()} fn Function to be bound.

\* @param {...\*} args Optional arguments to be prebound to the `fn` function call.

\* @returns {function()} Function that wraps the `fn` with all the specified bindings.

\*/

/\* jshint +W101 \*/

function bind(self, fn) {

var curryArgs = arguments.length > 2 ? sliceArgs(arguments, 2) : [];

if (isFunction(fn) && !(fn instanceof RegExp)) {

return curryArgs.length

? function() {

return arguments.length

? fn.apply(self, curryArgs.concat(slice.call(arguments, 0)))

: fn.apply(self, curryArgs);

}

: function() {

return arguments.length

? fn.apply(self, arguments)

: fn.call(self);

};

} else {

// in IE, native methods are not functions so they cannot be bound (note: they don't need to be)

return fn;

}

}

function toJsonReplacer(key, value) {

var val = value;

if (typeof key === 'string' && key.charAt(0) === '$') {

val = undefined;

} else if (isWindow(value)) {

val = '$WINDOW';

} else if (value && document === value) {

val = '$DOCUMENT';

} else if (isScope(value)) {

val = '$SCOPE';

}

return val;

}

/\*\*

\* @ngdoc function

\* @name angular.toJson

\* @module ng

\* @function

\*

\* @description

\* Serializes input into a JSON-formatted string. Properties with leading $ characters will be

\* stripped since angular uses this notation internally.

\*

\* @param {Object|Array|Date|string|number} obj Input to be serialized into JSON.

\* @param {boolean=} pretty If set to true, the JSON output will contain newlines and whitespace.

\* @returns {string|undefined} JSON-ified string representing `obj`.

\*/

function toJson(obj, pretty) {

if (typeof obj === 'undefined') return undefined;

return JSON.stringify(obj, toJsonReplacer, pretty ? ' ' : null);

}

/\*\*

\* @ngdoc function

\* @name angular.fromJson

\* @module ng

\* @function

\*

\* @description

\* Deserializes a JSON string.

\*

\* @param {string} json JSON string to deserialize.

\* @returns {Object|Array|string|number} Deserialized thingy.

\*/

function fromJson(json) {

return isString(json)

? JSON.parse(json)

: json;

}

function toBoolean(value) {

if (typeof value === 'function') {

value = true;

} else if (value && value.length !== 0) {

var v = lowercase("" + value);

value = !(v == 'f' || v == '0' || v == 'false' || v == 'no' || v == 'n' || v == '[]');

} else {

value = false;

}

return value;

}

/\*\*

\* @returns {string} Returns the string representation of the element.

\*/

function startingTag(element) {

element = jqLite(element).clone();

try {

// turns out IE does not let you set .html() on elements which

// are not allowed to have children. So we just ignore it.

element.empty();

} catch(e) {}

// As Per DOM Standards

var TEXT\_NODE = 3;

var elemHtml = jqLite('<div>').append(element).html();

try {

return element[0].nodeType === TEXT\_NODE ? lowercase(elemHtml) :

elemHtml.

match(/^(<[^>]+>)/)[1].

replace(/^<([\w\-]+)/, function(match, nodeName) { return '<' + lowercase(nodeName); });

} catch(e) {

return lowercase(elemHtml);

}

}

/////////////////////////////////////////////////

/\*\*

\* Tries to decode the URI component without throwing an exception.

\*

\* @private

\* @param str value potential URI component to check.

\* @returns {boolean} True if `value` can be decoded

\* with the decodeURIComponent function.

\*/

function tryDecodeURIComponent(value) {

try {

return decodeURIComponent(value);

} catch(e) {

// Ignore any invalid uri component

}

}

/\*\*

\* Parses an escaped url query string into key-value pairs.

\* @returns {Object.<string,boolean|Array>}

\*/

function parseKeyValue(/\*\*string\*/keyValue) {

var obj = {}, key\_value, key;

forEach((keyValue || "").split('&'), function(keyValue){

if ( keyValue ) {

key\_value = keyValue.split('=');

key = tryDecodeURIComponent(key\_value[0]);

if ( isDefined(key) ) {

var val = isDefined(key\_value[1]) ? tryDecodeURIComponent(key\_value[1]) : true;

if (!obj[key]) {

obj[key] = val;

} else if(isArray(obj[key])) {

obj[key].push(val);

} else {

obj[key] = [obj[key],val];

}

}

}

});

return obj;

}

function toKeyValue(obj) {

var parts = [];

forEach(obj, function(value, key) {

if (isArray(value)) {

forEach(value, function(arrayValue) {

parts.push(encodeUriQuery(key, true) +

(arrayValue === true ? '' : '=' + encodeUriQuery(arrayValue, true)));

});

} else {

parts.push(encodeUriQuery(key, true) +

(value === true ? '' : '=' + encodeUriQuery(value, true)));

}

});

return parts.length ? parts.join('&') : '';

}

/\*\*

\* We need our custom method because encodeURIComponent is too aggressive and doesn't follow

\* http://www.ietf.org/rfc/rfc3986.txt with regards to the character set (pchar) allowed in path

\* segments:

\* segment = \*pchar

\* pchar = unreserved / pct-encoded / sub-delims / ":" / "@"

\* pct-encoded = "%" HEXDIG HEXDIG

\* unreserved = ALPHA / DIGIT / "-" / "." / "\_" / "~"

\* sub-delims = "!" / "$" / "&" / "'" / "(" / ")"

\* / "\*" / "+" / "," / ";" / "="

\*/

function encodeUriSegment(val) {

return encodeUriQuery(val, true).

replace(/%26/gi, '&').

replace(/%3D/gi, '=').

replace(/%2B/gi, '+');

}

/\*\*

\* This method is intended for encoding \*key\* or \*value\* parts of query component. We need a custom

\* method because encodeURIComponent is too aggressive and encodes stuff that doesn't have to be

\* encoded per http://tools.ietf.org/html/rfc3986:

\* query = \*( pchar / "/" / "?" )

\* pchar = unreserved / pct-encoded / sub-delims / ":" / "@"

\* unreserved = ALPHA / DIGIT / "-" / "." / "\_" / "~"

\* pct-encoded = "%" HEXDIG HEXDIG

\* sub-delims = "!" / "$" / "&" / "'" / "(" / ")"

\* / "\*" / "+" / "," / ";" / "="

\*/

function encodeUriQuery(val, pctEncodeSpaces) {

return encodeURIComponent(val).

replace(/%40/gi, '@').

replace(/%3A/gi, ':').

replace(/%24/g, '$').

replace(/%2C/gi, ',').

replace(/%20/g, (pctEncodeSpaces ? '%20' : '+'));

}

/\*\*

\* @ngdoc directive

\* @name ngApp

\* @module ng

\*

\* @element ANY

\* @param {angular.Module} ngApp an optional application

\* {@link angular.module module} name to load.

\*

\* @description

\*

\* Use this directive to \*\*auto-bootstrap\*\* an AngularJS application. The `ngApp` directive

\* designates the \*\*root element\*\* of the application and is typically placed near the root element

\* of the page - e.g. on the `<body>` or `<html>` tags.

\*

\* Only one AngularJS application can be auto-bootstrapped per HTML document. The first `ngApp`

\* found in the document will be used to define the root element to auto-bootstrap as an

\* application. To run multiple applications in an HTML document you must manually bootstrap them using

\* {@link angular.bootstrap} instead. AngularJS applications cannot be nested within each other.

\*

\* You can specify an \*\*AngularJS module\*\* to be used as the root module for the application. This

\* module will be loaded into the {@link auto.$injector} when the application is bootstrapped and

\* should contain the application code needed or have dependencies on other modules that will

\* contain the code. See {@link angular.module} for more information.

\*

\* In the example below if the `ngApp` directive were not placed on the `html` element then the

\* document would not be compiled, the `AppController` would not be instantiated and the `{{ a+b }}`

\* would not be resolved to `3`.

\*

\* `ngApp` is the easiest, and most common, way to bootstrap an application.

\*

<example module="ngAppDemo">

<file name="index.html">

<div ng-controller="ngAppDemoController">

I can add: {{a}} + {{b}} = {{ a+b }}

</div>

</file>

<file name="script.js">

angular.module('ngAppDemo', []).controller('ngAppDemoController', function($scope) {

$scope.a = 1;

$scope.b = 2;

});

</file>

</example>

\*

\*/

function angularInit(element, bootstrap) {

var elements = [element],

appElement,

module,

names = ['ng:app', 'ng-app', 'x-ng-app', 'data-ng-app'],

NG\_APP\_CLASS\_REGEXP = /\sng[:\-]app(:\s\*([\w\d\_]+);?)?\s/;

function append(element) {

element && elements.push(element);

}

forEach(names, function(name) {

names[name] = true;

append(document.getElementById(name));

name = name.replace(':', '\\:');

if (element.querySelectorAll) {

forEach(element.querySelectorAll('.' + name), append);

forEach(element.querySelectorAll('.' + name + '\\:'), append);

forEach(element.querySelectorAll('[' + name + ']'), append);

}

});

forEach(elements, function(element) {

if (!appElement) {

var className = ' ' + element.className + ' ';

var match = NG\_APP\_CLASS\_REGEXP.exec(className);

if (match) {

appElement = element;

module = (match[2] || '').replace(/\s+/g, ',');

} else {

forEach(element.attributes, function(attr) {

if (!appElement && names[attr.name]) {

appElement = element;

module = attr.value;

}

});

}

}

});

if (appElement) {

bootstrap(appElement, module ? [module] : []);

}

}

/\*\*

\* @ngdoc function

\* @name angular.bootstrap

\* @module ng

\* @description

\* Use this function to manually start up angular application.

\*

\* See: {@link guide/bootstrap Bootstrap}

\*

\* Note that ngScenario-based end-to-end tests cannot use this function to bootstrap manually.

\* They must use {@link ng.directive:ngApp ngApp}.

\*

\* Angular will detect if it has been loaded into the browser more than once and only allow the

\* first loaded script to be bootstrapped and will report a warning to the browser console for

\* each of the subsequent scripts. This prevents strange results in applications, where otherwise

\* multiple instances of Angular try to work on the DOM.

\*

\* <example name="multi-bootstrap" module="multi-bootstrap">

\* <file name="index.html">

\* <script src="../../../angular.js"></script>

\* <div ng-controller="BrokenTable">

\* <table>

\* <tr>

\* <th ng-repeat="heading in headings">{{heading}}</th>

\* </tr>

\* <tr ng-repeat="filling in fillings">

\* <td ng-repeat="fill in filling">{{fill}}</td>

\* </tr>

\* </table>

\* </div>

\* </file>

\* <file name="controller.js">

\* var app = angular.module('multi-bootstrap', [])

\*

\* .controller('BrokenTable', function($scope) {

\* $scope.headings = ['One', 'Two', 'Three'];

\* $scope.fillings = [[1, 2, 3], ['A', 'B', 'C'], [7, 8, 9]];

\* });

\* </file>

\* <file name="protractor.js" type="protractor">

\* it('should only insert one table cell for each item in $scope.fillings', function() {

\* expect(element.all(by.css('td')).count())

\* .toBe(9);

\* });

\* </file>

\* </example>

\*

\* @param {DOMElement} element DOM element which is the root of angular application.

\* @param {Array<String|Function|Array>=} modules an array of modules to load into the application.

\* Each item in the array should be the name of a predefined module or a (DI annotated)

\* function that will be invoked by the injector as a run block.

\* See: {@link angular.module modules}

\* @returns {auto.$injector} Returns the newly created injector for this app.

\*/

function bootstrap(element, modules) {

var doBootstrap = function() {

element = jqLite(element);

if (element.injector()) {

var tag = (element[0] === document) ? 'document' : startingTag(element);

throw ngMinErr('btstrpd', "App Already Bootstrapped with this Element '{0}'", tag);

}

modules = modules || [];

modules.unshift(['$provide', function($provide) {

$provide.value('$rootElement', element);

}]);

modules.unshift('ng');

var injector = createInjector(modules);

injector.invoke(['$rootScope', '$rootElement', '$compile', '$injector', '$animate',

function(scope, element, compile, injector, animate) {

scope.$apply(function() {

element.data('$injector', injector);

compile(element)(scope);

});

}]

);

return injector;

};

var NG\_DEFER\_BOOTSTRAP = /^NG\_DEFER\_BOOTSTRAP!/;

if (window && !NG\_DEFER\_BOOTSTRAP.test(window.name)) {

return doBootstrap();

}

window.name = window.name.replace(NG\_DEFER\_BOOTSTRAP, '');

angular.resumeBootstrap = function(extraModules) {

forEach(extraModules, function(module) {

modules.push(module);

});

doBootstrap();

};

}

var SNAKE\_CASE\_REGEXP = /[A-Z]/g;

function snake\_case(name, separator){

separator = separator || '\_';

return name.replace(SNAKE\_CASE\_REGEXP, function(letter, pos) {

return (pos ? separator : '') + letter.toLowerCase();

});

}

function bindJQuery() {

// bind to jQuery if present;

jQuery = window.jQuery;

// reset to jQuery or default to us.

if (jQuery) {

jqLite = jQuery;

extend(jQuery.fn, {

scope: JQLitePrototype.scope,

isolateScope: JQLitePrototype.isolateScope,

controller: JQLitePrototype.controller,

injector: JQLitePrototype.injector,

inheritedData: JQLitePrototype.inheritedData

});

// Method signature:

// jqLitePatchJQueryRemove(name, dispatchThis, filterElems, getterIfNoArguments)

jqLitePatchJQueryRemove('remove', true, true, false);

jqLitePatchJQueryRemove('empty', false, false, false);

jqLitePatchJQueryRemove('html', false, false, true);

} else {

jqLite = JQLite;

}

angular.element = jqLite;

}

/\*\*

\* throw error if the argument is falsy.

\*/

function assertArg(arg, name, reason) {

if (!arg) {

throw ngMinErr('areq', "Argument '{0}' is {1}", (name || '?'), (reason || "required"));

}

return arg;

}

function assertArgFn(arg, name, acceptArrayAnnotation) {

if (acceptArrayAnnotation && isArray(arg)) {

arg = arg[arg.length - 1];

}

assertArg(isFunction(arg), name, 'not a function, got ' +

(arg && typeof arg == 'object' ? arg.constructor.name || 'Object' : typeof arg));

return arg;

}

/\*\*

\* throw error if the name given is hasOwnProperty

\* @param {String} name the name to test

\* @param {String} context the context in which the name is used, such as module or directive

\*/

function assertNotHasOwnProperty(name, context) {

if (name === 'hasOwnProperty') {

throw ngMinErr('badname', "hasOwnProperty is not a valid {0} name", context);

}

}

/\*\*

\* Return the value accessible from the object by path. Any undefined traversals are ignored

\* @param {Object} obj starting object

\* @param {String} path path to traverse

\* @param {boolean} [bindFnToScope=true]

\* @returns {Object} value as accessible by path

\*/

//TODO(misko): this function needs to be removed

function getter(obj, path, bindFnToScope) {

if (!path) return obj;

var keys = path.split('.');

var key;

var lastInstance = obj;

var len = keys.length;

for (var i = 0; i < len; i++) {

key = keys[i];

if (obj) {

obj = (lastInstance = obj)[key];

}

}

if (!bindFnToScope && isFunction(obj)) {

return bind(lastInstance, obj);

}

return obj;

}

/\*\*

\* Return the DOM siblings between the first and last node in the given array.

\* @param {Array} array like object

\* @returns {DOMElement} object containing the elements

\*/

function getBlockElements(nodes) {

var startNode = nodes[0],

endNode = nodes[nodes.length - 1];

if (startNode === endNode) {

return jqLite(startNode);

}

var element = startNode;

var elements = [element];

do {

element = element.nextSibling;

if (!element) break;

elements.push(element);

} while (element !== endNode);

return jqLite(elements);

}

/\*\*

\* @ngdoc type

\* @name angular.Module

\* @module ng

\* @description

\*

\* Interface for configuring angular {@link angular.module modules}.

\*/

function setupModuleLoader(window) {

var $injectorMinErr = minErr('$injector');

var ngMinErr = minErr('ng');

function ensure(obj, name, factory) {

return obj[name] || (obj[name] = factory());

}

var angular = ensure(window, 'angular', Object);

// We need to expose `angular.$$minErr` to modules such as `ngResource` that reference it during bootstrap

angular.$$minErr = angular.$$minErr || minErr;

return ensure(angular, 'module', function() {

/\*\* @type {Object.<string, angular.Module>} \*/

var modules = {};

/\*\*

\* @ngdoc function

\* @name angular.module

\* @module ng

\* @description

\*

\* The `angular.module` is a global place for creating, registering and retrieving Angular

\* modules.

\* All modules (angular core or 3rd party) that should be available to an application must be

\* registered using this mechanism.

\*

\* When passed two or more arguments, a new module is created. If passed only one argument, an

\* existing module (the name passed as the first argument to `module`) is retrieved.

\*

\*

\* # Module

\*

\* A module is a collection of services, directives, filters, and configuration information.

\* `angular.module` is used to configure the {@link auto.$injector $injector}.

\*

\* ```js

\* // Create a new module

\* var myModule = angular.module('myModule', []);

\*

\* // register a new service

\* myModule.value('appName', 'MyCoolApp');

\*

\* // configure existing services inside initialization blocks.

\* myModule.config(['$locationProvider', function($locationProvider) {

\* // Configure existing providers

\* $locationProvider.hashPrefix('!');

\* }]);

\* ```

\*

\* Then you can create an injector and load your modules like this:

\*

\* ```js

\* var injector = angular.injector(['ng', 'myModule'])

\* ```

\*

\* However it's more likely that you'll just use

\* {@link ng.directive:ngApp ngApp} or

\* {@link angular.bootstrap} to simplify this process for you.

\*

\* @param {!string} name The name of the module to create or retrieve.

<<<<<\* @param {!Array.<string>=} requires If specified then new module is being created. If

>>>>>\* unspecified then the module is being retrieved for further configuration.

\* @param {Function} configFn Optional configuration function for the module. Same as

\* {@link angular.Module#config Module#config()}.

\* @returns {module} new module with the {@link angular.Module} api.

\*/

return function module(name, requires, configFn) {

var assertNotHasOwnProperty = function(name, context) {

if (name === 'hasOwnProperty') {

throw ngMinErr('badname', 'hasOwnProperty is not a valid {0} name', context);

}

};

assertNotHasOwnProperty(name, 'module');

if (requires && modules.hasOwnProperty(name)) {

modules[name] = null;

}

return ensure(modules, name, function() {

if (!requires) {

throw $injectorMinErr('nomod', "Module '{0}' is not available! You either misspelled " +

"the module name or forgot to load it. If registering a module ensure that you " +

"specify the dependencies as the second argument.", name);

}

/\*\* @type {!Array.<Array.<\*>>} \*/

var invokeQueue = [];

/\*\* @type {!Array.<Function>} \*/

var runBlocks = [];

var config = invokeLater('$injector', 'invoke');

/\*\* @type {angular.Module} \*/

var moduleInstance = {

// Private state

\_invokeQueue: invokeQueue,

\_runBlocks: runBlocks,

/\*\*

\* @ngdoc property

\* @name angular.Module#requires

\* @module ng

\* @returns {Array.<string>} List of module names which must be loaded before this module.

\* @description

\* Holds the list of modules which the injector will load before the current module is

\* loaded.

\*/

requires: requires,

/\*\*

\* @ngdoc property

\* @name angular.Module#name

\* @module ng

\* @returns {string} Name of the module.

\* @description

\*/

name: name,

/\*\*

\* @ngdoc method

\* @name angular.Module#provider

\* @module ng

\* @param {string} name service name

\* @param {Function} providerType Construction function for creating new instance of the

\* service.

\* @description

\* See {@link auto.$provide#provider $provide.provider()}.

\*/

provider: invokeLater('$provide', 'provider'),

/\*\*

\* @ngdoc method

\* @name angular.Module#factory

\* @module ng

\* @param {string} name service name

\* @param {Function} providerFunction Function for creating new instance of the service.

\* @description

\* See {@link auto.$provide#factory $provide.factory()}.

\*/

factory: invokeLater('$provide', 'factory'),

/\*\*

\* @ngdoc method

\* @name angular.Module#service

\* @module ng

\* @param {string} name service name

\* @param {Function} constructor A constructor function that will be instantiated.

\* @description

\* See {@link auto.$provide#service $provide.service()}.

\*/

service: invokeLater('$provide', 'service'),

/\*\*

\* @ngdoc method

\* @name angular.Module#value

\* @module ng

\* @param {string} name service name

\* @param {\*} object Service instance object.

\* @description

\* See {@link auto.$provide#value $provide.value()}.

\*/

value: invokeLater('$provide', 'value'),

/\*\*

\* @ngdoc method

\* @name angular.Module#constant

\* @module ng

\* @param {string} name constant name

\* @param {\*} object Constant value.

\* @description

\* Because the constant are fixed, they get applied before other provide methods.

\* See {@link auto.$provide#constant $provide.constant()}.

\*/

constant: invokeLater('$provide', 'constant', 'unshift'),

/\*\*

\* @ngdoc method

\* @name angular.Module#animation

\* @module ng

\* @param {string} name animation name

\* @param {Function} animationFactory Factory function for creating new instance of an

\* animation.

\* @description

\*

\* \*\*NOTE\*\*: animations take effect only if the \*\*ngAnimate\*\* module is loaded.

\*

\*

\* Defines an animation hook that can be later used with

\* {@link ngAnimate.$animate $animate} service and directives that use this service.

\*

\* ```js

\* module.animation('.animation-name', function($inject1, $inject2) {

\* return {

\* eventName : function(element, done) {

\* //code to run the animation

\* //once complete, then run done()

\* return function cancellationFunction(element) {

\* //code to cancel the animation

\* }

\* }

\* }

\* })

\* ```

\*

\* See {@link ngAnimate.$animateProvider#register $animateProvider.register()} and

\* {@link ngAnimate ngAnimate module} for more information.

\*/

animation: invokeLater('$animateProvider', 'register'),

/\*\*

\* @ngdoc method

\* @name angular.Module#filter

\* @module ng

\* @param {string} name Filter name.

\* @param {Function} filterFactory Factory function for creating new instance of filter.

\* @description

\* See {@link ng.$filterProvider#register $filterProvider.register()}.

\*/

filter: invokeLater('$filterProvider', 'register'),

/\*\*

\* @ngdoc method

\* @name angular.Module#controller

\* @module ng

\* @param {string|Object} name Controller name, or an object map of controllers where the

\* keys are the names and the values are the constructors.

\* @param {Function} constructor Controller constructor function.

\* @description

\* See {@link ng.$controllerProvider#register $controllerProvider.register()}.

\*/

controller: invokeLater('$controllerProvider', 'register'),

/\*\*

\* @ngdoc method

\* @name angular.Module#directive

\* @module ng

\* @param {string|Object} name Directive name, or an object map of directives where the

\* keys are the names and the values are the factories.

\* @param {Function} directiveFactory Factory function for creating new instance of

\* directives.

\* @description

\* See {@link ng.$compileProvider#directive $compileProvider.directive()}.

\*/

directive: invokeLater('$compileProvider', 'directive'),

/\*\*

\* @ngdoc method

\* @name angular.Module#config

\* @module ng

\* @param {Function} configFn Execute this function on module load. Useful for service

\* configuration.

\* @description

\* Use this method to register work which needs to be performed on module loading.

\*/

config: config,

/\*\*

\* @ngdoc method

\* @name angular.Module#run

\* @module ng

\* @param {Function} initializationFn Execute this function after injector creation.

\* Useful for application initialization.

\* @description

\* Use this method to register work which should be performed when the injector is done

\* loading all modules.

\*/

run: function(block) {

runBlocks.push(block);

return this;

}

};

if (configFn) {

config(configFn);

}

return moduleInstance;

/\*\*

\* @param {string} provider

\* @param {string} method

\* @param {String=} insertMethod

\* @returns {angular.Module}

\*/

function invokeLater(provider, method, insertMethod) {

return function() {

invokeQueue[insertMethod || 'push']([provider, method, arguments]);

return moduleInstance;

};

}

});

};

});

}

/\* global

angularModule: true,

version: true,

$LocaleProvider,

$CompileProvider,

htmlAnchorDirective,

inputDirective,

inputDirective,

formDirective,

scriptDirective,

selectDirective,

styleDirective,

optionDirective,

ngBindDirective,

ngBindHtmlDirective,

ngBindTemplateDirective,

ngClassDirective,

ngClassEvenDirective,

ngClassOddDirective,

ngCspDirective,

ngCloakDirective,

ngControllerDirective,

ngFormDirective,

ngHideDirective,

ngIfDirective,

ngIncludeDirective,

ngIncludeFillContentDirective,

ngInitDirective,

ngNonBindableDirective,

ngPluralizeDirective,

ngRepeatDirective,

ngShowDirective,

ngStyleDirective,

ngSwitchDirective,

ngSwitchWhenDirective,

ngSwitchDefaultDirective,

ngOptionsDirective,

ngTranscludeDirective,

ngModelDirective,

ngListDirective,

ngChangeDirective,

requiredDirective,

requiredDirective,

ngValueDirective,

ngAttributeAliasDirectives,

ngEventDirectives,

$AnchorScrollProvider,

$AnimateProvider,

$BrowserProvider,

$CacheFactoryProvider,

$ControllerProvider,

$DocumentProvider,

$ExceptionHandlerProvider,

$FilterProvider,

$InterpolateProvider,

$IntervalProvider,

$HttpProvider,

$HttpBackendProvider,

$LocationProvider,

$LogProvider,

$ParseProvider,

$RootScopeProvider,

$QProvider,

$$SanitizeUriProvider,

$SceProvider,

$SceDelegateProvider,

$SnifferProvider,

$TemplateCacheProvider,

$TimeoutProvider,

$$RAFProvider,

$$AsyncCallbackProvider,

$WindowProvider

\*/

/\*\*

\* @ngdoc object

\* @name angular.version

\* @module ng

\* @description

\* An object that contains information about the current AngularJS version. This object has the

\* following properties:

\*

\* - `full` – `{string}` – Full version string, such as "0.9.18".

\* - `major` – `{number}` – Major version number, such as "0".

\* - `minor` – `{number}` – Minor version number, such as "9".

\* - `dot` – `{number}` – Dot version number, such as "18".

\* - `codeName` – `{string}` – Code name of the release, such as "jiggling-armfat".

\*/

var version = {

full: '1.2.16', // all of these placeholder strings will be replaced by grunt's

major: 1, // package task

minor: 2,

dot: 16,

codeName: 'badger-enumeration'

};

function publishExternalAPI(angular){

extend(angular, {

'bootstrap': bootstrap,

'copy': copy,

'extend': extend,

'equals': equals,

'element': jqLite,

'forEach': forEach,

'injector': createInjector,

'noop':noop,

'bind':bind,

'toJson': toJson,

'fromJson': fromJson,

'identity':identity,

'isUndefined': isUndefined,

'isDefined': isDefined,

'isString': isString,

'isFunction': isFunction,

'isObject': isObject,

'isNumber': isNumber,

'isElement': isElement,

'isArray': isArray,

'version': version,

'isDate': isDate,

'lowercase': lowercase,

'uppercase': uppercase,

'callbacks': {counter: 0},

'$$minErr': minErr,

'$$csp': csp

});

angularModule = setupModuleLoader(window);

try {

angularModule('ngLocale');

} catch (e) {

angularModule('ngLocale', []).provider('$locale', $LocaleProvider);

}

angularModule('ng', ['ngLocale'], ['$provide',

function ngModule($provide) {

// $$sanitizeUriProvider needs to be before $compileProvider as it is used by it.

$provide.provider({

$$sanitizeUri: $$SanitizeUriProvider

});

$provide.provider('$compile', $CompileProvider).

directive({

a: htmlAnchorDirective,

input: inputDirective,

textarea: inputDirective,

form: formDirective,

script: scriptDirective,

select: selectDirective,

style: styleDirective,

option: optionDirective,

ngBind: ngBindDirective,

ngBindHtml: ngBindHtmlDirective,

ngBindTemplate: ngBindTemplateDirective,

ngClass: ngClassDirective,

ngClassEven: ngClassEvenDirective,

ngClassOdd: ngClassOddDirective,

ngCloak: ngCloakDirective,

ngController: ngControllerDirective,

ngForm: ngFormDirective,

ngHide: ngHideDirective,

ngIf: ngIfDirective,

ngInclude: ngIncludeDirective,

ngInit: ngInitDirective,

ngNonBindable: ngNonBindableDirective,

ngPluralize: ngPluralizeDirective,

ngRepeat: ngRepeatDirective,

ngShow: ngShowDirective,

ngStyle: ngStyleDirective,

ngSwitch: ngSwitchDirective,

ngSwitchWhen: ngSwitchWhenDirective,

ngSwitchDefault: ngSwitchDefaultDirective,

ngOptions: ngOptionsDirective,

ngTransclude: ngTranscludeDirective,

ngModel: ngModelDirective,

ngList: ngListDirective,

ngChange: ngChangeDirective,

required: requiredDirective,

ngRequired: requiredDirective,

ngValue: ngValueDirective

}).

directive({

ngInclude: ngIncludeFillContentDirective

}).

directive(ngAttributeAliasDirectives).

directive(ngEventDirectives);

$provide.provider({

$anchorScroll: $AnchorScrollProvider,

$animate: $AnimateProvider,

$browser: $BrowserProvider,

$cacheFactory: $CacheFactoryProvider,

$controller: $ControllerProvider,

$document: $DocumentProvider,

$exceptionHandler: $ExceptionHandlerProvider,

$filter: $FilterProvider,

$interpolate: $InterpolateProvider,

$interval: $IntervalProvider,

$http: $HttpProvider,

$httpBackend: $HttpBackendProvider,

$location: $LocationProvider,

$log: $LogProvider,

$parse: $ParseProvider,

$rootScope: $RootScopeProvider,

$q: $QProvider,

$sce: $SceProvider,

$sceDelegate: $SceDelegateProvider,

$sniffer: $SnifferProvider,

$templateCache: $TemplateCacheProvider,

$timeout: $TimeoutProvider,

$window: $WindowProvider,

$$rAF: $$RAFProvider,

$$asyncCallback : $$AsyncCallbackProvider

});

}

]);

}

/\* global

-JQLitePrototype,

-addEventListenerFn,

-removeEventListenerFn,

-BOOLEAN\_ATTR

\*/

//////////////////////////////////

//JQLite

//////////////////////////////////

/\*\*

\* @ngdoc function

\* @name angular.element

\* @module ng

\* @function

\*

\* @description

\* Wraps a raw DOM element or HTML string as a [jQuery](http://jquery.com) element.

\*

\* If jQuery is available, `angular.element` is an alias for the

\* [jQuery](http://api.jquery.com/jQuery/) function. If jQuery is not available, `angular.element`

\* delegates to Angular's built-in subset of jQuery, called "jQuery lite" or "jqLite."

\*

\* <div class="alert alert-success">jqLite is a tiny, API-compatible subset of jQuery that allows

\* Angular to manipulate the DOM in a cross-browser compatible way. \*\*jqLite\*\* implements only the most

\* commonly needed functionality with the goal of having a very small footprint.</div>

\*

\* To use jQuery, simply load it before `DOMContentLoaded` event fired.

\*

\* <div class="alert">\*\*Note:\*\* all element references in Angular are always wrapped with jQuery or

\* jqLite; they are never raw DOM references.</div>

\*

\* ## Angular's jqLite

\* jqLite provides only the following jQuery methods:

\*

\* - [`addClass()`](http://api.jquery.com/addClass/)

\* - [`after()`](http://api.jquery.com/after/)

\* - [`append()`](http://api.jquery.com/append/)

\* - [`attr()`](http://api.jquery.com/attr/)

\* - [`bind()`](http://api.jquery.com/bind/) - Does not support namespaces, selectors or eventData

\* - [`children()`](http://api.jquery.com/children/) - Does not support selectors

\* - [`clone()`](http://api.jquery.com/clone/)

\* - [`contents()`](http://api.jquery.com/contents/)

\* - [`css()`](http://api.jquery.com/css/)

\* - [`data()`](http://api.jquery.com/data/)

\* - [`empty()`](http://api.jquery.com/empty/)

\* - [`eq()`](http://api.jquery.com/eq/)

\* - [`find()`](http://api.jquery.com/find/) - Limited to lookups by tag name

\* - [`hasClass()`](http://api.jquery.com/hasClass/)

\* - [`html()`](http://api.jquery.com/html/)

\* - [`next()`](http://api.jquery.com/next/) - Does not support selectors

\* - [`on()`](http://api.jquery.com/on/) - Does not support namespaces, selectors or eventData

\* - [`off()`](http://api.jquery.com/off/) - Does not support namespaces or selectors

\* - [`one()`](http://api.jquery.com/one/) - Does not support namespaces or selectors

\* - [`parent()`](http://api.jquery.com/parent/) - Does not support selectors

\* - [`prepend()`](http://api.jquery.com/prepend/)

\* - [`prop()`](http://api.jquery.com/prop/)

\* - [`ready()`](http://api.jquery.com/ready/)

\* - [`remove()`](http://api.jquery.com/remove/)

\* - [`removeAttr()`](http://api.jquery.com/removeAttr/)

\* - [`removeClass()`](http://api.jquery.com/removeClass/)

\* - [`removeData()`](http://api.jquery.com/removeData/)

\* - [`replaceWith()`](http://api.jquery.com/replaceWith/)

\* - [`text()`](http://api.jquery.com/text/)

\* - [`toggleClass()`](http://api.jquery.com/toggleClass/)

\* - [`triggerHandler()`](http://api.jquery.com/triggerHandler/) - Passes a dummy event object to handlers.

\* - [`unbind()`](http://api.jquery.com/unbind/) - Does not support namespaces

\* - [`val()`](http://api.jquery.com/val/)

\* - [`wrap()`](http://api.jquery.com/wrap/)

\*

\* ## jQuery/jqLite Extras

\* Angular also provides the following additional methods and events to both jQuery and jqLite:

\*

\* ### Events

\* - `$destroy` - AngularJS intercepts all jqLite/jQuery's DOM destruction apis and fires this event

\* on all DOM nodes being removed. This can be used to clean up any 3rd party bindings to the DOM

\* element before it is removed.

\*

\* ### Methods

\* - `controller(name)` - retrieves the controller of the current element or its parent. By default

\* retrieves controller associated with the `ngController` directive. If `name` is provided as

\* camelCase directive name, then the controller for this directive will be retrieved (e.g.

\* `'ngModel'`).

\* - `injector()` - retrieves the injector of the current element or its parent.

\* - `scope()` - retrieves the {@link ng.$rootScope.Scope scope} of the current

\* element or its parent.

\* - `isolateScope()` - retrieves an isolate {@link ng.$rootScope.Scope scope} if one is attached directly to the

\* current element. This getter should be used only on elements that contain a directive which starts a new isolate

\* scope. Calling `scope()` on this element always returns the original non-isolate scope.

\* - `inheritedData()` - same as `data()`, but walks up the DOM until a value is found or the top

\* parent element is reached.

\*

\* @param {string|DOMElement} element HTML string or DOMElement to be wrapped into jQuery.

\* @returns {Object} jQuery object.

\*/

var jqCache = JQLite.cache = {},

jqName = JQLite.expando = 'ng-' + new Date().getTime(),

jqId = 1,

addEventListenerFn = (window.document.addEventListener

? function(element, type, fn) {element.addEventListener(type, fn, false);}

: function(element, type, fn) {element.attachEvent('on' + type, fn);}),

removeEventListenerFn = (window.document.removeEventListener

? function(element, type, fn) {element.removeEventListener(type, fn, false); }

: function(element, type, fn) {element.detachEvent('on' + type, fn); });

/\*

\* !!! This is an undocumented "private" function !!!

\*/

var jqData = JQLite.\_data = function(node) {

//jQuery always returns an object on cache miss

return this.cache[node[this.expando]] || {};

};

function jqNextId() { return ++jqId; }

var SPECIAL\_CHARS\_REGEXP = /([\:\-\\_]+(.))/g;

var MOZ\_HACK\_REGEXP = /^moz([A-Z])/;

var jqLiteMinErr = minErr('jqLite');

/\*\*

\* Converts snake\_case to camelCase.

\* Also there is special case for Moz prefix starting with upper case letter.

\* @param name Name to normalize

\*/

function camelCase(name) {

return name.

replace(SPECIAL\_CHARS\_REGEXP, function(\_, separator, letter, offset) {

return offset ? letter.toUpperCase() : letter;

}).

replace(MOZ\_HACK\_REGEXP, 'Moz$1');

}

/////////////////////////////////////////////

// jQuery mutation patch

//

// In conjunction with bindJQuery intercepts all jQuery's DOM destruction apis and fires a

// $destroy event on all DOM nodes being removed.

//

/////////////////////////////////////////////

function jqLitePatchJQueryRemove(name, dispatchThis, filterElems, getterIfNoArguments) {

var originalJqFn = jQuery.fn[name];

originalJqFn = originalJqFn.$original || originalJqFn;

removePatch.$original = originalJqFn;

jQuery.fn[name] = removePatch;

function removePatch(param) {

// jshint -W040

var list = filterElems && param ? [this.filter(param)] : [this],

fireEvent = dispatchThis,

set, setIndex, setLength,

element, childIndex, childLength, children;

if (!getterIfNoArguments || param != null) {

while(list.length) {

set = list.shift();

for(setIndex = 0, setLength = set.length; setIndex < setLength; setIndex++) {

element = jqLite(set[setIndex]);

if (fireEvent) {

element.triggerHandler('$destroy');

} else {

fireEvent = !fireEvent;

}

for(childIndex = 0, childLength = (children = element.children()).length;

childIndex < childLength;

childIndex++) {

list.push(jQuery(children[childIndex]));

}

}

}

}

return originalJqFn.apply(this, arguments);

}

}

var SINGLE\_TAG\_REGEXP = /^<(\w+)\s\*\/?>(?:<\/\1>|)$/;

var HTML\_REGEXP = /<|&#?\w+;/;

var TAG\_NAME\_REGEXP = /<([\w:]+)/;

var XHTML\_TAG\_REGEXP = /<(?!area|br|col|embed|hr|img|input|link|meta|param)(([\w:]+)[^>]\*)\/>/gi;

var wrapMap = {

'option': [1, '<select multiple="multiple">', '</select>'],

'thead': [1, '<table>', '</table>'],

'col': [2, '<table><colgroup>', '</colgroup></table>'],

'tr': [2, '<table><tbody>', '</tbody></table>'],

'td': [3, '<table><tbody><tr>', '</tr></tbody></table>'],

'\_default': [0, "", ""]

};

wrapMap.optgroup = wrapMap.option;

wrapMap.tbody = wrapMap.tfoot = wrapMap.colgroup = wrapMap.caption = wrapMap.thead;

wrapMap.th = wrapMap.td;

function jqLiteIsTextNode(html) {

return !HTML\_REGEXP.test(html);

}

function jqLiteBuildFragment(html, context) {

var elem, tmp, tag, wrap,

fragment = context.createDocumentFragment(),

nodes = [], i, j, jj;

if (jqLiteIsTextNode(html)) {

// Convert non-html into a text node

nodes.push(context.createTextNode(html));

} else {

tmp = fragment.appendChild(context.createElement('div'));

// Convert html into DOM nodes

tag = (TAG\_NAME\_REGEXP.exec(html) || ["", ""])[1].toLowerCase();

wrap = wrapMap[tag] || wrapMap.\_default;

tmp.innerHTML = '<div>&#160;</div>' +

wrap[1] + html.replace(XHTML\_TAG\_REGEXP, "<$1></$2>") + wrap[2];

tmp.removeChild(tmp.firstChild);

// Descend through wrappers to the right content

i = wrap[0];

while (i--) {

tmp = tmp.lastChild;

}

for (j=0, jj=tmp.childNodes.length; j<jj; ++j) nodes.push(tmp.childNodes[j]);

tmp = fragment.firstChild;

tmp.textContent = "";

}

// Remove wrapper from fragment

fragment.textContent = "";

fragment.innerHTML = ""; // Clear inner HTML

return nodes;

}

function jqLiteParseHTML(html, context) {

context = context || document;

var parsed;

if ((parsed = SINGLE\_TAG\_REGEXP.exec(html))) {

return [context.createElement(parsed[1])];

}

return jqLiteBuildFragment(html, context);

}

/////////////////////////////////////////////

function JQLite(element) {

if (element instanceof JQLite) {

return element;

}

if (isString(element)) {

element = trim(element);

}

if (!(this instanceof JQLite)) {

if (isString(element) && element.charAt(0) != '<') {

throw jqLiteMinErr('nosel', 'Looking up elements via selectors is not supported by jqLite! See: http://docs.angularjs.org/api/angular.element');

}

return new JQLite(element);

}

if (isString(element)) {

jqLiteAddNodes(this, jqLiteParseHTML(element));

var fragment = jqLite(document.createDocumentFragment());

fragment.append(this);

} else {

jqLiteAddNodes(this, element);

}

}

function jqLiteClone(element) {

return element.cloneNode(true);

}

function jqLiteDealoc(element){

jqLiteRemoveData(element);

for ( var i = 0, children = element.childNodes || []; i < children.length; i++) {

jqLiteDealoc(children[i]);

}

}

function jqLiteOff(element, type, fn, unsupported) {

if (isDefined(unsupported)) throw jqLiteMinErr('offargs', 'jqLite#off() does not support the `selector` argument');

var events = jqLiteExpandoStore(element, 'events'),

handle = jqLiteExpandoStore(element, 'handle');

if (!handle) return; //no listeners registered

if (isUndefined(type)) {

forEach(events, function(eventHandler, type) {

removeEventListenerFn(element, type, eventHandler);

delete events[type];

});

} else {

forEach(type.split(' '), function(type) {

if (isUndefined(fn)) {

removeEventListenerFn(element, type, events[type]);

delete events[type];

} else {

arrayRemove(events[type] || [], fn);

}

});

}

}

function jqLiteRemoveData(element, name) {

var expandoId = element[jqName],

expandoStore = jqCache[expandoId];

if (expandoStore) {

if (name) {

delete jqCache[expandoId].data[name];

return;

}

if (expandoStore.handle) {

expandoStore.events.$destroy && expandoStore.handle({}, '$destroy');

jqLiteOff(element);

}

delete jqCache[expandoId];

element[jqName] = undefined; // ie does not allow deletion of attributes on elements.

}

}

function jqLiteExpandoStore(element, key, value) {

var expandoId = element[jqName],

expandoStore = jqCache[expandoId || -1];

if (isDefined(value)) {

if (!expandoStore) {

element[jqName] = expandoId = jqNextId();

expandoStore = jqCache[expandoId] = {};

}

expandoStore[key] = value;

} else {

return expandoStore && expandoStore[key];

}

}

function jqLiteData(element, key, value) {

var data = jqLiteExpandoStore(element, 'data'),

isSetter = isDefined(value),

keyDefined = !isSetter && isDefined(key),

isSimpleGetter = keyDefined && !isObject(key);

if (!data && !isSimpleGetter) {

jqLiteExpandoStore(element, 'data', data = {});

}

if (isSetter) {

data[key] = value;

} else {

if (keyDefined) {

if (isSimpleGetter) {

// don't create data in this case.

return data && data[key];

} else {

extend(data, key);

}

} else {

return data;

}

}

}

function jqLiteHasClass(element, selector) {

if (!element.getAttribute) return false;

return ((" " + (element.getAttribute('class') || '') + " ").replace(/[\n\t]/g, " ").

indexOf( " " + selector + " " ) > -1);

}

function jqLiteRemoveClass(element, cssClasses) {

if (cssClasses && element.setAttribute) {

forEach(cssClasses.split(' '), function(cssClass) {

element.setAttribute('class', trim(

(" " + (element.getAttribute('class') || '') + " ")

.replace(/[\n\t]/g, " ")

.replace(" " + trim(cssClass) + " ", " "))

);

});

}

}

function jqLiteAddClass(element, cssClasses) {

if (cssClasses && element.setAttribute) {

var existingClasses = (' ' + (element.getAttribute('class') || '') + ' ')

.replace(/[\n\t]/g, " ");

forEach(cssClasses.split(' '), function(cssClass) {

cssClass = trim(cssClass);

if (existingClasses.indexOf(' ' + cssClass + ' ') === -1) {

existingClasses += cssClass + ' ';

}

});

element.setAttribute('class', trim(existingClasses));

}

}

function jqLiteAddNodes(root, elements) {

if (elements) {

elements = (!elements.nodeName && isDefined(elements.length) && !isWindow(elements))

? elements

: [ elements ];

for(var i=0; i < elements.length; i++) {

root.push(elements[i]);

}

}

}

function jqLiteController(element, name) {

return jqLiteInheritedData(element, '$' + (name || 'ngController' ) + 'Controller');

}

function jqLiteInheritedData(element, name, value) {

element = jqLite(element);

// if element is the document object work with the html element instead

// this makes $(document).scope() possible

if(element[0].nodeType == 9) {

element = element.find('html');

}

var names = isArray(name) ? name : [name];

while (element.length) {

var node = element[0];

for (var i = 0, ii = names.length; i < ii; i++) {

if ((value = element.data(names[i])) !== undefined) return value;

}

// If dealing with a document fragment node with a host element, and no parent, use the host

// element as the parent. This enables directives within a Shadow DOM or polyfilled Shadow DOM

// to lookup parent controllers.

element = jqLite(node.parentNode || (node.nodeType === 11 && node.host));

}

}

function jqLiteEmpty(element) {

for (var i = 0, childNodes = element.childNodes; i < childNodes.length; i++) {

jqLiteDealoc(childNodes[i]);

}

while (element.firstChild) {

element.removeChild(element.firstChild);

}

}

//////////////////////////////////////////

// Functions which are declared directly.

//////////////////////////////////////////

var JQLitePrototype = JQLite.prototype = {

ready: function(fn) {

var fired = false;

function trigger() {

if (fired) return;

fired = true;

fn();

}

// check if document already is loaded

if (document.readyState === 'complete'){

setTimeout(trigger);

} else {

this.on('DOMContentLoaded', trigger); // works for modern browsers and IE9

// we can not use jqLite since we are not done loading and jQuery could be loaded later.

// jshint -W064

JQLite(window).on('load', trigger); // fallback to window.onload for others

// jshint +W064

}

},

toString: function() {

var value = [];

forEach(this, function(e){ value.push('' + e);});

return '[' + value.join(', ') + ']';

},

eq: function(index) {

return (index >= 0) ? jqLite(this[index]) : jqLite(this[this.length + index]);

},

length: 0,

push: push,

sort: [].sort,

splice: [].splice

};

//////////////////////////////////////////

// Functions iterating getter/setters.

// these functions return self on setter and

// value on get.

//////////////////////////////////////////

var BOOLEAN\_ATTR = {};

forEach('multiple,selected,checked,disabled,readOnly,required,open'.split(','), function(value) {

BOOLEAN\_ATTR[lowercase(value)] = value;

});

var BOOLEAN\_ELEMENTS = {};

forEach('input,select,option,textarea,button,form,details'.split(','), function(value) {

BOOLEAN\_ELEMENTS[uppercase(value)] = true;

});

function getBooleanAttrName(element, name) {

// check dom last since we will most likely fail on name

var booleanAttr = BOOLEAN\_ATTR[name.toLowerCase()];

// booleanAttr is here twice to minimize DOM access

return booleanAttr && BOOLEAN\_ELEMENTS[element.nodeName] && booleanAttr;

}

forEach({

data: jqLiteData,

inheritedData: jqLiteInheritedData,

scope: function(element) {

// Can't use jqLiteData here directly so we stay compatible with jQuery!

return jqLite(element).data('$scope') || jqLiteInheritedData(element.parentNode || element, ['$isolateScope', '$scope']);

},

isolateScope: function(element) {

// Can't use jqLiteData here directly so we stay compatible with jQuery!

return jqLite(element).data('$isolateScope') || jqLite(element).data('$isolateScopeNoTemplate');

},

controller: jqLiteController,

injector: function(element) {

return jqLiteInheritedData(element, '$injector');

},

removeAttr: function(element,name) {

element.removeAttribute(name);

},

hasClass: jqLiteHasClass,

css: function(element, name, value) {

name = camelCase(name);

if (isDefined(value)) {

element.style[name] = value;

} else {

var val;

if (msie <= 8) {

// this is some IE specific weirdness that jQuery 1.6.4 does not sure why

val = element.currentStyle && element.currentStyle[name];

if (val === '') val = 'auto';

}

val = val || element.style[name];

if (msie <= 8) {

// jquery weirdness :-/

val = (val === '') ? undefined : val;

}

return val;

}

},

attr: function(element, name, value){

var lowercasedName = lowercase(name);

if (BOOLEAN\_ATTR[lowercasedName]) {

if (isDefined(value)) {

if (!!value) {

element[name] = true;

element.setAttribute(name, lowercasedName);

} else {

element[name] = false;

element.removeAttribute(lowercasedName);

}

} else {

return (element[name] ||

(element.attributes.getNamedItem(name)|| noop).specified)

? lowercasedName

: undefined;

}

} else if (isDefined(value)) {

element.setAttribute(name, value);

} else if (element.getAttribute) {

// the extra argument "2" is to get the right thing for a.href in IE, see jQuery code

// some elements (e.g. Document) don't have get attribute, so return undefined

var ret = element.getAttribute(name, 2);

// normalize non-existing attributes to undefined (as jQuery)

return ret === null ? undefined : ret;

}

},

prop: function(element, name, value) {

if (isDefined(value)) {

element[name] = value;

} else {

return element[name];

}

},

text: (function() {

var NODE\_TYPE\_TEXT\_PROPERTY = [];

if (msie < 9) {

NODE\_TYPE\_TEXT\_PROPERTY[1] = 'innerText'; /\*\* Element \*\*/

NODE\_TYPE\_TEXT\_PROPERTY[3] = 'nodeValue'; /\*\* Text \*\*/

} else {

NODE\_TYPE\_TEXT\_PROPERTY[1] = /\*\* Element \*\*/

NODE\_TYPE\_TEXT\_PROPERTY[3] = 'textContent'; /\*\* Text \*\*/

}

getText.$dv = '';

return getText;

function getText(element, value) {

var textProp = NODE\_TYPE\_TEXT\_PROPERTY[element.nodeType];

if (isUndefined(value)) {

return textProp ? element[textProp] : '';

}

element[textProp] = value;

}

})(),

val: function(element, value) {

if (isUndefined(value)) {

if (nodeName\_(element) === 'SELECT' && element.multiple) {

var result = [];

forEach(element.options, function (option) {

if (option.selected) {

result.push(option.value || option.text);

}

});

return result.length === 0 ? null : result;

}

return element.value;

}

element.value = value;

},

html: function(element, value) {

if (isUndefined(value)) {

return element.innerHTML;

}

for (var i = 0, childNodes = element.childNodes; i < childNodes.length; i++) {

jqLiteDealoc(childNodes[i]);

}

element.innerHTML = value;

},

empty: jqLiteEmpty

}, function(fn, name){

/\*\*

\* Properties: writes return selection, reads return first value

\*/

JQLite.prototype[name] = function(arg1, arg2) {

var i, key;

// jqLiteHasClass has only two arguments, but is a getter-only fn, so we need to special-case it

// in a way that survives minification.

// jqLiteEmpty takes no arguments but is a setter.

if (fn !== jqLiteEmpty &&

(((fn.length == 2 && (fn !== jqLiteHasClass && fn !== jqLiteController)) ? arg1 : arg2) === undefined)) {

if (isObject(arg1)) {

// we are a write, but the object properties are the key/values

for (i = 0; i < this.length; i++) {

if (fn === jqLiteData) {

// data() takes the whole object in jQuery

fn(this[i], arg1);

} else {

for (key in arg1) {

fn(this[i], key, arg1[key]);

}

}

}

// return self for chaining

return this;

} else {

// we are a read, so read the first child.

var value = fn.$dv;

// Only if we have $dv do we iterate over all, otherwise it is just the first element.

var jj = (value === undefined) ? Math.min(this.length, 1) : this.length;

for (var j = 0; j < jj; j++) {

var nodeValue = fn(this[j], arg1, arg2);

value = value ? value + nodeValue : nodeValue;

}

return value;

}

} else {

// we are a write, so apply to all children

for (i = 0; i < this.length; i++) {

fn(this[i], arg1, arg2);

}

// return self for chaining

return this;

}

};

});

function createEventHandler(element, events) {

var eventHandler = function (event, type) {

if (!event.preventDefault) {

event.preventDefault = function() {

event.returnValue = false; //ie

};

}

if (!event.stopPropagation) {

event.stopPropagation = function() {

event.cancelBubble = true; //ie

};

}

if (!event.target) {

event.target = event.srcElement || document;

}

if (isUndefined(event.defaultPrevented)) {

var prevent = event.preventDefault;

event.preventDefault = function() {

event.defaultPrevented = true;

prevent.call(event);

};

event.defaultPrevented = false;

}

event.isDefaultPrevented = function() {

return event.defaultPrevented || event.returnValue === false;

};

// Copy event handlers in case event handlers array is modified during execution.

var eventHandlersCopy = shallowCopy(events[type || event.type] || []);

forEach(eventHandlersCopy, function(fn) {

fn.call(element, event);

});

// Remove monkey-patched methods (IE),

// as they would cause memory leaks in IE8.

if (msie <= 8) {

// IE7/8 does not allow to delete property on native object

event.preventDefault = null;

event.stopPropagation = null;

event.isDefaultPrevented = null;

} else {

// It shouldn't affect normal browsers (native methods are defined on prototype).

delete event.preventDefault;

delete event.stopPropagation;

delete event.isDefaultPrevented;

}

};

eventHandler.elem = element;

return eventHandler;

}

//////////////////////////////////////////

// Functions iterating traversal.

// These functions chain results into a single

// selector.

//////////////////////////////////////////

forEach({

removeData: jqLiteRemoveData,

dealoc: jqLiteDealoc,

on: function onFn(element, type, fn, unsupported){

if (isDefined(unsupported)) throw jqLiteMinErr('onargs', 'jqLite#on() does not support the `selector` or `eventData` parameters');

var events = jqLiteExpandoStore(element, 'events'),

handle = jqLiteExpandoStore(element, 'handle');

if (!events) jqLiteExpandoStore(element, 'events', events = {});

if (!handle) jqLiteExpandoStore(element, 'handle', handle = createEventHandler(element, events));

forEach(type.split(' '), function(type){

var eventFns = events[type];

if (!eventFns) {

if (type == 'mouseenter' || type == 'mouseleave') {

var contains = document.body.contains || document.body.compareDocumentPosition ?

function( a, b ) {

// jshint bitwise: false

var adown = a.nodeType === 9 ? a.documentElement : a,

bup = b && b.parentNode;

return a === bup || !!( bup && bup.nodeType === 1 && (

adown.contains ?

adown.contains( bup ) :

a.compareDocumentPosition && a.compareDocumentPosition( bup ) & 16

));

} :

function( a, b ) {

if ( b ) {

while ( (b = b.parentNode) ) {

if ( b === a ) {

return true;

}

}

}

return false;

};

events[type] = [];

// Refer to jQuery's implementation of mouseenter & mouseleave

// Read about mouseenter and mouseleave:

// http://www.quirksmode.org/js/events\_mouse.html#link8

var eventmap = { mouseleave : "mouseout", mouseenter : "mouseover"};

onFn(element, eventmap[type], function(event) {

var target = this, related = event.relatedTarget;

// For mousenter/leave call the handler if related is outside the target.

// NB: No relatedTarget if the mouse left/entered the browser window

if ( !related || (related !== target && !contains(target, related)) ){

handle(event, type);

}

});

} else {

addEventListenerFn(element, type, handle);

events[type] = [];

}

eventFns = events[type];

}

eventFns.push(fn);

});

},

off: jqLiteOff,

one: function(element, type, fn) {

element = jqLite(element);

//add the listener twice so that when it is called

//you can remove the original function and still be

//able to call element.off(ev, fn) normally

element.on(type, function onFn() {

element.off(type, fn);

element.off(type, onFn);

});

element.on(type, fn);

},

replaceWith: function(element, replaceNode) {

var index, parent = element.parentNode;

jqLiteDealoc(element);

forEach(new JQLite(replaceNode), function(node){

if (index) {

parent.insertBefore(node, index.nextSibling);

} else {

parent.replaceChild(node, element);

}

index = node;

});

},

children: function(element) {

var children = [];

forEach(element.childNodes, function(element){

if (element.nodeType === 1)

children.push(element);

});

return children;

},

contents: function(element) {

return element.contentDocument || element.childNodes || [];

},

append: function(element, node) {

forEach(new JQLite(node), function(child){

if (element.nodeType === 1 || element.nodeType === 11) {

element.appendChild(child);

}

});

},

prepend: function(element, node) {

if (element.nodeType === 1) {

var index = element.firstChild;

forEach(new JQLite(node), function(child){

element.insertBefore(child, index);

});

}

},

wrap: function(element, wrapNode) {

wrapNode = jqLite(wrapNode)[0];

var parent = element.parentNode;

if (parent) {

parent.replaceChild(wrapNode, element);

}

wrapNode.appendChild(element);

},

remove: function(element) {

jqLiteDealoc(element);

var parent = element.parentNode;

if (parent) parent.removeChild(element);

},

after: function(element, newElement) {

var index = element, parent = element.parentNode;

forEach(new JQLite(newElement), function(node){

parent.insertBefore(node, index.nextSibling);

index = node;

});

},

addClass: jqLiteAddClass,

removeClass: jqLiteRemoveClass,

toggleClass: function(element, selector, condition) {

if (selector) {

forEach(selector.split(' '), function(className){

var classCondition = condition;

if (isUndefined(classCondition)) {

classCondition = !jqLiteHasClass(element, className);

}

(classCondition ? jqLiteAddClass : jqLiteRemoveClass)(element, className);

});

}

},

parent: function(element) {

var parent = element.parentNode;

return parent && parent.nodeType !== 11 ? parent : null;

},

next: function(element) {

if (element.nextElementSibling) {

return element.nextElementSibling;

}

// IE8 doesn't have nextElementSibling

var elm = element.nextSibling;

while (elm != null && elm.nodeType !== 1) {

elm = elm.nextSibling;

}

return elm;

},

find: function(element, selector) {

if (element.getElementsByTagName) {

return element.getElementsByTagName(selector);

} else {

return [];

}

},

clone: jqLiteClone,

triggerHandler: function(element, eventName, eventData) {

var eventFns = (jqLiteExpandoStore(element, 'events') || {})[eventName];

eventData = eventData || [];

var event = [{

preventDefault: noop,

stopPropagation: noop

}];

forEach(eventFns, function(fn) {

fn.apply(element, event.concat(eventData));

});

}

}, function(fn, name){

/\*\*

\* chaining functions

\*/

JQLite.prototype[name] = function(arg1, arg2, arg3) {

var value;

for(var i=0; i < this.length; i++) {

if (isUndefined(value)) {

value = fn(this[i], arg1, arg2, arg3);

if (isDefined(value)) {

// any function which returns a value needs to be wrapped

value = jqLite(value);

}

} else {

jqLiteAddNodes(value, fn(this[i], arg1, arg2, arg3));

}

}

return isDefined(value) ? value : this;

};

// bind legacy bind/unbind to on/off

JQLite.prototype.bind = JQLite.prototype.on;

JQLite.prototype.unbind = JQLite.prototype.off;

});

/\*\*

\* Computes a hash of an 'obj'.

\* Hash of a:

\* string is string

\* number is number as string

\* object is either result of calling $$hashKey function on the object or uniquely generated id,

\* that is also assigned to the $$hashKey property of the object.

\*

\* @param obj

\* @returns {string} hash string such that the same input will have the same hash string.

\* The resulting string key is in 'type:hashKey' format.

\*/

function hashKey(obj) {

var objType = typeof obj,

key;

if (objType == 'object' && obj !== null) {

if (typeof (key = obj.$$hashKey) == 'function') {

// must invoke on object to keep the right this

key = obj.$$hashKey();

} else if (key === undefined) {

key = obj.$$hashKey = nextUid();

}

} else {

key = obj;

}

return objType + ':' + key;

}

/\*\*

\* HashMap which can use objects as keys

\*/

function HashMap(array){

forEach(array, this.put, this);

}

HashMap.prototype = {

/\*\*

\* Store key value pair

\* @param key key to store can be any type

\* @param value value to store can be any type

\*/

put: function(key, value) {

this[hashKey(key)] = value;

},

/\*\*

\* @param key

\* @returns {Object} the value for the key

\*/

get: function(key) {

return this[hashKey(key)];

},

/\*\*

\* Remove the key/value pair

\* @param key

\*/

remove: function(key) {

var value = this[key = hashKey(key)];

delete this[key];

return value;

}

};

/\*\*

\* @ngdoc function

\* @module ng

\* @name angular.injector

\* @function

\*

\* @description

\* Creates an injector function that can be used for retrieving services as well as for

\* dependency injection (see {@link guide/di dependency injection}).

\*

\* @param {Array.<string|Function>} modules A list of module functions or their aliases. See

\* {@link angular.module}. The `ng` module must be explicitly added.

\* @returns {function()} Injector function. See {@link auto.$injector $injector}.

\*

\* @example

\* Typical usage

\* ```js

\* // create an injector

\* var $injector = angular.injector(['ng']);

\*

\* // use the injector to kick off your application

\* // use the type inference to auto inject arguments, or use implicit injection

\* $injector.invoke(function($rootScope, $compile, $document){

\* $compile($document)($rootScope);

\* $rootScope.$digest();

\* });

\* ```

\*

\* Sometimes you want to get access to the injector of a currently running Angular app

\* from outside Angular. Perhaps, you want to inject and compile some markup after the

\* application has been bootstrapped. You can do this using extra `injector()` added

\* to JQuery/jqLite elements. See {@link angular.element}.

\*

\* \*This is fairly rare but could be the case if a third party library is injecting the

\* markup.\*

\*

\* In the following example a new block of HTML containing a `ng-controller`

\* directive is added to the end of the document body by JQuery. We then compile and link

\* it into the current AngularJS scope.

\*

\* ```js

\* var $div = $('<div ng-controller="MyCtrl">{{content.label}}</div>');

\* $(document.body).append($div);

\*

\* angular.element(document).injector().invoke(function($compile) {

\* var scope = angular.element($div).scope();

\* $compile($div)(scope);

\* });

\* ```

\*/

/\*\*

\* @ngdoc module

\* @name auto

\* @description

\*

\* Implicit module which gets automatically added to each {@link auto.$injector $injector}.

\*/

var FN\_ARGS = /^function\s\*[^\(]\*\(\s\*([^\)]\*)\)/m;

var FN\_ARG\_SPLIT = /,/;

var FN\_ARG = /^\s\*(\_?)(\S+?)\1\s\*$/;

var STRIP\_COMMENTS = /((\/\/.\*$)|(\/\\*[\s\S]\*?\\*\/))/mg;

var $injectorMinErr = minErr('$injector');

function annotate(fn) {

var $inject,

fnText,

argDecl,

last;

if (typeof fn == 'function') {

if (!($inject = fn.$inject)) {

$inject = [];

if (fn.length) {

fnText = fn.toString().replace(STRIP\_COMMENTS, '');

argDecl = fnText.match(FN\_ARGS);

forEach(argDecl[1].split(FN\_ARG\_SPLIT), function(arg){

arg.replace(FN\_ARG, function(all, underscore, name){

$inject.push(name);

});

});

}

fn.$inject = $inject;

}

} else if (isArray(fn)) {

last = fn.length - 1;

assertArgFn(fn[last], 'fn');

$inject = fn.slice(0, last);

} else {

assertArgFn(fn, 'fn', true);

}

return $inject;

}

///////////////////////////////////////

/\*\*

\* @ngdoc service

\* @name $injector

\* @function

\*

\* @description

\*

\* `$injector` is used to retrieve object instances as defined by

\* {@link auto.$provide provider}, instantiate types, invoke methods,

\* and load modules.

\*

\* The following always holds true:

\*

\* ```js

\* var $injector = angular.injector();

\* expect($injector.get('$injector')).toBe($injector);

\* expect($injector.invoke(function($injector){

\* return $injector;

\* }).toBe($injector);

\* ```

\*

\* # Injection Function Annotation

\*

\* JavaScript does not have annotations, and annotations are needed for dependency injection. The

\* following are all valid ways of annotating function with injection arguments and are equivalent.

\*

\* ```js

\* // inferred (only works if code not minified/obfuscated)

\* $injector.invoke(function(serviceA){});

\*

\* // annotated

\* function explicit(serviceA) {};

\* explicit.$inject = ['serviceA'];

\* $injector.invoke(explicit);

\*

\* // inline

\* $injector.invoke(['serviceA', function(serviceA){}]);

\* ```

\*

\* ## Inference

\*

\* In JavaScript calling `toString()` on a function returns the function definition. The definition

\* can then be parsed and the function arguments can be extracted. \*NOTE:\* This does not work with

\* minification, and obfuscation tools since these tools change the argument names.

\*

\* ## `$inject` Annotation

\* By adding a `$inject` property onto a function the injection parameters can be specified.

\*

\* ## Inline

\* As an array of injection names, where the last item in the array is the function to call.

\*/

/\*\*

\* @ngdoc method

\* @name $injector#get

\*

\* @description

\* Return an instance of the service.

\*

\* @param {string} name The name of the instance to retrieve.

\* @return {\*} The instance.

\*/

/\*\*

\* @ngdoc method

\* @name $injector#invoke

\*

\* @description

\* Invoke the method and supply the method arguments from the `$injector`.

\*

\* @param {!Function} fn The function to invoke. Function parameters are injected according to the

\* {@link guide/di $inject Annotation} rules.

\* @param {Object=} self The `this` for the invoked method.

\* @param {Object=} locals Optional object. If preset then any argument names are read from this

\* object first, before the `$injector` is consulted.

\* @returns {\*} the value returned by the invoked `fn` function.

\*/

/\*\*

\* @ngdoc method

\* @name $injector#has

\*

\* @description

\* Allows the user to query if the particular service exist.

\*

\* @param {string} Name of the service to query.

\* @returns {boolean} returns true if injector has given service.

\*/

/\*\*

\* @ngdoc method

\* @name $injector#instantiate

\* @description

\* Create a new instance of JS type. The method takes a constructor function invokes the new

\* operator and supplies all of the arguments to the constructor function as specified by the

\* constructor annotation.

\*

\* @param {Function} Type Annotated constructor function.

\* @param {Object=} locals Optional object. If preset then any argument names are read from this

\* object first, before the `$injector` is consulted.

\* @returns {Object} new instance of `Type`.

\*/

/\*\*

\* @ngdoc method

\* @name $injector#annotate

\*

\* @description

\* Returns an array of service names which the function is requesting for injection. This API is

\* used by the injector to determine which services need to be injected into the function when the

\* function is invoked. There are three ways in which the function can be annotated with the needed

\* dependencies.

\*

\* # Argument names

\*

\* The simplest form is to extract the dependencies from the arguments of the function. This is done

\* by converting the function into a string using `toString()` method and extracting the argument

\* names.

\* ```js

\* // Given

\* function MyController($scope, $route) {

\* // ...

\* }

\*

\* // Then

\* expect(injector.annotate(MyController)).toEqual(['$scope', '$route']);

\* ```

\*

\* This method does not work with code minification / obfuscation. For this reason the following

\* annotation strategies are supported.

\*

\* # The `$inject` property

\*

\* If a function has an `$inject` property and its value is an array of strings, then the strings

\* represent names of services to be injected into the function.

\* ```js

\* // Given

\* var MyController = function(obfuscatedScope, obfuscatedRoute) {

\* // ...

\* }

\* // Define function dependencies

\* MyController['$inject'] = ['$scope', '$route'];

\*

\* // Then

\* expect(injector.annotate(MyController)).toEqual(['$scope', '$route']);

\* ```

\*

\* # The array notation

\*

\* It is often desirable to inline Injected functions and that's when setting the `$inject` property

\* is very inconvenient. In these situations using the array notation to specify the dependencies in

\* a way that survives minification is a better choice:

\*

\* ```js

\* // We wish to write this (not minification / obfuscation safe)

\* injector.invoke(function($compile, $rootScope) {

\* // ...

\* });

\*

\* // We are forced to write break inlining

\* var tmpFn = function(obfuscatedCompile, obfuscatedRootScope) {

\* // ...

\* };

\* tmpFn.$inject = ['$compile', '$rootScope'];

\* injector.invoke(tmpFn);

\*

\* // To better support inline function the inline annotation is supported

\* injector.invoke(['$compile', '$rootScope', function(obfCompile, obfRootScope) {

\* // ...

\* }]);

\*

\* // Therefore

\* expect(injector.annotate(

\* ['$compile', '$rootScope', function(obfus\_$compile, obfus\_$rootScope) {}])

\* ).toEqual(['$compile', '$rootScope']);

\* ```

\*

\* @param {Function|Array.<string|Function>} fn Function for which dependent service names need to

\* be retrieved as described above.

\*

\* @returns {Array.<string>} The names of the services which the function requires.

\*/

/\*\*

\* @ngdoc object

\* @name $provide

\*

\* @description

\*

\* The {@link auto.$provide $provide} service has a number of methods for registering components

\* with the {@link auto.$injector $injector}. Many of these functions are also exposed on

\* {@link angular.Module}.

\*

\* An Angular \*\*service\*\* is a singleton object created by a \*\*service factory\*\*. These \*\*service

\* factories\*\* are functions which, in turn, are created by a \*\*service provider\*\*.

\* The \*\*service providers\*\* are constructor functions. When instantiated they must contain a

\* property called `$get`, which holds the \*\*service factory\*\* function.

\*

\* When you request a service, the {@link auto.$injector $injector} is responsible for finding the

\* correct \*\*service provider\*\*, instantiating it and then calling its `$get` \*\*service factory\*\*

\* function to get the instance of the \*\*service\*\*.

\*

\* Often services have no configuration options and there is no need to add methods to the service

\* provider. The provider will be no more than a constructor function with a `$get` property. For

\* these cases the {@link auto.$provide $provide} service has additional helper methods to register

\* services without specifying a provider.

\*

\* \* {@link auto.$provide#provider provider(provider)} - registers a \*\*service provider\*\* with the

\* {@link auto.$injector $injector}

\* \* {@link auto.$provide#constant constant(obj)} - registers a value/object that can be accessed by

\* providers and services.

\* \* {@link auto.$provide#value value(obj)} - registers a value/object that can only be accessed by

\* services, not providers.

\* \* {@link auto.$provide#factory factory(fn)} - registers a service \*\*factory function\*\*, `fn`,

\* that will be wrapped in a \*\*service provider\*\* object, whose `$get` property will contain the

\* given factory function.

\* \* {@link auto.$provide#service service(class)} - registers a \*\*constructor function\*\*, `class`

\* that will be wrapped in a \*\*service provider\*\* object, whose `$get` property will instantiate

\* a new object using the given constructor function.

\*

\* See the individual methods for more information and examples.

\*/

/\*\*

\* @ngdoc method

\* @name $provide#provider

\* @description

\*

\* Register a \*\*provider function\*\* with the {@link auto.$injector $injector}. Provider functions

\* are constructor functions, whose instances are responsible for "providing" a factory for a

\* service.

\*

\* Service provider names start with the name of the service they provide followed by `Provider`.

\* For example, the {@link ng.$log $log} service has a provider called

\* {@link ng.$logProvider $logProvider}.

\*

\* Service provider objects can have additional methods which allow configuration of the provider

\* and its service. Importantly, you can configure what kind of service is created by the `$get`

\* method, or how that service will act. For example, the {@link ng.$logProvider $logProvider} has a

\* method {@link ng.$logProvider#debugEnabled debugEnabled}

\* which lets you specify whether the {@link ng.$log $log} service will log debug messages to the

\* console or not.

\*

\* @param {string} name The name of the instance. NOTE: the provider will be available under `name +

'Provider'` key.

\* @param {(Object|function())} provider If the provider is:

\*

\* - `Object`: then it should have a `$get` method. The `$get` method will be invoked using

\* {@link auto.$injector#invoke $injector.invoke()} when an instance needs to be created.

\* - `Constructor`: a new instance of the provider will be created using

\* {@link auto.$injector#instantiate $injector.instantiate()}, then treated as `object`.

\*

\* @returns {Object} registered provider instance

\* @example

\*

\* The following example shows how to create a simple event tracking service and register it using

\* {@link auto.$provide#provider $provide.provider()}.

\*

\* ```js

\* // Define the eventTracker provider

\* function EventTrackerProvider() {

\* var trackingUrl = '/track';

\*

\* // A provider method for configuring where the tracked events should been saved

\* this.setTrackingUrl = function(url) {

\* trackingUrl = url;

\* };

\*

\* // The service factory function

\* this.$get = ['$http', function($http) {

\* var trackedEvents = {};

\* return {

\* // Call this to track an event

\* event: function(event) {

\* var count = trackedEvents[event] || 0;

\* count += 1;

\* trackedEvents[event] = count;

\* return count;

\* },

\* // Call this to save the tracked events to the trackingUrl

\* save: function() {

\* $http.post(trackingUrl, trackedEvents);

\* }

\* };

\* }];

\* }

\*

\* describe('eventTracker', function() {

\* var postSpy;

\*

\* beforeEach(module(function($provide) {

\* // Register the eventTracker provider

\* $provide.provider('eventTracker', EventTrackerProvider);

\* }));

\*

\* beforeEach(module(function(eventTrackerProvider) {

\* // Configure eventTracker provider

\* eventTrackerProvider.setTrackingUrl('/custom-track');

\* }));

\*

\* it('tracks events', inject(function(eventTracker) {

\* expect(eventTracker.event('login')).toEqual(1);

\* expect(eventTracker.event('login')).toEqual(2);

\* }));

\*

\* it('saves to the tracking url', inject(function(eventTracker, $http) {

\* postSpy = spyOn($http, 'post');

\* eventTracker.event('login');

\* eventTracker.save();

\* expect(postSpy).toHaveBeenCalled();

\* expect(postSpy.mostRecentCall.args[0]).not.toEqual('/track');

\* expect(postSpy.mostRecentCall.args[0]).toEqual('/custom-track');

\* expect(postSpy.mostRecentCall.args[1]).toEqual({ 'login': 1 });

\* }));

\* });

\* ```

\*/

/\*\*

\* @ngdoc method

\* @name $provide#factory

\* @description

\*

\* Register a \*\*service factory\*\*, which will be called to return the service instance.

\* This is short for registering a service where its provider consists of only a `$get` property,

\* which is the given service factory function.

\* You should use {@link auto.$provide#factory $provide.factory(getFn)} if you do not need to

\* configure your service in a provider.

\*

\* @param {string} name The name of the instance.

\* @param {function()} $getFn The $getFn for the instance creation. Internally this is a short hand

\* for `$provide.provider(name, {$get: $getFn})`.

\* @returns {Object} registered provider instance

\*

\* @example

\* Here is an example of registering a service

\* ```js

\* $provide.factory('ping', ['$http', function($http) {

\* return function ping() {

\* return $http.send('/ping');

\* };

\* }]);

\* ```

\* You would then inject and use this service like this:

\* ```js

\* someModule.controller('Ctrl', ['ping', function(ping) {

\* ping();

\* }]);

\* ```

\*/

/\*\*

\* @ngdoc method

\* @name $provide#service

\* @description

\*

\* Register a \*\*service constructor\*\*, which will be invoked with `new` to create the service

\* instance.

\* This is short for registering a service where its provider's `$get` property is the service

\* constructor function that will be used to instantiate the service instance.

\*

\* You should use {@link auto.$provide#service $provide.service(class)} if you define your service

\* as a type/class.

\*

\* @param {string} name The name of the instance.

\* @param {Function} constructor A class (constructor function) that will be instantiated.

\* @returns {Object} registered provider instance

\*

\* @example

\* Here is an example of registering a service using

\* {@link auto.$provide#service $provide.service(class)}.

\* ```js

\* var Ping = function($http) {

\* this.$http = $http;

\* };

\*

\* Ping.$inject = ['$http'];

\*

\* Ping.prototype.send = function() {

\* return this.$http.get('/ping');

\* };

\* $provide.service('ping', Ping);

\* ```

\* You would then inject and use this service like this:

\* ```js

\* someModule.controller('Ctrl', ['ping', function(ping) {

\* ping.send();

\* }]);

\* ```

\*/

/\*\*

\* @ngdoc method

\* @name $provide#value

\* @description

\*

\* Register a \*\*value service\*\* with the {@link auto.$injector $injector}, such as a string, a

\* number, an array, an object or a function. This is short for registering a service where its

\* provider's `$get` property is a factory function that takes no arguments and returns the \*\*value

\* service\*\*.

\*

\* Value services are similar to constant services, except that they cannot be injected into a

\* module configuration function (see {@link angular.Module#config}) but they can be overridden by

\* an Angular

\* {@link auto.$provide#decorator decorator}.

\*

\* @param {string} name The name of the instance.

\* @param {\*} value The value.

\* @returns {Object} registered provider instance

\*

\* @example

\* Here are some examples of creating value services.

\* ```js

\* $provide.value('ADMIN\_USER', 'admin');

\*

\* $provide.value('RoleLookup', { admin: 0, writer: 1, reader: 2 });

\*

\* $provide.value('halfOf', function(value) {

\* return value / 2;

\* });

\* ```

\*/

/\*\*

\* @ngdoc method

\* @name $provide#constant

\* @description

\*

\* Register a \*\*constant service\*\*, such as a string, a number, an array, an object or a function,

\* with the {@link auto.$injector $injector}. Unlike {@link auto.$provide#value value} it can be

\* injected into a module configuration function (see {@link angular.Module#config}) and it cannot

\* be overridden by an Angular {@link auto.$provide#decorator decorator}.

\*

\* @param {string} name The name of the constant.

\* @param {\*} value The constant value.

\* @returns {Object} registered instance

\*

\* @example

\* Here a some examples of creating constants:

\* ```js

\* $provide.constant('SHARD\_HEIGHT', 306);

\*

\* $provide.constant('MY\_COLOURS', ['red', 'blue', 'grey']);

\*

\* $provide.constant('double', function(value) {

\* return value \* 2;

\* });

\* ```

\*/

/\*\*

\* @ngdoc method

\* @name $provide#decorator

\* @description

\*

\* Register a \*\*service decorator\*\* with the {@link auto.$injector $injector}. A service decorator

\* intercepts the creation of a service, allowing it to override or modify the behaviour of the

\* service. The object returned by the decorator may be the original service, or a new service

\* object which replaces or wraps and delegates to the original service.

\*

\* @param {string} name The name of the service to decorate.

\* @param {function()} decorator This function will be invoked when the service needs to be

\* instantiated and should return the decorated service instance. The function is called using

\* the {@link auto.$injector#invoke injector.invoke} method and is therefore fully injectable.

\* Local injection arguments:

\*

\* \* `$delegate` - The original service instance, which can be monkey patched, configured,

\* decorated or delegated to.

\*

\* @example

\* Here we decorate the {@link ng.$log $log} service to convert warnings to errors by intercepting

\* calls to {@link ng.$log#error $log.warn()}.

\* ```js

\* $provide.decorator('$log', ['$delegate', function($delegate) {

\* $delegate.warn = $delegate.error;

\* return $delegate;

\* }]);

\* ```

\*/

function createInjector(modulesToLoad) {

var INSTANTIATING = {},

providerSuffix = 'Provider',

path = [],

loadedModules = new HashMap(),

providerCache = {

$provide: {

provider: supportObject(provider),

factory: supportObject(factory),

service: supportObject(service),

value: supportObject(value),

constant: supportObject(constant),

decorator: decorator

}

},

providerInjector = (providerCache.$injector =

createInternalInjector(providerCache, function() {

throw $injectorMinErr('unpr', "Unknown provider: {0}", path.join(' <- '));

})),

instanceCache = {},

instanceInjector = (instanceCache.$injector =

createInternalInjector(instanceCache, function(servicename) {

var provider = providerInjector.get(servicename + providerSuffix);

return instanceInjector.invoke(provider.$get, provider);

}));

forEach(loadModules(modulesToLoad), function(fn) { instanceInjector.invoke(fn || noop); });

return instanceInjector;

////////////////////////////////////

// $provider

////////////////////////////////////

function supportObject(delegate) {

return function(key, value) {

if (isObject(key)) {

forEach(key, reverseParams(delegate));

} else {

return delegate(key, value);

}

};

}

function provider(name, provider\_) {

assertNotHasOwnProperty(name, 'service');

if (isFunction(provider\_) || isArray(provider\_)) {

provider\_ = providerInjector.instantiate(provider\_);

}

if (!provider\_.$get) {

throw $injectorMinErr('pget', "Provider '{0}' must define $get factory method.", name);

}

return providerCache[name + providerSuffix] = provider\_;

}

function factory(name, factoryFn) { return provider(name, { $get: factoryFn }); }

function service(name, constructor) {

return factory(name, ['$injector', function($injector) {

return $injector.instantiate(constructor);

}]);

}

function value(name, val) { return factory(name, valueFn(val)); }

function constant(name, value) {

assertNotHasOwnProperty(name, 'constant');

providerCache[name] = value;

instanceCache[name] = value;

}

function decorator(serviceName, decorFn) {

var origProvider = providerInjector.get(serviceName + providerSuffix),

orig$get = origProvider.$get;

origProvider.$get = function() {

var origInstance = instanceInjector.invoke(orig$get, origProvider);

return instanceInjector.invoke(decorFn, null, {$delegate: origInstance});

};

}

////////////////////////////////////

// Module Loading

////////////////////////////////////

function loadModules(modulesToLoad){

var runBlocks = [], moduleFn, invokeQueue, i, ii;

forEach(modulesToLoad, function(module) {

if (loadedModules.get(module)) return;

loadedModules.put(module, true);

try {

if (isString(module)) {

moduleFn = angularModule(module);

runBlocks = runBlocks.concat(loadModules(moduleFn.requires)).concat(moduleFn.\_runBlocks);

for(invokeQueue = moduleFn.\_invokeQueue, i = 0, ii = invokeQueue.length; i < ii; i++) {

var invokeArgs = invokeQueue[i],

provider = providerInjector.get(invokeArgs[0]);

provider[invokeArgs[1]].apply(provider, invokeArgs[2]);

}

} else if (isFunction(module)) {

runBlocks.push(providerInjector.invoke(module));

} else if (isArray(module)) {

runBlocks.push(providerInjector.invoke(module));

} else {

assertArgFn(module, 'module');

}

} catch (e) {

if (isArray(module)) {

module = module[module.length - 1];

}

if (e.message && e.stack && e.stack.indexOf(e.message) == -1) {

// Safari & FF's stack traces don't contain error.message content

// unlike those of Chrome and IE

// So if stack doesn't contain message, we create a new string that contains both.

// Since error.stack is read-only in Safari, I'm overriding e and not e.stack here.

/\* jshint -W022 \*/

e = e.message + '\n' + e.stack;

}

throw $injectorMinErr('modulerr', "Failed to instantiate module {0} due to:\n{1}",

module, e.stack || e.message || e);

}

});

return runBlocks;

}

////////////////////////////////////

// internal Injector

////////////////////////////////////

function createInternalInjector(cache, factory) {

function getService(serviceName) {

if (cache.hasOwnProperty(serviceName)) {

if (cache[serviceName] === INSTANTIATING) {

throw $injectorMinErr('cdep', 'Circular dependency found: {0}', path.join(' <- '));

}

return cache[serviceName];

} else {

try {

path.unshift(serviceName);

cache[serviceName] = INSTANTIATING;

return cache[serviceName] = factory(serviceName);

} catch (err) {

if (cache[serviceName] === INSTANTIATING) {

delete cache[serviceName];

}

throw err;

} finally {

path.shift();

}

}

}

function invoke(fn, self, locals){

var args = [],

$inject = annotate(fn),

length, i,

key;

for(i = 0, length = $inject.length; i < length; i++) {

key = $inject[i];

if (typeof key !== 'string') {

throw $injectorMinErr('itkn',

'Incorrect injection token! Expected service name as string, got {0}', key);

}

args.push(

locals && locals.hasOwnProperty(key)

? locals[key]

: getService(key)

);

}

if (!fn.$inject) {

// this means that we must be an array.

fn = fn[length];

}

// http://jsperf.com/angularjs-invoke-apply-vs-switch

// #5388

return fn.apply(self, args);

}

function instantiate(Type, locals) {

var Constructor = function() {},

instance, returnedValue;

// Check if Type is annotated and use just the given function at n-1 as parameter

// e.g. someModule.factory('greeter', ['$window', function(renamed$window) {}]);

Constructor.prototype = (isArray(Type) ? Type[Type.length - 1] : Type).prototype;

instance = new Constructor();

returnedValue = invoke(Type, instance, locals);

return isObject(returnedValue) || isFunction(returnedValue) ? returnedValue : instance;

}

return {

invoke: invoke,

instantiate: instantiate,

get: getService,

annotate: annotate,

has: function(name) {

return providerCache.hasOwnProperty(name + providerSuffix) || cache.hasOwnProperty(name);

}

};

}

}

/\*\*

\* @ngdoc service

\* @name $anchorScroll

\* @kind function

\* @requires $window

\* @requires $location

\* @requires $rootScope

\*

\* @description

\* When called, it checks current value of `$location.hash()` and scroll to related element,

\* according to rules specified in

\* [Html5 spec](http://dev.w3.org/html5/spec/Overview.html#the-indicated-part-of-the-document).

\*

\* It also watches the `$location.hash()` and scrolls whenever it changes to match any anchor.

\* This can be disabled by calling `$anchorScrollProvider.disableAutoScrolling()`.

\*

\* @example

<example>

<file name="index.html">

<div id="scrollArea" ng-controller="ScrollCtrl">

<a ng-click="gotoBottom()">Go to bottom</a>

<a id="bottom"></a> You're at the bottom!

</div>

</file>

<file name="script.js">

function ScrollCtrl($scope, $location, $anchorScroll) {

$scope.gotoBottom = function (){

// set the location.hash to the id of

// the element you wish to scroll to.

$location.hash('bottom');

// call $anchorScroll()

$anchorScroll();

};

}

</file>

<file name="style.css">

#scrollArea {

height: 350px;

overflow: auto;

}

#bottom {

display: block;

margin-top: 2000px;

}

</file>

</example>

\*/

function $AnchorScrollProvider() {

var autoScrollingEnabled = true;

this.disableAutoScrolling = function() {

autoScrollingEnabled = false;

};

this.$get = ['$window', '$location', '$rootScope', function($window, $location, $rootScope) {

var document = $window.document;

// helper function to get first anchor from a NodeList

// can't use filter.filter, as it accepts only instances of Array

// and IE can't convert NodeList to an array using [].slice

// TODO(vojta): use filter if we change it to accept lists as well

function getFirstAnchor(list) {

var result = null;

forEach(list, function(element) {

if (!result && lowercase(element.nodeName) === 'a') result = element;

});

return result;

}

function scroll() {

var hash = $location.hash(), elm;

// empty hash, scroll to the top of the page

if (!hash) $window.scrollTo(0, 0);

// element with given id

else if ((elm = document.getElementById(hash))) elm.scrollIntoView();

// first anchor with given name :-D

else if ((elm = getFirstAnchor(document.getElementsByName(hash)))) elm.scrollIntoView();

// no element and hash == 'top', scroll to the top of the page

else if (hash === 'top') $window.scrollTo(0, 0);

}

// does not scroll when user clicks on anchor link that is currently on

// (no url change, no $location.hash() change), browser native does scroll

if (autoScrollingEnabled) {

$rootScope.$watch(function autoScrollWatch() {return $location.hash();},

function autoScrollWatchAction() {

$rootScope.$evalAsync(scroll);

});

}

return scroll;

}];

}

var $animateMinErr = minErr('$animate');

/\*\*

\* @ngdoc provider

\* @name $animateProvider

\*

\* @description

\* Default implementation of $animate that doesn't perform any animations, instead just

\* synchronously performs DOM

\* updates and calls done() callbacks.

\*

\* In order to enable animations the ngAnimate module has to be loaded.

\*

\* To see the functional implementation check out src/ngAnimate/animate.js

\*/

var $AnimateProvider = ['$provide', function($provide) {

this.$$selectors = {};

/\*\*

\* @ngdoc method

\* @name $animateProvider#register

\*

\* @description

\* Registers a new injectable animation factory function. The factory function produces the

\* animation object which contains callback functions for each event that is expected to be

\* animated.

\*

\* \* `eventFn`: `function(Element, doneFunction)` The element to animate, the `doneFunction`

\* must be called once the element animation is complete. If a function is returned then the

\* animation service will use this function to cancel the animation whenever a cancel event is

\* triggered.

\*

\*

\* ```js

\* return {

\* eventFn : function(element, done) {

\* //code to run the animation

\* //once complete, then run done()

\* return function cancellationFunction() {

\* //code to cancel the animation

\* }

\* }

\* }

\* ```

\*

\* @param {string} name The name of the animation.

\* @param {Function} factory The factory function that will be executed to return the animation

\* object.

\*/

this.register = function(name, factory) {

var key = name + '-animation';

if (name && name.charAt(0) != '.') throw $animateMinErr('notcsel',

"Expecting class selector starting with '.' got '{0}'.", name);

this.$$selectors[name.substr(1)] = key;

$provide.factory(key, factory);

};

/\*\*

\* @ngdoc method

\* @name $animateProvider#classNameFilter

\*

\* @description

\* Sets and/or returns the CSS class regular expression that is checked when performing

\* an animation. Upon bootstrap the classNameFilter value is not set at all and will

\* therefore enable $animate to attempt to perform an animation on any element.

\* When setting the classNameFilter value, animations will only be performed on elements

\* that successfully match the filter expression. This in turn can boost performance

\* for low-powered devices as well as applications containing a lot of structural operations.

\* @param {RegExp=} expression The className expression which will be checked against all animations

\* @return {RegExp} The current CSS className expression value. If null then there is no expression value

\*/

this.classNameFilter = function(expression) {

if(arguments.length === 1) {

this.$$classNameFilter = (expression instanceof RegExp) ? expression : null;

}

return this.$$classNameFilter;

};

this.$get = ['$timeout', '$$asyncCallback', function($timeout, $$asyncCallback) {

function async(fn) {

fn && $$asyncCallback(fn);

}

/\*\*

\*

\* @ngdoc service

\* @name $animate

\* @description The $animate service provides rudimentary DOM manipulation functions to

\* insert, remove and move elements within the DOM, as well as adding and removing classes.

\* This service is the core service used by the ngAnimate $animator service which provides

\* high-level animation hooks for CSS and JavaScript.

\*

\* $animate is available in the AngularJS core, however, the ngAnimate module must be included

\* to enable full out animation support. Otherwise, $animate will only perform simple DOM

\* manipulation operations.

\*

\* To learn more about enabling animation support, click here to visit the {@link ngAnimate

\* ngAnimate module page} as well as the {@link ngAnimate.$animate ngAnimate $animate service

\* page}.

\*/

return {

/\*\*

\*

\* @ngdoc method

\* @name $animate#enter

\* @function

\* @description Inserts the element into the DOM either after the `after` element or within

\* the `parent` element. Once complete, the done() callback will be fired (if provided).

\* @param {DOMElement} element the element which will be inserted into the DOM

\* @param {DOMElement} parent the parent element which will append the element as

\* a child (if the after element is not present)

\* @param {DOMElement} after the sibling element which will append the element

\* after itself

\* @param {Function=} done callback function that will be called after the element has been

\* inserted into the DOM

\*/

enter : function(element, parent, after, done) {

if (after) {

after.after(element);

} else {

if (!parent || !parent[0]) {

parent = after.parent();

}

parent.append(element);

}

async(done);

},

/\*\*

\*

\* @ngdoc method

\* @name $animate#leave

\* @function

\* @description Removes the element from the DOM. Once complete, the done() callback will be

\* fired (if provided).

\* @param {DOMElement} element the element which will be removed from the DOM

\* @param {Function=} done callback function that will be called after the element has been

\* removed from the DOM

\*/

leave : function(element, done) {

element.remove();

async(done);

},

/\*\*

\*

\* @ngdoc method

\* @name $animate#move

\* @function

\* @description Moves the position of the provided element within the DOM to be placed

\* either after the `after` element or inside of the `parent` element. Once complete, the

\* done() callback will be fired (if provided).

\*

\* @param {DOMElement} element the element which will be moved around within the

\* DOM

\* @param {DOMElement} parent the parent element where the element will be

\* inserted into (if the after element is not present)

\* @param {DOMElement} after the sibling element where the element will be

\* positioned next to

\* @param {Function=} done the callback function (if provided) that will be fired after the

\* element has been moved to its new position

\*/

move : function(element, parent, after, done) {

// Do not remove element before insert. Removing will cause data associated with the

// element to be dropped. Insert will implicitly do the remove.

this.enter(element, parent, after, done);

},

/\*\*

\*

\* @ngdoc method

\* @name $animate#addClass

\* @function

\* @description Adds the provided className CSS class value to the provided element. Once

\* complete, the done() callback will be fired (if provided).

\* @param {DOMElement} element the element which will have the className value

\* added to it

\* @param {string} className the CSS class which will be added to the element

\* @param {Function=} done the callback function (if provided) that will be fired after the

\* className value has been added to the element

\*/

addClass : function(element, className, done) {

className = isString(className) ?

className :

isArray(className) ? className.join(' ') : '';

forEach(element, function (element) {

jqLiteAddClass(element, className);

});

async(done);

},

/\*\*

\*

\* @ngdoc method

\* @name $animate#removeClass

\* @function

\* @description Removes the provided className CSS class value from the provided element.

\* Once complete, the done() callback will be fired (if provided).

\* @param {DOMElement} element the element which will have the className value

\* removed from it

\* @param {string} className the CSS class which will be removed from the element

\* @param {Function=} done the callback function (if provided) that will be fired after the

\* className value has been removed from the element

\*/

removeClass : function(element, className, done) {

className = isString(className) ?

className :

isArray(className) ? className.join(' ') : '';

forEach(element, function (element) {

jqLiteRemoveClass(element, className);

});

async(done);

},

/\*\*

\*

\* @ngdoc method

\* @name $animate#setClass

\* @function

\* @description Adds and/or removes the given CSS classes to and from the element.

\* Once complete, the done() callback will be fired (if provided).

\* @param {DOMElement} element the element which will it's CSS classes changed

\* removed from it

\* @param {string} add the CSS classes which will be added to the element

\* @param {string} remove the CSS class which will be removed from the element

\* @param {Function=} done the callback function (if provided) that will be fired after the

\* CSS classes have been set on the element

\*/

setClass : function(element, add, remove, done) {

forEach(element, function (element) {

jqLiteAddClass(element, add);

jqLiteRemoveClass(element, remove);

});

async(done);

},

enabled : noop

};

}];

}];

function $$AsyncCallbackProvider(){

this.$get = ['$$rAF', '$timeout', function($$rAF, $timeout) {

return $$rAF.supported

? function(fn) { return $$rAF(fn); }

: function(fn) {

return $timeout(fn, 0, false);

};

}];

}

/\*\*

\* ! This is a private undocumented service !

\*

\* @name $browser

\* @requires $log

\* @description

\* This object has two goals:

\*

\* - hide all the global state in the browser caused by the window object

\* - abstract away all the browser specific features and inconsistencies

\*

\* For tests we provide {@link ngMock.$browser mock implementation} of the `$browser`

\* service, which can be used for convenient testing of the application without the interaction with

\* the real browser apis.

\*/

/\*\*

\* @param {object} window The global window object.

\* @param {object} document jQuery wrapped document.

\* @param {function()} XHR XMLHttpRequest constructor.

\* @param {object} $log console.log or an object with the same interface.

\* @param {object} $sniffer $sniffer service

\*/

function Browser(window, document, $log, $sniffer) {

var self = this,

rawDocument = document[0],

location = window.location,

history = window.history,

setTimeout = window.setTimeout,

clearTimeout = window.clearTimeout,

pendingDeferIds = {};

self.isMock = false;

var outstandingRequestCount = 0;

var outstandingRequestCallbacks = [];

// TODO(vojta): remove this temporary api

self.$$completeOutstandingRequest = completeOutstandingRequest;

self.$$incOutstandingRequestCount = function() { outstandingRequestCount++; };

/\*\*

\* Executes the `fn` function(supports currying) and decrements the `outstandingRequestCallbacks`

\* counter. If the counter reaches 0, all the `outstandingRequestCallbacks` are executed.

\*/

function completeOutstandingRequest(fn) {

try {

fn.apply(null, sliceArgs(arguments, 1));

} finally {

outstandingRequestCount--;

if (outstandingRequestCount === 0) {

while(outstandingRequestCallbacks.length) {

try {

outstandingRequestCallbacks.pop()();

} catch (e) {

$log.error(e);

}

}

}

}

}

/\*\*

\* @private

\* Note: this method is used only by scenario runner

\* TODO(vojta): prefix this method with $$ ?

\* @param {function()} callback Function that will be called when no outstanding request

\*/

self.notifyWhenNoOutstandingRequests = function(callback) {

// force browser to execute all pollFns - this is needed so that cookies and other pollers fire

// at some deterministic time in respect to the test runner's actions. Leaving things up to the

// regular poller would result in flaky tests.

forEach(pollFns, function(pollFn){ pollFn(); });

if (outstandingRequestCount === 0) {

callback();

} else {

outstandingRequestCallbacks.push(callback);

}

};

//////////////////////////////////////////////////////////////

// Poll Watcher API

//////////////////////////////////////////////////////////////

var pollFns = [],

pollTimeout;

/\*\*

\* @name $browser#addPollFn

\*

\* @param {function()} fn Poll function to add

\*

\* @description

\* Adds a function to the list of functions that poller periodically executes,

\* and starts polling if not started yet.

\*

\* @returns {function()} the added function

\*/

self.addPollFn = function(fn) {

if (isUndefined(pollTimeout)) startPoller(100, setTimeout);

pollFns.push(fn);

return fn;

};

/\*\*

\* @param {number} interval How often should browser call poll functions (ms)

\* @param {function()} setTimeout Reference to a real or fake `setTimeout` function.

\*

\* @description

\* Configures the poller to run in the specified intervals, using the specified

\* setTimeout fn and kicks it off.

\*/

function startPoller(interval, setTimeout) {

(function check() {

forEach(pollFns, function(pollFn){ pollFn(); });

pollTimeout = setTimeout(check, interval);

})();

}

//////////////////////////////////////////////////////////////

// URL API

//////////////////////////////////////////////////////////////

var lastBrowserUrl = location.href,

baseElement = document.find('base'),

newLocation = null;

/\*\*

\* @name $browser#url

\*

\* @description

\* GETTER:

\* Without any argument, this method just returns current value of location.href.

\*

\* SETTER:

\* With at least one argument, this method sets url to new value.

\* If html5 history api supported, pushState/replaceState is used, otherwise

\* location.href/location.replace is used.

\* Returns its own instance to allow chaining

\*

\* NOTE: this api is intended for use only by the $location service. Please use the

\* {@link ng.$location $location service} to change url.

\*

\* @param {string} url New url (when used as setter)

\* @param {boolean=} replace Should new url replace current history record ?

\*/

self.url = function(url, replace) {

// Android Browser BFCache causes location, history reference to become stale.

if (location !== window.location) location = window.location;

if (history !== window.history) history = window.history;

// setter

if (url) {

if (lastBrowserUrl == url) return;

lastBrowserUrl = url;

if ($sniffer.history) {

if (replace) history.replaceState(null, '', url);

else {

history.pushState(null, '', url);

// Crazy Opera Bug: http://my.opera.com/community/forums/topic.dml?id=1185462

baseElement.attr('href', baseElement.attr('href'));

}

} else {

newLocation = url;

if (replace) {

location.replace(url);

} else {

location.href = url;

}

}

return self;

// getter

} else {

// - newLocation is a workaround for an IE7-9 issue with location.replace and location.href

// methods not updating location.href synchronously.

// - the replacement is a workaround for https://bugzilla.mozilla.org/show\_bug.cgi?id=407172

return newLocation || location.href.replace(/%27/g,"'");

}

};

var urlChangeListeners = [],

urlChangeInit = false;

function fireUrlChange() {

newLocation = null;

if (lastBrowserUrl == self.url()) return;

lastBrowserUrl = self.url();

forEach(urlChangeListeners, function(listener) {

listener(self.url());

});

}

/\*\*

\* @name $browser#onUrlChange

\*

\* @description

\* Register callback function that will be called, when url changes.

\*

\* It's only called when the url is changed from outside of angular:

\* - user types different url into address bar

\* - user clicks on history (forward/back) button

\* - user clicks on a link

\*

\* It's not called when url is changed by $browser.url() method

\*

\* The listener gets called with new url as parameter.

\*

\* NOTE: this api is intended for use only by the $location service. Please use the

\* {@link ng.$location $location service} to monitor url changes in angular apps.

\*

\* @param {function(string)} listener Listener function to be called when url changes.

\* @return {function(string)} Returns the registered listener fn - handy if the fn is anonymous.

\*/

self.onUrlChange = function(callback) {

// TODO(vojta): refactor to use node's syntax for events

if (!urlChangeInit) {

// We listen on both (hashchange/popstate) when available, as some browsers (e.g. Opera)

// don't fire popstate when user change the address bar and don't fire hashchange when url

// changed by push/replaceState

// html5 history api - popstate event

if ($sniffer.history) jqLite(window).on('popstate', fireUrlChange);

// hashchange event

if ($sniffer.hashchange) jqLite(window).on('hashchange', fireUrlChange);

// polling

else self.addPollFn(fireUrlChange);

urlChangeInit = true;

}

urlChangeListeners.push(callback);

return callback;

};

//////////////////////////////////////////////////////////////

// Misc API

//////////////////////////////////////////////////////////////

/\*\*

\* @name $browser#baseHref

\*

\* @description

\* Returns current <base href>

\* (always relative - without domain)

\*

\* @returns {string} The current base href

\*/

self.baseHref = function() {

var href = baseElement.attr('href');

return href ? href.replace(/^(https?\:)?\/\/[^\/]\*/, '') : '';

};

//////////////////////////////////////////////////////////////

// Cookies API

//////////////////////////////////////////////////////////////

var lastCookies = {};

var lastCookieString = '';

var cookiePath = self.baseHref();

/\*\*

\* @name $browser#cookies

\*

\* @param {string=} name Cookie name

\* @param {string=} value Cookie value

\*

\* @description

\* The cookies method provides a 'private' low level access to browser cookies.

\* It is not meant to be used directly, use the $cookie service instead.

\*

\* The return values vary depending on the arguments that the method was called with as follows:

\*

\* - cookies() -> hash of all cookies, this is NOT a copy of the internal state, so do not modify

\* it

\* - cookies(name, value) -> set name to value, if value is undefined delete the cookie

\* - cookies(name) -> the same as (name, undefined) == DELETES (no one calls it right now that

\* way)

\*

\* @returns {Object} Hash of all cookies (if called without any parameter)

\*/

self.cookies = function(name, value) {

/\* global escape: false, unescape: false \*/

var cookieLength, cookieArray, cookie, i, index;

if (name) {

if (value === undefined) {

rawDocument.cookie = escape(name) + "=;path=" + cookiePath +

";expires=Thu, 01 Jan 1970 00:00:00 GMT";

} else {

if (isString(value)) {

cookieLength = (rawDocument.cookie = escape(name) + '=' + escape(value) +

';path=' + cookiePath).length + 1;

// per http://www.ietf.org/rfc/rfc2109.txt browser must allow at minimum:

// - 300 cookies

// - 20 cookies per unique domain

// - 4096 bytes per cookie

if (cookieLength > 4096) {

$log.warn("Cookie '"+ name +

"' possibly not set or overflowed because it was too large ("+

cookieLength + " > 4096 bytes)!");

}

}

}

} else {

if (rawDocument.cookie !== lastCookieString) {

lastCookieString = rawDocument.cookie;

cookieArray = lastCookieString.split("; ");

lastCookies = {};

for (i = 0; i < cookieArray.length; i++) {

cookie = cookieArray[i];

index = cookie.indexOf('=');

if (index > 0) { //ignore nameless cookies

name = unescape(cookie.substring(0, index));

// the first value that is seen for a cookie is the most

// specific one. values for the same cookie name that

// follow are for less specific paths.

if (lastCookies[name] === undefined) {

lastCookies[name] = unescape(cookie.substring(index + 1));

}

}

}

}

return lastCookies;

}

};

/\*\*

\* @name $browser#defer

\* @param {function()} fn A function, who's execution should be deferred.

\* @param {number=} [delay=0] of milliseconds to defer the function execution.

\* @returns {\*} DeferId that can be used to cancel the task via `$browser.defer.cancel()`.

\*

\* @description

\* Executes a fn asynchronously via `setTimeout(fn, delay)`.

\*

\* Unlike when calling `setTimeout` directly, in test this function is mocked and instead of using

\* `setTimeout` in tests, the fns are queued in an array, which can be programmatically flushed

\* via `$browser.defer.flush()`.

\*

\*/

self.defer = function(fn, delay) {

var timeoutId;

outstandingRequestCount++;

timeoutId = setTimeout(function() {

delete pendingDeferIds[timeoutId];

completeOutstandingRequest(fn);

}, delay || 0);

pendingDeferIds[timeoutId] = true;

return timeoutId;

};

/\*\*

\* @name $browser#defer.cancel

\*

\* @description

\* Cancels a deferred task identified with `deferId`.

\*

\* @param {\*} deferId Token returned by the `$browser.defer` function.

\* @returns {boolean} Returns `true` if the task hasn't executed yet and was successfully

\* canceled.

\*/

self.defer.cancel = function(deferId) {

if (pendingDeferIds[deferId]) {

delete pendingDeferIds[deferId];

clearTimeout(deferId);

completeOutstandingRequest(noop);

return true;

}

return false;

};

}

function $BrowserProvider(){

this.$get = ['$window', '$log', '$sniffer', '$document',

function( $window, $log, $sniffer, $document){

return new Browser($window, $document, $log, $sniffer);

}];

}

/\*\*

\* @ngdoc service

\* @name $cacheFactory

\*

\* @description

\* Factory that constructs {@link $cacheFactory.Cache Cache} objects and gives access to

\* them.

\*

\* ```js

\*

\* var cache = $cacheFactory('cacheId');

\* expect($cacheFactory.get('cacheId')).toBe(cache);

\* expect($cacheFactory.get('noSuchCacheId')).not.toBeDefined();

\*

\* cache.put("key", "value");

\* cache.put("another key", "another value");

\*

\* // We've specified no options on creation

\* expect(cache.info()).toEqual({id: 'cacheId', size: 2});

\*

\* ```

\*

\*

\* @param {string} cacheId Name or id of the newly created cache.

\* @param {object=} options Options object that specifies the cache behavior. Properties:

\*

\* - `{number=}` `capacity` — turns the cache into LRU cache.

\*

\* @returns {object} Newly created cache object with the following set of methods:

\*

\* - `{object}` `info()` — Returns id, size, and options of cache.

\* - `{{\*}}` `put({string} key, {\*} value)` — Puts a new key-value pair into the cache and returns

\* it.

\* - `{{\*}}` `get({string} key)` — Returns cached value for `key` or undefined for cache miss.

\* - `{void}` `remove({string} key)` — Removes a key-value pair from the cache.

\* - `{void}` `removeAll()` — Removes all cached values.

\* - `{void}` `destroy()` — Removes references to this cache from $cacheFactory.

\*

\* @example

<example module="cacheExampleApp">

<file name="index.html">

<div ng-controller="CacheController">

<input ng-model="newCacheKey" placeholder="Key">

<input ng-model="newCacheValue" placeholder="Value">

<button ng-click="put(newCacheKey, newCacheValue)">Cache</button>

<p ng-if="keys.length">Cached Values</p>

<div ng-repeat="key in keys">

<span ng-bind="key"></span>

<span>: </span>

<b ng-bind="cache.get(key)"></b>

</div>

<p>Cache Info</p>

<div ng-repeat="(key, value) in cache.info()">

<span ng-bind="key"></span>

<span>: </span>

<b ng-bind="value"></b>

</div>

</div>

</file>

<file name="script.js">

angular.module('cacheExampleApp', []).

controller('CacheController', ['$scope', '$cacheFactory', function($scope, $cacheFactory) {

$scope.keys = [];

$scope.cache = $cacheFactory('cacheId');

$scope.put = function(key, value) {

$scope.cache.put(key, value);

$scope.keys.push(key);

};

}]);

</file>

<file name="style.css">

p {

margin: 10px 0 3px;

}

</file>

</example>

\*/

function $CacheFactoryProvider() {

this.$get = function() {

var caches = {};

function cacheFactory(cacheId, options) {

if (cacheId in caches) {

throw minErr('$cacheFactory')('iid', "CacheId '{0}' is already taken!", cacheId);

}

var size = 0,

stats = extend({}, options, {id: cacheId}),

data = {},

capacity = (options && options.capacity) || Number.MAX\_VALUE,

lruHash = {},

freshEnd = null,

staleEnd = null;

/\*\*

\* @ngdoc type

\* @name $cacheFactory.Cache

\*

\* @description

\* A cache object used to store and retrieve data, primarily used by

\* {@link $http $http} and the {@link ng.directive:script script} directive to cache

\* templates and other data.

\*

\* ```js

\* angular.module('superCache')

\* .factory('superCache', ['$cacheFactory', function($cacheFactory) {

\* return $cacheFactory('super-cache');

\* }]);

\* ```

\*

\* Example test:

\*

\* ```js

\* it('should behave like a cache', inject(function(superCache) {

\* superCache.put('key', 'value');

\* superCache.put('another key', 'another value');

\*

\* expect(superCache.info()).toEqual({

\* id: 'super-cache',

\* size: 2

\* });

\*

\* superCache.remove('another key');

\* expect(superCache.get('another key')).toBeUndefined();

\*

\* superCache.removeAll();

\* expect(superCache.info()).toEqual({

\* id: 'super-cache',

\* size: 0

\* });

\* }));

\* ```

\*/

return caches[cacheId] = {

/\*\*

\* @ngdoc method

\* @name $cacheFactory.Cache#put

\* @function

\*

\* @description

\* Inserts a named entry into the {@link $cacheFactory.Cache Cache} object to be

\* retrieved later, and incrementing the size of the cache if the key was not already

\* present in the cache. If behaving like an LRU cache, it will also remove stale

\* entries from the set.

\*

\* It will not insert undefined values into the cache.

\*

\* @param {string} key the key under which the cached data is stored.

\* @param {\*} value the value to store alongside the key. If it is undefined, the key

\* will not be stored.

\* @returns {\*} the value stored.

\*/

put: function(key, value) {

if (capacity < Number.MAX\_VALUE) {

var lruEntry = lruHash[key] || (lruHash[key] = {key: key});

refresh(lruEntry);

}

if (isUndefined(value)) return;

if (!(key in data)) size++;

data[key] = value;

if (size > capacity) {

this.remove(staleEnd.key);

}

return value;

},

/\*\*

\* @ngdoc method

\* @name $cacheFactory.Cache#get

\* @function

\*

\* @description

\* Retrieves named data stored in the {@link $cacheFactory.Cache Cache} object.

\*

\* @param {string} key the key of the data to be retrieved

\* @returns {\*} the value stored.

\*/

get: function(key) {

if (capacity < Number.MAX\_VALUE) {

var lruEntry = lruHash[key];

if (!lruEntry) return;

refresh(lruEntry);

}

return data[key];

},

/\*\*

\* @ngdoc method

\* @name $cacheFactory.Cache#remove

\* @function

\*

\* @description

\* Removes an entry from the {@link $cacheFactory.Cache Cache} object.

\*

\* @param {string} key the key of the entry to be removed

\*/

remove: function(key) {

if (capacity < Number.MAX\_VALUE) {

var lruEntry = lruHash[key];

if (!lruEntry) return;

if (lruEntry == freshEnd) freshEnd = lruEntry.p;

if (lruEntry == staleEnd) staleEnd = lruEntry.n;

link(lruEntry.n,lruEntry.p);

delete lruHash[key];

}

delete data[key];

size--;

},

/\*\*

\* @ngdoc method

\* @name $cacheFactory.Cache#removeAll

\* @function

\*

\* @description

\* Clears the cache object of any entries.

\*/

removeAll: function() {

data = {};

size = 0;

lruHash = {};

freshEnd = staleEnd = null;

},

/\*\*

\* @ngdoc method

\* @name $cacheFactory.Cache#destroy

\* @function

\*

\* @description

\* Destroys the {@link $cacheFactory.Cache Cache} object entirely,

\* removing it from the {@link $cacheFactory $cacheFactory} set.

\*/

destroy: function() {

data = null;

stats = null;

lruHash = null;

delete caches[cacheId];

},

/\*\*

\* @ngdoc method

\* @name $cacheFactory.Cache#info

\* @function

\*

\* @description

\* Retrieve information regarding a particular {@link $cacheFactory.Cache Cache}.

\*

\* @returns {object} an object with the following properties:

\* <ul>

\* <li>\*\*id\*\*: the id of the cache instance</li>

\* <li>\*\*size\*\*: the number of entries kept in the cache instance</li>

\* <li>\*\*...\*\*: any additional properties from the options object when creating the

\* cache.</li>

\* </ul>

\*/

info: function() {

return extend({}, stats, {size: size});

}

};

/\*\*

\* makes the `entry` the freshEnd of the LRU linked list

\*/

function refresh(entry) {

if (entry != freshEnd) {

if (!staleEnd) {

staleEnd = entry;

} else if (staleEnd == entry) {

staleEnd = entry.n;

}

link(entry.n, entry.p);

link(entry, freshEnd);

freshEnd = entry;

freshEnd.n = null;

}

}

/\*\*

\* bidirectionally links two entries of the LRU linked list

\*/

function link(nextEntry, prevEntry) {

if (nextEntry != prevEntry) {

if (nextEntry) nextEntry.p = prevEntry; //p stands for previous, 'prev' didn't minify

if (prevEntry) prevEntry.n = nextEntry; //n stands for next, 'next' didn't minify

}

}

}

/\*\*

\* @ngdoc method

\* @name $cacheFactory#info

\*

\* @description

\* Get information about all the of the caches that have been created

\*

\* @returns {Object} - key-value map of `cacheId` to the result of calling `cache#info`

\*/

cacheFactory.info = function() {

var info = {};

forEach(caches, function(cache, cacheId) {

info[cacheId] = cache.info();

});

return info;

};

/\*\*

\* @ngdoc method

\* @name $cacheFactory#get

\*

\* @description

\* Get access to a cache object by the `cacheId` used when it was created.

\*

\* @param {string} cacheId Name or id of a cache to access.

\* @returns {object} Cache object identified by the cacheId or undefined if no such cache.

\*/

cacheFactory.get = function(cacheId) {

return caches[cacheId];

};

return cacheFactory;

};

}

/\*\*

\* @ngdoc service

\* @name $templateCache

\*

\* @description

\* The first time a template is used, it is loaded in the template cache for quick retrieval. You

\* can load templates directly into the cache in a `script` tag, or by consuming the

\* `$templateCache` service directly.

\*

\* Adding via the `script` tag:

\*

\* ```html

\* <script type="text/ng-template" id="templateId.html">

\* <p>This is the content of the template</p>

\* </script>

\* ```

\*

\* \*\*Note:\*\* the `script` tag containing the template does not need to be included in the `head` of

\* the document, but it must be below the `ng-app` definition.

\*

\* Adding via the $templateCache service:

\*

\* ```js

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

\* myApp.run(function($templateCache) {

\* $templateCache.put('templateId.html', 'This is the content of the template');

\* });

\* ```

\*

\* To retrieve the template later, simply use it in your HTML:

\* ```html

\* <div ng-include=" 'templateId.html' "></div>

\* ```

\*

\* or get it via Javascript:

\* ```js

\* $templateCache.get('templateId.html')

\* ```

\*

\* See {@link ng.$cacheFactory $cacheFactory}.

\*

\*/

function $TemplateCacheProvider() {

this.$get = ['$cacheFactory', function($cacheFactory) {

return $cacheFactory('templates');

}];

}

/\* ! VARIABLE/FUNCTION NAMING CONVENTIONS THAT APPLY TO THIS FILE!

\*

\* DOM-related variables:

\*

\* - "node" - DOM Node

\* - "element" - DOM Element or Node

\* - "$node" or "$element" - jqLite-wrapped node or element

\*

\*

\* Compiler related stuff:

\*

\* - "linkFn" - linking fn of a single directive

\* - "nodeLinkFn" - function that aggregates all linking fns for a particular node

\* - "childLinkFn" - function that aggregates all linking fns for child nodes of a particular node

\* - "compositeLinkFn" - function that aggregates all linking fns for a compilation root (nodeList)

\*/

/\*\*

\* @ngdoc service

\* @name $compile

\* @function

\*

\* @description

\* Compiles an HTML string or DOM into a template and produces a template function, which

\* can then be used to link {@link ng.$rootScope.Scope `scope`} and the template together.

\*

\* The compilation is a process of walking the DOM tree and matching DOM elements to

\* {@link ng.$compileProvider#directive directives}.

\*

\* <div class="alert alert-warning">

\* \*\*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 {@link guide/directive directive guide}.

\* </div>

\*

\* ## Comprehensive Directive API

\*

\* There are many different options for a directive.

\*

\* The difference resides in the return value of the factory function.

\* You can either return a "Directive Definition Object" (see below) that defines the directive properties,

\* or just the `postLink` function (all other properties will have the default values).

\*

\* <div class="alert alert-success">

\* \*\*Best Practice:\*\* It's recommended to use the "directive definition object" form.

\* </div>

\*

\* Here's an example directive declared with a Directive Definition Object:

\*

\* ```js

\* var myModule = angular.module(...);

\*

\* myModule.directive('directiveName', function factory(injectables) {

\* var directiveDefinitionObject = {

\* priority: 0,

\* template: '<div></div>', // or // function(tElement, tAttrs) { ... },

\* // or

\* // templateUrl: 'directive.html', // or // function(tElement, tAttrs) { ... },

\* replace: false,

\* transclude: false,

\* restrict: 'A',

\* scope: false,

\* controller: function($scope, $element, $attrs, $transclude, otherInjectables) { ... },

\* controllerAs: 'stringAlias',

\* require: 'siblingDirectiveName', // or // ['^parentDirectiveName', '?optionalDirectiveName', '?^optionalParent'],

\* compile: function compile(tElement, tAttrs, transclude) {

\* return {

\* pre: function preLink(scope, iElement, iAttrs, controller) { ... },

\* post: function postLink(scope, iElement, iAttrs, controller) { ... }

\* }

\* // or

\* // return function postLink( ... ) { ... }

\* },

\* // or

\* // link: {

\* // pre: function preLink(scope, iElement, iAttrs, controller) { ... },

\* // post: function postLink(scope, iElement, iAttrs, controller) { ... }

\* // }

\* // or

\* // link: function postLink( ... ) { ... }

\* };

\* return directiveDefinitionObject;

\* });

\* ```

\*

\* <div class="alert alert-warning">

\* \*\*Note:\*\* Any unspecified options will use the default value. You can see the default values below.

\* </div>

\*

\* Therefore the above can be simplified as:

\*

\* ```js

\* var myModule = angular.module(...);

\*

\* myModule.directive('directiveName', function factory(injectables) {

\* var directiveDefinitionObject = {

\* link: function postLink(scope, iElement, iAttrs) { ... }

\* };

\* return directiveDefinitionObject;

\* // or

\* // return function postLink(scope, iElement, iAttrs) { ... }

\* });

\* ```

\*

\*

\*

\* ### Directive Definition Object

\*

\* The directive definition object provides instructions to the {@link ng.$compile

\* compiler}. The attributes are:

\*

\* #### `priority`

\* When there are multiple directives defined on a single DOM element, sometimes it

\* is necessary to specify the order in which the directives are applied. The `priority` is used

\* to sort the directives before their `compile` functions get called. Priority is defined as a

\* number. Directives with greater numerical `priority` are compiled first. Pre-link functions

\* are also run in priority order, but post-link functions are run in reverse order. The order

\* of directives with the same priority is undefined. The default priority is `0`.

\*

\* #### `terminal`

\* If set to true then the current `priority` will be the last set of directives

\* which will execute (any directives at the current priority will still execute

\* as the order of execution on same `priority` is undefined).

\*

\* #### `scope`

\* \*\*If set to `true`,\*\* then a new scope will be created for this directive. If multiple directives on the

\* same element request a new scope, only one new scope is created. The new scope rule does not

\* apply for the root of the template since the root of the template always gets a new scope.

\*

\* \*\*If set to `{}` (object hash),\*\* then a new "isolate" scope is created. The 'isolate' scope differs from

\* normal scope in that it does not prototypically inherit from the parent scope. This is useful

\* when creating reusable components, which should not accidentally read or modify data in the

\* parent scope.

\*

\* The 'isolate' scope takes an object hash which defines a set of local scope properties

\* derived from the parent scope. These local properties are useful for aliasing values for

\* templates. Locals definition is a hash of local scope property to its source:

\*

\* \* `@` or `@attr` - bind a local scope property to the value of DOM attribute. The result is

\* always a string since DOM attributes are strings. If no `attr` name is specified then the

\* attribute name is assumed to be the same as the local name.

\* Given `<widget my-attr="hello {{name}}">` and widget definition

\* of `scope: { localName:'@myAttr' }`, then widget scope property `localName` will reflect

\* the interpolated value of `hello {{name}}`. As the `name` attribute changes so will the

\* `localName` property on the widget scope. The `name` is read from the parent scope (not

\* component scope).

\*

\* \* `=` or `=attr` - set up bi-directional binding between a local scope property and the

\* parent scope property of name defined via the value of the `attr` attribute. If no `attr`

\* name is specified then the attribute name is assumed to be the same as the local name.

\* Given `<widget my-attr="parentModel">` and widget definition of

\* `scope: { localModel:'=myAttr' }`, then widget scope property `localModel` will reflect the

\* value of `parentModel` on the parent scope. Any changes to `parentModel` will be reflected

\* in `localModel` and any changes in `localModel` will reflect in `parentModel`. If the parent

\* scope property doesn't exist, it will throw a NON\_ASSIGNABLE\_MODEL\_EXPRESSION exception. You

\* can avoid this behavior using `=?` or `=?attr` in order to flag the property as optional.

\*

\* \* `&` or `&attr` - provides a way to execute an expression in the context of the parent scope.

\* If no `attr` name is specified then the attribute name is assumed to be the same as the

\* local name. Given `<widget my-attr="count = count + value">` and widget definition of

\* `scope: { localFn:'&myAttr' }`, then isolate scope property `localFn` will point to

\* a function wrapper for the `count = count + value` expression. Often it's desirable to

\* pass data from the isolated scope via an expression and to the parent scope, this can be

\* done by passing a map of local variable names and values into the expression wrapper fn.

\* For example, if the expression is `increment(amount)` then we can specify the amount value

\* by calling the `localFn` as `localFn({amount: 22})`.

\*

\*

\*

\* #### `controller`

\* Controller constructor function. The controller is instantiated before the

\* pre-linking phase and it is shared with other directives (see

\* `require` attribute). This allows the directives to communicate with each other and augment

\* each other's behavior. The controller is injectable (and supports bracket notation) with the following locals:

\*

\* \* `$scope` - Current scope associated with the element

\* \* `$element` - Current element

\* \* `$attrs` - Current attributes object for the element

\* \* `$transclude` - A transclude linking function pre-bound to the correct transclusion scope.

\* The scope can be overridden by an optional first argument.

\* `function([scope], cloneLinkingFn)`.

\*

\*

\* #### `require`

\* Require another directive and inject its controller as the fourth argument to the linking function. The

\* `require` takes a string name (or array of strings) of the directive(s) to pass in. If an array is used, the

\* injected argument will be an array in corresponding order. If no such directive can be

\* found, or if the directive does not have a controller, then an error is raised. The name can be prefixed with:

\*

\* \* (no prefix) - Locate the required controller on the current element. Throw an error if not found.

\* \* `?` - Attempt to locate the required controller or pass `null` to the `link` fn if not found.

\* \* `^` - Locate the required controller by searching the element's parents. Throw an error if not found.

\* \* `?^` - Attempt to locate the required controller by searching the element's parents or pass `null` to the

\* `link` fn if not found.

\*

\*

\* #### `controllerAs`

\* Controller alias at the directive scope. An alias for the controller so it

\* can be referenced at the directive template. The directive needs to define a scope for this

\* configuration to be used. Useful in the case when directive is used as component.

\*

\*

\* #### `restrict`

\* String of subset of `EACM` which restricts the directive to a specific directive

\* declaration style. If omitted, the default (attributes only) is used.

\*

\* \* `E` - Element name: `<my-directive></my-directive>`

\* \* `A` - Attribute (default): `<div my-directive="exp"></div>`

\* \* `C` - Class: `<div class="my-directive: exp;"></div>`

\* \* `M` - Comment: `<!-- directive: my-directive exp -->`

\*

\*

\* #### `template`

\* replace the current element with the contents of the HTML. The replacement process

\* migrates all of the attributes / classes from the old element to the new one. See the

\* {@link guide/directive#creating-custom-directives\_creating-directives\_template-expanding-directive

\* Directives Guide} for an example.

\*

\* You can specify `template` as a string representing the template or as a function which takes

\* two arguments `tElement` and `tAttrs` (described in the `compile` function api below) and

\* returns a string value representing the template.

\*

\*

\* #### `templateUrl`

\* Same as `template` but the template is loaded from the specified URL. Because

\* the template loading is asynchronous the compilation/linking is suspended until the template

\* is loaded.

\*

\* You can specify `templateUrl` as a string representing the URL or as a function which takes two

\* arguments `tElement` and `tAttrs` (described in the `compile` function api below) and returns

\* a string value representing the url. In either case, the template URL is passed through {@link

\* api/ng.$sce#getTrustedResourceUrl $sce.getTrustedResourceUrl}.

\*

\*

\* #### `replace`

\* specify where the template should be inserted. Defaults to `false`.

\*

\* \* `true` - the template will replace the current element.

\* \* `false` - the template will replace the contents of the current element.

\*

\*

\* #### `transclude`

\* compile the content of the element and make it available to the directive.

\* Typically used with {@link ng.directive:ngTransclude

\* ngTransclude}. The advantage of transclusion is that the linking function receives a

\* transclusion function which is pre-bound to the correct scope. In a typical setup the widget

\* creates an `isolate` scope, but the transclusion is not a child, but a sibling of the `isolate`

\* scope. This makes it possible for the widget to have private state, and the transclusion to

\* be bound to the parent (pre-`isolate`) scope.

\*

\* \* `true` - transclude the content of the directive.

\* \* `'element'` - transclude the whole element including any directives defined at lower priority.

\*

\*

\* #### `compile`

\*

\* ```js

\* function compile(tElement, tAttrs, transclude) { ... }

\* ```

\*

\* The compile function deals with transforming the template DOM. Since most directives do not do

\* template transformation, it is not used often. Examples that require compile functions are

\* directives that transform template DOM, such as {@link

\* api/ng.directive:ngRepeat ngRepeat}, or load the contents

\* asynchronously, such as {@link ngRoute.directive:ngView ngView}. The

\* compile function takes the following arguments.

\*

\* \* `tElement` - template element - The element where the directive has been declared. It is

\* safe to do template transformation on the element and child elements only.

\*

\* \* `tAttrs` - template attributes - Normalized list of attributes declared on this element shared

\* between all directive compile functions.

\*

\* \* `transclude` - [\*DEPRECATED\*!] A transclude linking function: `function(scope, cloneLinkingFn)`

\*

\* <div class="alert alert-warning">

\* \*\*Note:\*\* The template instance and the link instance may be different objects if the template has

\* been cloned. For this reason it is \*\*not\*\* safe to do anything other than DOM transformations that

\* apply to all cloned DOM nodes within the compile function. Specifically, DOM listener registration

\* should be done in a linking function rather than in a compile function.

\* </div>

\* <div class="alert alert-warning">

\* \*\*Note:\*\* The compile function cannot handle directives that recursively use themselves in their

\* own templates or compile functions. Compiling these directives results in an infinite loop and a

\* stack overflow errors.

\*

\* This can be avoided by manually using $compile in the postLink function to imperatively compile

\* a directive's template instead of relying on automatic template compilation via `template` or

\* `templateUrl` declaration or manual compilation inside the compile function.

\* </div>

\*

\* <div class="alert alert-error">

\* \*\*Note:\*\* The `transclude` function that is passed to the compile function is deprecated, as it

\* e.g. does not know about the right outer scope. Please use the transclude function that is passed

\* to the link function instead.

\* </div>

\* A compile function can have a return value which can be either a function or an object.

\*

\* \* returning a (post-link) function - is equivalent to registering the linking function via the

\* `link` property of the config object when the compile function is empty.

\*

\* \* returning an object with function(s) registered via `pre` and `post` properties - allows you to

\* control when a linking function should be called during the linking phase. See info about

\* pre-linking and post-linking functions below.

\*

\*

\* #### `link`

\* This property is used only if the `compile` property is not defined.

\*

\* ```js

\* function link(scope, iElement, iAttrs, controller, transcludeFn) { ... }

\* ```

\*

\* The link function is responsible for registering DOM listeners as well as updating the DOM. It is

\* executed after the template has been cloned. This is where most of the directive logic will be

\* put.

\*

\* \* `scope` - {@link ng.$rootScope.Scope Scope} - The scope to be used by the

\* directive for registering {@link ng.$rootScope.Scope#$watch watches}.

\*

\* \* `iElement` - instance element - The element where the directive is to be used. It is safe to

\* manipulate the children of the element only in `postLink` function since the children have

\* already been linked.

\*

\* \* `iAttrs` - instance attributes - Normalized list of attributes declared on this element shared

\* between all directive linking functions.

\*

\* \* `controller` - a controller instance - A controller instance if at least one directive on the

\* element defines a controller. The controller is shared among all the directives, which allows

\* the directives to use the controllers as a communication channel.

\*

\* \* `transcludeFn` - A transclude linking function pre-bound to the correct transclusion scope.

\* The scope can be overridden by an optional first argument. This is the same as the `$transclude`

\* parameter of directive controllers.

\* `function([scope], cloneLinkingFn)`.

\*

\*

\* #### Pre-linking function

\*

\* Executed before the child elements are linked. Not safe to do DOM transformation since the

\* compiler linking function will fail to locate the correct elements for linking.

\*

\* #### Post-linking function

\*

\* Executed after the child elements are linked. It is safe to do DOM transformation in the post-linking function.

\*

\* <a name="Attributes"></a>

\* ### Attributes

\*

\* The {@link ng.$compile.directive.Attributes Attributes} object - passed as a parameter in the

\* `link()` or `compile()` functions. It has a variety of uses.

\*

\* accessing \*Normalized attribute names:\*

\* Directives like 'ngBind' can be expressed in many ways: 'ng:bind', `data-ng-bind`, or 'x-ng-bind'.

\* the attributes object allows for normalized access to

\* the attributes.

\*

\* \* \*Directive inter-communication:\* All directives share the same instance of the attributes

\* object which allows the directives to use the attributes object as inter directive

\* communication.

\*

\* \* \*Supports interpolation:\* Interpolation attributes are assigned to the attribute object

\* allowing other directives to read the interpolated value.

\*

\* \* \*Observing interpolated attributes:\* Use `$observe` to observe the value changes of attributes

\* that contain interpolation (e.g. `src="{{bar}}"`). Not only is this very efficient but it's also

\* the only way to easily get the actual value because during the linking phase the interpolation

\* hasn't been evaluated yet and so the value is at this time set to `undefined`.

\*

\* ```js

\* function linkingFn(scope, elm, attrs, ctrl) {

\* // get the attribute value

\* console.log(attrs.ngModel);

\*

\* // change the attribute

\* attrs.$set('ngModel', 'new value');

\*

\* // observe changes to interpolated attribute

\* attrs.$observe('ngModel', function(value) {

\* console.log('ngModel has changed value to ' + value);

\* });

\* }

\* ```

\*

\* Below is an example using `$compileProvider`.

\*

\* <div class="alert alert-warning">

\* \*\*Note\*\*: Typically directives are registered with `module.directive`. The example below is

\* to illustrate how `$compile` works.

\* </div>

\*

<example module="compile">

<file name="index.html">

<script>

angular.module('compile', [], function($compileProvider) {

// configure new 'compile' directive by passing a directive

// factory function. The factory function injects the '$compile'

$compileProvider.directive('compile', function($compile) {

// directive factory creates a link function

return function(scope, element, attrs) {

scope.$watch(

function(scope) {

// watch the 'compile' expression for changes

return scope.$eval(attrs.compile);

},

function(value) {

// when the 'compile' expression changes

// assign it into the current DOM

element.html(value);

// compile the new DOM and link it to the current

// scope.

// NOTE: we only compile .childNodes so that

// we don't get into infinite loop compiling ourselves

$compile(element.contents())(scope);

}

);

};

})

});

function Ctrl($scope) {

$scope.name = 'Angular';

$scope.html = 'Hello {{name}}';

}

</script>

<div ng-controller="Ctrl">

<input ng-model="name"> <br>

<textarea ng-model="html"></textarea> <br>

<div compile="html"></div>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should auto compile', function() {

var textarea = $('textarea');

var output = $('div[compile]');

// The initial state reads 'Hello Angular'.

expect(output.getText()).toBe('Hello Angular');

textarea.clear();

textarea.sendKeys('{{name}}!');

expect(output.getText()).toBe('Angular!');

});

</file>

</example>

\*

\*

\* @param {string|DOMElement} element Element or HTML string to compile into a template function.

\* @param {function(angular.Scope, cloneAttachFn=)} transclude function available to directives.

\* @param {number} maxPriority only apply directives lower than given priority (Only effects the

\* root element(s), not their children)

\* @returns {function(scope, cloneAttachFn=)} a link function which is used to bind template

\* (a DOM element/tree) to a scope. Where:

\*

\* \* `scope` - A {@link ng.$rootScope.Scope Scope} to bind to.

\* \* `cloneAttachFn` - If `cloneAttachFn` is provided, then the link function will clone the

\* `template` and call the `cloneAttachFn` function allowing the caller to attach the

\* cloned elements to the DOM document at the appropriate place. The `cloneAttachFn` is

\* called as: <br> `cloneAttachFn(clonedElement, scope)` where:

\*

\* \* `clonedElement` - is a clone of the original `element` passed into the compiler.

\* \* `scope` - is the current scope with which the linking function is working with.

\*

\* Calling the linking function returns the element of the template. It is either the original

\* element passed in, or the clone of the element if the `cloneAttachFn` is provided.

\*

\* After linking the view is not updated until after a call to $digest which typically is done by

\* Angular automatically.

\*

\* If you need access to the bound view, there are two ways to do it:

\*

\* - If you are not asking the linking function to clone the template, create the DOM element(s)

\* before you send them to the compiler and keep this reference around.

\* ```js

\* var element = $compile('<p>{{total}}</p>')(scope);

\* ```

\*

\* - if on the other hand, you need the element to be cloned, the view reference from the original

\* example would not point to the clone, but rather to the original template that was cloned. In

\* this case, you can access the clone via the cloneAttachFn:

\* ```js

\* var templateElement = angular.element('<p>{{total}}</p>'),

\* scope = ....;

\*

\* var clonedElement = $compile(templateElement)(scope, function(clonedElement, scope) {

\* //attach the clone to DOM document at the right place

\* });

\*

\* //now we have reference to the cloned DOM via `clonedElement`

\* ```

\*

\*

\* For information on how the compiler works, see the

\* {@link guide/compiler Angular HTML Compiler} section of the Developer Guide.

\*/

var $compileMinErr = minErr('$compile');

/\*\*

\* @ngdoc provider

\* @name $compileProvider

\* @function

\*

\* @description

\*/

$CompileProvider.$inject = ['$provide', '$$sanitizeUriProvider'];

function $CompileProvider($provide, $$sanitizeUriProvider) {

var hasDirectives = {},

Suffix = 'Directive',

COMMENT\_DIRECTIVE\_REGEXP = /^\s\*directive\:\s\*([\d\w\-\_]+)\s+(.\*)$/,

CLASS\_DIRECTIVE\_REGEXP = /(([\d\w\-\_]+)(?:\:([^;]+))?;?)/;

// Ref: http://developers.whatwg.org/webappapis.html#event-handler-idl-attributes

// The assumption is that future DOM event attribute names will begin with

// 'on' and be composed of only English letters.

var EVENT\_HANDLER\_ATTR\_REGEXP = /^(on[a-z]+|formaction)$/;

/\*\*

\* @ngdoc method

\* @name $compileProvider#directive

\* @function

\*

\* @description

\* Register a new directive with the compiler.

\*

\* @param {string|Object} name Name of the directive in camel-case (i.e. <code>ngBind</code> which

\* will match as <code>ng-bind</code>), or an object map of directives where the keys are the

\* names and the values are the factories.

\* @param {Function|Array} directiveFactory An injectable directive factory function. See

\* {@link guide/directive} for more info.

\* @returns {ng.$compileProvider} Self for chaining.

\*/

this.directive = function registerDirective(name, directiveFactory) {

assertNotHasOwnProperty(name, 'directive');

if (isString(name)) {

assertArg(directiveFactory, 'directiveFactory');

if (!hasDirectives.hasOwnProperty(name)) {

hasDirectives[name] = [];

$provide.factory(name + Suffix, ['$injector', '$exceptionHandler',

function($injector, $exceptionHandler) {

var directives = [];

forEach(hasDirectives[name], function(directiveFactory, index) {

try {

var directive = $injector.invoke(directiveFactory);

if (isFunction(directive)) {

directive = { compile: valueFn(directive) };

} else if (!directive.compile && directive.link) {

directive.compile = valueFn(directive.link);

}

directive.priority = directive.priority || 0;

directive.index = index;

directive.name = directive.name || name;

directive.require = directive.require || (directive.controller && directive.name);

directive.restrict = directive.restrict || 'A';

directives.push(directive);

} catch (e) {

$exceptionHandler(e);

}

});

return directives;

}]);

}

hasDirectives[name].push(directiveFactory);

} else {

forEach(name, reverseParams(registerDirective));

}

return this;

};

/\*\*

\* @ngdoc method

\* @name $compileProvider#aHrefSanitizationWhitelist

\* @function

\*

\* @description

\* Retrieves or overrides the default regular expression that is used for whitelisting of safe

\* urls during a[href] sanitization.

\*

\* The sanitization is a security measure aimed at prevent XSS attacks via html links.

\*

\* Any url about to be assigned to a[href] via data-binding is first normalized and turned into

\* an absolute url. Afterwards, the url is matched against the `aHrefSanitizationWhitelist`

\* regular expression. If a match is found, the original url is written into the dom. Otherwise,

\* the absolute url is prefixed with `'unsafe:'` string and only then is it written into the DOM.

\*

\* @param {RegExp=} regexp New regexp to whitelist urls with.

\* @returns {RegExp|ng.$compileProvider} Current RegExp if called without value or self for

\* chaining otherwise.

\*/

this.aHrefSanitizationWhitelist = function(regexp) {

if (isDefined(regexp)) {

$$sanitizeUriProvider.aHrefSanitizationWhitelist(regexp);

return this;

} else {

return $$sanitizeUriProvider.aHrefSanitizationWhitelist();

}

};

/\*\*

\* @ngdoc method

\* @name $compileProvider#imgSrcSanitizationWhitelist

\* @function

\*

\* @description

\* Retrieves or overrides the default regular expression that is used for whitelisting of safe

\* urls during img[src] sanitization.

\*

\* The sanitization is a security measure aimed at prevent XSS attacks via html links.

\*

\* Any url about to be assigned to img[src] via data-binding is first normalized and turned into

\* an absolute url. Afterwards, the url is matched against the `imgSrcSanitizationWhitelist`

\* regular expression. If a match is found, the original url is written into the dom. Otherwise,

\* the absolute url is prefixed with `'unsafe:'` string and only then is it written into the DOM.

\*

\* @param {RegExp=} regexp New regexp to whitelist urls with.

\* @returns {RegExp|ng.$compileProvider} Current RegExp if called without value or self for

\* chaining otherwise.

\*/

this.imgSrcSanitizationWhitelist = function(regexp) {

if (isDefined(regexp)) {

$$sanitizeUriProvider.imgSrcSanitizationWhitelist(regexp);

return this;

} else {

return $$sanitizeUriProvider.imgSrcSanitizationWhitelist();

}

};

this.$get = [

'$injector', '$interpolate', '$exceptionHandler', '$http', '$templateCache', '$parse',

'$controller', '$rootScope', '$document', '$sce', '$animate', '$$sanitizeUri',

function($injector, $interpolate, $exceptionHandler, $http, $templateCache, $parse,

$controller, $rootScope, $document, $sce, $animate, $$sanitizeUri) {

var Attributes = function(element, attr) {

this.$$element = element;

this.$attr = attr || {};

};

Attributes.prototype = {

$normalize: directiveNormalize,

/\*\*

\* @ngdoc method

\* @name $compile.directive.Attributes#$addClass

\* @function

\*

\* @description

\* Adds the CSS class value specified by the classVal parameter to the element. If animations

\* are enabled then an animation will be triggered for the class addition.

\*

\* @param {string} classVal The className value that will be added to the element

\*/

$addClass : function(classVal) {

if(classVal && classVal.length > 0) {

$animate.addClass(this.$$element, classVal);

}

},

/\*\*

\* @ngdoc method

\* @name $compile.directive.Attributes#$removeClass

\* @function

\*

\* @description

\* Removes the CSS class value specified by the classVal parameter from the element. If

\* animations are enabled then an animation will be triggered for the class removal.

\*

\* @param {string} classVal The className value that will be removed from the element

\*/

$removeClass : function(classVal) {

if(classVal && classVal.length > 0) {

$animate.removeClass(this.$$element, classVal);

}

},

/\*\*

\* @ngdoc method

\* @name $compile.directive.Attributes#$updateClass

\* @function

\*

\* @description

\* Adds and removes the appropriate CSS class values to the element based on the difference

\* between the new and old CSS class values (specified as newClasses and oldClasses).

\*

\* @param {string} newClasses The current CSS className value

\* @param {string} oldClasses The former CSS className value

\*/

$updateClass : function(newClasses, oldClasses) {

var toAdd = tokenDifference(newClasses, oldClasses);

var toRemove = tokenDifference(oldClasses, newClasses);

if(toAdd.length === 0) {

$animate.removeClass(this.$$element, toRemove);

} else if(toRemove.length === 0) {

$animate.addClass(this.$$element, toAdd);

} else {

$animate.setClass(this.$$element, toAdd, toRemove);

}

},

/\*\*

\* Set a normalized attribute on the element in a way such that all directives

\* can share the attribute. This function properly handles boolean attributes.

\* @param {string} key Normalized key. (ie ngAttribute)

\* @param {string|boolean} value The value to set. If `null` attribute will be deleted.

\* @param {boolean=} writeAttr If false, does not write the value to DOM element attribute.

\* Defaults to true.

\* @param {string=} attrName Optional none normalized name. Defaults to key.

\*/

$set: function(key, value, writeAttr, attrName) {

// TODO: decide whether or not to throw an error if "class"

//is set through this function since it may cause $updateClass to

//become unstable.

var booleanKey = getBooleanAttrName(this.$$element[0], key),

normalizedVal,

nodeName;

if (booleanKey) {

this.$$element.prop(key, value);

attrName = booleanKey;

}

this[key] = value;

// translate normalized key to actual key

if (attrName) {

this.$attr[key] = attrName;

} else {

attrName = this.$attr[key];

if (!attrName) {

this.$attr[key] = attrName = snake\_case(key, '-');

}

}

nodeName = nodeName\_(this.$$element);

// sanitize a[href] and img[src] values

if ((nodeName === 'A' && key === 'href') ||

(nodeName === 'IMG' && key === 'src')) {

this[key] = value = $$sanitizeUri(value, key === 'src');

}

if (writeAttr !== false) {

if (value === null || value === undefined) {

this.$$element.removeAttr(attrName);

} else {

this.$$element.attr(attrName, value);

}

}

// fire observers

var $$observers = this.$$observers;

$$observers && forEach($$observers[key], function(fn) {

try {

fn(value);

} catch (e) {

$exceptionHandler(e);

}

});

},

/\*\*

\* @ngdoc method

\* @name $compile.directive.Attributes#$observe

\* @function

\*

\* @description

\* Observes an interpolated attribute.

\*

\* The observer function will be invoked once during the next `$digest` following

\* compilation. The observer is then invoked whenever the interpolated value

\* changes.

\*

\* @param {string} key Normalized key. (ie ngAttribute) .

\* @param {function(interpolatedValue)} fn Function that will be called whenever

the interpolated value of the attribute changes.

\* See the {@link guide/directive#Attributes Directives} guide for more info.

\* @returns {function()} the `fn` parameter.

\*/

$observe: function(key, fn) {

var attrs = this,

$$observers = (attrs.$$observers || (attrs.$$observers = {})),

listeners = ($$observers[key] || ($$observers[key] = []));

listeners.push(fn);

$rootScope.$evalAsync(function() {

if (!listeners.$$inter) {

// no one registered attribute interpolation function, so lets call it manually

fn(attrs[key]);

}

});

return fn;

}

};

var startSymbol = $interpolate.startSymbol(),

endSymbol = $interpolate.endSymbol(),

denormalizeTemplate = (startSymbol == '{{' || endSymbol == '}}')

? identity

: function denormalizeTemplate(template) {

return template.replace(/\{\{/g, startSymbol).replace(/}}/g, endSymbol);

},

NG\_ATTR\_BINDING = /^ngAttr[A-Z]/;

return compile;

//================================

function compile($compileNodes, transcludeFn, maxPriority, ignoreDirective,

previousCompileContext) {

if (!($compileNodes instanceof jqLite)) {

// jquery always rewraps, whereas we need to preserve the original selector so that we can

// modify it.

$compileNodes = jqLite($compileNodes);

}

// We can not compile top level text elements since text nodes can be merged and we will

// not be able to attach scope data to them, so we will wrap them in <span>

forEach($compileNodes, function(node, index){

if (node.nodeType == 3 /\* text node \*/ && node.nodeValue.match(/\S+/) /\* non-empty \*/ ) {

$compileNodes[index] = node = jqLite(node).wrap('<span></span>').parent()[0];

}

});

var compositeLinkFn =

compileNodes($compileNodes, transcludeFn, $compileNodes,

maxPriority, ignoreDirective, previousCompileContext);

safeAddClass($compileNodes, 'ng-scope');

return function publicLinkFn(scope, cloneConnectFn, transcludeControllers){

assertArg(scope, 'scope');

// important!!: we must call our jqLite.clone() since the jQuery one is trying to be smart

// and sometimes changes the structure of the DOM.

var $linkNode = cloneConnectFn

? JQLitePrototype.clone.call($compileNodes) // IMPORTANT!!!

: $compileNodes;

forEach(transcludeControllers, function(instance, name) {

$linkNode.data('$' + name + 'Controller', instance);

});

// Attach scope only to non-text nodes.

for(var i = 0, ii = $linkNode.length; i<ii; i++) {

var node = $linkNode[i],

nodeType = node.nodeType;

if (nodeType === 1 /\* element \*/ || nodeType === 9 /\* document \*/) {

$linkNode.eq(i).data('$scope', scope);

}

}

if (cloneConnectFn) cloneConnectFn($linkNode, scope);

if (compositeLinkFn) compositeLinkFn(scope, $linkNode, $linkNode);

return $linkNode;

};

}

function safeAddClass($element, className) {

try {

$element.addClass(className);

} catch(e) {

// ignore, since it means that we are trying to set class on

// SVG element, where class name is read-only.

}

}

/\*\*

\* Compile function matches each node in nodeList against the directives. Once all directives

\* for a particular node are collected their compile functions are executed. The compile

\* functions return values - the linking functions - are combined into a composite linking

\* function, which is the a linking function for the node.

\*

\* @param {NodeList} nodeList an array of nodes or NodeList to compile

\* @param {function(angular.Scope, cloneAttachFn=)} transcludeFn A linking function, where the

\* scope argument is auto-generated to the new child of the transcluded parent scope.

\* @param {DOMElement=} $rootElement If the nodeList is the root of the compilation tree then

\* the rootElement must be set the jqLite collection of the compile root. This is

\* needed so that the jqLite collection items can be replaced with widgets.

\* @param {number=} maxPriority Max directive priority.

\* @returns {Function} A composite linking function of all of the matched directives or null.

\*/

function compileNodes(nodeList, transcludeFn, $rootElement, maxPriority, ignoreDirective,

previousCompileContext) {

var linkFns = [],

attrs, directives, nodeLinkFn, childNodes, childLinkFn, linkFnFound;

for (var i = 0; i < nodeList.length; i++) {

attrs = new Attributes();

// we must always refer to nodeList[i] since the nodes can be replaced underneath us.

directives = collectDirectives(nodeList[i], [], attrs, i === 0 ? maxPriority : undefined,

ignoreDirective);

nodeLinkFn = (directives.length)

? applyDirectivesToNode(directives, nodeList[i], attrs, transcludeFn, $rootElement,

null, [], [], previousCompileContext)

: null;

if (nodeLinkFn && nodeLinkFn.scope) {

safeAddClass(jqLite(nodeList[i]), 'ng-scope');

}

childLinkFn = (nodeLinkFn && nodeLinkFn.terminal ||

!(childNodes = nodeList[i].childNodes) ||

!childNodes.length)

? null

: compileNodes(childNodes,

nodeLinkFn ? nodeLinkFn.transclude : transcludeFn);

linkFns.push(nodeLinkFn, childLinkFn);

linkFnFound = linkFnFound || nodeLinkFn || childLinkFn;

//use the previous context only for the first element in the virtual group

previousCompileContext = null;

}

// return a linking function if we have found anything, null otherwise

return linkFnFound ? compositeLinkFn : null;

function compositeLinkFn(scope, nodeList, $rootElement, boundTranscludeFn) {

var nodeLinkFn, childLinkFn, node, $node, childScope, childTranscludeFn, i, ii, n;

// copy nodeList so that linking doesn't break due to live list updates.

var nodeListLength = nodeList.length,

stableNodeList = new Array(nodeListLength);

for (i = 0; i < nodeListLength; i++) {

stableNodeList[i] = nodeList[i];

}

for(i = 0, n = 0, ii = linkFns.length; i < ii; n++) {

node = stableNodeList[n];

nodeLinkFn = linkFns[i++];

childLinkFn = linkFns[i++];

$node = jqLite(node);

if (nodeLinkFn) {

if (nodeLinkFn.scope) {

childScope = scope.$new();

$node.data('$scope', childScope);

} else {

childScope = scope;

}

childTranscludeFn = nodeLinkFn.transclude;

if (childTranscludeFn || (!boundTranscludeFn && transcludeFn)) {

nodeLinkFn(childLinkFn, childScope, node, $rootElement,

createBoundTranscludeFn(scope, childTranscludeFn || transcludeFn)

);

} else {

nodeLinkFn(childLinkFn, childScope, node, $rootElement, boundTranscludeFn);

}

} else if (childLinkFn) {

childLinkFn(scope, node.childNodes, undefined, boundTranscludeFn);

}

}

}

}

function createBoundTranscludeFn(scope, transcludeFn) {

return function boundTranscludeFn(transcludedScope, cloneFn, controllers) {

var scopeCreated = false;

if (!transcludedScope) {

transcludedScope = scope.$new();

transcludedScope.$$transcluded = true;

scopeCreated = true;

}

var clone = transcludeFn(transcludedScope, cloneFn, controllers);

if (scopeCreated) {

clone.on('$destroy', bind(transcludedScope, transcludedScope.$destroy));

}

return clone;

};

}

/\*\*

\* Looks for directives on the given node and adds them to the directive collection which is

\* sorted.

\*

\* @param node Node to search.

\* @param directives An array to which the directives are added to. This array is sorted before

\* the function returns.

\* @param attrs The shared attrs object which is used to populate the normalized attributes.

\* @param {number=} maxPriority Max directive priority.

\*/

function collectDirectives(node, directives, attrs, maxPriority, ignoreDirective) {

var nodeType = node.nodeType,

attrsMap = attrs.$attr,

match,

className;

switch(nodeType) {

case 1: /\* Element \*/

// use the node name: <directive>

addDirective(directives,

directiveNormalize(nodeName\_(node).toLowerCase()), 'E', maxPriority, ignoreDirective);

// iterate over the attributes

for (var attr, name, nName, ngAttrName, value, nAttrs = node.attributes,

j = 0, jj = nAttrs && nAttrs.length; j < jj; j++) {

var attrStartName = false;

var attrEndName = false;

attr = nAttrs[j];

if (!msie || msie >= 8 || attr.specified) {

name = attr.name;

// support ngAttr attribute binding

ngAttrName = directiveNormalize(name);

if (NG\_ATTR\_BINDING.test(ngAttrName)) {

name = snake\_case(ngAttrName.substr(6), '-');

}

var directiveNName = ngAttrName.replace(/(Start|End)$/, '');

if (ngAttrName === directiveNName + 'Start') {

attrStartName = name;

attrEndName = name.substr(0, name.length - 5) + 'end';

name = name.substr(0, name.length - 6);

}

nName = directiveNormalize(name.toLowerCase());

attrsMap[nName] = name;

attrs[nName] = value = trim(attr.value);

if (getBooleanAttrName(node, nName)) {

attrs[nName] = true; // presence means true

}

addAttrInterpolateDirective(node, directives, value, nName);

addDirective(directives, nName, 'A', maxPriority, ignoreDirective, attrStartName,

attrEndName);

}

}

// use class as directive

className = node.className;

if (isString(className) && className !== '') {

while (match = CLASS\_DIRECTIVE\_REGEXP.exec(className)) {

nName = directiveNormalize(match[2]);

if (addDirective(directives, nName, 'C', maxPriority, ignoreDirective)) {

attrs[nName] = trim(match[3]);

}

className = className.substr(match.index + match[0].length);

}

}

break;

case 3: /\* Text Node \*/

addTextInterpolateDirective(directives, node.nodeValue);

break;

case 8: /\* Comment \*/

try {

match = COMMENT\_DIRECTIVE\_REGEXP.exec(node.nodeValue);

if (match) {

nName = directiveNormalize(match[1]);

if (addDirective(directives, nName, 'M', maxPriority, ignoreDirective)) {

attrs[nName] = trim(match[2]);

}

}

} catch (e) {

// turns out that under some circumstances IE9 throws errors when one attempts to read

// comment's node value.

// Just ignore it and continue. (Can't seem to reproduce in test case.)

}

break;

}

directives.sort(byPriority);

return directives;

}

/\*\*

\* Given a node with an directive-start it collects all of the siblings until it finds

\* directive-end.

\* @param node

\* @param attrStart

\* @param attrEnd

\* @returns {\*}

\*/

function groupScan(node, attrStart, attrEnd) {

var nodes = [];

var depth = 0;

if (attrStart && node.hasAttribute && node.hasAttribute(attrStart)) {

var startNode = node;

do {

if (!node) {

throw $compileMinErr('uterdir',

"Unterminated attribute, found '{0}' but no matching '{1}' found.",

attrStart, attrEnd);

}

if (node.nodeType == 1 /\*\* Element \*\*/) {

if (node.hasAttribute(attrStart)) depth++;

if (node.hasAttribute(attrEnd)) depth--;

}

nodes.push(node);

node = node.nextSibling;

} while (depth > 0);

} else {

nodes.push(node);

}

return jqLite(nodes);

}

/\*\*

\* Wrapper for linking function which converts normal linking function into a grouped

\* linking function.

\* @param linkFn

\* @param attrStart

\* @param attrEnd

\* @returns {Function}

\*/

function groupElementsLinkFnWrapper(linkFn, attrStart, attrEnd) {

return function(scope, element, attrs, controllers, transcludeFn) {

element = groupScan(element[0], attrStart, attrEnd);

return linkFn(scope, element, attrs, controllers, transcludeFn);

};

}

/\*\*

\* Once the directives have been collected, their compile functions are executed. This method

\* is responsible for inlining directive templates as well as terminating the application

\* of the directives if the terminal directive has been reached.

\*

\* @param {Array} directives Array of collected directives to execute their compile function.

\* this needs to be pre-sorted by priority order.

\* @param {Node} compileNode The raw DOM node to apply the compile functions to

\* @param {Object} templateAttrs The shared attribute function

\* @param {function(angular.Scope, cloneAttachFn=)} transcludeFn A linking function, where the

\* scope argument is auto-generated to the new

\* child of the transcluded parent scope.

\* @param {JQLite} jqCollection If we are working on the root of the compile tree then this

\* argument has the root jqLite array so that we can replace nodes

\* on it.

\* @param {Object=} originalReplaceDirective An optional directive that will be ignored when

\* compiling the transclusion.

\* @param {Array.<Function>} preLinkFns

\* @param {Array.<Function>} postLinkFns

\* @param {Object} previousCompileContext Context used for previous compilation of the current

\* node

\* @returns {Function} linkFn

\*/

function applyDirectivesToNode(directives, compileNode, templateAttrs, transcludeFn,

jqCollection, originalReplaceDirective, preLinkFns, postLinkFns,

previousCompileContext) {

previousCompileContext = previousCompileContext || {};

var terminalPriority = -Number.MAX\_VALUE,

newScopeDirective,

controllerDirectives = previousCompileContext.controllerDirectives,

newIsolateScopeDirective = previousCompileContext.newIsolateScopeDirective,

templateDirective = previousCompileContext.templateDirective,

nonTlbTranscludeDirective = previousCompileContext.nonTlbTranscludeDirective,

hasTranscludeDirective = false,

hasElementTranscludeDirective = previousCompileContext.hasElementTranscludeDirective,

$compileNode = templateAttrs.$$element = jqLite(compileNode),

directive,

directiveName,

$template,

replaceDirective = originalReplaceDirective,

childTranscludeFn = transcludeFn,

linkFn,

directiveValue;

// executes all directives on the current element

for(var i = 0, ii = directives.length; i < ii; i++) {

directive = directives[i];

var attrStart = directive.$$start;

var attrEnd = directive.$$end;

// collect multiblock sections

if (attrStart) {

$compileNode = groupScan(compileNode, attrStart, attrEnd);

}

$template = undefined;

if (terminalPriority > directive.priority) {

break; // prevent further processing of directives

}

if (directiveValue = directive.scope) {

newScopeDirective = newScopeDirective || directive;

// skip the check for directives with async templates, we'll check the derived sync

// directive when the template arrives

if (!directive.templateUrl) {

assertNoDuplicate('new/isolated scope', newIsolateScopeDirective, directive,

$compileNode);

if (isObject(directiveValue)) {

newIsolateScopeDirective = directive;

}

}

}

directiveName = directive.name;

if (!directive.templateUrl && directive.controller) {

directiveValue = directive.controller;

controllerDirectives = controllerDirectives || {};

assertNoDuplicate("'" + directiveName + "' controller",

controllerDirectives[directiveName], directive, $compileNode);

controllerDirectives[directiveName] = directive;

}

if (directiveValue = directive.transclude) {

hasTranscludeDirective = true;

// Special case ngIf and ngRepeat so that we don't complain about duplicate transclusion.

// This option should only be used by directives that know how to safely handle element transclusion,

// where the transcluded nodes are added or replaced after linking.

if (!directive.$$tlb) {

assertNoDuplicate('transclusion', nonTlbTranscludeDirective, directive, $compileNode);

nonTlbTranscludeDirective = directive;

}

if (directiveValue == 'element') {

hasElementTranscludeDirective = true;

terminalPriority = directive.priority;

$template = groupScan(compileNode, attrStart, attrEnd);

$compileNode = templateAttrs.$$element =

jqLite(document.createComment(' ' + directiveName + ': ' +

templateAttrs[directiveName] + ' '));

compileNode = $compileNode[0];

replaceWith(jqCollection, jqLite(sliceArgs($template)), compileNode);

childTranscludeFn = compile($template, transcludeFn, terminalPriority,

replaceDirective && replaceDirective.name, {

// Don't pass in:

// - controllerDirectives - otherwise we'll create duplicates controllers

// - newIsolateScopeDirective or templateDirective - combining templates with

// element transclusion doesn't make sense.

//

// We need only nonTlbTranscludeDirective so that we prevent putting transclusion

// on the same element more than once.

nonTlbTranscludeDirective: nonTlbTranscludeDirective

});

} else {

$template = jqLite(jqLiteClone(compileNode)).contents();

$compileNode.empty(); // clear contents

childTranscludeFn = compile($template, transcludeFn);

}

}

if (directive.template) {

assertNoDuplicate('template', templateDirective, directive, $compileNode);

templateDirective = directive;

directiveValue = (isFunction(directive.template))

? directive.template($compileNode, templateAttrs)

: directive.template;

directiveValue = denormalizeTemplate(directiveValue);

if (directive.replace) {

replaceDirective = directive;

if (jqLiteIsTextNode(directiveValue)) {

$template = [];

} else {

$template = jqLite(directiveValue);

}

compileNode = $template[0];

if ($template.length != 1 || compileNode.nodeType !== 1) {

throw $compileMinErr('tplrt',

"Template for directive '{0}' must have exactly one root element. {1}",

directiveName, '');

}

replaceWith(jqCollection, $compileNode, compileNode);

var newTemplateAttrs = {$attr: {}};

// combine directives from the original node and from the template:

// - take the array of directives for this element

// - split it into two parts, those that already applied (processed) and those that weren't (unprocessed)

// - collect directives from the template and sort them by priority

// - combine directives as: processed + template + unprocessed

var templateDirectives = collectDirectives(compileNode, [], newTemplateAttrs);

var unprocessedDirectives = directives.splice(i + 1, directives.length - (i + 1));

if (newIsolateScopeDirective) {

markDirectivesAsIsolate(templateDirectives);

}

directives = directives.concat(templateDirectives).concat(unprocessedDirectives);

mergeTemplateAttributes(templateAttrs, newTemplateAttrs);

ii = directives.length;

} else {

$compileNode.html(directiveValue);

}

}

if (directive.templateUrl) {

assertNoDuplicate('template', templateDirective, directive, $compileNode);

templateDirective = directive;

if (directive.replace) {

replaceDirective = directive;

}

nodeLinkFn = compileTemplateUrl(directives.splice(i, directives.length - i), $compileNode,

templateAttrs, jqCollection, childTranscludeFn, preLinkFns, postLinkFns, {

controllerDirectives: controllerDirectives,

newIsolateScopeDirective: newIsolateScopeDirective,

templateDirective: templateDirective,

nonTlbTranscludeDirective: nonTlbTranscludeDirective

});

ii = directives.length;

} else if (directive.compile) {

try {

linkFn = directive.compile($compileNode, templateAttrs, childTranscludeFn);

if (isFunction(linkFn)) {

addLinkFns(null, linkFn, attrStart, attrEnd);

} else if (linkFn) {

addLinkFns(linkFn.pre, linkFn.post, attrStart, attrEnd);

}

} catch (e) {

$exceptionHandler(e, startingTag($compileNode));

}

}

if (directive.terminal) {

nodeLinkFn.terminal = true;

terminalPriority = Math.max(terminalPriority, directive.priority);

}

}

nodeLinkFn.scope = newScopeDirective && newScopeDirective.scope === true;

nodeLinkFn.transclude = hasTranscludeDirective && childTranscludeFn;

previousCompileContext.hasElementTranscludeDirective = hasElementTranscludeDirective;

// might be normal or delayed nodeLinkFn depending on if templateUrl is present

return nodeLinkFn;

////////////////////

function addLinkFns(pre, post, attrStart, attrEnd) {

if (pre) {

if (attrStart) pre = groupElementsLinkFnWrapper(pre, attrStart, attrEnd);

pre.require = directive.require;

if (newIsolateScopeDirective === directive || directive.$$isolateScope) {

pre = cloneAndAnnotateFn(pre, {isolateScope: true});

}

preLinkFns.push(pre);

}

if (post) {

if (attrStart) post = groupElementsLinkFnWrapper(post, attrStart, attrEnd);

post.require = directive.require;

if (newIsolateScopeDirective === directive || directive.$$isolateScope) {

post = cloneAndAnnotateFn(post, {isolateScope: true});

}

postLinkFns.push(post);

}

}

function getControllers(require, $element, elementControllers) {

var value, retrievalMethod = 'data', optional = false;

if (isString(require)) {

while((value = require.charAt(0)) == '^' || value == '?') {

require = require.substr(1);

if (value == '^') {

retrievalMethod = 'inheritedData';

}

optional = optional || value == '?';

}

value = null;

if (elementControllers && retrievalMethod === 'data') {

value = elementControllers[require];

}

value = value || $element[retrievalMethod]('$' + require + 'Controller');

if (!value && !optional) {

throw $compileMinErr('ctreq',

"Controller '{0}', required by directive '{1}', can't be found!",

require, directiveName);

}

return value;

} else if (isArray(require)) {

value = [];

forEach(require, function(require) {

value.push(getControllers(require, $element, elementControllers));

});

}

return value;

}

function nodeLinkFn(childLinkFn, scope, linkNode, $rootElement, boundTranscludeFn) {

var attrs, $element, i, ii, linkFn, controller, isolateScope, elementControllers = {}, transcludeFn;

if (compileNode === linkNode) {

attrs = templateAttrs;

} else {

attrs = shallowCopy(templateAttrs, new Attributes(jqLite(linkNode), templateAttrs.$attr));

}

$element = attrs.$$element;

if (newIsolateScopeDirective) {

var LOCAL\_REGEXP = /^\s\*([@=&])(\??)\s\*(\w\*)\s\*$/;

var $linkNode = jqLite(linkNode);

isolateScope = scope.$new(true);

if (templateDirective && (templateDirective === newIsolateScopeDirective.$$originalDirective)) {

$linkNode.data('$isolateScope', isolateScope) ;

} else {

$linkNode.data('$isolateScopeNoTemplate', isolateScope);

}

safeAddClass($linkNode, 'ng-isolate-scope');

forEach(newIsolateScopeDirective.scope, function(definition, scopeName) {

var match = definition.match(LOCAL\_REGEXP) || [],

attrName = match[3] || scopeName,

optional = (match[2] == '?'),

mode = match[1], // @, =, or &

lastValue,

parentGet, parentSet, compare;

isolateScope.$$isolateBindings[scopeName] = mode + attrName;

switch (mode) {

case '@':

attrs.$observe(attrName, function(value) {

isolateScope[scopeName] = value;

});

attrs.$$observers[attrName].$$scope = scope;

if( attrs[attrName] ) {

// If the attribute has been provided then we trigger an interpolation to ensure

// the value is there for use in the link fn

isolateScope[scopeName] = $interpolate(attrs[attrName])(scope);

}

break;

case '=':

if (optional && !attrs[attrName]) {

return;

}

parentGet = $parse(attrs[attrName]);

if (parentGet.literal) {

compare = equals;

} else {

compare = function(a,b) { return a === b; };

}

parentSet = parentGet.assign || function() {

// reset the change, or we will throw this exception on every $digest

lastValue = isolateScope[scopeName] = parentGet(scope);

throw $compileMinErr('nonassign',

"Expression '{0}' used with directive '{1}' is non-assignable!",

attrs[attrName], newIsolateScopeDirective.name);

};

lastValue = isolateScope[scopeName] = parentGet(scope);

isolateScope.$watch(function parentValueWatch() {

var parentValue = parentGet(scope);

if (!compare(parentValue, isolateScope[scopeName])) {

// we are out of sync and need to copy

if (!compare(parentValue, lastValue)) {

// parent changed and it has precedence

isolateScope[scopeName] = parentValue;

} else {

// if the parent can be assigned then do so

parentSet(scope, parentValue = isolateScope[scopeName]);

}

}

return lastValue = parentValue;

}, null, parentGet.literal);

break;

case '&':

parentGet = $parse(attrs[attrName]);

isolateScope[scopeName] = function(locals) {

return parentGet(scope, locals);

};

break;

default:

throw $compileMinErr('iscp',

"Invalid isolate scope definition for directive '{0}'." +

" Definition: {... {1}: '{2}' ...}",

newIsolateScopeDirective.name, scopeName, definition);

}

});

}

transcludeFn = boundTranscludeFn && controllersBoundTransclude;

if (controllerDirectives) {

forEach(controllerDirectives, function(directive) {

var locals = {

$scope: directive === newIsolateScopeDirective || directive.$$isolateScope ? isolateScope : scope,

$element: $element,

$attrs: attrs,

$transclude: transcludeFn

}, controllerInstance;

controller = directive.controller;

if (controller == '@') {

controller = attrs[directive.name];

}

controllerInstance = $controller(controller, locals);

// For directives with element transclusion the element is a comment,

// but jQuery .data doesn't support attaching data to comment nodes as it's hard to

// clean up (http://bugs.jquery.com/ticket/8335).

// Instead, we save the controllers for the element in a local hash and attach to .data

// later, once we have the actual element.

elementControllers[directive.name] = controllerInstance;

if (!hasElementTranscludeDirective) {

$element.data('$' + directive.name + 'Controller', controllerInstance);

}

if (directive.controllerAs) {

locals.$scope[directive.controllerAs] = controllerInstance;

}

});

}

// PRELINKING

for(i = 0, ii = preLinkFns.length; i < ii; i++) {

try {

linkFn = preLinkFns[i];

linkFn(linkFn.isolateScope ? isolateScope : scope, $element, attrs,

linkFn.require && getControllers(linkFn.require, $element, elementControllers), transcludeFn);

} catch (e) {

$exceptionHandler(e, startingTag($element));

}

}

// RECURSION

// We only pass the isolate scope, if the isolate directive has a template,

// otherwise the child elements do not belong to the isolate directive.

var scopeToChild = scope;

if (newIsolateScopeDirective && (newIsolateScopeDirective.template || newIsolateScopeDirective.templateUrl === null)) {

scopeToChild = isolateScope;

}

childLinkFn && childLinkFn(scopeToChild, linkNode.childNodes, undefined, boundTranscludeFn);

// POSTLINKING

for(i = postLinkFns.length - 1; i >= 0; i--) {

try {

linkFn = postLinkFns[i];

linkFn(linkFn.isolateScope ? isolateScope : scope, $element, attrs,

linkFn.require && getControllers(linkFn.require, $element, elementControllers), transcludeFn);

} catch (e) {

$exceptionHandler(e, startingTag($element));

}

}

// This is the function that is injected as `$transclude`.

function controllersBoundTransclude(scope, cloneAttachFn) {

var transcludeControllers;

// no scope passed

if (arguments.length < 2) {

cloneAttachFn = scope;

scope = undefined;

}

if (hasElementTranscludeDirective) {

transcludeControllers = elementControllers;

}

return boundTranscludeFn(scope, cloneAttachFn, transcludeControllers);

}

}

}

function markDirectivesAsIsolate(directives) {

// mark all directives as needing isolate scope.

for (var j = 0, jj = directives.length; j < jj; j++) {

directives[j] = inherit(directives[j], {$$isolateScope: true});

}

}

/\*\*

\* looks up the directive and decorates it with exception handling and proper parameters. We

\* call this the boundDirective.

\*

\* @param {string} name name of the directive to look up.

\* @param {string} location The directive must be found in specific format.

\* String containing any of theses characters:

\*

\* \* `E`: element name

\* \* `A': attribute

\* \* `C`: class

\* \* `M`: comment

\* @returns {boolean} true if directive was added.

\*/

function addDirective(tDirectives, name, location, maxPriority, ignoreDirective, startAttrName,

endAttrName) {

if (name === ignoreDirective) return null;

var match = null;

if (hasDirectives.hasOwnProperty(name)) {

for(var directive, directives = $injector.get(name + Suffix),

i = 0, ii = directives.length; i<ii; i++) {

try {

directive = directives[i];

if ( (maxPriority === undefined || maxPriority > directive.priority) &&

directive.restrict.indexOf(location) != -1) {

if (startAttrName) {

directive = inherit(directive, {$$start: startAttrName, $$end: endAttrName});

}

tDirectives.push(directive);

match = directive;

}

} catch(e) { $exceptionHandler(e); }

}

}

return match;

}

/\*\*

\* When the element is replaced with HTML template then the new attributes

\* on the template need to be merged with the existing attributes in the DOM.

\* The desired effect is to have both of the attributes present.

\*

\* @param {object} dst destination attributes (original DOM)

\* @param {object} src source attributes (from the directive template)

\*/

function mergeTemplateAttributes(dst, src) {

var srcAttr = src.$attr,

dstAttr = dst.$attr,

$element = dst.$$element;

// reapply the old attributes to the new element

forEach(dst, function(value, key) {

if (key.charAt(0) != '$') {

if (src[key]) {

value += (key === 'style' ? ';' : ' ') + src[key];

}

dst.$set(key, value, true, srcAttr[key]);

}

});

// copy the new attributes on the old attrs object

forEach(src, function(value, key) {

if (key == 'class') {

safeAddClass($element, value);

dst['class'] = (dst['class'] ? dst['class'] + ' ' : '') + value;

} else if (key == 'style') {

$element.attr('style', $element.attr('style') + ';' + value);

dst['style'] = (dst['style'] ? dst['style'] + ';' : '') + value;

// `dst` will never contain hasOwnProperty as DOM parser won't let it.

// You will get an "InvalidCharacterError: DOM Exception 5" error if you

// have an attribute like "has-own-property" or "data-has-own-property", etc.

} else if (key.charAt(0) != '$' && !dst.hasOwnProperty(key)) {

dst[key] = value;

dstAttr[key] = srcAttr[key];

}

});

}

function compileTemplateUrl(directives, $compileNode, tAttrs,

$rootElement, childTranscludeFn, preLinkFns, postLinkFns, previousCompileContext) {

var linkQueue = [],

afterTemplateNodeLinkFn,

afterTemplateChildLinkFn,

beforeTemplateCompileNode = $compileNode[0],

origAsyncDirective = directives.shift(),

// The fact that we have to copy and patch the directive seems wrong!

derivedSyncDirective = extend({}, origAsyncDirective, {

templateUrl: null, transclude: null, replace: null, $$originalDirective: origAsyncDirective

}),

templateUrl = (isFunction(origAsyncDirective.templateUrl))

? origAsyncDirective.templateUrl($compileNode, tAttrs)

: origAsyncDirective.templateUrl;

$compileNode.empty();

$http.get($sce.getTrustedResourceUrl(templateUrl), {cache: $templateCache}).

success(function(content) {

var compileNode, tempTemplateAttrs, $template, childBoundTranscludeFn;

content = denormalizeTemplate(content);

if (origAsyncDirective.replace) {

if (jqLiteIsTextNode(content)) {

$template = [];

} else {

$template = jqLite(content);

}

compileNode = $template[0];

if ($template.length != 1 || compileNode.nodeType !== 1) {

throw $compileMinErr('tplrt',

"Template for directive '{0}' must have exactly one root element. {1}",

origAsyncDirective.name, templateUrl);

}

tempTemplateAttrs = {$attr: {}};

replaceWith($rootElement, $compileNode, compileNode);

var templateDirectives = collectDirectives(compileNode, [], tempTemplateAttrs);

if (isObject(origAsyncDirective.scope)) {

markDirectivesAsIsolate(templateDirectives);

}

directives = templateDirectives.concat(directives);

mergeTemplateAttributes(tAttrs, tempTemplateAttrs);

} else {

compileNode = beforeTemplateCompileNode;

$compileNode.html(content);

}

directives.unshift(derivedSyncDirective);

afterTemplateNodeLinkFn = applyDirectivesToNode(directives, compileNode, tAttrs,

childTranscludeFn, $compileNode, origAsyncDirective, preLinkFns, postLinkFns,

previousCompileContext);

forEach($rootElement, function(node, i) {

if (node == compileNode) {

$rootElement[i] = $compileNode[0];

}

});

afterTemplateChildLinkFn = compileNodes($compileNode[0].childNodes, childTranscludeFn);

while(linkQueue.length) {

var scope = linkQueue.shift(),

beforeTemplateLinkNode = linkQueue.shift(),

linkRootElement = linkQueue.shift(),

boundTranscludeFn = linkQueue.shift(),

linkNode = $compileNode[0];

if (beforeTemplateLinkNode !== beforeTemplateCompileNode) {

var oldClasses = beforeTemplateLinkNode.className;

if (!(previousCompileContext.hasElementTranscludeDirective &&

origAsyncDirective.replace)) {

// it was cloned therefore we have to clone as well.

linkNode = jqLiteClone(compileNode);

}

replaceWith(linkRootElement, jqLite(beforeTemplateLinkNode), linkNode);

// Copy in CSS classes from original node

safeAddClass(jqLite(linkNode), oldClasses);

}

if (afterTemplateNodeLinkFn.transclude) {

childBoundTranscludeFn = createBoundTranscludeFn(scope, afterTemplateNodeLinkFn.transclude);

} else {

childBoundTranscludeFn = boundTranscludeFn;

}

afterTemplateNodeLinkFn(afterTemplateChildLinkFn, scope, linkNode, $rootElement,

childBoundTranscludeFn);

}

linkQueue = null;

}).

error(function(response, code, headers, config) {

throw $compileMinErr('tpload', 'Failed to load template: {0}', config.url);

});

return function delayedNodeLinkFn(ignoreChildLinkFn, scope, node, rootElement, boundTranscludeFn) {

if (linkQueue) {

linkQueue.push(scope);

linkQueue.push(node);

linkQueue.push(rootElement);

linkQueue.push(boundTranscludeFn);

} else {

afterTemplateNodeLinkFn(afterTemplateChildLinkFn, scope, node, rootElement, boundTranscludeFn);

}

};

}

/\*\*

\* Sorting function for bound directives.

\*/

function byPriority(a, b) {

var diff = b.priority - a.priority;

if (diff !== 0) return diff;

if (a.name !== b.name) return (a.name < b.name) ? -1 : 1;

return a.index - b.index;

}

function assertNoDuplicate(what, previousDirective, directive, element) {

if (previousDirective) {

throw $compileMinErr('multidir', 'Multiple directives [{0}, {1}] asking for {2} on: {3}',

previousDirective.name, directive.name, what, startingTag(element));

}

}

function addTextInterpolateDirective(directives, text) {

var interpolateFn = $interpolate(text, true);

if (interpolateFn) {

directives.push({

priority: 0,

compile: valueFn(function textInterpolateLinkFn(scope, node) {

var parent = node.parent(),

bindings = parent.data('$binding') || [];

bindings.push(interpolateFn);

safeAddClass(parent.data('$binding', bindings), 'ng-binding');

scope.$watch(interpolateFn, function interpolateFnWatchAction(value) {

node[0].nodeValue = value;

});

})

});

}

}

function getTrustedContext(node, attrNormalizedName) {

if (attrNormalizedName == "srcdoc") {

return $sce.HTML;

}

var tag = nodeName\_(node);

// maction[xlink:href] can source SVG. It's not limited to <maction>.

if (attrNormalizedName == "xlinkHref" ||

(tag == "FORM" && attrNormalizedName == "action") ||

(tag != "IMG" && (attrNormalizedName == "src" ||

attrNormalizedName == "ngSrc"))) {

return $sce.RESOURCE\_URL;

}

}

function addAttrInterpolateDirective(node, directives, value, name) {

var interpolateFn = $interpolate(value, true);

// no interpolation found -> ignore

if (!interpolateFn) return;

if (name === "multiple" && nodeName\_(node) === "SELECT") {

throw $compileMinErr("selmulti",

"Binding to the 'multiple' attribute is not supported. Element: {0}",

startingTag(node));

}

directives.push({

priority: 100,

compile: function() {

return {

pre: function attrInterpolatePreLinkFn(scope, element, attr) {

var $$observers = (attr.$$observers || (attr.$$observers = {}));

if (EVENT\_HANDLER\_ATTR\_REGEXP.test(name)) {

throw $compileMinErr('nodomevents',

"Interpolations for HTML DOM event attributes are disallowed. Please use the " +

"ng- versions (such as ng-click instead of onclick) instead.");

}

// we need to interpolate again, in case the attribute value has been updated

// (e.g. by another directive's compile function)

interpolateFn = $interpolate(attr[name], true, getTrustedContext(node, name));

// if attribute was updated so that there is no interpolation going on we don't want to

// register any observers

if (!interpolateFn) return;

// TODO(i): this should likely be attr.$set(name, iterpolateFn(scope) so that we reset the

// actual attr value

attr[name] = interpolateFn(scope);

($$observers[name] || ($$observers[name] = [])).$$inter = true;

(attr.$$observers && attr.$$observers[name].$$scope || scope).

$watch(interpolateFn, function interpolateFnWatchAction(newValue, oldValue) {

//special case for class attribute addition + removal

//so that class changes can tap into the animation

//hooks provided by the $animate service. Be sure to

//skip animations when the first digest occurs (when

//both the new and the old values are the same) since

//the CSS classes are the non-interpolated values

if(name === 'class' && newValue != oldValue) {

attr.$updateClass(newValue, oldValue);

} else {

attr.$set(name, newValue);

}

});

}

};

}

});

}

/\*\*

\* This is a special jqLite.replaceWith, which can replace items which

\* have no parents, provided that the containing jqLite collection is provided.

\*

\* @param {JqLite=} $rootElement The root of the compile tree. Used so that we can replace nodes

\* in the root of the tree.

\* @param {JqLite} elementsToRemove The jqLite element which we are going to replace. We keep

\* the shell, but replace its DOM node reference.

\* @param {Node} newNode The new DOM node.

\*/

function replaceWith($rootElement, elementsToRemove, newNode) {

var firstElementToRemove = elementsToRemove[0],

removeCount = elementsToRemove.length,

parent = firstElementToRemove.parentNode,

i, ii;

if ($rootElement) {

for(i = 0, ii = $rootElement.length; i < ii; i++) {

if ($rootElement[i] == firstElementToRemove) {

$rootElement[i++] = newNode;

for (var j = i, j2 = j + removeCount - 1,

jj = $rootElement.length;

j < jj; j++, j2++) {

if (j2 < jj) {

$rootElement[j] = $rootElement[j2];

} else {

delete $rootElement[j];

}

}

$rootElement.length -= removeCount - 1;

break;

}

}

}

if (parent) {

parent.replaceChild(newNode, firstElementToRemove);

}

var fragment = document.createDocumentFragment();

fragment.appendChild(firstElementToRemove);

newNode[jqLite.expando] = firstElementToRemove[jqLite.expando];

for (var k = 1, kk = elementsToRemove.length; k < kk; k++) {

var element = elementsToRemove[k];

jqLite(element).remove(); // must do this way to clean up expando

fragment.appendChild(element);

delete elementsToRemove[k];

}

elementsToRemove[0] = newNode;

elementsToRemove.length = 1;

}

function cloneAndAnnotateFn(fn, annotation) {

return extend(function() { return fn.apply(null, arguments); }, fn, annotation);

}

}];

}

var PREFIX\_REGEXP = /^(x[\:\-\_]|data[\:\-\_])/i;

/\*\*

\* Converts all accepted directives format into proper directive name.

\* All of these will become 'myDirective':

\* my:Directive

\* my-directive

\* x-my-directive

\* data-my:directive

\*

\* Also there is special case for Moz prefix starting with upper case letter.

\* @param name Name to normalize

\*/

function directiveNormalize(name) {

return camelCase(name.replace(PREFIX\_REGEXP, ''));

}

/\*\*

\* @ngdoc type

\* @name $compile.directive.Attributes

\*

\* @description

\* A shared object between directive compile / linking functions which contains normalized DOM

\* element attributes. The values reflect current binding state `{{ }}`. The normalization is

\* needed since all of these are treated as equivalent in Angular:

\*

\* <span ng:bind="a" ng-bind="a" data-ng-bind="a" x-ng-bind="a">

\*/

/\*\*

\* @ngdoc property

\* @name $compile.directive.Attributes#$attr

\* @returns {object} A map of DOM element attribute names to the normalized name. This is

\* needed to do reverse lookup from normalized name back to actual name.

\*/

/\*\*

\* @ngdoc method

\* @name $compile.directive.Attributes#$set

\* @function

\*

\* @description

\* Set DOM element attribute value.

\*

\*

\* @param {string} name Normalized element attribute name of the property to modify. The name is

\* reverse-translated using the {@link ng.$compile.directive.Attributes#$attr $attr}

\* property to the original name.

\* @param {string} value Value to set the attribute to. The value can be an interpolated string.

\*/

/\*\*

\* Closure compiler type information

\*/

function nodesetLinkingFn(

/\* angular.Scope \*/ scope,

/\* NodeList \*/ nodeList,

/\* Element \*/ rootElement,

/\* function(Function) \*/ boundTranscludeFn

){}

function directiveLinkingFn(

/\* nodesetLinkingFn \*/ nodesetLinkingFn,

/\* angular.Scope \*/ scope,

/\* Node \*/ node,

/\* Element \*/ rootElement,

/\* function(Function) \*/ boundTranscludeFn

){}

function tokenDifference(str1, str2) {

var values = '',

tokens1 = str1.split(/\s+/),

tokens2 = str2.split(/\s+/);

outer:

for(var i = 0; i < tokens1.length; i++) {

var token = tokens1[i];

for(var j = 0; j < tokens2.length; j++) {

if(token == tokens2[j]) continue outer;

}

values += (values.length > 0 ? ' ' : '') + token;

}

return values;

}

/\*\*

\* @ngdoc provider

\* @name $controllerProvider

\* @description

\* The {@link ng.$controller $controller service} is used by Angular to create new

\* controllers.

\*

\* This provider allows controller registration via the

\* {@link ng.$controllerProvider#register register} method.

\*/

function $ControllerProvider() {

var controllers = {},

CNTRL\_REG = /^(\S+)(\s+as\s+(\w+))?$/;

/\*\*

\* @ngdoc method

\* @name $controllerProvider#register

\* @param {string|Object} name Controller name, or an object map of controllers where the keys are

\* the names and the values are the constructors.

\* @param {Function|Array} constructor Controller constructor fn (optionally decorated with DI

\* annotations in the array notation).

\*/

this.register = function(name, constructor) {

assertNotHasOwnProperty(name, 'controller');

if (isObject(name)) {

extend(controllers, name);

} else {

controllers[name] = constructor;

}

};

this.$get = ['$injector', '$window', function($injector, $window) {

/\*\*

\* @ngdoc service

\* @name $controller

\* @requires $injector

\*

\* @param {Function|string} constructor If called with a function then it's considered to be the

\* controller constructor function. Otherwise it's considered to be a string which is used

\* to retrieve the controller constructor using the following steps:

\*

\* \* check if a controller with given name is registered via `$controllerProvider`

\* \* check if evaluating the string on the current scope returns a constructor

\* \* check `window[constructor]` on the global `window` object

\*

\* @param {Object} locals Injection locals for Controller.

\* @return {Object} Instance of given controller.

\*

\* @description

\* `$controller` service is responsible for instantiating controllers.

\*

\* It's just a simple call to {@link auto.$injector $injector}, but extracted into

\* a service, so that one can override this service with [BC version](https://gist.github.com/1649788).

\*/

return function(expression, locals) {

var instance, match, constructor, identifier;

if(isString(expression)) {

match = expression.match(CNTRL\_REG),

constructor = match[1],

identifier = match[3];

expression = controllers.hasOwnProperty(constructor)

? controllers[constructor]

: getter(locals.$scope, constructor, true) || getter($window, constructor, true);

assertArgFn(expression, constructor, true);

}

instance = $injector.instantiate(expression, locals);

if (identifier) {

if (!(locals && typeof locals.$scope == 'object')) {

throw minErr('$controller')('noscp',

"Cannot export controller '{0}' as '{1}'! No $scope object provided via `locals`.",

constructor || expression.name, identifier);

}

locals.$scope[identifier] = instance;

}

return instance;

};

}];

}

/\*\*

\* @ngdoc service

\* @name $document

\* @requires $window

\*

\* @description

\* A {@link angular.element jQuery or jqLite} wrapper for the browser's `window.document` object.

\*

\* @example

<example>

<file name="index.html">

<div ng-controller="MainCtrl">

<p>$document title: <b ng-bind="title"></b></p>

<p>window.document title: <b ng-bind="windowTitle"></b></p>

</div>

</file>

<file name="script.js">

function MainCtrl($scope, $document) {

$scope.title = $document[0].title;

$scope.windowTitle = angular.element(window.document)[0].title;

}

</file>

</example>

\*/

function $DocumentProvider(){

this.$get = ['$window', function(window){

return jqLite(window.document);

}];

}

/\*\*

\* @ngdoc service

\* @name $exceptionHandler

\* @requires ng.$log

\*

\* @description

\* Any uncaught exception in angular expressions is delegated to this service.

\* The default implementation simply delegates to `$log.error` which logs it into

\* the browser console.

\*

\* In unit tests, if `angular-mocks.js` is loaded, this service is overridden by

\* {@link ngMock.$exceptionHandler mock $exceptionHandler} which aids in testing.

\*

\* ## Example:

\*

\* ```js

\* angular.module('exceptionOverride', []).factory('$exceptionHandler', function () {

\* return function (exception, cause) {

\* exception.message += ' (caused by "' + cause + '")';

\* throw exception;

\* };

\* });

\* ```

\*

\* This example will override the normal action of `$exceptionHandler`, to make angular

\* exceptions fail hard when they happen, instead of just logging to the console.

\*

\* @param {Error} exception Exception associated with the error.

\* @param {string=} cause optional information about the context in which

\* the error was thrown.

\*

\*/

function $ExceptionHandlerProvider() {

this.$get = ['$log', function($log) {

return function(exception, cause) {

$log.error.apply($log, arguments);

};

}];

}

/\*\*

\* Parse headers into key value object

\*

\* @param {string} headers Raw headers as a string

\* @returns {Object} Parsed headers as key value object

\*/

function parseHeaders(headers) {

var parsed = {}, key, val, i;

if (!headers) return parsed;

forEach(headers.split('\n'), function(line) {

i = line.indexOf(':');

key = lowercase(trim(line.substr(0, i)));

val = trim(line.substr(i + 1));

if (key) {

if (parsed[key]) {

parsed[key] += ', ' + val;

} else {

parsed[key] = val;

}

}

});

return parsed;

}

/\*\*

\* Returns a function that provides access to parsed headers.

\*

\* Headers are lazy parsed when first requested.

\* @see parseHeaders

\*

\* @param {(string|Object)} headers Headers to provide access to.

\* @returns {function(string=)} Returns a getter function which if called with:

\*

\* - if called with single an argument returns a single header value or null

\* - if called with no arguments returns an object containing all headers.

\*/

function headersGetter(headers) {

var headersObj = isObject(headers) ? headers : undefined;

return function(name) {

if (!headersObj) headersObj = parseHeaders(headers);

if (name) {

return headersObj[lowercase(name)] || null;

}

return headersObj;

};

}

/\*\*

\* Chain all given functions

\*

\* This function is used for both request and response transforming

\*

\* @param {\*} data Data to transform.

\* @param {function(string=)} headers Http headers getter fn.

\* @param {(Function|Array.<Function>)} fns Function or an array of functions.

\* @returns {\*} Transformed data.

\*/

function transformData(data, headers, fns) {

if (isFunction(fns))

return fns(data, headers);

forEach(fns, function(fn) {

data = fn(data, headers);

});

return data;

}

function isSuccess(status) {

return 200 <= status && status < 300;

}

function $HttpProvider() {

var JSON\_START = /^\s\*(\[|\{[^\{])/,

JSON\_END = /[\}\]]\s\*$/,

PROTECTION\_PREFIX = /^\)\]\}',?\n/,

CONTENT\_TYPE\_APPLICATION\_JSON = {'Content-Type': 'application/json;charset=utf-8'};

var defaults = this.defaults = {

// transform incoming response data

transformResponse: [function(data) {

if (isString(data)) {

// strip json vulnerability protection prefix

data = data.replace(PROTECTION\_PREFIX, '');

if (JSON\_START.test(data) && JSON\_END.test(data))

data = fromJson(data);

}

return data;

}],

// transform outgoing request data

transformRequest: [function(d) {

return isObject(d) && !isFile(d) && !isBlob(d) ? toJson(d) : d;

}],

// default headers

headers: {

common: {

'Accept': 'application/json, text/plain, \*/\*'

},

post: copy(CONTENT\_TYPE\_APPLICATION\_JSON),

put: copy(CONTENT\_TYPE\_APPLICATION\_JSON),

patch: copy(CONTENT\_TYPE\_APPLICATION\_JSON)

},

xsrfCookieName: 'XSRF-TOKEN',

xsrfHeaderName: 'X-XSRF-TOKEN'

};

/\*\*

\* Are ordered by request, i.e. they are applied in the same order as the

\* array, on request, but reverse order, on response.

\*/

var interceptorFactories = this.interceptors = [];

/\*\*

\* For historical reasons, response interceptors are ordered by the order in which

\* they are applied to the response. (This is the opposite of interceptorFactories)

\*/

var responseInterceptorFactories = this.responseInterceptors = [];

this.$get = ['$httpBackend', '$browser', '$cacheFactory', '$rootScope', '$q', '$injector',

function($httpBackend, $browser, $cacheFactory, $rootScope, $q, $injector) {

var defaultCache = $cacheFactory('$http');

/\*\*

\* Interceptors stored in reverse order. Inner interceptors before outer interceptors.

\* The reversal is needed so that we can build up the interception chain around the

\* server request.

\*/

var reversedInterceptors = [];

forEach(interceptorFactories, function(interceptorFactory) {

reversedInterceptors.unshift(isString(interceptorFactory)

? $injector.get(interceptorFactory) : $injector.invoke(interceptorFactory));

});

forEach(responseInterceptorFactories, function(interceptorFactory, index) {

var responseFn = isString(interceptorFactory)

? $injector.get(interceptorFactory)

: $injector.invoke(interceptorFactory);

/\*\*

\* Response interceptors go before "around" interceptors (no real reason, just

\* had to pick one.) But they are already reversed, so we can't use unshift, hence

\* the splice.

\*/

reversedInterceptors.splice(index, 0, {

response: function(response) {

return responseFn($q.when(response));

},

responseError: function(response) {

return responseFn($q.reject(response));

}

});

});

/\*\*

\* @ngdoc service

\* @kind function

\* @name $http

\* @requires ng.$httpBackend

\* @requires $cacheFactory

\* @requires $rootScope

\* @requires $q

\* @requires $injector

\*

\* @description

\* The `$http` service is a core Angular service that facilitates communication with the remote

\* HTTP servers via the browser's [XMLHttpRequest](https://developer.mozilla.org/en/xmlhttprequest)

\* object or via [JSONP](http://en.wikipedia.org/wiki/JSONP).

\*

\* For unit testing applications that use `$http` service, see

\* {@link ngMock.$httpBackend $httpBackend mock}.

\*

\* For a higher level of abstraction, please check out the {@link ngResource.$resource

\* $resource} service.

\*

\* The $http API is based on the {@link ng.$q deferred/promise APIs} exposed by

\* the $q service. While for simple usage patterns this doesn't matter much, for advanced usage

\* it is important to familiarize yourself with these APIs and the guarantees they provide.

\*

\*

\* # General usage

\* The `$http` service is a function which takes a single argument — a configuration object —

\* that is used to generate an HTTP request and returns a {@link ng.$q promise}

\* with two $http specific methods: `success` and `error`.

\*

\* ```js

\* $http({method: 'GET', url: '/someUrl'}).

\* success(function(data, status, headers, config) {

\* // this callback will be called asynchronously

\* // when the response is available

\* }).

\* error(function(data, status, headers, config) {

\* // called asynchronously if an error occurs

\* // or server returns response with an error status.

\* });

\* ```

\*

\* Since the returned value of calling the $http function is a `promise`, you can also use

\* the `then` method to register callbacks, and these callbacks will receive a single argument –

\* an object representing the response. See the API signature and type info below for more

\* details.

\*

\* A response status code between 200 and 299 is considered a success status and

\* will result in the success callback being called. Note that if the response is a redirect,

\* XMLHttpRequest will transparently follow it, meaning that the error callback will not be

\* called for such responses.

\*

\* # Writing Unit Tests that use $http

\* When unit testing (using {@link ngMock ngMock}), it is necessary to call

\* {@link ngMock.$httpBackend#flush $httpBackend.flush()} to flush each pending

\* request using trained responses.

\*

\* ```

\* $httpBackend.expectGET(...);

\* $http.get(...);

\* $httpBackend.flush();

\* ```

\*

\* # Shortcut methods

\*

\* Shortcut methods are also available. All shortcut methods require passing in the URL, and

\* request data must be passed in for POST/PUT requests.

\*

\* ```js

\* $http.get('/someUrl').success(successCallback);

\* $http.post('/someUrl', data).success(successCallback);

\* ```

\*

\* Complete list of shortcut methods:

\*

\* - {@link ng.$http#get $http.get}

\* - {@link ng.$http#head $http.head}

\* - {@link ng.$http#post $http.post}

\* - {@link ng.$http#put $http.put}

\* - {@link ng.$http#delete $http.delete}

\* - {@link ng.$http#jsonp $http.jsonp}

\*

\*

\* # Setting HTTP Headers

\*

\* The $http service will automatically add certain HTTP headers to all requests. These defaults

\* can be fully configured by accessing the `$httpProvider.defaults.headers` configuration

\* object, which currently contains this default configuration:

\*

\* - `$httpProvider.defaults.headers.common` (headers that are common for all requests):

\* - `Accept: application/json, text/plain, \* / \*`

\* - `$httpProvider.defaults.headers.post`: (header defaults for POST requests)

\* - `Content-Type: application/json`

\* - `$httpProvider.defaults.headers.put` (header defaults for PUT requests)

\* - `Content-Type: application/json`

\*

\* To add or overwrite these defaults, simply add or remove a property from these configuration

\* objects. To add headers for an HTTP method other than POST or PUT, simply add a new object

\* with the lowercased HTTP method name as the key, e.g.

\* `$httpProvider.defaults.headers.get = { 'My-Header' : 'value' }.

\*

\* The defaults can also be set at runtime via the `$http.defaults` object in the same

\* fashion. For example:

\*

\* ```

\* module.run(function($http) {

\* $http.defaults.headers.common.Authorization = 'Basic YmVlcDpib29w'

\* });

\* ```

\*

\* In addition, you can supply a `headers` property in the config object passed when

\* calling `$http(config)`, which overrides the defaults without changing them globally.

\*

\*

\* # Transforming Requests and Responses

\*

\* Both requests and responses can be transformed using transform functions. By default, Angular

\* applies these transformations:

\*

\* Request transformations:

\*

\* - If the `data` property of the request configuration object contains an object, serialize it

\* into JSON format.

\*

\* Response transformations:

\*

\* - If XSRF prefix is detected, strip it (see Security Considerations section below).

\* - If JSON response is detected, deserialize it using a JSON parser.

\*

\* To globally augment or override the default transforms, modify the

\* `$httpProvider.defaults.transformRequest` and `$httpProvider.defaults.transformResponse`

\* properties. These properties are by default an array of transform functions, which allows you

\* to `push` or `unshift` a new transformation function into the transformation chain. You can

\* also decide to completely override any default transformations by assigning your

\* transformation functions to these properties directly without the array wrapper. These defaults

\* are again available on the $http factory at run-time, which may be useful if you have run-time

\* services you wish to be involved in your transformations.

\*

\* Similarly, to locally override the request/response transforms, augment the

\* `transformRequest` and/or `transformResponse` properties of the configuration object passed

\* into `$http`.

\*

\*

\* # Caching

\*

\* To enable caching, set the request configuration `cache` property to `true` (to use default

\* cache) or to a custom cache object (built with {@link ng.$cacheFactory `$cacheFactory`}).

\* When the cache is enabled, `$http` stores the response from the server in the specified

\* cache. The next time the same request is made, the response is served from the cache without

\* sending a request to the server.

\*

\* Note that even if the response is served from cache, delivery of the data is asynchronous in

\* the same way that real requests are.

\*

\* If there are multiple GET requests for the same URL that should be cached using the same

\* cache, but the cache is not populated yet, only one request to the server will be made and

\* the remaining requests will be fulfilled using the response from the first request.

\*

\* You can change the default cache to a new object (built with

\* {@link ng.$cacheFactory `$cacheFactory`}) by updating the

\* {@link ng.$http#properties\_defaults `$http.defaults.cache`} property. All requests who set

\* their `cache` property to `true` will now use this cache object.

\*

\* If you set the default cache to `false` then only requests that specify their own custom

\* cache object will be cached.

\*

\* # Interceptors

\*

\* Before you start creating interceptors, be sure to understand the

\* {@link ng.$q $q and deferred/promise APIs}.

\*

\* For purposes of global error handling, authentication, or any kind of synchronous or

\* asynchronous pre-processing of request or postprocessing of responses, it is desirable to be

\* able to intercept requests before they are handed to the server and

\* responses before they are handed over to the application code that

\* initiated these requests. The interceptors leverage the {@link ng.$q

\* promise APIs} to fulfill this need for both synchronous and asynchronous pre-processing.

\*

\* The interceptors are service factories that are registered with the `$httpProvider` by

\* adding them to the `$httpProvider.interceptors` array. The factory is called and

\* injected with dependencies (if specified) and returns the interceptor.

\*

\* There are two kinds of interceptors (and two kinds of rejection interceptors):

\*

\* \* `request`: interceptors get called with http `config` object. The function is free to

\* modify the `config` or create a new one. The function needs to return the `config`

\* directly or as a promise.

\* \* `requestError`: interceptor gets called when a previous interceptor threw an error or

\* resolved with a rejection.

\* \* `response`: interceptors get called with http `response` object. The function is free to

\* modify the `response` or create a new one. The function needs to return the `response`

\* directly or as a promise.

\* \* `responseError`: interceptor gets called when a previous interceptor threw an error or

\* resolved with a rejection.

\*

\*

\* ```js

\* // register the interceptor as a service

\* $provide.factory('myHttpInterceptor', function($q, dependency1, dependency2) {

\* return {

\* // optional method

\* 'request': function(config) {

\* // do something on success

\* return config || $q.when(config);

\* },

\*

\* // optional method

\* 'requestError': function(rejection) {

\* // do something on error

\* if (canRecover(rejection)) {

\* return responseOrNewPromise

\* }

\* return $q.reject(rejection);

\* },

\*

\*

\*

\* // optional method

\* 'response': function(response) {

\* // do something on success

\* return response || $q.when(response);

\* },

\*

\* // optional method

\* 'responseError': function(rejection) {

\* // do something on error

\* if (canRecover(rejection)) {

\* return responseOrNewPromise

\* }

\* return $q.reject(rejection);

\* }

\* };

\* });

\*

\* $httpProvider.interceptors.push('myHttpInterceptor');

\*

\*

\* // alternatively, register the interceptor via an anonymous factory

\* $httpProvider.interceptors.push(function($q, dependency1, dependency2) {

\* return {

\* 'request': function(config) {

\* // same as above

\* },

\*

\* 'response': function(response) {

\* // same as above

\* }

\* };

\* });

\* ```

\*

\* # Response interceptors (DEPRECATED)

\*

\* Before you start creating interceptors, be sure to understand the

\* {@link ng.$q $q and deferred/promise APIs}.

\*

\* For purposes of global error handling, authentication or any kind of synchronous or

\* asynchronous preprocessing of received responses, it is desirable to be able to intercept

\* responses for http requests before they are handed over to the application code that

\* initiated these requests. The response interceptors leverage the {@link ng.$q

\* promise apis} to fulfil this need for both synchronous and asynchronous preprocessing.

\*

\* The interceptors are service factories that are registered with the $httpProvider by

\* adding them to the `$httpProvider.responseInterceptors` array. The factory is called and

\* injected with dependencies (if specified) and returns the interceptor — a function that

\* takes a {@link ng.$q promise} and returns the original or a new promise.

\*

\* ```js

\* // register the interceptor as a service

\* $provide.factory('myHttpInterceptor', function($q, dependency1, dependency2) {

\* return function(promise) {

\* return promise.then(function(response) {

\* // do something on success

\* return response;

\* }, function(response) {

\* // do something on error

\* if (canRecover(response)) {

\* return responseOrNewPromise

\* }

\* return $q.reject(response);

\* });

\* }

\* });

\*

\* $httpProvider.responseInterceptors.push('myHttpInterceptor');

\*

\*

\* // register the interceptor via an anonymous factory

\* $httpProvider.responseInterceptors.push(function($q, dependency1, dependency2) {

\* return function(promise) {

\* // same as above

\* }

\* });

\* ```

\*

\*

\* # Security Considerations

\*

\* When designing web applications, consider security threats from:

\*

\* - [JSON vulnerability](http://haacked.com/archive/2008/11/20/anatomy-of-a-subtle-json-vulnerability.aspx)

\* - [XSRF](http://en.wikipedia.org/wiki/Cross-site\_request\_forgery)

\*

\* Both server and the client must cooperate in order to eliminate these threats. Angular comes

\* pre-configured with strategies that address these issues, but for this to work backend server

\* cooperation is required.

\*

\* ## JSON Vulnerability Protection

\*

\* A [JSON vulnerability](http://haacked.com/archive/2008/11/20/anatomy-of-a-subtle-json-vulnerability.aspx)

\* allows third party website to turn your JSON resource URL into

\* [JSONP](http://en.wikipedia.org/wiki/JSONP) request under some conditions. To

\* counter this your server can prefix all JSON requests with following string `")]}',\n"`.

\* Angular will automatically strip the prefix before processing it as JSON.

\*

\* For example if your server needs to return:

\* ```js

\* ['one','two']

\* ```

\*

\* which is vulnerable to attack, your server can return:

\* ```js

\* )]}',

\* ['one','two']

\* ```

\*

\* Angular will strip the prefix, before processing the JSON.

\*

\*

\* ## Cross Site Request Forgery (XSRF) Protection

\*

\* [XSRF](http://en.wikipedia.org/wiki/Cross-site\_request\_forgery) is a technique by which

\* an unauthorized site can gain your user's private data. Angular provides a mechanism

\* to counter XSRF. When performing XHR requests, the $http service reads a token from a cookie

\* (by default, `XSRF-TOKEN`) and sets it as an HTTP header (`X-XSRF-TOKEN`). Since only

\* JavaScript that runs on your domain could read the cookie, your server can be assured that

\* the XHR came from JavaScript running on your domain. The header will not be set for

\* cross-domain requests.

\*

\* To take advantage of this, your server needs to set a token in a JavaScript readable session

\* cookie called `XSRF-TOKEN` on the first HTTP GET request. On subsequent XHR requests the

\* server can verify that the cookie matches `X-XSRF-TOKEN` HTTP header, and therefore be sure

\* that only JavaScript running on your domain could have sent the request. The token must be

\* unique for each user and must be verifiable by the server (to prevent the JavaScript from

\* making up its own tokens). We recommend that the token is a digest of your site's

\* authentication cookie with a [salt](https://en.wikipedia.org/wiki/Salt\_(cryptography))

\* for added security.

\*

\* The name of the headers can be specified using the xsrfHeaderName and xsrfCookieName

\* properties of either $httpProvider.defaults at config-time, $http.defaults at run-time,

\* or the per-request config object.

\*

\*

\* @param {object} config Object describing the request to be made and how it should be

\* processed. The object has following properties:

\*

\* - \*\*method\*\* – `{string}` – HTTP method (e.g. 'GET', 'POST', etc)

\* - \*\*url\*\* – `{string}` – Absolute or relative URL of the resource that is being requested.

\* - \*\*params\*\* – `{Object.<string|Object>}` – Map of strings or objects which will be turned

\* to `?key1=value1&key2=value2` after the url. If the value is not a string, it will be

\* JSONified.

\* - \*\*data\*\* – `{string|Object}` – Data to be sent as the request message data.

\* - \*\*headers\*\* – `{Object}` – Map of strings or functions which return strings representing

\* HTTP headers to send to the server. If the return value of a function is null, the

\* header will not be sent.

\* - \*\*xsrfHeaderName\*\* – `{string}` – Name of HTTP header to populate with the XSRF token.

\* - \*\*xsrfCookieName\*\* – `{string}` – Name of cookie containing the XSRF token.

\* - \*\*transformRequest\*\* –

\* `{function(data, headersGetter)|Array.<function(data, headersGetter)>}` –

\* transform function or an array of such functions. The transform function takes the http

\* request body and headers and returns its transformed (typically serialized) version.

\* - \*\*transformResponse\*\* –

\* `{function(data, headersGetter)|Array.<function(data, headersGetter)>}` –

\* transform function or an array of such functions. The transform function takes the http

\* response body and headers and returns its transformed (typically deserialized) version.

\* - \*\*cache\*\* – `{boolean|Cache}` – If true, a default $http cache will be used to cache the

\* GET request, otherwise if a cache instance built with

\* {@link ng.$cacheFactory $cacheFactory}, this cache will be used for

\* caching.

\* - \*\*timeout\*\* – `{number|Promise}` – timeout in milliseconds, or {@link ng.$q promise}

\* that should abort the request when resolved.

\* - \*\*withCredentials\*\* - `{boolean}` - whether to to set the `withCredentials` flag on the

\* XHR object. See [requests with credentials]https://developer.mozilla.org/en/http\_access\_control#section\_5

\* for more information.

\* - \*\*responseType\*\* - `{string}` - see

\* [requestType](https://developer.mozilla.org/en-US/docs/DOM/XMLHttpRequest#responseType).

\*

\* @returns {HttpPromise} Returns a {@link ng.$q promise} object with the

\* standard `then` method and two http specific methods: `success` and `error`. The `then`

\* method takes two arguments a success and an error callback which will be called with a

\* response object. The `success` and `error` methods take a single argument - a function that

\* will be called when the request succeeds or fails respectively. The arguments passed into

\* these functions are destructured representation of the response object passed into the

\* `then` method. The response object has these properties:

\*

\* - \*\*data\*\* – `{string|Object}` – The response body transformed with the transform

\* functions.

\* - \*\*status\*\* – `{number}` – HTTP status code of the response.

\* - \*\*headers\*\* – `{function([headerName])}` – Header getter function.

\* - \*\*config\*\* – `{Object}` – The configuration object that was used to generate the request.

\* - \*\*statusText\*\* – `{string}` – HTTP status text of the response.

\*

\* @property {Array.<Object>} pendingRequests Array of config objects for currently pending

\* requests. This is primarily meant to be used for debugging purposes.

\*

\*

\* @example

<example>

<file name="index.html">

<div ng-controller="FetchCtrl">

<select ng-model="method">

<option>GET</option>

<option>JSONP</option>

</select>

<input type="text" ng-model="url" size="80"/>

<button id="fetchbtn" ng-click="fetch()">fetch</button><br>

<button id="samplegetbtn" ng-click="updateModel('GET', 'http-hello.html')">Sample GET</button>

<button id="samplejsonpbtn"

ng-click="updateModel('JSONP',

'http://angularjs.org/greet.php?callback=JSON\_CALLBACK&name=Super%20Hero')">

Sample JSONP

</button>

<button id="invalidjsonpbtn"

ng-click="updateModel('JSONP', 'http://angularjs.org/doesntexist&callback=JSON\_CALLBACK')">

Invalid JSONP

</button>

<pre>http status code: {{status}}</pre>

<pre>http response data: {{data}}</pre>

</div>

</file>

<file name="script.js">

function FetchCtrl($scope, $http, $templateCache) {

$scope.method = 'GET';

$scope.url = 'http-hello.html';

$scope.fetch = function() {

$scope.code = null;

$scope.response = null;

$http({method: $scope.method, url: $scope.url, cache: $templateCache}).

success(function(data, status) {

$scope.status = status;

$scope.data = data;

}).

error(function(data, status) {

$scope.data = data || "Request failed";

$scope.status = status;

});

};

$scope.updateModel = function(method, url) {

$scope.method = method;

$scope.url = url;

};

}

</file>

<file name="http-hello.html">

Hello, $http!

</file>

<file name="protractor.js" type="protractor">

var status = element(by.binding('status'));

var data = element(by.binding('data'));

var fetchBtn = element(by.id('fetchbtn'));

var sampleGetBtn = element(by.id('samplegetbtn'));

var sampleJsonpBtn = element(by.id('samplejsonpbtn'));

var invalidJsonpBtn = element(by.id('invalidjsonpbtn'));

it('should make an xhr GET request', function() {

sampleGetBtn.click();

fetchBtn.click();

expect(status.getText()).toMatch('200');

expect(data.getText()).toMatch(/Hello, \$http!/);

});

it('should make a JSONP request to angularjs.org', function() {

sampleJsonpBtn.click();

fetchBtn.click();

expect(status.getText()).toMatch('200');

expect(data.getText()).toMatch(/Super Hero!/);

});

it('should make JSONP request to invalid URL and invoke the error handler',

function() {

invalidJsonpBtn.click();

fetchBtn.click();

expect(status.getText()).toMatch('0');

expect(data.getText()).toMatch('Request failed');

});

</file>

</example>

\*/

function $http(requestConfig) {

var config = {

method: 'get',

transformRequest: defaults.transformRequest,

transformResponse: defaults.transformResponse

};

var headers = mergeHeaders(requestConfig);

extend(config, requestConfig);

config.headers = headers;

config.method = uppercase(config.method);

var xsrfValue = urlIsSameOrigin(config.url)

? $browser.cookies()[config.xsrfCookieName || defaults.xsrfCookieName]

: undefined;

if (xsrfValue) {

headers[(config.xsrfHeaderName || defaults.xsrfHeaderName)] = xsrfValue;

}

var serverRequest = function(config) {

headers = config.headers;

var reqData = transformData(config.data, headersGetter(headers), config.transformRequest);

// strip content-type if data is undefined

if (isUndefined(config.data)) {

forEach(headers, function(value, header) {

if (lowercase(header) === 'content-type') {

delete headers[header];

}

});

}

if (isUndefined(config.withCredentials) && !isUndefined(defaults.withCredentials)) {

config.withCredentials = defaults.withCredentials;

}

// send request

return sendReq(config, reqData, headers).then(transformResponse, transformResponse);

};

var chain = [serverRequest, undefined];

var promise = $q.when(config);

// apply interceptors

forEach(reversedInterceptors, function(interceptor) {

if (interceptor.request || interceptor.requestError) {

chain.unshift(interceptor.request, interceptor.requestError);

}

if (interceptor.response || interceptor.responseError) {

chain.push(interceptor.response, interceptor.responseError);

}

});

while(chain.length) {

var thenFn = chain.shift();

var rejectFn = chain.shift();

promise = promise.then(thenFn, rejectFn);

}

promise.success = function(fn) {

promise.then(function(response) {

fn(response.data, response.status, response.headers, config);

});

return promise;

};

promise.error = function(fn) {

promise.then(null, function(response) {

fn(response.data, response.status, response.headers, config);

});

return promise;

};

return promise;

function transformResponse(response) {

// make a copy since the response must be cacheable

var resp = extend({}, response, {

data: transformData(response.data, response.headers, config.transformResponse)

});

return (isSuccess(response.status))

? resp

: $q.reject(resp);

}

function mergeHeaders(config) {

var defHeaders = defaults.headers,

reqHeaders = extend({}, config.headers),

defHeaderName, lowercaseDefHeaderName, reqHeaderName;

defHeaders = extend({}, defHeaders.common, defHeaders[lowercase(config.method)]);

// execute if header value is function

execHeaders(defHeaders);

execHeaders(reqHeaders);

// using for-in instead of forEach to avoid unecessary iteration after header has been found

defaultHeadersIteration:

for (defHeaderName in defHeaders) {

lowercaseDefHeaderName = lowercase(defHeaderName);

for (reqHeaderName in reqHeaders) {

if (lowercase(reqHeaderName) === lowercaseDefHeaderName) {

continue defaultHeadersIteration;

}

}

reqHeaders[defHeaderName] = defHeaders[defHeaderName];

}

return reqHeaders;

function execHeaders(headers) {

var headerContent;

forEach(headers, function(headerFn, header) {

if (isFunction(headerFn)) {

headerContent = headerFn();

if (headerContent != null) {

headers[header] = headerContent;

} else {

delete headers[header];

}

}

});

}

}

}

$http.pendingRequests = [];

/\*\*

\* @ngdoc method

\* @name $http#get

\*

\* @description

\* Shortcut method to perform `GET` request.

\*

\* @param {string} url Relative or absolute URL specifying the destination of the request

\* @param {Object=} config Optional configuration object

\* @returns {HttpPromise} Future object

\*/

/\*\*

\* @ngdoc method

\* @name $http#delete

\*

\* @description

\* Shortcut method to perform `DELETE` request.

\*

\* @param {string} url Relative or absolute URL specifying the destination of the request

\* @param {Object=} config Optional configuration object

\* @returns {HttpPromise} Future object

\*/

/\*\*

\* @ngdoc method

\* @name $http#head

\*

\* @description

\* Shortcut method to perform `HEAD` request.

\*

\* @param {string} url Relative or absolute URL specifying the destination of the request

\* @param {Object=} config Optional configuration object

\* @returns {HttpPromise} Future object

\*/

/\*\*

\* @ngdoc method

\* @name $http#jsonp

\*

\* @description

\* Shortcut method to perform `JSONP` request.

\*

\* @param {string} url Relative or absolute URL specifying the destination of the request.

\* Should contain `JSON\_CALLBACK` string.

\* @param {Object=} config Optional configuration object

\* @returns {HttpPromise} Future object

\*/

createShortMethods('get', 'delete', 'head', 'jsonp');

/\*\*

\* @ngdoc method

\* @name $http#post

\*

\* @description

\* Shortcut method to perform `POST` request.

\*

\* @param {string} url Relative or absolute URL specifying the destination of the request

\* @param {\*} data Request content

\* @param {Object=} config Optional configuration object

\* @returns {HttpPromise} Future object

\*/

/\*\*

\* @ngdoc method

\* @name $http#put

\*

\* @description

\* Shortcut method to perform `PUT` request.

\*

\* @param {string} url Relative or absolute URL specifying the destination of the request

\* @param {\*} data Request content

\* @param {Object=} config Optional configuration object

\* @returns {HttpPromise} Future object

\*/

createShortMethodsWithData('post', 'put');

/\*\*

\* @ngdoc property

\* @name $http#defaults

\*

\* @description

\* Runtime equivalent of the `$httpProvider.defaults` property. Allows configuration of

\* default headers, withCredentials as well as request and response transformations.

\*

\* See "Setting HTTP Headers" and "Transforming Requests and Responses" sections above.

\*/

$http.defaults = defaults;

return $http;

function createShortMethods(names) {

forEach(arguments, function(name) {

$http[name] = function(url, config) {

return $http(extend(config || {}, {

method: name,

url: url

}));

};

});

}

function createShortMethodsWithData(name) {

forEach(arguments, function(name) {

$http[name] = function(url, data, config) {

return $http(extend(config || {}, {

method: name,

url: url,

data: data

}));

};

});

}

/\*\*

\* Makes the request.

\*

\* !!! ACCESSES CLOSURE VARS:

\* $httpBackend, defaults, $log, $rootScope, defaultCache, $http.pendingRequests

\*/

function sendReq(config, reqData, reqHeaders) {

var deferred = $q.defer(),

promise = deferred.promise,

cache,

cachedResp,

url = buildUrl(config.url, config.params);

$http.pendingRequests.push(config);

promise.then(removePendingReq, removePendingReq);

if ((config.cache || defaults.cache) && config.cache !== false && config.method == 'GET') {

cache = isObject(config.cache) ? config.cache

: isObject(defaults.cache) ? defaults.cache

: defaultCache;

}

if (cache) {

cachedResp = cache.get(url);

if (isDefined(cachedResp)) {

if (cachedResp.then) {

// cached request has already been sent, but there is no response yet

cachedResp.then(removePendingReq, removePendingReq);

return cachedResp;

} else {

// serving from cache

if (isArray(cachedResp)) {

resolvePromise(cachedResp[1], cachedResp[0], copy(cachedResp[2]), cachedResp[3]);

} else {

resolvePromise(cachedResp, 200, {}, 'OK');

}

}

} else {

// put the promise for the non-transformed response into cache as a placeholder

cache.put(url, promise);

}

}

// if we won't have the response in cache, send the request to the backend

if (isUndefined(cachedResp)) {

$httpBackend(config.method, url, reqData, done, reqHeaders, config.timeout,

config.withCredentials, config.responseType);

}

return promise;

/\*\*

\* Callback registered to $httpBackend():

\* - caches the response if desired

\* - resolves the raw $http promise

\* - calls $apply

\*/

function done(status, response, headersString, statusText) {

if (cache) {

if (isSuccess(status)) {

cache.put(url, [status, response, parseHeaders(headersString), statusText]);

} else {

// remove promise from the cache

cache.remove(url);

}

}

resolvePromise(response, status, headersString, statusText);

if (!$rootScope.$$phase) $rootScope.$apply();

}

/\*\*

\* Resolves the raw $http promise.

\*/

function resolvePromise(response, status, headers, statusText) {

// normalize internal statuses to 0

status = Math.max(status, 0);

(isSuccess(status) ? deferred.resolve : deferred.reject)({

data: response,

status: status,

headers: headersGetter(headers),

config: config,

statusText : statusText

});

}

function removePendingReq() {

var idx = indexOf($http.pendingRequests, config);

if (idx !== -1) $http.pendingRequests.splice(idx, 1);

}

}

function buildUrl(url, params) {

if (!params) return url;

var parts = [];

forEachSorted(params, function(value, key) {

if (value === null || isUndefined(value)) return;

if (!isArray(value)) value = [value];

forEach(value, function(v) {

if (isObject(v)) {

v = toJson(v);

}

parts.push(encodeUriQuery(key) + '=' +

encodeUriQuery(v));

});

});

if(parts.length > 0) {

url += ((url.indexOf('?') == -1) ? '?' : '&') + parts.join('&');

}

return url;

}

}];

}

function createXhr(method) {

//if IE and the method is not RFC2616 compliant, or if XMLHttpRequest

//is not available, try getting an ActiveXObject. Otherwise, use XMLHttpRequest

//if it is available

if (msie <= 8 && (!method.match(/^(get|post|head|put|delete|options)$/i) ||

!window.XMLHttpRequest)) {

return new window.ActiveXObject("Microsoft.XMLHTTP");

} else if (window.XMLHttpRequest) {

return new window.XMLHttpRequest();

}

throw minErr('$httpBackend')('noxhr', "This browser does not support XMLHttpRequest.");

}

/\*\*

\* @ngdoc service

\* @name $httpBackend

\* @requires $window

\* @requires $document

\*

\* @description

\* HTTP backend used by the {@link ng.$http service} that delegates to

\* XMLHttpRequest object or JSONP and deals with browser incompatibilities.

\*

\* You should never need to use this service directly, instead use the higher-level abstractions:

\* {@link ng.$http $http} or {@link ngResource.$resource $resource}.

\*

\* During testing this implementation is swapped with {@link ngMock.$httpBackend mock

\* $httpBackend} which can be trained with responses.

\*/

function $HttpBackendProvider() {

this.$get = ['$browser', '$window', '$document', function($browser, $window, $document) {

return createHttpBackend($browser, createXhr, $browser.defer, $window.angular.callbacks, $document[0]);

}];

}

function createHttpBackend($browser, createXhr, $browserDefer, callbacks, rawDocument) {

var ABORTED = -1;

// TODO(vojta): fix the signature

return function(method, url, post, callback, headers, timeout, withCredentials, responseType) {

var status;

$browser.$$incOutstandingRequestCount();

url = url || $browser.url();

if (lowercase(method) == 'jsonp') {

var callbackId = '\_' + (callbacks.counter++).toString(36);

callbacks[callbackId] = function(data) {

callbacks[callbackId].data = data;

};

var jsonpDone = jsonpReq(url.replace('JSON\_CALLBACK', 'angular.callbacks.' + callbackId),

function() {

if (callbacks[callbackId].data) {

completeRequest(callback, 200, callbacks[callbackId].data);

} else {

completeRequest(callback, status || -2);

}

callbacks[callbackId] = angular.noop;

});

} else {

var xhr = createXhr(method);

xhr.open(method, url, true);

forEach(headers, function(value, key) {

if (isDefined(value)) {

xhr.setRequestHeader(key, value);

}

});

// In IE6 and 7, this might be called synchronously when xhr.send below is called and the

// response is in the cache. the promise api will ensure that to the app code the api is

// always async

xhr.onreadystatechange = function() {

// onreadystatechange might get called multiple times with readyState === 4 on mobile webkit caused by

// xhrs that are resolved while the app is in the background (see #5426).

// since calling completeRequest sets the `xhr` variable to null, we just check if it's not null before

// continuing

//

// we can't set xhr.onreadystatechange to undefined or delete it because that breaks IE8 (method=PATCH) and

// Safari respectively.

if (xhr && xhr.readyState == 4) {

var responseHeaders = null,

response = null;

if(status !== ABORTED) {

responseHeaders = xhr.getAllResponseHeaders();

// responseText is the old-school way of retrieving response (supported by IE8 & 9)

// response/responseType properties were introduced in XHR Level2 spec (supported by IE10)

response = ('response' in xhr) ? xhr.response : xhr.responseText;

}

completeRequest(callback,

status || xhr.status,

response,

responseHeaders,

xhr.statusText || '');

}

};

if (withCredentials) {

xhr.withCredentials = true;

}

if (responseType) {

try {

xhr.responseType = responseType;

} catch (e) {

// WebKit added support for the json responseType value on 09/03/2013

// https://bugs.webkit.org/show\_bug.cgi?id=73648. Versions of Safari prior to 7 are

// known to throw when setting the value "json" as the response type. Other older

// browsers implementing the responseType

//

// The json response type can be ignored if not supported, because JSON payloads are

// parsed on the client-side regardless.

if (responseType !== 'json') {

throw e;

}

}

}

xhr.send(post || null);

}

if (timeout > 0) {

var timeoutId = $browserDefer(timeoutRequest, timeout);

} else if (timeout && timeout.then) {

timeout.then(timeoutRequest);

}

function timeoutRequest() {

status = ABORTED;

jsonpDone && jsonpDone();

xhr && xhr.abort();

}

function completeRequest(callback, status, response, headersString, statusText) {

// cancel timeout and subsequent timeout promise resolution

timeoutId && $browserDefer.cancel(timeoutId);

jsonpDone = xhr = null;

// fix status code when it is 0 (0 status is undocumented).

// Occurs when accessing file resources or on Android 4.1 stock browser

// while retrieving files from application cache.

if (status === 0) {

status = response ? 200 : urlResolve(url).protocol == 'file' ? 404 : 0;

}

// normalize IE bug (http://bugs.jquery.com/ticket/1450)

status = status === 1223 ? 204 : status;

statusText = statusText || '';

callback(status, response, headersString, statusText);

$browser.$$completeOutstandingRequest(noop);

}

};

function jsonpReq(url, done) {

// we can't use jQuery/jqLite here because jQuery does crazy shit with script elements, e.g.:

// - fetches local scripts via XHR and evals them

// - adds and immediately removes script elements from the document

var script = rawDocument.createElement('script'),

doneWrapper = function() {

script.onreadystatechange = script.onload = script.onerror = null;

rawDocument.body.removeChild(script);

if (done) done();

};

script.type = 'text/javascript';

script.src = url;

if (msie && msie <= 8) {

script.onreadystatechange = function() {

if (/loaded|complete/.test(script.readyState)) {

doneWrapper();

}

};

} else {

script.onload = script.onerror = function() {

doneWrapper();

};

}

rawDocument.body.appendChild(script);

return doneWrapper;

}

}

var $interpolateMinErr = minErr('$interpolate');

/\*\*

\* @ngdoc provider

\* @name $interpolateProvider

\* @function

\*

\* @description

\*

\* Used for configuring the interpolation markup. Defaults to `{{` and `}}`.

\*

\* @example

<example module="customInterpolationApp">

<file name="index.html">

<script>

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

customInterpolationApp.config(function($interpolateProvider) {

$interpolateProvider.startSymbol('//');

$interpolateProvider.endSymbol('//');

});

customInterpolationApp.controller('DemoController', function DemoController() {

this.label = "This binding is brought you by // interpolation symbols.";

});

</script>

<div ng-app="App" ng-controller="DemoController as demo">

//demo.label//

</div>

</file>

<file name="protractor.js" type="protractor">

it('should interpolate binding with custom symbols', function() {

expect(element(by.binding('demo.label')).getText()).toBe('This binding is brought you by // interpolation symbols.');

});

</file>

</example>

\*/

function $InterpolateProvider() {

var startSymbol = '{{';

var endSymbol = '}}';

/\*\*

\* @ngdoc method

\* @name $interpolateProvider#startSymbol

\* @description

\* Symbol to denote start of expression in the interpolated string. Defaults to `{{`.

\*

\* @param {string=} value new value to set the starting symbol to.

\* @returns {string|self} Returns the symbol when used as getter and self if used as setter.

\*/

this.startSymbol = function(value){

if (value) {

startSymbol = value;

return this;

} else {

return startSymbol;

}

};

/\*\*

\* @ngdoc method

\* @name $interpolateProvider#endSymbol

\* @description

\* Symbol to denote the end of expression in the interpolated string. Defaults to `}}`.

\*

\* @param {string=} value new value to set the ending symbol to.

\* @returns {string|self} Returns the symbol when used as getter and self if used as setter.

\*/

this.endSymbol = function(value){

if (value) {

endSymbol = value;

return this;

} else {

return endSymbol;

}

};

this.$get = ['$parse', '$exceptionHandler', '$sce', function($parse, $exceptionHandler, $sce) {

var startSymbolLength = startSymbol.length,

endSymbolLength = endSymbol.length;

/\*\*

\* @ngdoc service

\* @name $interpolate

\* @function

\*

\* @requires $parse

\* @requires $sce

\*

\* @description

\*

\* Compiles a string with markup into an interpolation function. This service is used by the

\* HTML {@link ng.$compile $compile} service for data binding. See

\* {@link ng.$interpolateProvider $interpolateProvider} for configuring the

\* interpolation markup.

\*

\*

\* ```js

\* var $interpolate = ...; // injected

\* var exp = $interpolate('Hello {{name | uppercase}}!');

\* expect(exp({name:'Angular'}).toEqual('Hello ANGULAR!');

\* ```

\*

\*

\* @param {string} text The text with markup to interpolate.

\* @param {boolean=} mustHaveExpression if set to true then the interpolation string must have

\* embedded expression in order to return an interpolation function. Strings with no

\* embedded expression will return null for the interpolation function.

\* @param {string=} trustedContext when provided, the returned function passes the interpolated

\* result through {@link ng.$sce#getTrusted $sce.getTrusted(interpolatedResult,

\* trustedContext)} before returning it. Refer to the {@link ng.$sce $sce} service that

\* provides Strict Contextual Escaping for details.

\* @returns {function(context)} an interpolation function which is used to compute the

\* interpolated string. The function has these parameters:

\*

\* \* `context`: an object against which any expressions embedded in the strings are evaluated

\* against.

\*

\*/

function $interpolate(text, mustHaveExpression, trustedContext) {

var startIndex,

endIndex,

index = 0,

parts = [],

length = text.length,

hasInterpolation = false,

fn,

exp,

concat = [];

while(index < length) {

if ( ((startIndex = text.indexOf(startSymbol, index)) != -1) &&

((endIndex = text.indexOf(endSymbol, startIndex + startSymbolLength)) != -1) ) {

(index != startIndex) && parts.push(text.substring(index, startIndex));

parts.push(fn = $parse(exp = text.substring(startIndex + startSymbolLength, endIndex)));

fn.exp = exp;

index = endIndex + endSymbolLength;

hasInterpolation = true;

} else {

// we did not find anything, so we have to add the remainder to the parts array

(index != length) && parts.push(text.substring(index));

index = length;

}

}

if (!(length = parts.length)) {

// we added, nothing, must have been an empty string.

parts.push('');

length = 1;

}

// Concatenating expressions makes it hard to reason about whether some combination of

// concatenated values are unsafe to use and could easily lead to XSS. By requiring that a

// single expression be used for iframe[src], object[src], etc., we ensure that the value

// that's used is assigned or constructed by some JS code somewhere that is more testable or

// make it obvious that you bound the value to some user controlled value. This helps reduce

// the load when auditing for XSS issues.

if (trustedContext && parts.length > 1) {

throw $interpolateMinErr('noconcat',

"Error while interpolating: {0}\nStrict Contextual Escaping disallows " +

"interpolations that concatenate multiple expressions when a trusted value is " +

"required. See http://docs.angularjs.org/api/ng.$sce", text);

}

if (!mustHaveExpression || hasInterpolation) {

concat.length = length;

fn = function(context) {

try {

for(var i = 0, ii = length, part; i<ii; i++) {

if (typeof (part = parts[i]) == 'function') {

part = part(context);

if (trustedContext) {

part = $sce.getTrusted(trustedContext, part);

} else {

part = $sce.valueOf(part);

}

if (part === null || isUndefined(part)) {

part = '';

} else if (typeof part != 'string') {

part = toJson(part);

}

}

concat[i] = part;

}

return concat.join('');

}

catch(err) {

var newErr = $interpolateMinErr('interr', "Can't interpolate: {0}\n{1}", text,

err.toString());

$exceptionHandler(newErr);

}

};

fn.exp = text;

fn.parts = parts;

return fn;

}

}

/\*\*

\* @ngdoc method

\* @name $interpolate#startSymbol

\* @description

\* Symbol to denote the start of expression in the interpolated string. Defaults to `{{`.

\*

\* Use {@link ng.$interpolateProvider#startSymbol $interpolateProvider#startSymbol} to change

\* the symbol.

\*

\* @returns {string} start symbol.

\*/

$interpolate.startSymbol = function() {

return startSymbol;

};

/\*\*

\* @ngdoc method

\* @name $interpolate#endSymbol

\* @description

\* Symbol to denote the end of expression in the interpolated string. Defaults to `}}`.

\*

\* Use {@link ng.$interpolateProvider#endSymbol $interpolateProvider#endSymbol} to change

\* the symbol.

\*

\* @returns {string} end symbol.

\*/

$interpolate.endSymbol = function() {

return endSymbol;

};

return $interpolate;

}];

}

function $IntervalProvider() {

this.$get = ['$rootScope', '$window', '$q',

function($rootScope, $window, $q) {

var intervals = {};

/\*\*

\* @ngdoc service

\* @name $interval

\*

\* @description

\* Angular's wrapper for `window.setInterval`. The `fn` function is executed every `delay`

\* milliseconds.

\*

\* The return value of registering an interval function is a promise. This promise will be

\* notified upon each tick of the interval, and will be resolved after `count` iterations, or

\* run indefinitely if `count` is not defined. The value of the notification will be the

\* number of iterations that have run.

\* To cancel an interval, call `$interval.cancel(promise)`.

\*

\* In tests you can use {@link ngMock.$interval#flush `$interval.flush(millis)`} to

\* move forward by `millis` milliseconds and trigger any functions scheduled to run in that

\* time.

\*

\* <div class="alert alert-warning">

\* \*\*Note\*\*: Intervals created by this service must be explicitly destroyed when you are finished

\* with them. In particular they are not automatically destroyed when a controller's scope or a

\* directive's element are destroyed.

\* You should take this into consideration and make sure to always cancel the interval at the

\* appropriate moment. See the example below for more details on how and when to do this.

\* </div>

\*

\* @param {function()} fn A function that should be called repeatedly.

\* @param {number} delay Number of milliseconds between each function call.

\* @param {number=} [count=0] Number of times to repeat. If not set, or 0, will repeat

\* indefinitely.

\* @param {boolean=} [invokeApply=true] If set to `false` skips model dirty checking, otherwise

\* will invoke `fn` within the {@link ng.$rootScope.Scope#$apply $apply} block.

\* @returns {promise} A promise which will be notified on each iteration.

\*

\* @example

\* <example module="time">

\* <file name="index.html">

\* <script>

\* function Ctrl2($scope,$interval) {

\* $scope.format = 'M/d/yy h:mm:ss a';

\* $scope.blood\_1 = 100;

\* $scope.blood\_2 = 120;

\*

\* var stop;

\* $scope.fight = function() {

\* // Don't start a new fight if we are already fighting

\* if ( angular.isDefined(stop) ) return;

\*

\* stop = $interval(function() {

\* if ($scope.blood\_1 > 0 && $scope.blood\_2 > 0) {

\* $scope.blood\_1 = $scope.blood\_1 - 3;

\* $scope.blood\_2 = $scope.blood\_2 - 4;

\* } else {

\* $scope.stopFight();

\* }

\* }, 100);

\* };

\*

\* $scope.stopFight = function() {

\* if (angular.isDefined(stop)) {

\* $interval.cancel(stop);

\* stop = undefined;

\* }

\* };

\*

\* $scope.resetFight = function() {

\* $scope.blood\_1 = 100;

\* $scope.blood\_2 = 120;

\* }

\*

\* $scope.$on('$destroy', function() {

\* // Make sure that the interval is destroyed too

\* $scope.stopFight();

\* });

\* }

\*

\* angular.module('time', [])

\* // Register the 'myCurrentTime' directive factory method.

\* // We inject $interval and dateFilter service since the factory method is DI.

\* .directive('myCurrentTime', function($interval, dateFilter) {

\* // return the directive link function. (compile function not needed)

\* return function(scope, element, attrs) {

\* var format, // date format

\* stopTime; // so that we can cancel the time updates

\*

\* // used to update the UI

\* function updateTime() {

\* element.text(dateFilter(new Date(), format));

\* }

\*

\* // watch the expression, and update the UI on change.

\* scope.$watch(attrs.myCurrentTime, function(value) {

\* format = value;

\* updateTime();

\* });

\*

\* stopTime = $interval(updateTime, 1000);

\*

\* // listen on DOM destroy (removal) event, and cancel the next UI update

\* // to prevent updating time ofter the DOM element was removed.

\* element.bind('$destroy', function() {

\* $interval.cancel(stopTime);

\* });

\* }

\* });

\* </script>

\*

\* <div>

\* <div ng-controller="Ctrl2">

\* Date format: <input ng-model="format"> <hr/>

\* Current time is: <span my-current-time="format"></span>

\* <hr/>

\* Blood 1 : <font color='red'>{{blood\_1}}</font>

\* Blood 2 : <font color='red'>{{blood\_2}}</font>

\* <button type="button" data-ng-click="fight()">Fight</button>

\* <button type="button" data-ng-click="stopFight()">StopFight</button>

\* <button type="button" data-ng-click="resetFight()">resetFight</button>

\* </div>

\* </div>

\*

\* </file>

\* </example>

\*/

function interval(fn, delay, count, invokeApply) {

var setInterval = $window.setInterval,

clearInterval = $window.clearInterval,

deferred = $q.defer(),

promise = deferred.promise,

iteration = 0,

skipApply = (isDefined(invokeApply) && !invokeApply);

count = isDefined(count) ? count : 0;

promise.then(null, null, fn);

promise.$$intervalId = setInterval(function tick() {

deferred.notify(iteration++);

if (count > 0 && iteration >= count) {

deferred.resolve(iteration);

clearInterval(promise.$$intervalId);

delete intervals[promise.$$intervalId];

}

if (!skipApply) $rootScope.$apply();

}, delay);

intervals[promise.$$intervalId] = deferred;

return promise;

}

/\*\*

\* @ngdoc method

\* @name $interval#cancel

\*

\* @description

\* Cancels a task associated with the `promise`.

\*

\* @param {promise} promise returned by the `$interval` function.

\* @returns {boolean} Returns `true` if the task was successfully canceled.

\*/

interval.cancel = function(promise) {

if (promise && promise.$$intervalId in intervals) {

intervals[promise.$$intervalId].reject('canceled');

clearInterval(promise.$$intervalId);

delete intervals[promise.$$intervalId];

return true;

}

return false;

};

return interval;

}];

}

/\*\*

\* @ngdoc service

\* @name $locale

\*

\* @description

\* $locale service provides localization rules for various Angular components. As of right now the

\* only public api is:

\*

\* \* `id` – `{string}` – locale id formatted as `languageId-countryId` (e.g. `en-us`)

\*/

function $LocaleProvider(){

this.$get = function() {

return {

id: 'en-us',

NUMBER\_FORMATS: {

DECIMAL\_SEP: '.',

GROUP\_SEP: ',',

PATTERNS: [

{ // Decimal Pattern

minInt: 1,

minFrac: 0,

maxFrac: 3,

posPre: '',

posSuf: '',

negPre: '-',

negSuf: '',

gSize: 3,

lgSize: 3

},{ //Currency Pattern

minInt: 1,

minFrac: 2,

maxFrac: 2,

posPre: '\u00A4',

posSuf: '',

negPre: '(\u00A4',

negSuf: ')',

gSize: 3,

lgSize: 3

}

],

CURRENCY\_SYM: '$'

},

DATETIME\_FORMATS: {

MONTH:

'January,February,March,April,May,June,July,August,September,October,November,December'

.split(','),

SHORTMONTH: 'Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec'.split(','),

DAY: 'Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday'.split(','),

SHORTDAY: 'Sun,Mon,Tue,Wed,Thu,Fri,Sat'.split(','),

AMPMS: ['AM','PM'],

medium: 'MMM d, y h:mm:ss a',

short: 'M/d/yy h:mm a',

fullDate: 'EEEE, MMMM d, y',

longDate: 'MMMM d, y',

mediumDate: 'MMM d, y',

shortDate: 'M/d/yy',

mediumTime: 'h:mm:ss a',

shortTime: 'h:mm a'

},

pluralCat: function(num) {

if (num === 1) {

return 'one';

}

return 'other';

}

};

};

}

var PATH\_MATCH = /^([^\?#]\*)(\?([^#]\*))?(#(.\*))?$/,

DEFAULT\_PORTS = {'http': 80, 'https': 443, 'ftp': 21};

var $locationMinErr = minErr('$location');

/\*\*

\* Encode path using encodeUriSegment, ignoring forward slashes

\*

\* @param {string} path Path to encode

\* @returns {string}

\*/

function encodePath(path) {

var segments = path.split('/'),

i = segments.length;

while (i--) {

segments[i] = encodeUriSegment(segments[i]);

}

return segments.join('/');

}

function parseAbsoluteUrl(absoluteUrl, locationObj, appBase) {

var parsedUrl = urlResolve(absoluteUrl, appBase);

locationObj.$$protocol = parsedUrl.protocol;

locationObj.$$host = parsedUrl.hostname;

locationObj.$$port = int(parsedUrl.port) || DEFAULT\_PORTS[parsedUrl.protocol] || null;

}

function parseAppUrl(relativeUrl, locationObj, appBase) {

var prefixed = (relativeUrl.charAt(0) !== '/');

if (prefixed) {

relativeUrl = '/' + relativeUrl;

}

var match = urlResolve(relativeUrl, appBase);

locationObj.$$path = decodeURIComponent(prefixed && match.pathname.charAt(0) === '/' ?

match.pathname.substring(1) : match.pathname);

locationObj.$$search = parseKeyValue(match.search);

locationObj.$$hash = decodeURIComponent(match.hash);

// make sure path starts with '/';

if (locationObj.$$path && locationObj.$$path.charAt(0) != '/') {

locationObj.$$path = '/' + locationObj.$$path;

}

}

/\*\*

\*

\* @param {string} begin

\* @param {string} whole

\* @returns {string} returns text from whole after begin or undefined if it does not begin with

\* expected string.

\*/

function beginsWith(begin, whole) {

if (whole.indexOf(begin) === 0) {

return whole.substr(begin.length);

}

}

function stripHash(url) {

var index = url.indexOf('#');

return index == -1 ? url : url.substr(0, index);

}

function stripFile(url) {

return url.substr(0, stripHash(url).lastIndexOf('/') + 1);

}

/\* return the server only (scheme://host:port) \*/

function serverBase(url) {

return url.substring(0, url.indexOf('/', url.indexOf('//') + 2));

}

/\*\*

\* LocationHtml5Url represents an url

\* This object is exposed as $location service when HTML5 mode is enabled and supported

\*

\* @constructor

\* @param {string} appBase application base URL

\* @param {string} basePrefix url path prefix

\*/

function LocationHtml5Url(appBase, basePrefix) {

this.$$html5 = true;

basePrefix = basePrefix || '';

var appBaseNoFile = stripFile(appBase);

parseAbsoluteUrl(appBase, this, appBase);

/\*\*

\* Parse given html5 (regular) url string into properties

\* @param {string} newAbsoluteUrl HTML5 url

\* @private

\*/

this.$$parse = function(url) {

var pathUrl = beginsWith(appBaseNoFile, url);

if (!isString(pathUrl)) {

throw $locationMinErr('ipthprfx', 'Invalid url "{0}", missing path prefix "{1}".', url,

appBaseNoFile);

}

parseAppUrl(pathUrl, this, appBase);

if (!this.$$path) {

this.$$path = '/';

}

this.$$compose();

};

/\*\*

\* Compose url and update `absUrl` property

\* @private

\*/

this.$$compose = function() {

var search = toKeyValue(this.$$search),

hash = this.$$hash ? '#' + encodeUriSegment(this.$$hash) : '';

this.$$url = encodePath(this.$$path) + (search ? '?' + search : '') + hash;

this.$$absUrl = appBaseNoFile + this.$$url.substr(1); // first char is always '/'

};

this.$$rewrite = function(url) {

var appUrl, prevAppUrl;

if ( (appUrl = beginsWith(appBase, url)) !== undefined ) {

prevAppUrl = appUrl;

if ( (appUrl = beginsWith(basePrefix, appUrl)) !== undefined ) {

return appBaseNoFile + (beginsWith('/', appUrl) || appUrl);

} else {

return appBase + prevAppUrl;

}

} else if ( (appUrl = beginsWith(appBaseNoFile, url)) !== undefined ) {

return appBaseNoFile + appUrl;

} else if (appBaseNoFile == url + '/') {

return appBaseNoFile;

}

};

}

/\*\*

\* LocationHashbangUrl represents url

\* This object is exposed as $location service when developer doesn't opt into html5 mode.

\* It also serves as the base class for html5 mode fallback on legacy browsers.

\*

\* @constructor

\* @param {string} appBase application base URL

\* @param {string} hashPrefix hashbang prefix

\*/

function LocationHashbangUrl(appBase, hashPrefix) {

var appBaseNoFile = stripFile(appBase);

parseAbsoluteUrl(appBase, this, appBase);

/\*\*

\* Parse given hashbang url into properties

\* @param {string} url Hashbang url

\* @private

\*/

this.$$parse = function(url) {

var withoutBaseUrl = beginsWith(appBase, url) || beginsWith(appBaseNoFile, url);

var withoutHashUrl = withoutBaseUrl.charAt(0) == '#'

? beginsWith(hashPrefix, withoutBaseUrl)

: (this.$$html5)

? withoutBaseUrl

: '';

if (!isString(withoutHashUrl)) {

throw $locationMinErr('ihshprfx', 'Invalid url "{0}", missing hash prefix "{1}".', url,

hashPrefix);

}

parseAppUrl(withoutHashUrl, this, appBase);

this.$$path = removeWindowsDriveName(this.$$path, withoutHashUrl, appBase);

this.$$compose();

/\*

\* In Windows, on an anchor node on documents loaded from

\* the filesystem, the browser will return a pathname

\* prefixed with the drive name ('/C:/path') when a

\* pathname without a drive is set:

\* \* a.setAttribute('href', '/foo')

\* \* a.pathname === '/C:/foo' //true

\*

\* Inside of Angular, we're always using pathnames that

\* do not include drive names for routing.

\*/

function removeWindowsDriveName (path, url, base) {

/\*

Matches paths for file protocol on windows,

such as /C:/foo/bar, and captures only /foo/bar.

\*/

var windowsFilePathExp = /^\/?.\*?:(\/.\*)/;

var firstPathSegmentMatch;

//Get the relative path from the input URL.

if (url.indexOf(base) === 0) {

url = url.replace(base, '');

}

/\*

\* The input URL intentionally contains a

\* first path segment that ends with a colon.

\*/

if (windowsFilePathExp.exec(url)) {

return path;

}

firstPathSegmentMatch = windowsFilePathExp.exec(path);

return firstPathSegmentMatch ? firstPathSegmentMatch[1] : path;

}

};

/\*\*

\* Compose hashbang url and update `absUrl` property

\* @private

\*/

this.$$compose = function() {

var search = toKeyValue(this.$$search),

hash = this.$$hash ? '#' + encodeUriSegment(this.$$hash) : '';

this.$$url = encodePath(this.$$path) + (search ? '?' + search : '') + hash;

this.$$absUrl = appBase + (this.$$url ? hashPrefix + this.$$url : '');

};

this.$$rewrite = function(url) {

if(stripHash(appBase) == stripHash(url)) {

return url;

}

};

}

/\*\*

\* LocationHashbangUrl represents url

\* This object is exposed as $location service when html5 history api is enabled but the browser

\* does not support it.

\*

\* @constructor

\* @param {string} appBase application base URL

\* @param {string} hashPrefix hashbang prefix

\*/

function LocationHashbangInHtml5Url(appBase, hashPrefix) {

this.$$html5 = true;

LocationHashbangUrl.apply(this, arguments);

var appBaseNoFile = stripFile(appBase);

this.$$rewrite = function(url) {

var appUrl;

if ( appBase == stripHash(url) ) {

return url;

} else if ( (appUrl = beginsWith(appBaseNoFile, url)) ) {

return appBase + hashPrefix + appUrl;

} else if ( appBaseNoFile === url + '/') {

return appBaseNoFile;

}

};

}

LocationHashbangInHtml5Url.prototype =

LocationHashbangUrl.prototype =

LocationHtml5Url.prototype = {

/\*\*

\* Are we in html5 mode?

\* @private

\*/

$$html5: false,

/\*\*

\* Has any change been replacing ?

\* @private

\*/

$$replace: false,

/\*\*

\* @ngdoc method

\* @name $location#absUrl

\*

\* @description

\* This method is getter only.

\*

\* Return full url representation with all segments encoded according to rules specified in

\* [RFC 3986](http://www.ietf.org/rfc/rfc3986.txt).

\*

\* @return {string} full url

\*/

absUrl: locationGetter('$$absUrl'),

/\*\*

\* @ngdoc method

\* @name $location#url

\*

\* @description

\* This method is getter / setter.

\*

\* Return url (e.g. `/path?a=b#hash`) when called without any parameter.

\*

\* Change path, search and hash, when called with parameter and return `$location`.

\*

\* @param {string=} url New url without base prefix (e.g. `/path?a=b#hash`)

\* @param {string=} replace The path that will be changed

\* @return {string} url

\*/

url: function(url, replace) {

if (isUndefined(url))

return this.$$url;

var match = PATH\_MATCH.exec(url);

if (match[1]) this.path(decodeURIComponent(match[1]));

if (match[2] || match[1]) this.search(match[3] || '');

this.hash(match[5] || '', replace);

return this;

},

/\*\*

\* @ngdoc method

\* @name $location#protocol

\*

\* @description

\* This method is getter only.

\*

\* Return protocol of current url.

\*

\* @return {string} protocol of current url

\*/

protocol: locationGetter('$$protocol'),

/\*\*

\* @ngdoc method

\* @name $location#host

\*

\* @description

\* This method is getter only.

\*

\* Return host of current url.

\*

\* @return {string} host of current url.

\*/

host: locationGetter('$$host'),

/\*\*

\* @ngdoc method

\* @name $location#port

\*

\* @description

\* This method is getter only.

\*

\* Return port of current url.

\*

\* @return {Number} port

\*/

port: locationGetter('$$port'),

/\*\*

\* @ngdoc method

\* @name $location#path

\*

\* @description

\* This method is getter / setter.

\*

\* Return path of current url when called without any parameter.

\*

\* Change path when called with parameter and return `$location`.

\*

\* Note: Path should always begin with forward slash (/), this method will add the forward slash

\* if it is missing.

\*

\* @param {string=} path New path

\* @return {string} path

\*/

path: locationGetterSetter('$$path', function(path) {

return path.charAt(0) == '/' ? path : '/' + path;

}),

/\*\*

\* @ngdoc method

\* @name $location#search

\*

\* @description

\* This method is getter / setter.

\*

\* Return search part (as object) of current url when called without any parameter.

\*

\* Change search part when called with parameter and return `$location`.

\*

\* @param {string|Object.<string>|Object.<Array.<string>>} search New search params - string or

\* hash object. Hash object may contain an array of values, which will be decoded as duplicates in

\* the url.

\*

\* @param {(string|Array<string>)=} paramValue If `search` is a string, then `paramValue` will override only a

\* single search parameter. If `paramValue` is an array, it will set the parameter as a

\* comma-separated value. If `paramValue` is `null`, the parameter will be deleted.

\*

\* @return {string} search

\*/

search: function(search, paramValue) {

switch (arguments.length) {

case 0:

return this.$$search;

case 1:

if (isString(search)) {

this.$$search = parseKeyValue(search);

} else if (isObject(search)) {

this.$$search = search;

} else {

throw $locationMinErr('isrcharg',

'The first argument of the `$location#search()` call must be a string or an object.');

}

break;

default:

if (isUndefined(paramValue) || paramValue === null) {

delete this.$$search[search];

} else {

this.$$search[search] = paramValue;

}

}

this.$$compose();

return this;

},

/\*\*

\* @ngdoc method

\* @name $location#hash

\*

\* @description

\* This method is getter / setter.

\*

\* Return hash fragment when called without any parameter.

\*

\* Change hash fragment when called with parameter and return `$location`.

\*

\* @param {string=} hash New hash fragment

\* @return {string} hash

\*/

hash: locationGetterSetter('$$hash', identity),

/\*\*

\* @ngdoc method

\* @name $location#replace

\*

\* @description

\* If called, all changes to $location during current `$digest` will be replacing current history

\* record, instead of adding new one.

\*/

replace: function() {

this.$$replace = true;

return this;

}

};

function locationGetter(property) {

return function() {

return this[property];

};

}

function locationGetterSetter(property, preprocess) {

return function(value) {

if (isUndefined(value))

return this[property];

this[property] = preprocess(value);

this.$$compose();

return this;

};

}

/\*\*

\* @ngdoc service

\* @name $location

\*

\* @requires $rootElement

\*

\* @description

\* The $location service parses the URL in the browser address bar (based on the

\* [window.location](https://developer.mozilla.org/en/window.location)) and makes the URL

\* available to your application. Changes to the URL in the address bar are reflected into

\* $location service and changes to $location are reflected into the browser address bar.

\*

\* \*\*The $location service:\*\*

\*

\* - Exposes the current URL in the browser address bar, so you can

\* - Watch and observe the URL.

\* - Change the URL.

\* - Synchronizes the URL with the browser when the user

\* - Changes the address bar.

\* - Clicks the back or forward button (or clicks a History link).

\* - Clicks on a link.

\* - Represents the URL object as a set of methods (protocol, host, port, path, search, hash).

\*

\* For more information see {@link guide/$location Developer Guide: Using $location}

\*/

/\*\*

\* @ngdoc provider

\* @name $locationProvider

\* @description

\* Use the `$locationProvider` to configure how the application deep linking paths are stored.

\*/

function $LocationProvider(){

var hashPrefix = '',

html5Mode = false;

/\*\*

\* @ngdoc property

\* @name $locationProvider#hashPrefix

\* @description

\* @param {string=} prefix Prefix for hash part (containing path and search)

\* @returns {\*} current value if used as getter or itself (chaining) if used as setter

\*/

this.hashPrefix = function(prefix) {

if (isDefined(prefix)) {

hashPrefix = prefix;

return this;

} else {

return hashPrefix;

}

};

/\*\*

\* @ngdoc property

\* @name $locationProvider#html5Mode

\* @description

\* @param {boolean=} mode Use HTML5 strategy if available.

\* @returns {\*} current value if used as getter or itself (chaining) if used as setter

\*/

this.html5Mode = function(mode) {

if (isDefined(mode)) {

html5Mode = mode;

return this;

} else {

return html5Mode;

}

};

/\*\*

\* @ngdoc event

\* @name $location#$locationChangeStart

\* @eventType broadcast on root scope

\* @description

\* Broadcasted before a URL will change. This change can be prevented by calling

\* `preventDefault` method of the event. See {@link ng.$rootScope.Scope#$on} for more

\* details about event object. Upon successful change

\* {@link ng.$location#events\_$locationChangeSuccess $locationChangeSuccess} is fired.

\*

\* @param {Object} angularEvent Synthetic event object.

\* @param {string} newUrl New URL

\* @param {string=} oldUrl URL that was before it was changed.

\*/

/\*\*

\* @ngdoc event

\* @name $location#$locationChangeSuccess

\* @eventType broadcast on root scope

\* @description

\* Broadcasted after a URL was changed.

\*

\* @param {Object} angularEvent Synthetic event object.

\* @param {string} newUrl New URL

\* @param {string=} oldUrl URL that was before it was changed.

\*/

this.$get = ['$rootScope', '$browser', '$sniffer', '$rootElement',

function( $rootScope, $browser, $sniffer, $rootElement) {

var $location,

LocationMode,

baseHref = $browser.baseHref(), // if base[href] is undefined, it defaults to ''

initialUrl = $browser.url(),

appBase;

if (html5Mode) {

appBase = serverBase(initialUrl) + (baseHref || '/');

LocationMode = $sniffer.history ? LocationHtml5Url : LocationHashbangInHtml5Url;

} else {

appBase = stripHash(initialUrl);

LocationMode = LocationHashbangUrl;

}

$location = new LocationMode(appBase, '#' + hashPrefix);

$location.$$parse($location.$$rewrite(initialUrl));

$rootElement.on('click', function(event) {

// TODO(vojta): rewrite link when opening in new tab/window (in legacy browser)

// currently we open nice url link and redirect then

if (event.ctrlKey || event.metaKey || event.which == 2) return;

var elm = jqLite(event.target);

// traverse the DOM up to find first A tag

while (lowercase(elm[0].nodeName) !== 'a') {

// ignore rewriting if no A tag (reached root element, or no parent - removed from document)

if (elm[0] === $rootElement[0] || !(elm = elm.parent())[0]) return;

}

var absHref = elm.prop('href');

if (isObject(absHref) && absHref.toString() === '[object SVGAnimatedString]') {

// SVGAnimatedString.animVal should be identical to SVGAnimatedString.baseVal, unless during

// an animation.

absHref = urlResolve(absHref.animVal).href;

}

var rewrittenUrl = $location.$$rewrite(absHref);

if (absHref && !elm.attr('target') && rewrittenUrl && !event.isDefaultPrevented()) {

event.preventDefault();

if (rewrittenUrl != $browser.url()) {

// update location manually

$location.$$parse(rewrittenUrl);

$rootScope.$apply();

// hack to work around FF6 bug 684208 when scenario runner clicks on links

window.angular['ff-684208-preventDefault'] = true;

}

}

});

// rewrite hashbang url <> html5 url

if ($location.absUrl() != initialUrl) {

$browser.url($location.absUrl(), true);

}

// update $location when $browser url changes

$browser.onUrlChange(function(newUrl) {

if ($location.absUrl() != newUrl) {

$rootScope.$evalAsync(function() {

var oldUrl = $location.absUrl();

$location.$$parse(newUrl);

if ($rootScope.$broadcast('$locationChangeStart', newUrl,

oldUrl).defaultPrevented) {

$location.$$parse(oldUrl);

$browser.url(oldUrl);

} else {

afterLocationChange(oldUrl);

}

});

if (!$rootScope.$$phase) $rootScope.$digest();

}

});

// update browser

var changeCounter = 0;

$rootScope.$watch(function $locationWatch() {

var oldUrl = $browser.url();

var currentReplace = $location.$$replace;

if (!changeCounter || oldUrl != $location.absUrl()) {

changeCounter++;

$rootScope.$evalAsync(function() {

if ($rootScope.$broadcast('$locationChangeStart', $location.absUrl(), oldUrl).

defaultPrevented) {

$location.$$parse(oldUrl);

} else {

$browser.url($location.absUrl(), currentReplace);

afterLocationChange(oldUrl);

}

});

}

$location.$$replace = false;

return changeCounter;

});

return $location;

function afterLocationChange(oldUrl) {

$rootScope.$broadcast('$locationChangeSuccess', $location.absUrl(), oldUrl);

}

}];

}

/\*\*

\* @ngdoc service

\* @name $log

\* @requires $window

\*

\* @description

\* Simple service for logging. Default implementation safely writes the message

\* into the browser's console (if present).

\*

\* The main purpose of this service is to simplify debugging and troubleshooting.

\*

\* The default is to log `debug` messages. You can use

\* {@link ng.$logProvider ng.$logProvider#debugEnabled} to change this.

\*

\* @example

<example>

<file name="script.js">

function LogCtrl($scope, $log) {

$scope.$log = $log;

$scope.message = 'Hello World!';

}

</file>

<file name="index.html">

<div ng-controller="LogCtrl">

<p>Reload this page with open console, enter text and hit the log button...</p>

Message:

<input type="text" ng-model="message"/>

<button ng-click="$log.log(message)">log</button>

<button ng-click="$log.warn(message)">warn</button>

<button ng-click="$log.info(message)">info</button>

<button ng-click="$log.error(message)">error</button>

</div>

</file>

</example>

\*/

/\*\*

\* @ngdoc provider

\* @name $logProvider

\* @description

\* Use the `$logProvider` to configure how the application logs messages

\*/

function $LogProvider(){

var debug = true,

self = this;

/\*\*

\* @ngdoc property

\* @name $logProvider#debugEnabled

\* @description

\* @param {boolean=} flag enable or disable debug level messages

\* @returns {\*} current value if used as getter or itself (chaining) if used as setter

\*/

this.debugEnabled = function(flag) {

if (isDefined(flag)) {

debug = flag;

return this;

} else {

return debug;

}

};

this.$get = ['$window', function($window){

return {

/\*\*

\* @ngdoc method

\* @name $log#log

\*

\* @description

\* Write a log message

\*/

log: consoleLog('log'),

/\*\*

\* @ngdoc method

\* @name $log#info

\*

\* @description

\* Write an information message

\*/

info: consoleLog('info'),

/\*\*

\* @ngdoc method

\* @name $log#warn

\*

\* @description

\* Write a warning message

\*/

warn: consoleLog('warn'),

/\*\*

\* @ngdoc method

\* @name $log#error

\*

\* @description

\* Write an error message

\*/

error: consoleLog('error'),

/\*\*

\* @ngdoc method

\* @name $log#debug

\*

\* @description

\* Write a debug message

\*/

debug: (function () {

var fn = consoleLog('debug');

return function() {

if (debug) {

fn.apply(self, arguments);

}

};

}())

};

function formatError(arg) {

if (arg instanceof Error) {

if (arg.stack) {

arg = (arg.message && arg.stack.indexOf(arg.message) === -1)

? 'Error: ' + arg.message + '\n' + arg.stack

: arg.stack;

} else if (arg.sourceURL) {

arg = arg.message + '\n' + arg.sourceURL + ':' + arg.line;

}

}

return arg;

}

function consoleLog(type) {

var console = $window.console || {},

logFn = console[type] || console.log || noop,

hasApply = false;

// Note: reading logFn.apply throws an error in IE11 in IE8 document mode.

// The reason behind this is that console.log has type "object" in IE8...

try {

hasApply = !!logFn.apply;

} catch (e) {}

if (hasApply) {

return function() {

var args = [];

forEach(arguments, function(arg) {

args.push(formatError(arg));

});

return logFn.apply(console, args);

};

}

// we are IE which either doesn't have window.console => this is noop and we do nothing,

// or we are IE where console.log doesn't have apply so we log at least first 2 args

return function(arg1, arg2) {

logFn(arg1, arg2 == null ? '' : arg2);

};

}

}];

}

var $parseMinErr = minErr('$parse');

var promiseWarningCache = {};

var promiseWarning;

// Sandboxing Angular Expressions

// ------------------------------

// Angular expressions are generally considered safe because these expressions only have direct

// access to $scope and locals. However, one can obtain the ability to execute arbitrary JS code by

// obtaining a reference to native JS functions such as the Function constructor.

//

// As an example, consider the following Angular expression:

//

// {}.toString.constructor(alert("evil JS code"))

//

// We want to prevent this type of access. For the sake of performance, during the lexing phase we

// disallow any "dotted" access to any member named "constructor".

//

// For reflective calls (a[b]) we check that the value of the lookup is not the Function constructor

// while evaluating the expression, which is a stronger but more expensive test. Since reflective

// calls are expensive anyway, this is not such a big deal compared to static dereferencing.

//

// This sandboxing technique is not perfect and doesn't aim to be. The goal is to prevent exploits

// against the expression language, but not to prevent exploits that were enabled by exposing

// sensitive JavaScript or browser apis on Scope. Exposing such objects on a Scope is never a good

// practice and therefore we are not even trying to protect against interaction with an object

// explicitly exposed in this way.

//

// A developer could foil the name check by aliasing the Function constructor under a different

// name on the scope.

//

// In general, it is not possible to access a Window object from an angular expression unless a

// window or some DOM object that has a reference to window is published onto a Scope.

function ensureSafeMemberName(name, fullExpression) {

if (name === "constructor") {

throw $parseMinErr('isecfld',

'Referencing "constructor" field in Angular expressions is disallowed! Expression: {0}',

fullExpression);

}

return name;

}

function ensureSafeObject(obj, fullExpression) {

// nifty check if obj is Function that is fast and works across iframes and other contexts

if (obj) {

if (obj.constructor === obj) {

throw $parseMinErr('isecfn',

'Referencing Function in Angular expressions is disallowed! Expression: {0}',

fullExpression);

} else if (// isWindow(obj)

obj.document && obj.location && obj.alert && obj.setInterval) {

throw $parseMinErr('isecwindow',

'Referencing the Window in Angular expressions is disallowed! Expression: {0}',

fullExpression);

} else if (// isElement(obj)

obj.children && (obj.nodeName || (obj.prop && obj.attr && obj.find))) {

throw $parseMinErr('isecdom',

'Referencing DOM nodes in Angular expressions is disallowed! Expression: {0}',

fullExpression);

}

}

return obj;

}

var OPERATORS = {

/\* jshint bitwise : false \*/

'null':function(){return null;},

'true':function(){return true;},

'false':function(){return false;},

undefined:noop,

'+':function(self, locals, a,b){

a=a(self, locals); b=b(self, locals);

if (isDefined(a)) {

if (isDefined(b)) {

return a + b;

}

return a;

}

return isDefined(b)?b:undefined;},

'-':function(self, locals, a,b){

a=a(self, locals); b=b(self, locals);

return (isDefined(a)?a:0)-(isDefined(b)?b:0);

},

'\*':function(self, locals, a,b){return a(self, locals)\*b(self, locals);},

'/':function(self, locals, a,b){return a(self, locals)/b(self, locals);},

'%':function(self, locals, a,b){return a(self, locals)%b(self, locals);},

'^':function(self, locals, a,b){return a(self, locals)^b(self, locals);},

'=':noop,

'===':function(self, locals, a, b){return a(self, locals)===b(self, locals);},

'!==':function(self, locals, a, b){return a(self, locals)!==b(self, locals);},

'==':function(self, locals, a,b){return a(self, locals)==b(self, locals);},

'!=':function(self, locals, a,b){return a(self, locals)!=b(self, locals);},

'<':function(self, locals, a,b){return a(self, locals)<b(self, locals);},

'>':function(self, locals, a,b){return a(self, locals)>b(self, locals);},

'<=':function(self, locals, a,b){return a(self, locals)<=b(self, locals);},

'>=':function(self, locals, a,b){return a(self, locals)>=b(self, locals);},

'&&':function(self, locals, a,b){return a(self, locals)&&b(self, locals);},

'||':function(self, locals, a,b){return a(self, locals)||b(self, locals);},

'&':function(self, locals, a,b){return a(self, locals)&b(self, locals);},

// '|':function(self, locals, a,b){return a|b;},

'|':function(self, locals, a,b){return b(self, locals)(self, locals, a(self, locals));},

'!':function(self, locals, a){return !a(self, locals);}

};

/\* jshint bitwise: true \*/

var ESCAPE = {"n":"\n", "f":"\f", "r":"\r", "t":"\t", "v":"\v", "'":"'", '"':'"'};

/////////////////////////////////////////

/\*\*

\* @constructor

\*/

var Lexer = function (options) {

this.options = options;

};

Lexer.prototype = {

constructor: Lexer,

lex: function (text) {

this.text = text;

this.index = 0;

this.ch = undefined;

this.lastCh = ':'; // can start regexp

this.tokens = [];

var token;

var json = [];

while (this.index < this.text.length) {

this.ch = this.text.charAt(this.index);

if (this.is('"\'')) {

this.readString(this.ch);

} else if (this.isNumber(this.ch) || this.is('.') && this.isNumber(this.peek())) {

this.readNumber();

} else if (this.isIdent(this.ch)) {

this.readIdent();

// identifiers can only be if the preceding char was a { or ,

if (this.was('{,') && json[0] === '{' &&

(token = this.tokens[this.tokens.length - 1])) {

token.json = token.text.indexOf('.') === -1;

}

} else if (this.is('(){}[].,;:?')) {

this.tokens.push({

index: this.index,

text: this.ch,

json: (this.was(':[,') && this.is('{[')) || this.is('}]:,')

});

if (this.is('{[')) json.unshift(this.ch);

if (this.is('}]')) json.shift();

this.index++;

} else if (this.isWhitespace(this.ch)) {

this.index++;

continue;

} else {

var ch2 = this.ch + this.peek();

var ch3 = ch2 + this.peek(2);

var fn = OPERATORS[this.ch];

var fn2 = OPERATORS[ch2];

var fn3 = OPERATORS[ch3];

if (fn3) {

this.tokens.push({index: this.index, text: ch3, fn: fn3});

this.index += 3;

} else if (fn2) {

this.tokens.push({index: this.index, text: ch2, fn: fn2});

this.index += 2;

} else if (fn) {

this.tokens.push({

index: this.index,

text: this.ch,

fn: fn,

json: (this.was('[,:') && this.is('+-'))

});

this.index += 1;

} else {

this.throwError('Unexpected next character ', this.index, this.index + 1);

}

}

this.lastCh = this.ch;

}

return this.tokens;

},

is: function(chars) {

return chars.indexOf(this.ch) !== -1;

},

was: function(chars) {

return chars.indexOf(this.lastCh) !== -1;

},

peek: function(i) {

var num = i || 1;

return (this.index + num < this.text.length) ? this.text.charAt(this.index + num) : false;

},

isNumber: function(ch) {

return ('0' <= ch && ch <= '9');

},

isWhitespace: function(ch) {

// IE treats non-breaking space as \u00A0

return (ch === ' ' || ch === '\r' || ch === '\t' ||

ch === '\n' || ch === '\v' || ch === '\u00A0');

},

isIdent: function(ch) {

return ('a' <= ch && ch <= 'z' ||

'A' <= ch && ch <= 'Z' ||

'\_' === ch || ch === '$');

},

isExpOperator: function(ch) {

return (ch === '-' || ch === '+' || this.isNumber(ch));

},

throwError: function(error, start, end) {

end = end || this.index;

var colStr = (isDefined(start)

? 's ' + start + '-' + this.index + ' [' + this.text.substring(start, end) + ']'

: ' ' + end);

throw $parseMinErr('lexerr', 'Lexer Error: {0} at column{1} in expression [{2}].',

error, colStr, this.text);

},

readNumber: function() {

var number = '';

var start = this.index;

while (this.index < this.text.length) {

var ch = lowercase(this.text.charAt(this.index));

if (ch == '.' || this.isNumber(ch)) {

number += ch;

} else {

var peekCh = this.peek();

if (ch == 'e' && this.isExpOperator(peekCh)) {

number += ch;

} else if (this.isExpOperator(ch) &&

peekCh && this.isNumber(peekCh) &&

number.charAt(number.length - 1) == 'e') {

number += ch;

} else if (this.isExpOperator(ch) &&

(!peekCh || !this.isNumber(peekCh)) &&

number.charAt(number.length - 1) == 'e') {

this.throwError('Invalid exponent');

} else {

break;

}

}

this.index++;

}

number = 1 \* number;

this.tokens.push({

index: start,

text: number,

json: true,

fn: function() { return number; }

});

},

readIdent: function() {

var parser = this;

var ident = '';

var start = this.index;

var lastDot, peekIndex, methodName, ch;

while (this.index < this.text.length) {

ch = this.text.charAt(this.index);

if (ch === '.' || this.isIdent(ch) || this.isNumber(ch)) {

if (ch === '.') lastDot = this.index;

ident += ch;

} else {

break;

}

this.index++;

}

//check if this is not a method invocation and if it is back out to last dot

if (lastDot) {

peekIndex = this.index;

while (peekIndex < this.text.length) {

ch = this.text.charAt(peekIndex);

if (ch === '(') {

methodName = ident.substr(lastDot - start + 1);

ident = ident.substr(0, lastDot - start);

this.index = peekIndex;

break;

}

if (this.isWhitespace(ch)) {

peekIndex++;

} else {

break;

}

}

}

var token = {

index: start,

text: ident

};

// OPERATORS is our own object so we don't need to use special hasOwnPropertyFn

if (OPERATORS.hasOwnProperty(ident)) {

token.fn = OPERATORS[ident];

token.json = OPERATORS[ident];

} else {

var getter = getterFn(ident, this.options, this.text);

token.fn = extend(function(self, locals) {

return (getter(self, locals));

}, {

assign: function(self, value) {

return setter(self, ident, value, parser.text, parser.options);

}

});

}

this.tokens.push(token);

if (methodName) {

this.tokens.push({

index:lastDot,

text: '.',

json: false

});

this.tokens.push({

index: lastDot + 1,

text: methodName,

json: false

});

}

},

readString: function(quote) {

var start = this.index;

this.index++;

var string = '';

var rawString = quote;

var escape = false;

while (this.index < this.text.length) {

var ch = this.text.charAt(this.index);

rawString += ch;

if (escape) {

if (ch === 'u') {

var hex = this.text.substring(this.index + 1, this.index + 5);

if (!hex.match(/[\da-f]{4}/i))

this.throwError('Invalid unicode escape [\\u' + hex + ']');

this.index += 4;

string += String.fromCharCode(parseInt(hex, 16));

} else {

var rep = ESCAPE[ch];

if (rep) {

string += rep;

} else {

string += ch;

}

}

escape = false;

} else if (ch === '\\') {

escape = true;

} else if (ch === quote) {

this.index++;

this.tokens.push({

index: start,

text: rawString,

string: string,

json: true,

fn: function() { return string; }

});

return;

} else {

string += ch;

}

this.index++;

}

this.throwError('Unterminated quote', start);

}

};

/\*\*

\* @constructor

\*/

var Parser = function (lexer, $filter, options) {

this.lexer = lexer;

this.$filter = $filter;

this.options = options;

};

Parser.ZERO = extend(function () {

return 0;

}, {

constant: true

});

Parser.prototype = {

constructor: Parser,

parse: function (text, json) {

this.text = text;

//TODO(i): strip all the obsolte json stuff from this file

this.json = json;

this.tokens = this.lexer.lex(text);

if (json) {

// The extra level of aliasing is here, just in case the lexer misses something, so that

// we prevent any accidental execution in JSON.

this.assignment = this.logicalOR;

this.functionCall =

this.fieldAccess =

this.objectIndex =

this.filterChain = function() {

this.throwError('is not valid json', {text: text, index: 0});

};

}

var value = json ? this.primary() : this.statements();

if (this.tokens.length !== 0) {

this.throwError('is an unexpected token', this.tokens[0]);

}

value.literal = !!value.literal;

value.constant = !!value.constant;

return value;

},

primary: function () {

var primary;

if (this.expect('(')) {

primary = this.filterChain();

this.consume(')');

} else if (this.expect('[')) {

primary = this.arrayDeclaration();

} else if (this.expect('{')) {

primary = this.object();

} else {

var token = this.expect();

primary = token.fn;

if (!primary) {

this.throwError('not a primary expression', token);

}

if (token.json) {

primary.constant = true;

primary.literal = true;

}

}

var next, context;

while ((next = this.expect('(', '[', '.'))) {

if (next.text === '(') {

primary = this.functionCall(primary, context);

context = null;

} else if (next.text === '[') {

context = primary;

primary = this.objectIndex(primary);

} else if (next.text === '.') {

context = primary;

primary = this.fieldAccess(primary);

} else {

this.throwError('IMPOSSIBLE');

}

}

return primary;

},

throwError: function(msg, token) {

throw $parseMinErr('syntax',

'Syntax Error: Token \'{0}\' {1} at column {2} of the expression [{3}] starting at [{4}].',

token.text, msg, (token.index + 1), this.text, this.text.substring(token.index));

},

peekToken: function() {

if (this.tokens.length === 0)

throw $parseMinErr('ueoe', 'Unexpected end of expression: {0}', this.text);

return this.tokens[0];

},

peek: function(e1, e2, e3, e4) {

if (this.tokens.length > 0) {

var token = this.tokens[0];

var t = token.text;

if (t === e1 || t === e2 || t === e3 || t === e4 ||

(!e1 && !e2 && !e3 && !e4)) {

return token;

}

}

return false;

},

expect: function(e1, e2, e3, e4){

var token = this.peek(e1, e2, e3, e4);

if (token) {

if (this.json && !token.json) {

this.throwError('is not valid json', token);

}

this.tokens.shift();

return token;

}

return false;

},

consume: function(e1){

if (!this.expect(e1)) {

this.throwError('is unexpected, expecting [' + e1 + ']', this.peek());

}

},

unaryFn: function(fn, right) {

return extend(function(self, locals) {

return fn(self, locals, right);

}, {

constant:right.constant

});

},

ternaryFn: function(left, middle, right){

return extend(function(self, locals){

return left(self, locals) ? middle(self, locals) : right(self, locals);

}, {

constant: left.constant && middle.constant && right.constant

});

},

binaryFn: function(left, fn, right) {

return extend(function(self, locals) {

return fn(self, locals, left, right);

}, {

constant:left.constant && right.constant

});

},

statements: function() {

var statements = [];

while (true) {

if (this.tokens.length > 0 && !this.peek('}', ')', ';', ']'))

statements.push(this.filterChain());

if (!this.expect(';')) {

// optimize for the common case where there is only one statement.

// TODO(size): maybe we should not support multiple statements?

return (statements.length === 1)

? statements[0]

: function(self, locals) {

var value;

for (var i = 0; i < statements.length; i++) {

var statement = statements[i];

if (statement) {

value = statement(self, locals);

}

}

return value;

};

}

}

},

filterChain: function() {

var left = this.expression();

var token;

while (true) {

if ((token = this.expect('|'))) {

left = this.binaryFn(left, token.fn, this.filter());

} else {

return left;

}

}

},

filter: function() {

var token = this.expect();

var fn = this.$filter(token.text);

var argsFn = [];

while (true) {

if ((token = this.expect(':'))) {

argsFn.push(this.expression());

} else {

var fnInvoke = function(self, locals, input) {

var args = [input];

for (var i = 0; i < argsFn.length; i++) {

args.push(argsFn[i](self, locals));

}

return fn.apply(self, args);

};

return function() {

return fnInvoke;

};

}

}

},

expression: function() {

return this.assignment();

},

assignment: function() {

var left = this.ternary();

var right;

var token;

if ((token = this.expect('='))) {

if (!left.assign) {

this.throwError('implies assignment but [' +

this.text.substring(0, token.index) + '] can not be assigned to', token);

}

right = this.ternary();

return function(scope, locals) {

return left.assign(scope, right(scope, locals), locals);

};

}

return left;

},

ternary: function() {

var left = this.logicalOR();

var middle;

var token;

if ((token = this.expect('?'))) {

middle = this.ternary();

if ((token = this.expect(':'))) {

return this.ternaryFn(left, middle, this.ternary());

} else {

this.throwError('expected :', token);

}

} else {

return left;

}

},

logicalOR: function() {

var left = this.logicalAND();

var token;

while (true) {

if ((token = this.expect('||'))) {

left = this.binaryFn(left, token.fn, this.logicalAND());

} else {

return left;

}

}

},

logicalAND: function() {

var left = this.equality();

var token;

if ((token = this.expect('&&'))) {

left = this.binaryFn(left, token.fn, this.logicalAND());

}

return left;

},

equality: function() {

var left = this.relational();

var token;

if ((token = this.expect('==','!=','===','!=='))) {

left = this.binaryFn(left, token.fn, this.equality());

}

return left;

},

relational: function() {

var left = this.additive();

var token;

if ((token = this.expect('<', '>', '<=', '>='))) {

left = this.binaryFn(left, token.fn, this.relational());

}

return left;

},

additive: function() {

var left = this.multiplicative();

var token;

while ((token = this.expect('+','-'))) {

left = this.binaryFn(left, token.fn, this.multiplicative());

}

return left;

},

multiplicative: function() {

var left = this.unary();

var token;

while ((token = this.expect('\*','/','%'))) {

left = this.binaryFn(left, token.fn, this.unary());

}

return left;

},

unary: function() {

var token;

if (this.expect('+')) {

return this.primary();

} else if ((token = this.expect('-'))) {

return this.binaryFn(Parser.ZERO, token.fn, this.unary());

} else if ((token = this.expect('!'))) {

return this.unaryFn(token.fn, this.unary());

} else {

return this.primary();

}

},

fieldAccess: function(object) {

var parser = this;

var field = this.expect().text;

var getter = getterFn(field, this.options, this.text);

return extend(function(scope, locals, self) {

return getter(self || object(scope, locals));

}, {

assign: function(scope, value, locals) {

return setter(object(scope, locals), field, value, parser.text, parser.options);

}

});

},

objectIndex: function(obj) {

var parser = this;

var indexFn = this.expression();

this.consume(']');

return extend(function(self, locals) {

var o = obj(self, locals),

i = indexFn(self, locals),

v, p;

if (!o) return undefined;

v = ensureSafeObject(o[i], parser.text);

if (v && v.then && parser.options.unwrapPromises) {

p = v;

if (!('$$v' in v)) {

p.$$v = undefined;

p.then(function(val) { p.$$v = val; });

}

v = v.$$v;

}

return v;

}, {

assign: function(self, value, locals) {

var key = indexFn(self, locals);

// prevent overwriting of Function.constructor which would break ensureSafeObject check

var safe = ensureSafeObject(obj(self, locals), parser.text);

return safe[key] = value;

}

});

},

functionCall: function(fn, contextGetter) {

var argsFn = [];

if (this.peekToken().text !== ')') {

do {

argsFn.push(this.expression());

} while (this.expect(','));

}

this.consume(')');

var parser = this;

return function(scope, locals) {

var args = [];

var context = contextGetter ? contextGetter(scope, locals) : scope;

for (var i = 0; i < argsFn.length; i++) {

args.push(argsFn[i](scope, locals));

}

var fnPtr = fn(scope, locals, context) || noop;

ensureSafeObject(context, parser.text);

ensureSafeObject(fnPtr, parser.text);

// IE stupidity! (IE doesn't have apply for some native functions)

var v = fnPtr.apply

? fnPtr.apply(context, args)

: fnPtr(args[0], args[1], args[2], args[3], args[4]);

return ensureSafeObject(v, parser.text);

};

},

// This is used with json array declaration

arrayDeclaration: function () {

var elementFns = [];

var allConstant = true;

if (this.peekToken().text !== ']') {

do {

if (this.peek(']')) {

// Support trailing commas per ES5.1.

break;

}

var elementFn = this.expression();

elementFns.push(elementFn);

if (!elementFn.constant) {

allConstant = false;

}

} while (this.expect(','));

}

this.consume(']');

return extend(function(self, locals) {

var array = [];

for (var i = 0; i < elementFns.length; i++) {

array.push(elementFns[i](self, locals));

}

return array;

}, {

literal: true,

constant: allConstant

});

},

object: function () {

var keyValues = [];

var allConstant = true;

if (this.peekToken().text !== '}') {

do {

if (this.peek('}')) {

// Support trailing commas per ES5.1.

break;

}

var token = this.expect(),

key = token.string || token.text;

this.consume(':');

var value = this.expression();

keyValues.push({key: key, value: value});

if (!value.constant) {

allConstant = false;

}

} while (this.expect(','));

}

this.consume('}');

return extend(function(self, locals) {

var object = {};

for (var i = 0; i < keyValues.length; i++) {

var keyValue = keyValues[i];

object[keyValue.key] = keyValue.value(self, locals);

}

return object;

}, {

literal: true,

constant: allConstant

});

}

};

//////////////////////////////////////////////////

// Parser helper functions

//////////////////////////////////////////////////

function setter(obj, path, setValue, fullExp, options) {

//needed?

options = options || {};

var element = path.split('.'), key;

for (var i = 0; element.length > 1; i++) {

key = ensureSafeMemberName(element.shift(), fullExp);

var propertyObj = obj[key];

if (!propertyObj) {

propertyObj = {};

obj[key] = propertyObj;

}

obj = propertyObj;

if (obj.then && options.unwrapPromises) {

promiseWarning(fullExp);

if (!("$$v" in obj)) {

(function(promise) {

promise.then(function(val) { promise.$$v = val; }); }

)(obj);

}

if (obj.$$v === undefined) {

obj.$$v = {};

}

obj = obj.$$v;

}

}

key = ensureSafeMemberName(element.shift(), fullExp);

obj[key] = setValue;

return setValue;

}

var getterFnCache = {};

/\*\*

\* Implementation of the "Black Hole" variant from:

\* - http://jsperf.com/angularjs-parse-getter/4

\* - http://jsperf.com/path-evaluation-simplified/7

\*/

function cspSafeGetterFn(key0, key1, key2, key3, key4, fullExp, options) {

ensureSafeMemberName(key0, fullExp);

ensureSafeMemberName(key1, fullExp);

ensureSafeMemberName(key2, fullExp);

ensureSafeMemberName(key3, fullExp);

ensureSafeMemberName(key4, fullExp);

return !options.unwrapPromises

? function cspSafeGetter(scope, locals) {

var pathVal = (locals && locals.hasOwnProperty(key0)) ? locals : scope;

if (pathVal == null) return pathVal;

pathVal = pathVal[key0];

if (!key1) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key1];

if (!key2) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key2];

if (!key3) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key3];

if (!key4) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key4];

return pathVal;

}

: function cspSafePromiseEnabledGetter(scope, locals) {

var pathVal = (locals && locals.hasOwnProperty(key0)) ? locals : scope,

promise;

if (pathVal == null) return pathVal;

pathVal = pathVal[key0];

if (pathVal && pathVal.then) {

promiseWarning(fullExp);

if (!("$$v" in pathVal)) {

promise = pathVal;

promise.$$v = undefined;

promise.then(function(val) { promise.$$v = val; });

}

pathVal = pathVal.$$v;

}

if (!key1) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key1];

if (pathVal && pathVal.then) {

promiseWarning(fullExp);

if (!("$$v" in pathVal)) {

promise = pathVal;

promise.$$v = undefined;

promise.then(function(val) { promise.$$v = val; });

}

pathVal = pathVal.$$v;

}

if (!key2) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key2];

if (pathVal && pathVal.then) {

promiseWarning(fullExp);

if (!("$$v" in pathVal)) {

promise = pathVal;

promise.$$v = undefined;

promise.then(function(val) { promise.$$v = val; });

}

pathVal = pathVal.$$v;

}

if (!key3) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key3];

if (pathVal && pathVal.then) {

promiseWarning(fullExp);

if (!("$$v" in pathVal)) {

promise = pathVal;

promise.$$v = undefined;

promise.then(function(val) { promise.$$v = val; });

}

pathVal = pathVal.$$v;

}

if (!key4) return pathVal;

if (pathVal == null) return undefined;

pathVal = pathVal[key4];

if (pathVal && pathVal.then) {

promiseWarning(fullExp);

if (!("$$v" in pathVal)) {

promise = pathVal;

promise.$$v = undefined;

promise.then(function(val) { promise.$$v = val; });

}

pathVal = pathVal.$$v;

}

return pathVal;

};

}

function simpleGetterFn1(key0, fullExp) {

ensureSafeMemberName(key0, fullExp);

return function simpleGetterFn1(scope, locals) {

if (scope == null) return undefined;

return ((locals && locals.hasOwnProperty(key0)) ? locals : scope)[key0];

};

}

function simpleGetterFn2(key0, key1, fullExp) {

ensureSafeMemberName(key0, fullExp);

ensureSafeMemberName(key1, fullExp);

return function simpleGetterFn2(scope, locals) {

if (scope == null) return undefined;

scope = ((locals && locals.hasOwnProperty(key0)) ? locals : scope)[key0];

return scope == null ? undefined : scope[key1];

};

}

function getterFn(path, options, fullExp) {

// Check whether the cache has this getter already.

// We can use hasOwnProperty directly on the cache because we ensure,

// see below, that the cache never stores a path called 'hasOwnProperty'

if (getterFnCache.hasOwnProperty(path)) {

return getterFnCache[path];

}

var pathKeys = path.split('.'),

pathKeysLength = pathKeys.length,

fn;

// When we have only 1 or 2 tokens, use optimized special case closures.

// http://jsperf.com/angularjs-parse-getter/6

if (!options.unwrapPromises && pathKeysLength === 1) {

fn = simpleGetterFn1(pathKeys[0], fullExp);

} else if (!options.unwrapPromises && pathKeysLength === 2) {

fn = simpleGetterFn2(pathKeys[0], pathKeys[1], fullExp);

} else if (options.csp) {

if (pathKeysLength < 6) {

fn = cspSafeGetterFn(pathKeys[0], pathKeys[1], pathKeys[2], pathKeys[3], pathKeys[4], fullExp,

options);

} else {

fn = function(scope, locals) {

var i = 0, val;

do {

val = cspSafeGetterFn(pathKeys[i++], pathKeys[i++], pathKeys[i++], pathKeys[i++],

pathKeys[i++], fullExp, options)(scope, locals);

locals = undefined; // clear after first iteration

scope = val;

} while (i < pathKeysLength);

return val;

};

}

} else {

var code = 'var p;\n';

forEach(pathKeys, function(key, index) {

ensureSafeMemberName(key, fullExp);

code += 'if(s == null) return undefined;\n' +

's='+ (index

// we simply dereference 's' on any .dot notation

? 's'

// but if we are first then we check locals first, and if so read it first

: '((k&&k.hasOwnProperty("' + key + '"))?k:s)') + '["' + key + '"]' + ';\n' +

(options.unwrapPromises

? 'if (s && s.then) {\n' +

' pw("' + fullExp.replace(/(["\r\n])/g, '\\$1') + '");\n' +

' if (!("$$v" in s)) {\n' +

' p=s;\n' +

' p.$$v = undefined;\n' +

' p.then(function(v) {p.$$v=v;});\n' +

'}\n' +

' s=s.$$v\n' +

'}\n'

: '');

});

code += 'return s;';

/\* jshint -W054 \*/

var evaledFnGetter = new Function('s', 'k', 'pw', code); // s=scope, k=locals, pw=promiseWarning

/\* jshint +W054 \*/

evaledFnGetter.toString = valueFn(code);

fn = options.unwrapPromises ? function(scope, locals) {

return evaledFnGetter(scope, locals, promiseWarning);

} : evaledFnGetter;

}

// Only cache the value if it's not going to mess up the cache object

// This is more performant that using Object.prototype.hasOwnProperty.call

if (path !== 'hasOwnProperty') {

getterFnCache[path] = fn;

}

return fn;

}

///////////////////////////////////

/\*\*

\* @ngdoc service

\* @name $parse

\* @kind function

\*

\* @description

\*

\* Converts Angular {@link guide/expression expression} into a function.

\*

\* ```js

\* var getter = $parse('user.name');

\* var setter = getter.assign;

\* var context = {user:{name:'angular'}};

\* var locals = {user:{name:'local'}};

\*

\* expect(getter(context)).toEqual('angular');

\* setter(context, 'newValue');

\* expect(context.user.name).toEqual('newValue');

\* expect(getter(context, locals)).toEqual('local');

\* ```

\*

\*

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*

\* The returned function also has the following properties:

\* \* `literal` – `{boolean}` – whether the expression's top-level node is a JavaScript

\* literal.

\* \* `constant` – `{boolean}` – whether the expression is made entirely of JavaScript

\* constant literals.

\* \* `assign` – `{?function(context, value)}` – if the expression is assignable, this will be

\* set to a function to change its value on the given context.

\*

\*/

/\*\*

\* @ngdoc provider

\* @name $parseProvider

\* @function

\*

\* @description

\* `$parseProvider` can be used for configuring the default behavior of the {@link ng.$parse $parse}

\* service.

\*/

function $ParseProvider() {

var cache = {};

var $parseOptions = {

csp: false,

unwrapPromises: false,

logPromiseWarnings: true

};

/\*\*

\* @deprecated Promise unwrapping via $parse is deprecated and will be removed in the future.

\*

\* @ngdoc method

\* @name $parseProvider#unwrapPromises

\* @description

\*

\* \*\*This feature is deprecated, see deprecation notes below for more info\*\*

\*

\* If set to true (default is false), $parse will unwrap promises automatically when a promise is

\* found at any part of the expression. In other words, if set to true, the expression will always

\* result in a non-promise value.

\*

\* While the promise is unresolved, it's treated as undefined, but once resolved and fulfilled,

\* the fulfillment value is used in place of the promise while evaluating the expression.

\*

\* \*\*Deprecation notice\*\*

\*

\* This is a feature that didn't prove to be wildly useful or popular, primarily because of the

\* dichotomy between data access in templates (accessed as raw values) and controller code

\* (accessed as promises).

\*

\* In most code we ended up resolving promises manually in controllers anyway and thus unifying

\* the model access there.

\*

\* Other downsides of automatic promise unwrapping:

\*

\* - when building components it's often desirable to receive the raw promises

\* - adds complexity and slows down expression evaluation

\* - makes expression code pre-generation unattractive due to the amount of code that needs to be

\* generated

\* - makes IDE auto-completion and tool support hard

\*

\* \*\*Warning Logs\*\*

\*

\* If the unwrapping is enabled, Angular will log a warning about each expression that unwraps a

\* promise (to reduce the noise, each expression is logged only once). To disable this logging use

\* `$parseProvider.logPromiseWarnings(false)` api.

\*

\*

\* @param {boolean=} value New value.

\* @returns {boolean|self} Returns the current setting when used as getter and self if used as

\* setter.

\*/

this.unwrapPromises = function(value) {

if (isDefined(value)) {

$parseOptions.unwrapPromises = !!value;

return this;

} else {

return $parseOptions.unwrapPromises;

}

};

/\*\*

\* @deprecated Promise unwrapping via $parse is deprecated and will be removed in the future.

\*

\* @ngdoc method

\* @name $parseProvider#logPromiseWarnings

\* @description

\*

\* Controls whether Angular should log a warning on any encounter of a promise in an expression.

\*

\* The default is set to `true`.

\*

\* This setting applies only if `$parseProvider.unwrapPromises` setting is set to true as well.

\*

\* @param {boolean=} value New value.

\* @returns {boolean|self} Returns the current setting when used as getter and self if used as

\* setter.

\*/

this.logPromiseWarnings = function(value) {

if (isDefined(value)) {

$parseOptions.logPromiseWarnings = value;

return this;

} else {

return $parseOptions.logPromiseWarnings;

}

};

this.$get = ['$filter', '$sniffer', '$log', function($filter, $sniffer, $log) {

$parseOptions.csp = $sniffer.csp;

promiseWarning = function promiseWarningFn(fullExp) {

if (!$parseOptions.logPromiseWarnings || promiseWarningCache.hasOwnProperty(fullExp)) return;

promiseWarningCache[fullExp] = true;

$log.warn('[$parse] Promise found in the expression `' + fullExp + '`. ' +

'Automatic unwrapping of promises in Angular expressions is deprecated.');

};

return function(exp) {

var parsedExpression;

switch (typeof exp) {

case 'string':

if (cache.hasOwnProperty(exp)) {

return cache[exp];

}

var lexer = new Lexer($parseOptions);

var parser = new Parser(lexer, $filter, $parseOptions);

parsedExpression = parser.parse(exp, false);

if (exp !== 'hasOwnProperty') {

// Only cache the value if it's not going to mess up the cache object

// This is more performant that using Object.prototype.hasOwnProperty.call

cache[exp] = parsedExpression;

}

return parsedExpression;

case 'function':

return exp;

default:

return noop;

}

};

}];

}

/\*\*

\* @ngdoc service

\* @name $q

\* @requires $rootScope

\*

\* @description

\* A promise/deferred implementation inspired by [Kris Kowal's Q](https://github.com/kriskowal/q).

\*

\* [The CommonJS Promise proposal](http://wiki.commonjs.org/wiki/Promises) describes a promise as an

\* interface for interacting with an object that represents the result of an action that is

\* performed asynchronously, and may or may not be finished at any given point in time.

\*

\* From the perspective of dealing with error handling, deferred and promise APIs are to

\* asynchronous programming what `try`, `catch` and `throw` keywords are to synchronous programming.

\*

\* ```js

\* // for the purpose of this example let's assume that variables `$q`, `scope` and `okToGreet`

\* // are available in the current lexical scope (they could have been injected or passed in).

\*

\* function asyncGreet(name) {

\* var deferred = $q.defer();

\*

\* setTimeout(function() {

\* // since this fn executes async in a future turn of the event loop, we need to wrap

\* // our code into an $apply call so that the model changes are properly observed.

\* scope.$apply(function() {

\* deferred.notify('About to greet ' + name + '.');

\*

\* if (okToGreet(name)) {

\* deferred.resolve('Hello, ' + name + '!');

\* } else {

\* deferred.reject('Greeting ' + name + ' is not allowed.');

\* }

\* });

\* }, 1000);

\*

\* return deferred.promise;

\* }

\*

\* var promise = asyncGreet('Robin Hood');

\* promise.then(function(greeting) {

\* alert('Success: ' + greeting);

\* }, function(reason) {

\* alert('Failed: ' + reason);

\* }, function(update) {

\* alert('Got notification: ' + update);

\* });

\* ```

\*

\* At first it might not be obvious why this extra complexity is worth the trouble. The payoff

\* comes in the way of guarantees that promise and deferred APIs make, see

\* https://github.com/kriskowal/uncommonjs/blob/master/promises/specification.md.

\*

\* Additionally the promise api allows for composition that is very hard to do with the

\* traditional callback ([CPS](http://en.wikipedia.org/wiki/Continuation-passing\_style)) approach.

\* For more on this please see the [Q documentation](https://github.com/kriskowal/q) especially the

\* section on serial or parallel joining of promises.

\*

\*

\* # The Deferred API

\*

\* A new instance of deferred is constructed by calling `$q.defer()`.

\*

\* The purpose of the deferred object is to expose the associated Promise instance as well as APIs

\* that can be used for signaling the successful or unsuccessful completion, as well as the status

\* of the task.

\*

\* \*\*Methods\*\*

\*

\* - `resolve(value)` – resolves the derived promise with the `value`. If the value is a rejection

\* constructed via `$q.reject`, the promise will be rejected instead.

\* - `reject(reason)` – rejects the derived promise with the `reason`. This is equivalent to

\* resolving it with a rejection constructed via `$q.reject`.

\* - `notify(value)` - provides updates on the status of the promise's execution. This may be called

\* multiple times before the promise is either resolved or rejected.

\*

\* \*\*Properties\*\*

\*

\* - promise – `{Promise}` – promise object associated with this deferred.

\*

\*

\* # The Promise API

\*

\* A new promise instance is created when a deferred instance is created and can be retrieved by

\* calling `deferred.promise`.

\*

\* The purpose of the promise object is to allow for interested parties to get access to the result

\* of the deferred task when it completes.

\*

\* \*\*Methods\*\*

\*

\* - `then(successCallback, errorCallback, notifyCallback)` – regardless of when the promise was or

\* will be resolved or rejected, `then` calls one of the success or error callbacks asynchronously

\* as soon as the result is available. The callbacks are called with a single argument: the result

\* or rejection reason. Additionally, the notify callback may be called zero or more times to

\* provide a progress indication, before the promise is resolved or rejected.

\*

\* This method \*returns a new promise\* which is resolved or rejected via the return value of the

\* `successCallback`, `errorCallback`. It also notifies via the return value of the

\* `notifyCallback` method. The promise can not be resolved or rejected from the notifyCallback

\* method.

\*

\* - `catch(errorCallback)` – shorthand for `promise.then(null, errorCallback)`

\*

\* - `finally(callback)` – allows you to observe either the fulfillment or rejection of a promise,

\* but to do so without modifying the final value. This is useful to release resources or do some

\* clean-up that needs to be done whether the promise was rejected or resolved. See the [full

\* specification](https://github.com/kriskowal/q/wiki/API-Reference#promisefinallycallback) for

\* more information.

\*

\* Because `finally` is a reserved word in JavaScript and reserved keywords are not supported as

\* property names by ES3, you'll need to invoke the method like `promise['finally'](callback)` to

\* make your code IE8 and Android 2.x compatible.

\*

\* # Chaining promises

\*

\* Because calling the `then` method of a promise returns a new derived promise, it is easily

\* possible to create a chain of promises:

\*

\* ```js

\* promiseB = promiseA.then(function(result) {

\* return result + 1;

\* });

\*

\* // promiseB will be resolved immediately after promiseA is resolved and its value

\* // will be the result of promiseA incremented by 1

\* ```

\*

\* It is possible to create chains of any length and since a promise can be resolved with another

\* promise (which will defer its resolution further), it is possible to pause/defer resolution of

\* the promises at any point in the chain. This makes it possible to implement powerful APIs like

\* $http's response interceptors.

\*

\*

\* # Differences between Kris Kowal's Q and $q

\*

\* There are two main differences:

\*

\* - $q is integrated with the {@link ng.$rootScope.Scope} Scope model observation

\* mechanism in angular, which means faster propagation of resolution or rejection into your

\* models and avoiding unnecessary browser repaints, which would result in flickering UI.

\* - Q has many more features than $q, but that comes at a cost of bytes. $q is tiny, but contains

\* all the important functionality needed for common async tasks.

\*

\* # Testing

\*

\* ```js

\* it('should simulate promise', inject(function($q, $rootScope) {

\* var deferred = $q.defer();

\* var promise = deferred.promise;

\* var resolvedValue;

\*

\* promise.then(function(value) { resolvedValue = value; });

\* expect(resolvedValue).toBeUndefined();

\*

\* // Simulate resolving of promise

\* deferred.resolve(123);

\* // Note that the 'then' function does not get called synchronously.

\* // This is because we want the promise API to always be async, whether or not

\* // it got called synchronously or asynchronously.

\* expect(resolvedValue).toBeUndefined();

\*

\* // Propagate promise resolution to 'then' functions using $apply().

\* $rootScope.$apply();

\* expect(resolvedValue).toEqual(123);

\* }));

\* ```

\*/

function $QProvider() {

this.$get = ['$rootScope', '$exceptionHandler', function($rootScope, $exceptionHandler) {

return qFactory(function(callback) {

$rootScope.$evalAsync(callback);

}, $exceptionHandler);

}];

}

/\*\*

\* Constructs a promise manager.

\*

\* @param {function(Function)} nextTick Function for executing functions in the next turn.

\* @param {function(...\*)} exceptionHandler Function into which unexpected exceptions are passed for

\* debugging purposes.

\* @returns {object} Promise manager.

\*/

function qFactory(nextTick, exceptionHandler) {

/\*\*

\* @ngdoc method

\* @name $q#defer

\* @function

\*

\* @description

\* Creates a `Deferred` object which represents a task which will finish in the future.

\*

\* @returns {Deferred} Returns a new instance of deferred.

\*/

var defer = function() {

var pending = [],

value, deferred;

deferred = {

resolve: function(val) {

if (pending) {

var callbacks = pending;

pending = undefined;

value = ref(val);

if (callbacks.length) {

nextTick(function() {

var callback;

for (var i = 0, ii = callbacks.length; i < ii; i++) {

callback = callbacks[i];

value.then(callback[0], callback[1], callback[2]);

}

});

}

}

},

reject: function(reason) {

deferred.resolve(createInternalRejectedPromise(reason));

},

notify: function(progress) {

if (pending) {

var callbacks = pending;

if (pending.length) {

nextTick(function() {

var callback;

for (var i = 0, ii = callbacks.length; i < ii; i++) {

callback = callbacks[i];

callback[2](progress);

}

});

}

}

},

promise: {

then: function(callback, errback, progressback) {

var result = defer();

var wrappedCallback = function(value) {

try {

result.resolve((isFunction(callback) ? callback : defaultCallback)(value));

} catch(e) {

result.reject(e);

exceptionHandler(e);

}

};

var wrappedErrback = function(reason) {

try {

result.resolve((isFunction(errback) ? errback : defaultErrback)(reason));

} catch(e) {

result.reject(e);

exceptionHandler(e);

}

};

var wrappedProgressback = function(progress) {

try {

result.notify((isFunction(progressback) ? progressback : defaultCallback)(progress));

} catch(e) {

exceptionHandler(e);

}

};

if (pending) {

pending.push([wrappedCallback, wrappedErrback, wrappedProgressback]);

} else {

value.then(wrappedCallback, wrappedErrback, wrappedProgressback);

}

return result.promise;

},

"catch": function(callback) {

return this.then(null, callback);

},

"finally": function(callback) {

function makePromise(value, resolved) {

var result = defer();

if (resolved) {

result.resolve(value);

} else {

result.reject(value);

}

return result.promise;

}

function handleCallback(value, isResolved) {

var callbackOutput = null;

try {

callbackOutput = (callback ||defaultCallback)();

} catch(e) {

return makePromise(e, false);

}

if (callbackOutput && isFunction(callbackOutput.then)) {

return callbackOutput.then(function() {

return makePromise(value, isResolved);

}, function(error) {

return makePromise(error, false);

});

} else {

return makePromise(value, isResolved);

}

}

return this.then(function(value) {

return handleCallback(value, true);

}, function(error) {

return handleCallback(error, false);

});

}

}

};

return deferred;

};

var ref = function(value) {

if (value && isFunction(value.then)) return value;

return {

then: function(callback) {

var result = defer();

nextTick(function() {

result.resolve(callback(value));

});

return result.promise;

}

};

};

/\*\*

\* @ngdoc method

\* @name $q#reject

\* @function

\*

\* @description

\* Creates a promise that is resolved as rejected with the specified `reason`. This api should be

\* used to forward rejection in a chain of promises. If you are dealing with the last promise in

\* a promise chain, you don't need to worry about it.

\*

\* When comparing deferreds/promises to the familiar behavior of try/catch/throw, think of

\* `reject` as the `throw` keyword in JavaScript. This also means that if you "catch" an error via

\* a promise error callback and you want to forward the error to the promise derived from the

\* current promise, you have to "rethrow" the error by returning a rejection constructed via

\* `reject`.

\*

\* ```js

\* promiseB = promiseA.then(function(result) {

\* // success: do something and resolve promiseB

\* // with the old or a new result

\* return result;

\* }, function(reason) {

\* // error: handle the error if possible and

\* // resolve promiseB with newPromiseOrValue,

\* // otherwise forward the rejection to promiseB

\* if (canHandle(reason)) {

\* // handle the error and recover

\* return newPromiseOrValue;

\* }

\* return $q.reject(reason);

\* });

\* ```

\*

\* @param {\*} reason Constant, message, exception or an object representing the rejection reason.

\* @returns {Promise} Returns a promise that was already resolved as rejected with the `reason`.

\*/

var reject = function(reason) {

var result = defer();

result.reject(reason);

return result.promise;

};

var createInternalRejectedPromise = function(reason) {

return {

then: function(callback, errback) {

var result = defer();

nextTick(function() {

try {

result.resolve((isFunction(errback) ? errback : defaultErrback)(reason));

} catch(e) {

result.reject(e);

exceptionHandler(e);

}

});

return result.promise;

}

};

};

/\*\*

\* @ngdoc method

\* @name $q#when

\* @function

\*

\* @description

\* Wraps an object that might be a value or a (3rd party) then-able promise into a $q promise.

\* This is useful when you are dealing with an object that might or might not be a promise, or if

\* the promise comes from a source that can't be trusted.

\*

\* @param {\*} value Value or a promise

\* @returns {Promise} Returns a promise of the passed value or promise

\*/

var when = function(value, callback, errback, progressback) {

var result = defer(),

done;

var wrappedCallback = function(value) {

try {

return (isFunction(callback) ? callback : defaultCallback)(value);

} catch (e) {

exceptionHandler(e);

return reject(e);

}

};

var wrappedErrback = function(reason) {

try {

return (isFunction(errback) ? errback : defaultErrback)(reason);

} catch (e) {

exceptionHandler(e);

return reject(e);

}

};

var wrappedProgressback = function(progress) {

try {

return (isFunction(progressback) ? progressback : defaultCallback)(progress);

} catch (e) {

exceptionHandler(e);

}

};

nextTick(function() {

ref(value).then(function(value) {

if (done) return;

done = true;

result.resolve(ref(value).then(wrappedCallback, wrappedErrback, wrappedProgressback));

}, function(reason) {

if (done) return;

done = true;

result.resolve(wrappedErrback(reason));

}, function(progress) {

if (done) return;

result.notify(wrappedProgressback(progress));

});

});

return result.promise;

};

function defaultCallback(value) {

return value;

}

function defaultErrback(reason) {

return reject(reason);

}

/\*\*

\* @ngdoc method

\* @name $q#all

\* @function

\*

\* @description

\* Combines multiple promises into a single promise that is resolved when all of the input

\* promises are resolved.

\*

\* @param {Array.<Promise>|Object.<Promise>} promises An array or hash of promises.

\* @returns {Promise} Returns a single promise that will be resolved with an array/hash of values,

\* each value corresponding to the promise at the same index/key in the `promises` array/hash.

\* If any of the promises is resolved with a rejection, this resulting promise will be rejected

\* with the same rejection value.

\*/

function all(promises) {

var deferred = defer(),

counter = 0,

results = isArray(promises) ? [] : {};

forEach(promises, function(promise, key) {

counter++;

ref(promise).then(function(value) {

if (results.hasOwnProperty(key)) return;

results[key] = value;

if (!(--counter)) deferred.resolve(results);

}, function(reason) {

if (results.hasOwnProperty(key)) return;

deferred.reject(reason);

});

});

if (counter === 0) {

deferred.resolve(results);

}

return deferred.promise;

}

return {

defer: defer,

reject: reject,

when: when,

all: all

};

}

function $$RAFProvider(){ //rAF

this.$get = ['$window', '$timeout', function($window, $timeout) {

var requestAnimationFrame = $window.requestAnimationFrame ||

$window.webkitRequestAnimationFrame ||

$window.mozRequestAnimationFrame;

var cancelAnimationFrame = $window.cancelAnimationFrame ||

$window.webkitCancelAnimationFrame ||

$window.mozCancelAnimationFrame ||

$window.webkitCancelRequestAnimationFrame;

var rafSupported = !!requestAnimationFrame;

var raf = rafSupported

? function(fn) {

var id = requestAnimationFrame(fn);

return function() {

cancelAnimationFrame(id);

};

}

: function(fn) {

var timer = $timeout(fn, 16.66, false); // 1000 / 60 = 16.666

return function() {

$timeout.cancel(timer);

};

};

raf.supported = rafSupported;

return raf;

}];

}

/\*\*

\* DESIGN NOTES

\*

\* The design decisions behind the scope are heavily favored for speed and memory consumption.

\*

\* The typical use of scope is to watch the expressions, which most of the time return the same

\* value as last time so we optimize the operation.

\*

\* Closures construction is expensive in terms of speed as well as memory:

\* - No closures, instead use prototypical inheritance for API

\* - Internal state needs to be stored on scope directly, which means that private state is

\* exposed as $$\_\_\_\_ properties

\*

\* Loop operations are optimized by using while(count--) { ... }

\* - this means that in order to keep the same order of execution as addition we have to add

\* items to the array at the beginning (shift) instead of at the end (push)

\*

\* Child scopes are created and removed often

\* - Using an array would be slow since inserts in middle are expensive so we use linked list

\*

\* There are few watches then a lot of observers. This is why you don't want the observer to be

\* implemented in the same way as watch. Watch requires return of initialization function which

\* are expensive to construct.

\*/

/\*\*

\* @ngdoc provider

\* @name $rootScopeProvider

\* @description

\*

\* Provider for the $rootScope service.

\*/

/\*\*

\* @ngdoc method

\* @name $rootScopeProvider#digestTtl

\* @description

\*

\* Sets the number of `$digest` iterations the scope should attempt to execute before giving up and

\* assuming that the model is unstable.

\*

\* The current default is 10 iterations.

\*

\* In complex applications it's possible that the dependencies between `$watch`s will result in

\* several digest iterations. However if an application needs more than the default 10 digest

\* iterations for its model to stabilize then you should investigate what is causing the model to

\* continuously change during the digest.

\*

\* Increasing the TTL could have performance implications, so you should not change it without

\* proper justification.

\*

\* @param {number} limit The number of digest iterations.

\*/

/\*\*

\* @ngdoc service

\* @name $rootScope

\* @description

\*

\* Every application has a single root {@link ng.$rootScope.Scope scope}.

\* All other scopes are descendant scopes of the root scope. Scopes provide separation

\* between the model and the view, via a mechanism for watching the model for changes.

\* They also provide an event emission/broadcast and subscription facility. See the

\* {@link guide/scope developer guide on scopes}.

\*/

function $RootScopeProvider(){

var TTL = 10;

var $rootScopeMinErr = minErr('$rootScope');

var lastDirtyWatch = null;

this.digestTtl = function(value) {

if (arguments.length) {

TTL = value;

}

return TTL;

};

this.$get = ['$injector', '$exceptionHandler', '$parse', '$browser',

function( $injector, $exceptionHandler, $parse, $browser) {

/\*\*

\* @ngdoc type

\* @name $rootScope.Scope

\*

\* @description

\* A root scope can be retrieved using the {@link ng.$rootScope $rootScope} key from the

\* {@link auto.$injector $injector}. Child scopes are created using the

\* {@link ng.$rootScope.Scope#$new $new()} method. (Most scopes are created automatically when

\* compiled HTML template is executed.)

\*

\* Here is a simple scope snippet to show how you can interact with the scope.

\* ```html

\* <file src="./test/ng/rootScopeSpec.js" tag="docs1" />

\* ```

\*

\* # Inheritance

\* A scope can inherit from a parent scope, as in this example:

\* ```js

var parent = $rootScope;

var child = parent.$new();

parent.salutation = "Hello";

child.name = "World";

expect(child.salutation).toEqual('Hello');

child.salutation = "Welcome";

expect(child.salutation).toEqual('Welcome');

expect(parent.salutation).toEqual('Hello');

\* ```

\*

\*

\* @param {Object.<string, function()>=} providers Map of service factory which need to be

\* provided for the current scope. Defaults to {@link ng}.

\* @param {Object.<string, \*>=} instanceCache Provides pre-instantiated services which should

\* append/override services provided by `providers`. This is handy

\* when unit-testing and having the need to override a default

\* service.

\* @returns {Object} Newly created scope.

\*

\*/

function Scope() {

this.$id = nextUid();

this.$$phase = this.$parent = this.$$watchers =

this.$$nextSibling = this.$$prevSibling =

this.$$childHead = this.$$childTail = null;

this['this'] = this.$root = this;

this.$$destroyed = false;

this.$$asyncQueue = [];

this.$$postDigestQueue = [];

this.$$listeners = {};

this.$$listenerCount = {};

this.$$isolateBindings = {};

}

/\*\*

\* @ngdoc property

\* @name $rootScope.Scope#$id

\* @returns {number} Unique scope ID (monotonically increasing alphanumeric sequence) useful for

\* debugging.

\*/

Scope.prototype = {

constructor: Scope,

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$new

\* @function

\*

\* @description

\* Creates a new child {@link ng.$rootScope.Scope scope}.

\*

\* The parent scope will propagate the {@link ng.$rootScope.Scope#$digest $digest()} and

\* {@link ng.$rootScope.Scope#$digest $digest()} events. The scope can be removed from the

\* scope hierarchy using {@link ng.$rootScope.Scope#$destroy $destroy()}.

\*

\* {@link ng.$rootScope.Scope#$destroy $destroy()} must be called on a scope when it is

\* desired for the scope and its child scopes to be permanently detached from the parent and

\* thus stop participating in model change detection and listener notification by invoking.

\*

\* @param {boolean} isolate If true, then the scope does not prototypically inherit from the

\* parent scope. The scope is isolated, as it can not see parent scope properties.

\* When creating widgets, it is useful for the widget to not accidentally read parent

\* state.

\*

\* @returns {Object} The newly created child scope.

\*

\*/

$new: function(isolate) {

var ChildScope,

child;

if (isolate) {

child = new Scope();

child.$root = this.$root;

// ensure that there is just one async queue per $rootScope and its children

child.$$asyncQueue = this.$$asyncQueue;

child.$$postDigestQueue = this.$$postDigestQueue;

} else {

ChildScope = function() {}; // should be anonymous; This is so that when the minifier munges

// the name it does not become random set of chars. This will then show up as class

// name in the web inspector.

ChildScope.prototype = this;

child = new ChildScope();

child.$id = nextUid();

}

child['this'] = child;

child.$$listeners = {};

child.$$listenerCount = {};

child.$parent = this;

child.$$watchers = child.$$nextSibling = child.$$childHead = child.$$childTail = null;

child.$$prevSibling = this.$$childTail;

if (this.$$childHead) {

this.$$childTail.$$nextSibling = child;

this.$$childTail = child;

} else {

this.$$childHead = this.$$childTail = child;

}

return child;

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$watch

\* @function

\*

\* @description

\* Registers a `listener` callback to be executed whenever the `watchExpression` changes.

\*

\* - The `watchExpression` is called on every call to {@link ng.$rootScope.Scope#$digest

\* $digest()} and should return the value that will be watched. (Since

\* {@link ng.$rootScope.Scope#$digest $digest()} reruns when it detects changes the

\* `watchExpression` can execute multiple times per

\* {@link ng.$rootScope.Scope#$digest $digest()} and should be idempotent.)

\* - The `listener` is called only when the value from the current `watchExpression` and the

\* previous call to `watchExpression` are not equal (with the exception of the initial run,

\* see below). The inequality is determined according to

\* {@link angular.equals} function. To save the value of the object for later comparison,

\* the {@link angular.copy} function is used. It also means that watching complex options

\* will have adverse memory and performance implications.

\* - The watch `listener` may change the model, which may trigger other `listener`s to fire.

\* This is achieved by rerunning the watchers until no changes are detected. The rerun

\* iteration limit is 10 to prevent an infinite loop deadlock.

\*

\*

\* If you want to be notified whenever {@link ng.$rootScope.Scope#$digest $digest} is called,

\* you can register a `watchExpression` function with no `listener`. (Since `watchExpression`

\* can execute multiple times per {@link ng.$rootScope.Scope#$digest $digest} cycle when a

\* change is detected, be prepared for multiple calls to your listener.)

\*

\* After a watcher is registered with the scope, the `listener` fn is called asynchronously

\* (via {@link ng.$rootScope.Scope#$evalAsync $evalAsync}) to initialize the

\* watcher. In rare cases, this is undesirable because the listener is called when the result

\* of `watchExpression` didn't change. To detect this scenario within the `listener` fn, you

\* can compare the `newVal` and `oldVal`. If these two values are identical (`===`) then the

\* listener was called due to initialization.

\*

\* The example below contains an illustration of using a function as your $watch listener

\*

\*

\* # Example

\* ```js

// let's assume that scope was dependency injected as the $rootScope

var scope = $rootScope;

scope.name = 'misko';

scope.counter = 0;

expect(scope.counter).toEqual(0);

scope.$watch('name', function(newValue, oldValue) {

scope.counter = scope.counter + 1;

});

expect(scope.counter).toEqual(0);

scope.$digest();

// no variable change

expect(scope.counter).toEqual(0);

scope.name = 'adam';

scope.$digest();

expect(scope.counter).toEqual(1);

// Using a listener function

var food;

scope.foodCounter = 0;

expect(scope.foodCounter).toEqual(0);

scope.$watch(

// This is the listener function

function() { return food; },

// This is the change handler

function(newValue, oldValue) {

if ( newValue !== oldValue ) {

// Only increment the counter if the value changed

scope.foodCounter = scope.foodCounter + 1;

}

}

);

// No digest has been run so the counter will be zero

expect(scope.foodCounter).toEqual(0);

// Run the digest but since food has not changed count will still be zero

scope.$digest();

expect(scope.foodCounter).toEqual(0);

// Update food and run digest. Now the counter will increment

food = 'cheeseburger';

scope.$digest();

expect(scope.foodCounter).toEqual(1);

\* ```

\*

\*

\*

\* @param {(function()|string)} watchExpression Expression that is evaluated on each

\* {@link ng.$rootScope.Scope#$digest $digest} cycle. A change in the return value triggers

\* a call to the `listener`.

\*

\* - `string`: Evaluated as {@link guide/expression expression}

\* - `function(scope)`: called with current `scope` as a parameter.

\* @param {(function()|string)=} listener Callback called whenever the return value of

\* the `watchExpression` changes.

\*

\* - `string`: Evaluated as {@link guide/expression expression}

\* - `function(newValue, oldValue, scope)`: called with current and previous values as

\* parameters.

\*

\* @param {boolean=} objectEquality Compare for object equality using {@link angular.equals} instead of

\* comparing for reference equality.

\* @returns {function()} Returns a deregistration function for this listener.

\*/

$watch: function(watchExp, listener, objectEquality) {

var scope = this,

get = compileToFn(watchExp, 'watch'),

array = scope.$$watchers,

watcher = {

fn: listener,

last: initWatchVal,

get: get,

exp: watchExp,

eq: !!objectEquality

};

lastDirtyWatch = null;

// in the case user pass string, we need to compile it, do we really need this ?

if (!isFunction(listener)) {

var listenFn = compileToFn(listener || noop, 'listener');

watcher.fn = function(newVal, oldVal, scope) {listenFn(scope);};

}

if (typeof watchExp == 'string' && get.constant) {

var originalFn = watcher.fn;

watcher.fn = function(newVal, oldVal, scope) {

originalFn.call(this, newVal, oldVal, scope);

arrayRemove(array, watcher);

};

}

if (!array) {

array = scope.$$watchers = [];

}

// we use unshift since we use a while loop in $digest for speed.

// the while loop reads in reverse order.

array.unshift(watcher);

return function() {

arrayRemove(array, watcher);

lastDirtyWatch = null;

};

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$watchCollection

\* @function

\*

\* @description

\* Shallow watches the properties of an object and fires whenever any of the properties change

\* (for arrays, this implies watching the array items; for object maps, this implies watching

\* the properties). If a change is detected, the `listener` callback is fired.

\*

\* - The `obj` collection is observed via standard $watch operation and is examined on every

\* call to $digest() to see if any items have been added, removed, or moved.

\* - The `listener` is called whenever anything within the `obj` has changed. Examples include

\* adding, removing, and moving items belonging to an object or array.

\*

\*

\* # Example

\* ```js

$scope.names = ['igor', 'matias', 'misko', 'james'];

$scope.dataCount = 4;

$scope.$watchCollection('names', function(newNames, oldNames) {

$scope.dataCount = newNames.length;

});

expect($scope.dataCount).toEqual(4);

$scope.$digest();

//still at 4 ... no changes

expect($scope.dataCount).toEqual(4);

$scope.names.pop();

$scope.$digest();

//now there's been a change

expect($scope.dataCount).toEqual(3);

\* ```

\*

\*

\* @param {string|function(scope)} obj Evaluated as {@link guide/expression expression}. The

\* expression value should evaluate to an object or an array which is observed on each

\* {@link ng.$rootScope.Scope#$digest $digest} cycle. Any shallow change within the

\* collection will trigger a call to the `listener`.

\*

\* @param {function(newCollection, oldCollection, scope)} listener a callback function called

\* when a change is detected.

\* - The `newCollection` object is the newly modified data obtained from the `obj` expression

\* - The `oldCollection` object is a copy of the former collection data.

\* Due to performance considerations, the`oldCollection` value is computed only if the

\* `listener` function declares two or more arguments.

\* - The `scope` argument refers to the current scope.

\*

\* @returns {function()} Returns a de-registration function for this listener. When the

\* de-registration function is executed, the internal watch operation is terminated.

\*/

$watchCollection: function(obj, listener) {

var self = this;

// the current value, updated on each dirty-check run

var newValue;

// a shallow copy of the newValue from the last dirty-check run,

// updated to match newValue during dirty-check run

var oldValue;

// a shallow copy of the newValue from when the last change happened

var veryOldValue;

// only track veryOldValue if the listener is asking for it

var trackVeryOldValue = (listener.length > 1);

var changeDetected = 0;

var objGetter = $parse(obj);

var internalArray = [];

var internalObject = {};

var initRun = true;

var oldLength = 0;

function $watchCollectionWatch() {

newValue = objGetter(self);

var newLength, key;

if (!isObject(newValue)) { // if primitive

if (oldValue !== newValue) {

oldValue = newValue;

changeDetected++;

}

} else if (isArrayLike(newValue)) {

if (oldValue !== internalArray) {

// we are transitioning from something which was not an array into array.

oldValue = internalArray;

oldLength = oldValue.length = 0;

changeDetected++;

}

newLength = newValue.length;

if (oldLength !== newLength) {

// if lengths do not match we need to trigger change notification

changeDetected++;

oldValue.length = oldLength = newLength;

}

// copy the items to oldValue and look for changes.

for (var i = 0; i < newLength; i++) {

var bothNaN = (oldValue[i] !== oldValue[i]) &&

(newValue[i] !== newValue[i]);

if (!bothNaN && (oldValue[i] !== newValue[i])) {

changeDetected++;

oldValue[i] = newValue[i];

}

}

} else {

if (oldValue !== internalObject) {

// we are transitioning from something which was not an object into object.

oldValue = internalObject = {};

oldLength = 0;

changeDetected++;

}

// copy the items to oldValue and look for changes.

newLength = 0;

for (key in newValue) {

if (newValue.hasOwnProperty(key)) {

newLength++;

if (oldValue.hasOwnProperty(key)) {

if (oldValue[key] !== newValue[key]) {

changeDetected++;

oldValue[key] = newValue[key];

}

} else {

oldLength++;

oldValue[key] = newValue[key];

changeDetected++;

}

}

}

if (oldLength > newLength) {

// we used to have more keys, need to find them and destroy them.

changeDetected++;

for(key in oldValue) {

if (oldValue.hasOwnProperty(key) && !newValue.hasOwnProperty(key)) {

oldLength--;

delete oldValue[key];

}

}

}

}

return changeDetected;

}

function $watchCollectionAction() {

if (initRun) {

initRun = false;

listener(newValue, newValue, self);

} else {

listener(newValue, veryOldValue, self);

}

// make a copy for the next time a collection is changed

if (trackVeryOldValue) {

if (!isObject(newValue)) {

//primitive

veryOldValue = newValue;

} else if (isArrayLike(newValue)) {

veryOldValue = new Array(newValue.length);

for (var i = 0; i < newValue.length; i++) {

veryOldValue[i] = newValue[i];

}

} else { // if object

veryOldValue = {};

for (var key in newValue) {

if (hasOwnProperty.call(newValue, key)) {

veryOldValue[key] = newValue[key];

}

}

}

}

}

return this.$watch($watchCollectionWatch, $watchCollectionAction);

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$digest

\* @function

\*

\* @description

\* Processes all of the {@link ng.$rootScope.Scope#$watch watchers} of the current scope and

\* its children. Because a {@link ng.$rootScope.Scope#$watch watcher}'s listener can change

\* the model, the `$digest()` keeps calling the {@link ng.$rootScope.Scope#$watch watchers}

\* until no more listeners are firing. This means that it is possible to get into an infinite

\* loop. This function will throw `'Maximum iteration limit exceeded.'` if the number of

\* iterations exceeds 10.

\*

\* Usually, you don't call `$digest()` directly in

\* {@link ng.directive:ngController controllers} or in

\* {@link ng.$compileProvider#directive directives}.

\* Instead, you should call {@link ng.$rootScope.Scope#$apply $apply()} (typically from within

\* a {@link ng.$compileProvider#directive directives}), which will force a `$digest()`.

\*

\* If you want to be notified whenever `$digest()` is called,

\* you can register a `watchExpression` function with

\* {@link ng.$rootScope.Scope#$watch $watch()} with no `listener`.

\*

\* In unit tests, you may need to call `$digest()` to simulate the scope life cycle.

\*

\* # Example

\* ```js

var scope = ...;

scope.name = 'misko';

scope.counter = 0;

expect(scope.counter).toEqual(0);

scope.$watch('name', function(newValue, oldValue) {

scope.counter = scope.counter + 1;

});

expect(scope.counter).toEqual(0);

scope.$digest();

// no variable change

expect(scope.counter).toEqual(0);

scope.name = 'adam';

scope.$digest();

expect(scope.counter).toEqual(1);

\* ```

\*

\*/

$digest: function() {

var watch, value, last,

watchers,

asyncQueue = this.$$asyncQueue,

postDigestQueue = this.$$postDigestQueue,

length,

dirty, ttl = TTL,

next, current, target = this,

watchLog = [],

logIdx, logMsg, asyncTask;

beginPhase('$digest');

lastDirtyWatch = null;

do { // "while dirty" loop

dirty = false;

current = target;

while(asyncQueue.length) {

try {

asyncTask = asyncQueue.shift();

asyncTask.scope.$eval(asyncTask.expression);

} catch (e) {

clearPhase();

$exceptionHandler(e);

}

lastDirtyWatch = null;

}

traverseScopesLoop:

do { // "traverse the scopes" loop

if ((watchers = current.$$watchers)) {

// process our watches

length = watchers.length;

while (length--) {

try {

watch = watchers[length];

// Most common watches are on primitives, in which case we can short

// circuit it with === operator, only when === fails do we use .equals

if (watch) {

if ((value = watch.get(current)) !== (last = watch.last) &&

!(watch.eq

? equals(value, last)

: (typeof value == 'number' && typeof last == 'number'

&& isNaN(value) && isNaN(last)))) {

dirty = true;

lastDirtyWatch = watch;

watch.last = watch.eq ? copy(value) : value;

watch.fn(value, ((last === initWatchVal) ? value : last), current);

if (ttl < 5) {

logIdx = 4 - ttl;

if (!watchLog[logIdx]) watchLog[logIdx] = [];

logMsg = (isFunction(watch.exp))

? 'fn: ' + (watch.exp.name || watch.exp.toString())

: watch.exp;

logMsg += '; newVal: ' + toJson(value) + '; oldVal: ' + toJson(last);

watchLog[logIdx].push(logMsg);

}

} else if (watch === lastDirtyWatch) {

// If the most recently dirty watcher is now clean, short circuit since the remaining watchers

// have already been tested.

dirty = false;

break traverseScopesLoop;

}

}

} catch (e) {

clearPhase();

$exceptionHandler(e);

}

}

}

// Insanity Warning: scope depth-first traversal

// yes, this code is a bit crazy, but it works and we have tests to prove it!

// this piece should be kept in sync with the traversal in $broadcast

if (!(next = (current.$$childHead ||

(current !== target && current.$$nextSibling)))) {

while(current !== target && !(next = current.$$nextSibling)) {

current = current.$parent;

}

}

} while ((current = next));

// `break traverseScopesLoop;` takes us to here

if((dirty || asyncQueue.length) && !(ttl--)) {

clearPhase();

throw $rootScopeMinErr('infdig',

'{0} $digest() iterations reached. Aborting!\n' +

'Watchers fired in the last 5 iterations: {1}',

TTL, toJson(watchLog));

}

} while (dirty || asyncQueue.length);

clearPhase();

while(postDigestQueue.length) {

try {

postDigestQueue.shift()();

} catch (e) {

$exceptionHandler(e);

}

}

},

/\*\*

\* @ngdoc event

\* @name $rootScope.Scope#$destroy

\* @eventType broadcast on scope being destroyed

\*

\* @description

\* Broadcasted when a scope and its children are being destroyed.

\*

\* Note that, in AngularJS, there is also a `$destroy` jQuery event, which can be used to

\* clean up DOM bindings before an element is removed from the DOM.

\*/

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$destroy

\* @function

\*

\* @description

\* Removes the current scope (and all of its children) from the parent scope. Removal implies

\* that calls to {@link ng.$rootScope.Scope#$digest $digest()} will no longer

\* propagate to the current scope and its children. Removal also implies that the current

\* scope is eligible for garbage collection.

\*

\* The `$destroy()` is usually used by directives such as

\* {@link ng.directive:ngRepeat ngRepeat} for managing the

\* unrolling of the loop.

\*

\* Just before a scope is destroyed, a `$destroy` event is broadcasted on this scope.

\* Application code can register a `$destroy` event handler that will give it a chance to

\* perform any necessary cleanup.

\*

\* Note that, in AngularJS, there is also a `$destroy` jQuery event, which can be used to

\* clean up DOM bindings before an element is removed from the DOM.

\*/

$destroy: function() {

// we can't destroy the root scope or a scope that has been already destroyed

if (this.$$destroyed) return;

var parent = this.$parent;

this.$broadcast('$destroy');

this.$$destroyed = true;

if (this === $rootScope) return;

forEach(this.$$listenerCount, bind(null, decrementListenerCount, this));

// sever all the references to parent scopes (after this cleanup, the current scope should

// not be retained by any of our references and should be eligible for garbage collection)

if (parent.$$childHead == this) parent.$$childHead = this.$$nextSibling;

if (parent.$$childTail == this) parent.$$childTail = this.$$prevSibling;

if (this.$$prevSibling) this.$$prevSibling.$$nextSibling = this.$$nextSibling;

if (this.$$nextSibling) this.$$nextSibling.$$prevSibling = this.$$prevSibling;

// All of the code below is bogus code that works around V8's memory leak via optimized code

// and inline caches.

//

// see:

// - https://code.google.com/p/v8/issues/detail?id=2073#c26

// - https://github.com/angular/angular.js/issues/6794#issuecomment-38648909

// - https://github.com/angular/angular.js/issues/1313#issuecomment-10378451

this.$parent = this.$$nextSibling = this.$$prevSibling = this.$$childHead =

this.$$childTail = this.$root = null;

// don't reset these to null in case some async task tries to register a listener/watch/task

this.$$listeners = {};

this.$$watchers = this.$$asyncQueue = this.$$postDigestQueue = [];

// prevent NPEs since these methods have references to properties we nulled out

this.$destroy = this.$digest = this.$apply = noop;

this.$on = this.$watch = function() { return noop; };

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$eval

\* @function

\*

\* @description

\* Executes the `expression` on the current scope and returns the result. Any exceptions in

\* the expression are propagated (uncaught). This is useful when evaluating Angular

\* expressions.

\*

\* # Example

\* ```js

var scope = ng.$rootScope.Scope();

scope.a = 1;

scope.b = 2;

expect(scope.$eval('a+b')).toEqual(3);

expect(scope.$eval(function(scope){ return scope.a + scope.b; })).toEqual(3);

\* ```

\*

\* @param {(string|function())=} expression An angular expression to be executed.

\*

\* - `string`: execute using the rules as defined in {@link guide/expression expression}.

\* - `function(scope)`: execute the function with the current `scope` parameter.

\*

\* @param {(object)=} locals Local variables object, useful for overriding values in scope.

\* @returns {\*} The result of evaluating the expression.

\*/

$eval: function(expr, locals) {

return $parse(expr)(this, locals);

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$evalAsync

\* @function

\*

\* @description

\* Executes the expression on the current scope at a later point in time.

\*

\* The `$evalAsync` makes no guarantees as to when the `expression` will be executed, only

\* that:

\*

\* - it will execute after the function that scheduled the evaluation (preferably before DOM

\* rendering).

\* - at least one {@link ng.$rootScope.Scope#$digest $digest cycle} will be performed after

\* `expression` execution.

\*

\* Any exceptions from the execution of the expression are forwarded to the

\* {@link ng.$exceptionHandler $exceptionHandler} service.

\*

\* \_\_Note:\_\_ if this function is called outside of a `$digest` cycle, a new `$digest` cycle

\* will be scheduled. However, it is encouraged to always call code that changes the model

\* from within an `$apply` call. That includes code evaluated via `$evalAsync`.

\*

\* @param {(string|function())=} expression An angular expression to be executed.

\*

\* - `string`: execute using the rules as defined in {@link guide/expression expression}.

\* - `function(scope)`: execute the function with the current `scope` parameter.

\*

\*/

$evalAsync: function(expr) {

// if we are outside of an $digest loop and this is the first time we are scheduling async

// task also schedule async auto-flush

if (!$rootScope.$$phase && !$rootScope.$$asyncQueue.length) {

$browser.defer(function() {

if ($rootScope.$$asyncQueue.length) {

$rootScope.$digest();

}

});

}

this.$$asyncQueue.push({scope: this, expression: expr});

},

$$postDigest : function(fn) {

this.$$postDigestQueue.push(fn);

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$apply

\* @function

\*

\* @description

\* `$apply()` is used to execute an expression in angular from outside of the angular

\* framework. (For example from browser DOM events, setTimeout, XHR or third party libraries).

\* Because we are calling into the angular framework we need to perform proper scope life

\* cycle of {@link ng.$exceptionHandler exception handling},

\* {@link ng.$rootScope.Scope#$digest executing watches}.

\*

\* ## Life cycle

\*

\* # Pseudo-Code of `$apply()`

\* ```js

function $apply(expr) {

try {

return $eval(expr);

} catch (e) {

$exceptionHandler(e);

} finally {

$root.$digest();

}

}

\* ```

\*

\*

\* Scope's `$apply()` method transitions through the following stages:

\*

\* 1. The {@link guide/expression expression} is executed using the

\* {@link ng.$rootScope.Scope#$eval $eval()} method.

\* 2. Any exceptions from the execution of the expression are forwarded to the

\* {@link ng.$exceptionHandler $exceptionHandler} service.

\* 3. The {@link ng.$rootScope.Scope#$watch watch} listeners are fired immediately after the

\* expression was executed using the {@link ng.$rootScope.Scope#$digest $digest()} method.

\*

\*

\* @param {(string|function())=} exp An angular expression to be executed.

\*

\* - `string`: execute using the rules as defined in {@link guide/expression expression}.

\* - `function(scope)`: execute the function with current `scope` parameter.

\*

\* @returns {\*} The result of evaluating the expression.

\*/

$apply: function(expr) {

try {

beginPhase('$apply');

return this.$eval(expr);

} catch (e) {

$exceptionHandler(e);

} finally {

clearPhase();

try {

$rootScope.$digest();

} catch (e) {

$exceptionHandler(e);

throw e;

}

}

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$on

\* @function

\*

\* @description

\* Listens on events of a given type. See {@link ng.$rootScope.Scope#$emit $emit} for

\* discussion of event life cycle.

\*

\* The event listener function format is: `function(event, args...)`. The `event` object

\* passed into the listener has the following attributes:

\*

\* - `targetScope` - `{Scope}`: the scope on which the event was `$emit`-ed or

\* `$broadcast`-ed.

\* - `currentScope` - `{Scope}`: the current scope which is handling the event.

\* - `name` - `{string}`: name of the event.

\* - `stopPropagation` - `{function=}`: calling `stopPropagation` function will cancel

\* further event propagation (available only for events that were `$emit`-ed).

\* - `preventDefault` - `{function}`: calling `preventDefault` sets `defaultPrevented` flag

\* to true.

\* - `defaultPrevented` - `{boolean}`: true if `preventDefault` was called.

\*

\* @param {string} name Event name to listen on.

\* @param {function(event, ...args)} listener Function to call when the event is emitted.

\* @returns {function()} Returns a deregistration function for this listener.

\*/

$on: function(name, listener) {

var namedListeners = this.$$listeners[name];

if (!namedListeners) {

this.$$listeners[name] = namedListeners = [];

}

namedListeners.push(listener);

var current = this;

do {

if (!current.$$listenerCount[name]) {

current.$$listenerCount[name] = 0;

}

current.$$listenerCount[name]++;

} while ((current = current.$parent));

var self = this;

return function() {

namedListeners[indexOf(namedListeners, listener)] = null;

decrementListenerCount(self, 1, name);

};

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$emit

\* @function

\*

\* @description

\* Dispatches an event `name` upwards through the scope hierarchy notifying the

\* registered {@link ng.$rootScope.Scope#$on} listeners.

\*

\* The event life cycle starts at the scope on which `$emit` was called. All

\* {@link ng.$rootScope.Scope#$on listeners} listening for `name` event on this scope get

\* notified. Afterwards, the event traverses upwards toward the root scope and calls all

\* registered listeners along the way. The event will stop propagating if one of the listeners

\* cancels it.

\*

\* Any exception emitted from the {@link ng.$rootScope.Scope#$on listeners} will be passed

\* onto the {@link ng.$exceptionHandler $exceptionHandler} service.

\*

\* @param {string} name Event name to emit.

\* @param {...\*} args Optional one or more arguments which will be passed onto the event listeners.

\* @return {Object} Event object (see {@link ng.$rootScope.Scope#$on}).

\*/

$emit: function(name, args) {

var empty = [],

namedListeners,

scope = this,

stopPropagation = false,

event = {

name: name,

targetScope: scope,

stopPropagation: function() {stopPropagation = true;},

preventDefault: function() {

event.defaultPrevented = true;

},

defaultPrevented: false

},

listenerArgs = concat([event], arguments, 1),

i, length;

do {

namedListeners = scope.$$listeners[name] || empty;

event.currentScope = scope;

for (i=0, length=namedListeners.length; i<length; i++) {

// if listeners were deregistered, defragment the array

if (!namedListeners[i]) {

namedListeners.splice(i, 1);

i--;

length--;

continue;

}

try {

//allow all listeners attached to the current scope to run

namedListeners[i].apply(null, listenerArgs);

} catch (e) {

$exceptionHandler(e);

}

}

//if any listener on the current scope stops propagation, prevent bubbling

if (stopPropagation) return event;

//traverse upwards

scope = scope.$parent;

} while (scope);

return event;

},

/\*\*

\* @ngdoc method

\* @name $rootScope.Scope#$broadcast

\* @function

\*

\* @description

\* Dispatches an event `name` downwards to all child scopes (and their children) notifying the

\* registered {@link ng.$rootScope.Scope#$on} listeners.

\*

\* The event life cycle starts at the scope on which `$broadcast` was called. All

\* {@link ng.$rootScope.Scope#$on listeners} listening for `name` event on this scope get

\* notified. Afterwards, the event propagates to all direct and indirect scopes of the current

\* scope and calls all registered listeners along the way. The event cannot be canceled.

\*

\* Any exception emitted from the {@link ng.$rootScope.Scope#$on listeners} will be passed

\* onto the {@link ng.$exceptionHandler $exceptionHandler} service.

\*

\* @param {string} name Event name to broadcast.

\* @param {...\*} args Optional one or more arguments which will be passed onto the event listeners.

\* @return {Object} Event object, see {@link ng.$rootScope.Scope#$on}

\*/

$broadcast: function(name, args) {

var target = this,

current = target,

next = target,

event = {

name: name,

targetScope: target,

preventDefault: function() {

event.defaultPrevented = true;

},

defaultPrevented: false

},

listenerArgs = concat([event], arguments, 1),

listeners, i, length;

//down while you can, then up and next sibling or up and next sibling until back at root

while ((current = next)) {

event.currentScope = current;

listeners = current.$$listeners[name] || [];

for (i=0, length = listeners.length; i<length; i++) {

// if listeners were deregistered, defragment the array

if (!listeners[i]) {

listeners.splice(i, 1);

i--;

length--;

continue;

}

try {

listeners[i].apply(null, listenerArgs);

} catch(e) {

$exceptionHandler(e);

}

}

// Insanity Warning: scope depth-first traversal

// yes, this code is a bit crazy, but it works and we have tests to prove it!

// this piece should be kept in sync with the traversal in $digest

// (though it differs due to having the extra check for $$listenerCount)

if (!(next = ((current.$$listenerCount[name] && current.$$childHead) ||

(current !== target && current.$$nextSibling)))) {

while(current !== target && !(next = current.$$nextSibling)) {

current = current.$parent;

}

}

}

return event;

}

};

var $rootScope = new Scope();

return $rootScope;

function beginPhase(phase) {

if ($rootScope.$$phase) {

throw $rootScopeMinErr('inprog', '{0} already in progress', $rootScope.$$phase);

}

$rootScope.$$phase = phase;

}

function clearPhase() {

$rootScope.$$phase = null;

}

function compileToFn(exp, name) {

var fn = $parse(exp);

assertArgFn(fn, name);

return fn;

}

function decrementListenerCount(current, count, name) {

do {

current.$$listenerCount[name] -= count;

if (current.$$listenerCount[name] === 0) {

delete current.$$listenerCount[name];

}

} while ((current = current.$parent));

}

/\*\*

\* function used as an initial value for watchers.

\* because it's unique we can easily tell it apart from other values

\*/

function initWatchVal() {}

}];

}

/\*\*

\* @description

\* Private service to sanitize uris for links and images. Used by $compile and $sanitize.

\*/

function $$SanitizeUriProvider() {

var aHrefSanitizationWhitelist = /^\s\*(https?|ftp|mailto|tel|file):/,

imgSrcSanitizationWhitelist = /^\s\*(https?|ftp|file):|data:image\//;

/\*\*

\* @description

\* Retrieves or overrides the default regular expression that is used for whitelisting of safe

\* urls during a[href] sanitization.

\*

\* The sanitization is a security measure aimed at prevent XSS attacks via html links.

\*

\* Any url about to be assigned to a[href] via data-binding is first normalized and turned into

\* an absolute url. Afterwards, the url is matched against the `aHrefSanitizationWhitelist`

\* regular expression. If a match is found, the original url is written into the dom. Otherwise,

\* the absolute url is prefixed with `'unsafe:'` string and only then is it written into the DOM.

\*

\* @param {RegExp=} regexp New regexp to whitelist urls with.

\* @returns {RegExp|ng.$compileProvider} Current RegExp if called without value or self for

\* chaining otherwise.

\*/

this.aHrefSanitizationWhitelist = function(regexp) {

if (isDefined(regexp)) {

aHrefSanitizationWhitelist = regexp;

return this;

}

return aHrefSanitizationWhitelist;

};

/\*\*

\* @description

\* Retrieves or overrides the default regular expression that is used for whitelisting of safe

\* urls during img[src] sanitization.

\*

\* The sanitization is a security measure aimed at prevent XSS attacks via html links.

\*

\* Any url about to be assigned to img[src] via data-binding is first normalized and turned into

\* an absolute url. Afterwards, the url is matched against the `imgSrcSanitizationWhitelist`

\* regular expression. If a match is found, the original url is written into the dom. Otherwise,

\* the absolute url is prefixed with `'unsafe:'` string and only then is it written into the DOM.

\*

\* @param {RegExp=} regexp New regexp to whitelist urls with.

\* @returns {RegExp|ng.$compileProvider} Current RegExp if called without value or self for

\* chaining otherwise.

\*/

this.imgSrcSanitizationWhitelist = function(regexp) {

if (isDefined(regexp)) {

imgSrcSanitizationWhitelist = regexp;

return this;

}

return imgSrcSanitizationWhitelist;

};

this.$get = function() {

return function sanitizeUri(uri, isImage) {

var regex = isImage ? imgSrcSanitizationWhitelist : aHrefSanitizationWhitelist;

var normalizedVal;

// NOTE: urlResolve() doesn't support IE < 8 so we don't sanitize for that case.

if (!msie || msie >= 8 ) {

normalizedVal = urlResolve(uri).href;

if (normalizedVal !== '' && !normalizedVal.match(regex)) {

return 'unsafe:'+normalizedVal;

}

}

return uri;

};

};

}

var $sceMinErr = minErr('$sce');

var SCE\_CONTEXTS = {

HTML: 'html',

CSS: 'css',

URL: 'url',

// RESOURCE\_URL is a subtype of URL used in contexts where a privileged resource is sourced from a

// url. (e.g. ng-include, script src, templateUrl)

RESOURCE\_URL: 'resourceUrl',

JS: 'js'

};

// Helper functions follow.

// Copied from:

// http://docs.closure-library.googlecode.com/git/closure\_goog\_string\_string.js.source.html#line962

// Prereq: s is a string.

function escapeForRegexp(s) {

return s.replace(/([-()\[\]{}+?\*.$\^|,:#<!\\])/g, '\\$1').

replace(/\x08/g, '\\x08');

}

function adjustMatcher(matcher) {

if (matcher === 'self') {

return matcher;

} else if (isString(matcher)) {

// Strings match exactly except for 2 wildcards - '\*' and '\*\*'.

// '\*' matches any character except those from the set ':/.?&'.

// '\*\*' matches any character (like .\* in a RegExp).

// More than 2 \*'s raises an error as it's ill defined.

if (matcher.indexOf('\*\*\*') > -1) {

throw $sceMinErr('iwcard',

'Illegal sequence \*\*\* in string matcher. String: {0}', matcher);

}

matcher = escapeForRegexp(matcher).

replace('\\\*\\\*', '.\*').

replace('\\\*', '[^:/.?&;]\*');

return new RegExp('^' + matcher + '$');

} else if (isRegExp(matcher)) {

// The only other type of matcher allowed is a Regexp.

// Match entire URL / disallow partial matches.

// Flags are reset (i.e. no global, ignoreCase or multiline)

return new RegExp('^' + matcher.source + '$');

} else {

throw $sceMinErr('imatcher',

'Matchers may only be "self", string patterns or RegExp objects');

}

}

function adjustMatchers(matchers) {

var adjustedMatchers = [];

if (isDefined(matchers)) {

forEach(matchers, function(matcher) {

adjustedMatchers.push(adjustMatcher(matcher));

});

}

return adjustedMatchers;

}

/\*\*

\* @ngdoc service

\* @name $sceDelegate

\* @function

\*

\* @description

\*

\* `$sceDelegate` is a service that is used by the `$sce` service to provide {@link ng.$sce Strict

\* Contextual Escaping (SCE)} services to AngularJS.

\*

\* Typically, you would configure or override the {@link ng.$sceDelegate $sceDelegate} instead of

\* the `$sce` service to customize the way Strict Contextual Escaping works in AngularJS. This is

\* because, while the `$sce` provides numerous shorthand methods, etc., you really only need to

\* override 3 core functions (`trustAs`, `getTrusted` and `valueOf`) to replace the way things

\* work because `$sce` delegates to `$sceDelegate` for these operations.

\*

\* Refer {@link ng.$sceDelegateProvider $sceDelegateProvider} to configure this service.

\*

\* The default instance of `$sceDelegate` should work out of the box with little pain. While you

\* can override it completely to change the behavior of `$sce`, the common case would

\* involve configuring the {@link ng.$sceDelegateProvider $sceDelegateProvider} instead by setting

\* your own whitelists and blacklists for trusting URLs used for loading AngularJS resources such as

\* templates. Refer {@link ng.$sceDelegateProvider#resourceUrlWhitelist

\* $sceDelegateProvider.resourceUrlWhitelist} and {@link

\* ng.$sceDelegateProvider#resourceUrlBlacklist $sceDelegateProvider.resourceUrlBlacklist}

\*/

/\*\*

\* @ngdoc provider

\* @name $sceDelegateProvider

\* @description

\*

\* The `$sceDelegateProvider` provider allows developers to configure the {@link ng.$sceDelegate

\* $sceDelegate} service. This allows one to get/set the whitelists and blacklists used to ensure

\* that the URLs used for sourcing Angular templates are safe. Refer {@link

\* ng.$sceDelegateProvider#resourceUrlWhitelist $sceDelegateProvider.resourceUrlWhitelist} and

\* {@link ng.$sceDelegateProvider#resourceUrlBlacklist $sceDelegateProvider.resourceUrlBlacklist}

\*

\* For the general details about this service in Angular, read the main page for {@link ng.$sce

\* Strict Contextual Escaping (SCE)}.

\*

\* \*\*Example\*\*: Consider the following case. <a name="example"></a>

\*

\* - your app is hosted at url `http://myapp.example.com/`

\* - but some of your templates are hosted on other domains you control such as

\* `http://srv01.assets.example.com/`,  `http://srv02.assets.example.com/`, etc.

\* - and you have an open redirect at `http://myapp.example.com/clickThru?...`.

\*

\* Here is what a secure configuration for this scenario might look like:

\*

\* <pre class="prettyprint">

\* angular.module('myApp', []).config(function($sceDelegateProvider) {

\* $sceDelegateProvider.resourceUrlWhitelist([

\* // Allow same origin resource loads.

\* 'self',

\* // Allow loading from our assets domain. Notice the difference between \* and \*\*.

\* 'http://srv\*.assets.example.com/\*\*']);

\*

\* // The blacklist overrides the whitelist so the open redirect here is blocked.

\* $sceDelegateProvider.resourceUrlBlacklist([

\* 'http://myapp.example.com/clickThru\*\*']);

\* });

\* </pre>

\*/

function $SceDelegateProvider() {

this.SCE\_CONTEXTS = SCE\_CONTEXTS;

// Resource URLs can also be trusted by policy.

var resourceUrlWhitelist = ['self'],

resourceUrlBlacklist = [];

/\*\*

\* @ngdoc method

\* @name $sceDelegateProvider#resourceUrlWhitelist

\* @function

\*

\* @param {Array=} whitelist When provided, replaces the resourceUrlWhitelist with the value

\* provided. This must be an array or null. A snapshot of this array is used so further

\* changes to the array are ignored.

\*

\* Follow {@link ng.$sce#resourceUrlPatternItem this link} for a description of the items

\* allowed in this array.

\*

\* Note: \*\*an empty whitelist array will block all URLs\*\*!

\*

\* @return {Array} the currently set whitelist array.

\*

\* The \*\*default value\*\* when no whitelist has been explicitly set is `['self']` allowing only

\* same origin resource requests.

\*

\* @description

\* Sets/Gets the whitelist of trusted resource URLs.

\*/

this.resourceUrlWhitelist = function (value) {

if (arguments.length) {

resourceUrlWhitelist = adjustMatchers(value);

}

return resourceUrlWhitelist;

};

/\*\*

\* @ngdoc method

\* @name $sceDelegateProvider#resourceUrlBlacklist

\* @function

\*

\* @param {Array=} blacklist When provided, replaces the resourceUrlBlacklist with the value

\* provided. This must be an array or null. A snapshot of this array is used so further

\* changes to the array are ignored.

\*

\* Follow {@link ng.$sce#resourceUrlPatternItem this link} for a description of the items

\* allowed in this array.

\*

\* The typical usage for the blacklist is to \*\*block

\* [open redirects](http://cwe.mitre.org/data/definitions/601.html)\*\* served by your domain as

\* these would otherwise be trusted but actually return content from the redirected domain.

\*

\* Finally, \*\*the blacklist overrides the whitelist\*\* and has the final say.

\*

\* @return {Array} the currently set blacklist array.

\*

\* The \*\*default value\*\* when no whitelist has been explicitly set is the empty array (i.e. there

\* is no blacklist.)

\*

\* @description

\* Sets/Gets the blacklist of trusted resource URLs.

\*/

this.resourceUrlBlacklist = function (value) {

if (arguments.length) {

resourceUrlBlacklist = adjustMatchers(value);

}

return resourceUrlBlacklist;

};

this.$get = ['$injector', function($injector) {

var htmlSanitizer = function htmlSanitizer(html) {

throw $sceMinErr('unsafe', 'Attempting to use an unsafe value in a safe context.');

};

if ($injector.has('$sanitize')) {

htmlSanitizer = $injector.get('$sanitize');

}

function matchUrl(matcher, parsedUrl) {

if (matcher === 'self') {

return urlIsSameOrigin(parsedUrl);

} else {

// definitely a regex. See adjustMatchers()

return !!matcher.exec(parsedUrl.href);

}

}

function isResourceUrlAllowedByPolicy(url) {

var parsedUrl = urlResolve(url.toString());

var i, n, allowed = false;

// Ensure that at least one item from the whitelist allows this url.

for (i = 0, n = resourceUrlWhitelist.length; i < n; i++) {

if (matchUrl(resourceUrlWhitelist[i], parsedUrl)) {

allowed = true;

break;

}

}

if (allowed) {

// Ensure that no item from the blacklist blocked this url.

for (i = 0, n = resourceUrlBlacklist.length; i < n; i++) {

if (matchUrl(resourceUrlBlacklist[i], parsedUrl)) {

allowed = false;

break;

}

}

}

return allowed;

}

function generateHolderType(Base) {

var holderType = function TrustedValueHolderType(trustedValue) {

this.$$unwrapTrustedValue = function() {

return trustedValue;

};

};

if (Base) {

holderType.prototype = new Base();

}

holderType.prototype.valueOf = function sceValueOf() {

return this.$$unwrapTrustedValue();

};

holderType.prototype.toString = function sceToString() {

return this.$$unwrapTrustedValue().toString();

};

return holderType;

}

var trustedValueHolderBase = generateHolderType(),

byType = {};

byType[SCE\_CONTEXTS.HTML] = generateHolderType(trustedValueHolderBase);

byType[SCE\_CONTEXTS.CSS] = generateHolderType(trustedValueHolderBase);

byType[SCE\_CONTEXTS.URL] = generateHolderType(trustedValueHolderBase);

byType[SCE\_CONTEXTS.JS] = generateHolderType(trustedValueHolderBase);

byType[SCE\_CONTEXTS.RESOURCE\_URL] = generateHolderType(byType[SCE\_CONTEXTS.URL]);

/\*\*

\* @ngdoc method

\* @name $sceDelegate#trustAs

\*

\* @description

\* Returns an object that is trusted by angular for use in specified strict

\* contextual escaping contexts (such as ng-bind-html, ng-include, any src

\* attribute interpolation, any dom event binding attribute interpolation

\* such as for onclick, etc.) that uses the provided value.

\* See {@link ng.$sce $sce} for enabling strict contextual escaping.

\*

\* @param {string} type The kind of context in which this value is safe for use. e.g. url,

\* resourceUrl, html, js and css.

\* @param {\*} value The value that that should be considered trusted/safe.

\* @returns {\*} A value that can be used to stand in for the provided `value` in places

\* where Angular expects a $sce.trustAs() return value.

\*/

function trustAs(type, trustedValue) {

var Constructor = (byType.hasOwnProperty(type) ? byType[type] : null);

if (!Constructor) {

throw $sceMinErr('icontext',

'Attempted to trust a value in invalid context. Context: {0}; Value: {1}',

type, trustedValue);

}

if (trustedValue === null || trustedValue === undefined || trustedValue === '') {

return trustedValue;

}

// All the current contexts in SCE\_CONTEXTS happen to be strings. In order to avoid trusting

// mutable objects, we ensure here that the value passed in is actually a string.

if (typeof trustedValue !== 'string') {

throw $sceMinErr('itype',

'Attempted to trust a non-string value in a content requiring a string: Context: {0}',

type);

}

return new Constructor(trustedValue);

}

/\*\*

\* @ngdoc method

\* @name $sceDelegate#valueOf

\*

\* @description

\* If the passed parameter had been returned by a prior call to {@link ng.$sceDelegate#trustAs

\* `$sceDelegate.trustAs`}, returns the value that had been passed to {@link

\* ng.$sceDelegate#trustAs `$sceDelegate.trustAs`}.

\*

\* If the passed parameter is not a value that had been returned by {@link

\* ng.$sceDelegate#trustAs `$sceDelegate.trustAs`}, returns it as-is.

\*

\* @param {\*} value The result of a prior {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs`}

\* call or anything else.

\* @returns {\*} The `value` that was originally provided to {@link ng.$sceDelegate#trustAs

\* `$sceDelegate.trustAs`} if `value` is the result of such a call. Otherwise, returns

\* `value` unchanged.

\*/

function valueOf(maybeTrusted) {

if (maybeTrusted instanceof trustedValueHolderBase) {

return maybeTrusted.$$unwrapTrustedValue();

} else {

return maybeTrusted;

}

}

/\*\*

\* @ngdoc method

\* @name $sceDelegate#getTrusted

\*

\* @description

\* Takes the result of a {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs`} call and

\* returns the originally supplied value if the queried context type is a supertype of the

\* created type. If this condition isn't satisfied, throws an exception.

\*

\* @param {string} type The kind of context in which this value is to be used.

\* @param {\*} maybeTrusted The result of a prior {@link ng.$sceDelegate#trustAs

\* `$sceDelegate.trustAs`} call.

\* @returns {\*} The value the was originally provided to {@link ng.$sceDelegate#trustAs

\* `$sceDelegate.trustAs`} if valid in this context. Otherwise, throws an exception.

\*/

function getTrusted(type, maybeTrusted) {

if (maybeTrusted === null || maybeTrusted === undefined || maybeTrusted === '') {

return maybeTrusted;

}

var constructor = (byType.hasOwnProperty(type) ? byType[type] : null);

if (constructor && maybeTrusted instanceof constructor) {

return maybeTrusted.$$unwrapTrustedValue();

}

// If we get here, then we may only take one of two actions.

// 1. sanitize the value for the requested type, or

// 2. throw an exception.

if (type === SCE\_CONTEXTS.RESOURCE\_URL) {

if (isResourceUrlAllowedByPolicy(maybeTrusted)) {

return maybeTrusted;

} else {

throw $sceMinErr('insecurl',

'Blocked loading resource from url not allowed by $sceDelegate policy. URL: {0}',

maybeTrusted.toString());

}

} else if (type === SCE\_CONTEXTS.HTML) {

return htmlSanitizer(maybeTrusted);

}

throw $sceMinErr('unsafe', 'Attempting to use an unsafe value in a safe context.');

}

return { trustAs: trustAs,

getTrusted: getTrusted,

valueOf: valueOf };

}];

}

/\*\*

\* @ngdoc provider

\* @name $sceProvider

\* @description

\*

\* The $sceProvider provider allows developers to configure the {@link ng.$sce $sce} service.

\* - enable/disable Strict Contextual Escaping (SCE) in a module

\* - override the default implementation with a custom delegate

\*

\* Read more about {@link ng.$sce Strict Contextual Escaping (SCE)}.

\*/

/\* jshint maxlen: false\*/

/\*\*

\* @ngdoc service

\* @name $sce

\* @function

\*

\* @description

\*

\* `$sce` is a service that provides Strict Contextual Escaping services to AngularJS.

\*

\* # Strict Contextual Escaping

\*

\* Strict Contextual Escaping (SCE) is a mode in which AngularJS requires bindings in certain

\* contexts to result in a value that is marked as safe to use for that context. One example of

\* such a context is binding arbitrary html controlled by the user via `ng-bind-html`. We refer

\* to these contexts as privileged or SCE contexts.

\*

\* As of version 1.2, Angular ships with SCE enabled by default.

\*

\* Note: When enabled (the default), IE8 in quirks mode is not supported. In this mode, IE8 allows

\* one to execute arbitrary javascript by the use of the expression() syntax. Refer

\* <http://blogs.msdn.com/b/ie/archive/2008/10/16/ending-expressions.aspx> to learn more about them.

\* You can ensure your document is in standards mode and not quirks mode by adding `<!doctype html>`

\* to the top of your HTML document.

\*

\* SCE assists in writing code in way that (a) is secure by default and (b) makes auditing for

\* security vulnerabilities such as XSS, clickjacking, etc. a lot easier.

\*

\* Here's an example of a binding in a privileged context:

\*

\* <pre class="prettyprint">

\* <input ng-model="userHtml">

\* <div ng-bind-html="userHtml">

\* </pre>

\*

\* Notice that `ng-bind-html` is bound to `userHtml` controlled by the user. With SCE

\* disabled, this application allows the user to render arbitrary HTML into the DIV.

\* In a more realistic example, one may be rendering user comments, blog articles, etc. via

\* bindings. (HTML is just one example of a context where rendering user controlled input creates

\* security vulnerabilities.)

\*

\* For the case of HTML, you might use a library, either on the client side, or on the server side,

\* to sanitize unsafe HTML before binding to the value and rendering it in the document.

\*

\* How would you ensure that every place that used these types of bindings was bound to a value that

\* was sanitized by your library (or returned as safe for rendering by your server?) How can you

\* ensure that you didn't accidentally delete the line that sanitized the value, or renamed some

\* properties/fields and forgot to update the binding to the sanitized value?

\*

\* To be secure by default, you want to ensure that any such bindings are disallowed unless you can

\* determine that something explicitly says it's safe to use a value for binding in that

\* context. You can then audit your code (a simple grep would do) to ensure that this is only done

\* for those values that you can easily tell are safe - because they were received from your server,

\* sanitized by your library, etc. You can organize your codebase to help with this - perhaps

\* allowing only the files in a specific directory to do this. Ensuring that the internal API

\* exposed by that code doesn't markup arbitrary values as safe then becomes a more manageable task.

\*

\* In the case of AngularJS' SCE service, one uses {@link ng.$sce#trustAs $sce.trustAs}

\* (and shorthand methods such as {@link ng.$sce#trustAsHtml $sce.trustAsHtml}, etc.) to

\* obtain values that will be accepted by SCE / privileged contexts.

\*

\*

\* ## How does it work?

\*

\* In privileged contexts, directives and code will bind to the result of {@link ng.$sce#getTrusted

\* $sce.getTrusted(context, value)} rather than to the value directly. Directives use {@link

\* ng.$sce#parse $sce.parseAs} rather than `$parse` to watch attribute bindings, which performs the

\* {@link ng.$sce#getTrusted $sce.getTrusted} behind the scenes on non-constant literals.

\*

\* As an example, {@link ng.directive:ngBindHtml ngBindHtml} uses {@link

\* ng.$sce#parseAsHtml $sce.parseAsHtml(binding expression)}. Here's the actual code (slightly

\* simplified):

\*

\* <pre class="prettyprint">

\* var ngBindHtmlDirective = ['$sce', function($sce) {

\* return function(scope, element, attr) {

\* scope.$watch($sce.parseAsHtml(attr.ngBindHtml), function(value) {

\* element.html(value || '');

\* });

\* };

\* }];

\* </pre>

\*

\* ## Impact on loading templates

\*

\* This applies both to the {@link ng.directive:ngInclude `ng-include`} directive as well as

\* `templateUrl`'s specified by {@link guide/directive directives}.

\*

\* By default, Angular only loads templates from the same domain and protocol as the application

\* document. This is done by calling {@link ng.$sce#getTrustedResourceUrl

\* $sce.getTrustedResourceUrl} on the template URL. To load templates from other domains and/or

\* protocols, you may either either {@link ng.$sceDelegateProvider#resourceUrlWhitelist whitelist

\* them} or {@link ng.$sce#trustAsResourceUrl wrap it} into a trusted value.

\*

\* \*Please note\*:

\* The browser's

\* [Same Origin Policy](https://code.google.com/p/browsersec/wiki/Part2#Same-origin\_policy\_for\_XMLHttpRequest)

\* and [Cross-Origin Resource Sharing (CORS)](http://www.w3.org/TR/cors/)

\* policy apply in addition to this and may further restrict whether the template is successfully

\* loaded. This means that without the right CORS policy, loading templates from a different domain

\* won't work on all browsers. Also, loading templates from `file://` URL does not work on some

\* browsers.

\*

\* ## This feels like too much overhead for the developer?

\*

\* It's important to remember that SCE only applies to interpolation expressions.

\*

\* If your expressions are constant literals, they're automatically trusted and you don't need to

\* call `$sce.trustAs` on them (remember to include the `ngSanitize` module) (e.g.

\* `<div ng-bind-html="'<b>implicitly trusted</b>'"></div>`) just works.

\*

\* Additionally, `a[href]` and `img[src]` automatically sanitize their URLs and do not pass them

\* through {@link ng.$sce#getTrusted $sce.getTrusted}. SCE doesn't play a role here.

\*

\* The included {@link ng.$sceDelegate $sceDelegate} comes with sane defaults to allow you to load

\* templates in `ng-include` from your application's domain without having to even know about SCE.

\* It blocks loading templates from other domains or loading templates over http from an https

\* served document. You can change these by setting your own custom {@link

\* ng.$sceDelegateProvider#resourceUrlWhitelist whitelists} and {@link

\* ng.$sceDelegateProvider#resourceUrlBlacklist blacklists} for matching such URLs.

\*

\* This significantly reduces the overhead. It is far easier to pay the small overhead and have an

\* application that's secure and can be audited to verify that with much more ease than bolting

\* security onto an application later.

\*

\* <a name="contexts"></a>

\* ## What trusted context types are supported?

\*

\* | Context | Notes |

\* |---------------------|----------------|

\* | `$sce.HTML` | For HTML that's safe to source into the application. The {@link ng.directive:ngBindHtml ngBindHtml} directive uses this context for bindings. |

\* | `$sce.CSS` | For CSS that's safe to source into the application. Currently unused. Feel free to use it in your own directives. |

\* | `$sce.URL` | For URLs that are safe to follow as links. Currently unused (`<a href=` and `<img src=` sanitize their urls and don't constitute an SCE context. |

\* | `$sce.RESOURCE\_URL` | For URLs that are not only safe to follow as links, but whose contents are also safe to include in your application. Examples include `ng-include`, `src` / `ngSrc` bindings for tags other than `IMG` (e.g. `IFRAME`, `OBJECT`, etc.) <br><br>Note that `$sce.RESOURCE\_URL` makes a stronger statement about the URL than `$sce.URL` does and therefore contexts requiring values trusted for `$sce.RESOURCE\_URL` can be used anywhere that values trusted for `$sce.URL` are required. |

\* | `$sce.JS` | For JavaScript that is safe to execute in your application's context. Currently unused. Feel free to use it in your own directives. |

\*

\* ## Format of items in {@link ng.$sceDelegateProvider#resourceUrlWhitelist resourceUrlWhitelist}/{@link ng.$sceDelegateProvider#resourceUrlBlacklist Blacklist} <a name="resourceUrlPatternItem"></a>

\*

\* Each element in these arrays must be one of the following:

\*

\* - \*\*'self'\*\*

\* - The special \*\*string\*\*, `'self'`, can be used to match against all URLs of the \*\*same

\* domain\*\* as the application document using the \*\*same protocol\*\*.

\* - \*\*String\*\* (except the special value `'self'`)

\* - The string is matched against the full \*normalized / absolute URL\* of the resource

\* being tested (substring matches are not good enough.)

\* - There are exactly \*\*two wildcard sequences\*\* - `\*` and `\*\*`. All other characters

\* match themselves.

\* - `\*`: matches zero or more occurrences of any character other than one of the following 6

\* characters: '`:`', '`/`', '`.`', '`?`', '`&`' and ';'. It's a useful wildcard for use

\* in a whitelist.

\* - `\*\*`: matches zero or more occurrences of \*any\* character. As such, it's not

\* not appropriate to use in for a scheme, domain, etc. as it would match too much. (e.g.

\* http://\*\*.example.com/ would match http://evil.com/?ignore=.example.com/ and that might

\* not have been the intention.) It's usage at the very end of the path is ok. (e.g.

\* http://foo.example.com/templates/\*\*).

\* - \*\*RegExp\*\* (\*see caveat below\*)

\* - \*Caveat\*: While regular expressions are powerful and offer great flexibility, their syntax

\* (and all the inevitable escaping) makes them \*harder to maintain\*. It's easy to

\* accidentally introduce a bug when one updates a complex expression (imho, all regexes should

\* have good test coverage.). For instance, the use of `.` in the regex is correct only in a

\* small number of cases. A `.` character in the regex used when matching the scheme or a

\* subdomain could be matched against a `:` or literal `.` that was likely not intended. It

\* is highly recommended to use the string patterns and only fall back to regular expressions

\* if they as a last resort.

\* - The regular expression must be an instance of RegExp (i.e. not a string.) It is

\* matched against the \*\*entire\*\* \*normalized / absolute URL\* of the resource being tested

\* (even when the RegExp did not have the `^` and `$` codes.) In addition, any flags

\* present on the RegExp (such as multiline, global, ignoreCase) are ignored.

\* - If you are generating your JavaScript from some other templating engine (not

\* recommended, e.g. in issue [#4006](https://github.com/angular/angular.js/issues/4006)),

\* remember to escape your regular expression (and be aware that you might need more than

\* one level of escaping depending on your templating engine and the way you interpolated

\* the value.) Do make use of your platform's escaping mechanism as it might be good

\* enough before coding your own. e.g. Ruby has

\* [Regexp.escape(str)](http://www.ruby-doc.org/core-2.0.0/Regexp.html#method-c-escape)

\* and Python has [re.escape](http://docs.python.org/library/re.html#re.escape).

\* Javascript lacks a similar built in function for escaping. Take a look at Google

\* Closure library's [goog.string.regExpEscape(s)](

\* http://docs.closure-library.googlecode.com/git/closure\_goog\_string\_string.js.source.html#line962).

\*

\* Refer {@link ng.$sceDelegateProvider $sceDelegateProvider} for an example.

\*

\* ## Show me an example using SCE.

\*

\* @example

<example module="mySceApp" deps="angular-sanitize.js">

<file name="index.html">

<div ng-controller="myAppController as myCtrl">

<i ng-bind-html="myCtrl.explicitlyTrustedHtml" id="explicitlyTrustedHtml"></i><br><br>

<b>User comments</b><br>

By default, HTML that isn't explicitly trusted (e.g. Alice's comment) is sanitized when

$sanitize is available. If $sanitize isn't available, this results in an error instead of an

exploit.

<div class="well">

<div ng-repeat="userComment in myCtrl.userComments">

<b>{{userComment.name}}</b>:

<span ng-bind-html="userComment.htmlComment" class="htmlComment"></span>

<br>

</div>

</div>

</div>

</file>

<file name="script.js">

var mySceApp = angular.module('mySceApp', ['ngSanitize']);

mySceApp.controller("myAppController", function myAppController($http, $templateCache, $sce) {

var self = this;

$http.get("test\_data.json", {cache: $templateCache}).success(function(userComments) {

self.userComments = userComments;

});

self.explicitlyTrustedHtml = $sce.trustAsHtml(

'<span onmouseover="this.textContent=&quot;Explicitly trusted HTML bypasses ' +

'sanitization.&quot;">Hover over this text.</span>');

});

</file>

<file name="test\_data.json">

[

{ "name": "Alice",

"htmlComment":

"<span onmouseover='this.textContent=\"PWN3D!\"'>Is <i>anyone</i> reading this?</span>"

},

{ "name": "Bob",

"htmlComment": "<i>Yes!</i> Am I the only other one?"

}

]

</file>

<file name="protractor.js" type="protractor">

describe('SCE doc demo', function() {

it('should sanitize untrusted values', function() {

expect(element(by.css('.htmlComment')).getInnerHtml())

.toBe('<span>Is <i>anyone</i> reading this?</span>');

});

it('should NOT sanitize explicitly trusted values', function() {

expect(element(by.id('explicitlyTrustedHtml')).getInnerHtml()).toBe(

'<span onmouseover="this.textContent=&quot;Explicitly trusted HTML bypasses ' +

'sanitization.&quot;">Hover over this text.</span>');

});

});

</file>

</example>

\*

\*

\*

\* ## Can I disable SCE completely?

\*

\* Yes, you can. However, this is strongly discouraged. SCE gives you a lot of security benefits

\* for little coding overhead. It will be much harder to take an SCE disabled application and

\* either secure it on your own or enable SCE at a later stage. It might make sense to disable SCE

\* for cases where you have a lot of existing code that was written before SCE was introduced and

\* you're migrating them a module at a time.

\*

\* That said, here's how you can completely disable SCE:

\*

\* <pre class="prettyprint">

\* angular.module('myAppWithSceDisabledmyApp', []).config(function($sceProvider) {

\* // Completely disable SCE. For demonstration purposes only!

\* // Do not use in new projects.

\* $sceProvider.enabled(false);

\* });

\* </pre>

\*

\*/

/\* jshint maxlen: 100 \*/

function $SceProvider() {

var enabled = true;

/\*\*

\* @ngdoc method

\* @name $sceProvider#enabled

\* @function

\*

\* @param {boolean=} value If provided, then enables/disables SCE.

\* @return {boolean} true if SCE is enabled, false otherwise.

\*

\* @description

\* Enables/disables SCE and returns the current value.

\*/

this.enabled = function (value) {

if (arguments.length) {

enabled = !!value;

}

return enabled;

};

/\* Design notes on the default implementation for SCE.

\*

\* The API contract for the SCE delegate

\* -------------------------------------

\* The SCE delegate object must provide the following 3 methods:

\*

\* - trustAs(contextEnum, value)

\* This method is used to tell the SCE service that the provided value is OK to use in the

\* contexts specified by contextEnum. It must return an object that will be accepted by

\* getTrusted() for a compatible contextEnum and return this value.

\*

\* - valueOf(value)

\* For values that were not produced by trustAs(), return them as is. For values that were

\* produced by trustAs(), return the corresponding input value to trustAs. Basically, if

\* trustAs is wrapping the given values into some type, this operation unwraps it when given

\* such a value.

\*

\* - getTrusted(contextEnum, value)

\* This function should return the a value that is safe to use in the context specified by

\* contextEnum or throw and exception otherwise.

\*

\* NOTE: This contract deliberately does NOT state that values returned by trustAs() must be

\* opaque or wrapped in some holder object. That happens to be an implementation detail. For

\* instance, an implementation could maintain a registry of all trusted objects by context. In

\* such a case, trustAs() would return the same object that was passed in. getTrusted() would

\* return the same object passed in if it was found in the registry under a compatible context or

\* throw an exception otherwise. An implementation might only wrap values some of the time based

\* on some criteria. getTrusted() might return a value and not throw an exception for special

\* constants or objects even if not wrapped. All such implementations fulfill this contract.

\*

\*

\* A note on the inheritance model for SCE contexts

\* ------------------------------------------------

\* I've used inheritance and made RESOURCE\_URL wrapped types a subtype of URL wrapped types. This

\* is purely an implementation details.

\*

\* The contract is simply this:

\*

\* getTrusted($sce.RESOURCE\_URL, value) succeeding implies that getTrusted($sce.URL, value)

\* will also succeed.

\*

\* Inheritance happens to capture this in a natural way. In some future, we

\* may not use inheritance anymore. That is OK because no code outside of

\* sce.js and sceSpecs.js would need to be aware of this detail.

\*/

this.$get = ['$parse', '$sniffer', '$sceDelegate', function(

$parse, $sniffer, $sceDelegate) {

// Prereq: Ensure that we're not running in IE8 quirks mode. In that mode, IE allows

// the "expression(javascript expression)" syntax which is insecure.

if (enabled && $sniffer.msie && $sniffer.msieDocumentMode < 8) {

throw $sceMinErr('iequirks',

'Strict Contextual Escaping does not support Internet Explorer version < 9 in quirks ' +

'mode. You can fix this by adding the text <!doctype html> to the top of your HTML ' +

'document. See http://docs.angularjs.org/api/ng.$sce for more information.');

}

var sce = copy(SCE\_CONTEXTS);

/\*\*

\* @ngdoc method

\* @name $sce#isEnabled

\* @function

\*

\* @return {Boolean} true if SCE is enabled, false otherwise. If you want to set the value, you

\* have to do it at module config time on {@link ng.$sceProvider $sceProvider}.

\*

\* @description

\* Returns a boolean indicating if SCE is enabled.

\*/

sce.isEnabled = function () {

return enabled;

};

sce.trustAs = $sceDelegate.trustAs;

sce.getTrusted = $sceDelegate.getTrusted;

sce.valueOf = $sceDelegate.valueOf;

if (!enabled) {

sce.trustAs = sce.getTrusted = function(type, value) { return value; };

sce.valueOf = identity;

}

/\*\*

\* @ngdoc method

\* @name $sce#parse

\*

\* @description

\* Converts Angular {@link guide/expression expression} into a function. This is like {@link

\* ng.$parse $parse} and is identical when the expression is a literal constant. Otherwise, it

\* wraps the expression in a call to {@link ng.$sce#getTrusted $sce.getTrusted(\*type\*,

\* \*result\*)}

\*

\* @param {string} type The kind of SCE context in which this result will be used.

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*/

sce.parseAs = function sceParseAs(type, expr) {

var parsed = $parse(expr);

if (parsed.literal && parsed.constant) {

return parsed;

} else {

return function sceParseAsTrusted(self, locals) {

return sce.getTrusted(type, parsed(self, locals));

};

}

};

/\*\*

\* @ngdoc method

\* @name $sce#trustAs

\*

\* @description

\* Delegates to {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs`}. As such,

\* returns an object that is trusted by angular for use in specified strict contextual

\* escaping contexts (such as ng-bind-html, ng-include, any src attribute

\* interpolation, any dom event binding attribute interpolation such as for onclick, etc.)

\* that uses the provided value. See \* {@link ng.$sce $sce} for enabling strict contextual

\* escaping.

\*

\* @param {string} type The kind of context in which this value is safe for use. e.g. url,

\* resource\_url, html, js and css.

\* @param {\*} value The value that that should be considered trusted/safe.

\* @returns {\*} A value that can be used to stand in for the provided `value` in places

\* where Angular expects a $sce.trustAs() return value.

\*/

/\*\*

\* @ngdoc method

\* @name $sce#trustAsHtml

\*

\* @description

\* Shorthand method. `$sce.trustAsHtml(value)` →

\* {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs($sce.HTML, value)`}

\*

\* @param {\*} value The value to trustAs.

\* @returns {\*} An object that can be passed to {@link ng.$sce#getTrustedHtml

\* $sce.getTrustedHtml(value)} to obtain the original value. (privileged directives

\* only accept expressions that are either literal constants or are the

\* return value of {@link ng.$sce#trustAs $sce.trustAs}.)

\*/

/\*\*

\* @ngdoc method

\* @name $sce#trustAsUrl

\*

\* @description

\* Shorthand method. `$sce.trustAsUrl(value)` →

\* {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs($sce.URL, value)`}

\*

\* @param {\*} value The value to trustAs.

\* @returns {\*} An object that can be passed to {@link ng.$sce#getTrustedUrl

\* $sce.getTrustedUrl(value)} to obtain the original value. (privileged directives

\* only accept expressions that are either literal constants or are the

\* return value of {@link ng.$sce#trustAs $sce.trustAs}.)

\*/

/\*\*

\* @ngdoc method

\* @name $sce#trustAsResourceUrl

\*

\* @description

\* Shorthand method. `$sce.trustAsResourceUrl(value)` →

\* {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs($sce.RESOURCE\_URL, value)`}

\*

\* @param {\*} value The value to trustAs.

\* @returns {\*} An object that can be passed to {@link ng.$sce#getTrustedResourceUrl

\* $sce.getTrustedResourceUrl(value)} to obtain the original value. (privileged directives

\* only accept expressions that are either literal constants or are the return

\* value of {@link ng.$sce#trustAs $sce.trustAs}.)

\*/

/\*\*

\* @ngdoc method

\* @name $sce#trustAsJs

\*

\* @description

\* Shorthand method. `$sce.trustAsJs(value)` →

\* {@link ng.$sceDelegate#trustAs `$sceDelegate.trustAs($sce.JS, value)`}

\*

\* @param {\*} value The value to trustAs.

\* @returns {\*} An object that can be passed to {@link ng.$sce#getTrustedJs

\* $sce.getTrustedJs(value)} to obtain the original value. (privileged directives

\* only accept expressions that are either literal constants or are the

\* return value of {@link ng.$sce#trustAs $sce.trustAs}.)

\*/

/\*\*

\* @ngdoc method

\* @name $sce#getTrusted

\*

\* @description

\* Delegates to {@link ng.$sceDelegate#getTrusted `$sceDelegate.getTrusted`}. As such,

\* takes the result of a {@link ng.$sce#trustAs `$sce.trustAs`}() call and returns the

\* originally supplied value if the queried context type is a supertype of the created type.

\* If this condition isn't satisfied, throws an exception.

\*

\* @param {string} type The kind of context in which this value is to be used.

\* @param {\*} maybeTrusted The result of a prior {@link ng.$sce#trustAs `$sce.trustAs`}

\* call.

\* @returns {\*} The value the was originally provided to

\* {@link ng.$sce#trustAs `$sce.trustAs`} if valid in this context.

\* Otherwise, throws an exception.

\*/

/\*\*

\* @ngdoc method

\* @name $sce#getTrustedHtml

\*

\* @description

\* Shorthand method. `$sce.getTrustedHtml(value)` →

\* {@link ng.$sceDelegate#getTrusted `$sceDelegate.getTrusted($sce.HTML, value)`}

\*

\* @param {\*} value The value to pass to `$sce.getTrusted`.

\* @returns {\*} The return value of `$sce.getTrusted($sce.HTML, value)`

\*/

/\*\*

\* @ngdoc method

\* @name $sce#getTrustedCss

\*

\* @description

\* Shorthand method. `$sce.getTrustedCss(value)` →

\* {@link ng.$sceDelegate#getTrusted `$sceDelegate.getTrusted($sce.CSS, value)`}

\*

\* @param {\*} value The value to pass to `$sce.getTrusted`.

\* @returns {\*} The return value of `$sce.getTrusted($sce.CSS, value)`

\*/

/\*\*

\* @ngdoc method

\* @name $sce#getTrustedUrl

\*

\* @description

\* Shorthand method. `$sce.getTrustedUrl(value)` →

\* {@link ng.$sceDelegate#getTrusted `$sceDelegate.getTrusted($sce.URL, value)`}

\*

\* @param {\*} value The value to pass to `$sce.getTrusted`.

\* @returns {\*} The return value of `$sce.getTrusted($sce.URL, value)`

\*/

/\*\*

\* @ngdoc method

\* @name $sce#getTrustedResourceUrl

\*

\* @description

\* Shorthand method. `$sce.getTrustedResourceUrl(value)` →

\* {@link ng.$sceDelegate#getTrusted `$sceDelegate.getTrusted($sce.RESOURCE\_URL, value)`}

\*

\* @param {\*} value The value to pass to `$sceDelegate.getTrusted`.

\* @returns {\*} The return value of `$sce.getTrusted($sce.RESOURCE\_URL, value)`

\*/

/\*\*

\* @ngdoc method

\* @name $sce#getTrustedJs

\*

\* @description

\* Shorthand method. `$sce.getTrustedJs(value)` →

\* {@link ng.$sceDelegate#getTrusted `$sceDelegate.getTrusted($sce.JS, value)`}

\*

\* @param {\*} value The value to pass to `$sce.getTrusted`.

\* @returns {\*} The return value of `$sce.getTrusted($sce.JS, value)`

\*/

/\*\*

\* @ngdoc method

\* @name $sce#parseAsHtml

\*

\* @description

\* Shorthand method. `$sce.parseAsHtml(expression string)` →

\* {@link ng.$sce#parse `$sce.parseAs($sce.HTML, value)`}

\*

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*/

/\*\*

\* @ngdoc method

\* @name $sce#parseAsCss

\*

\* @description

\* Shorthand method. `$sce.parseAsCss(value)` →

\* {@link ng.$sce#parse `$sce.parseAs($sce.CSS, value)`}

\*

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*/

/\*\*

\* @ngdoc method

\* @name $sce#parseAsUrl

\*

\* @description

\* Shorthand method. `$sce.parseAsUrl(value)` →

\* {@link ng.$sce#parse `$sce.parseAs($sce.URL, value)`}

\*

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*/

/\*\*

\* @ngdoc method

\* @name $sce#parseAsResourceUrl

\*

\* @description

\* Shorthand method. `$sce.parseAsResourceUrl(value)` →

\* {@link ng.$sce#parse `$sce.parseAs($sce.RESOURCE\_URL, value)`}

\*

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*/

/\*\*

\* @ngdoc method

\* @name $sce#parseAsJs

\*

\* @description

\* Shorthand method. `$sce.parseAsJs(value)` →

\* {@link ng.$sce#parse `$sce.parseAs($sce.JS, value)`}

\*

\* @param {string} expression String expression to compile.

\* @returns {function(context, locals)} a function which represents the compiled expression:

\*

\* \* `context` – `{object}` – an object against which any expressions embedded in the strings

\* are evaluated against (typically a scope object).

\* \* `locals` – `{object=}` – local variables context object, useful for overriding values in

\* `context`.

\*/

// Shorthand delegations.

var parse = sce.parseAs,

getTrusted = sce.getTrusted,

trustAs = sce.trustAs;

forEach(SCE\_CONTEXTS, function (enumValue, name) {

var lName = lowercase(name);

sce[camelCase("parse\_as\_" + lName)] = function (expr) {

return parse(enumValue, expr);

};

sce[camelCase("get\_trusted\_" + lName)] = function (value) {

return getTrusted(enumValue, value);

};

sce[camelCase("trust\_as\_" + lName)] = function (value) {

return trustAs(enumValue, value);

};

});

return sce;

}];

}

/\*\*

\* !!! This is an undocumented "private" service !!!

\*

\* @name $sniffer

\* @requires $window

\* @requires $document

\*

\* @property {boolean} history Does the browser support html5 history api ?

\* @property {boolean} hashchange Does the browser support hashchange event ?

\* @property {boolean} transitions Does the browser support CSS transition events ?

\* @property {boolean} animations Does the browser support CSS animation events ?

\*

\* @description

\* This is very simple implementation of testing browser's features.

\*/

function $SnifferProvider() {

this.$get = ['$window', '$document', function($window, $document) {

var eventSupport = {},

android =

int((/android (\d+)/.exec(lowercase(($window.navigator || {}).userAgent)) || [])[1]),

boxee = /Boxee/i.test(($window.navigator || {}).userAgent),

document = $document[0] || {},

documentMode = document.documentMode,

vendorPrefix,

vendorRegex = /^(Moz|webkit|O|ms)(?=[A-Z])/,

bodyStyle = document.body && document.body.style,

transitions = false,

animations = false,

match;

if (bodyStyle) {

for(var prop in bodyStyle) {

if(match = vendorRegex.exec(prop)) {

vendorPrefix = match[0];

vendorPrefix = vendorPrefix.substr(0, 1).toUpperCase() + vendorPrefix.substr(1);

break;

}

}

if(!vendorPrefix) {

vendorPrefix = ('WebkitOpacity' in bodyStyle) && 'webkit';

}

transitions = !!(('transition' in bodyStyle) || (vendorPrefix + 'Transition' in bodyStyle));

animations = !!(('animation' in bodyStyle) || (vendorPrefix + 'Animation' in bodyStyle));

if (android && (!transitions||!animations)) {

transitions = isString(document.body.style.webkitTransition);

animations = isString(document.body.style.webkitAnimation);

}

}

return {

// Android has history.pushState, but it does not update location correctly

// so let's not use the history API at all.

// http://code.google.com/p/android/issues/detail?id=17471

// https://github.com/angular/angular.js/issues/904

// older webkit browser (533.9) on Boxee box has exactly the same problem as Android has

// so let's not use the history API also

// We are purposefully using `!(android < 4)` to cover the case when `android` is undefined

// jshint -W018

history: !!($window.history && $window.history.pushState && !(android < 4) && !boxee),

// jshint +W018

hashchange: 'onhashchange' in $window &&

// IE8 compatible mode lies

(!documentMode || documentMode > 7),

hasEvent: function(event) {

// IE9 implements 'input' event it's so fubared that we rather pretend that it doesn't have

// it. In particular the event is not fired when backspace or delete key are pressed or

// when cut operation is performed.

if (event == 'input' && msie == 9) return false;

if (isUndefined(eventSupport[event])) {

var divElm = document.createElement('div');

eventSupport[event] = 'on' + event in divElm;

}

return eventSupport[event];

},

csp: csp(),

vendorPrefix: vendorPrefix,

transitions : transitions,

animations : animations,

android: android,

msie : msie,

msieDocumentMode: documentMode

};

}];

}

function $TimeoutProvider() {

this.$get = ['$rootScope', '$browser', '$q', '$exceptionHandler',

function($rootScope, $browser, $q, $exceptionHandler) {

var deferreds = {};

/\*\*

\* @ngdoc service

\* @name $timeout

\*

\* @description

\* Angular's wrapper for `window.setTimeout`. The `fn` function is wrapped into a try/catch

\* block and delegates any exceptions to

\* {@link ng.$exceptionHandler $exceptionHandler} service.

\*

\* The return value of registering a timeout function is a promise, which will be resolved when

\* the timeout is reached and the timeout function is executed.

\*

\* To cancel a timeout request, call `$timeout.cancel(promise)`.

\*

\* In tests you can use {@link ngMock.$timeout `$timeout.flush()`} to

\* synchronously flush the queue of deferred functions.

\*

\* @param {function()} fn A function, whose execution should be delayed.

\* @param {number=} [delay=0] Delay in milliseconds.

\* @param {boolean=} [invokeApply=true] If set to `false` skips model dirty checking, otherwise

\* will invoke `fn` within the {@link ng.$rootScope.Scope#$apply $apply} block.

\* @returns {Promise} Promise that will be resolved when the timeout is reached. The value this

\* promise will be resolved with is the return value of the `fn` function.

\*

\*/

function timeout(fn, delay, invokeApply) {

var deferred = $q.defer(),

promise = deferred.promise,

skipApply = (isDefined(invokeApply) && !invokeApply),

timeoutId;

timeoutId = $browser.defer(function() {

try {

deferred.resolve(fn());

} catch(e) {

deferred.reject(e);

$exceptionHandler(e);

}

finally {

delete deferreds[promise.$$timeoutId];

}

if (!skipApply) $rootScope.$apply();

}, delay);

promise.$$timeoutId = timeoutId;

deferreds[timeoutId] = deferred;

return promise;

}

/\*\*

\* @ngdoc method

\* @name $timeout#cancel

\*

\* @description

\* Cancels a task associated with the `promise`. As a result of this, the promise will be

\* resolved with a rejection.

\*

\* @param {Promise=} promise Promise returned by the `$timeout` function.

\* @returns {boolean} Returns `true` if the task hasn't executed yet and was successfully

\* canceled.

\*/

timeout.cancel = function(promise) {

if (promise && promise.$$timeoutId in deferreds) {

deferreds[promise.$$timeoutId].reject('canceled');

delete deferreds[promise.$$timeoutId];

return $browser.defer.cancel(promise.$$timeoutId);

}

return false;

};

return timeout;

}];

}

// NOTE: The usage of window and document instead of $window and $document here is

// deliberate. This service depends on the specific behavior of anchor nodes created by the

// browser (resolving and parsing URLs) that is unlikely to be provided by mock objects and

// cause us to break tests. In addition, when the browser resolves a URL for XHR, it

// doesn't know about mocked locations and resolves URLs to the real document - which is

// exactly the behavior needed here. There is little value is mocking these out for this

// service.

var urlParsingNode = document.createElement("a");

var originUrl = urlResolve(window.location.href, true);

/\*\*

\*

\* Implementation Notes for non-IE browsers

\* ----------------------------------------

\* Assigning a URL to the href property of an anchor DOM node, even one attached to the DOM,

\* results both in the normalizing and parsing of the URL. Normalizing means that a relative

\* URL will be resolved into an absolute URL in the context of the application document.

\* Parsing means that the anchor node's host, hostname, protocol, port, pathname and related

\* properties are all populated to reflect the normalized URL. This approach has wide

\* compatibility - Safari 1+, Mozilla 1+, Opera 7+,e etc. See

\* http://www.aptana.com/reference/html/api/HTMLAnchorElement.html

\*

\* Implementation Notes for IE

\* ---------------------------

\* IE >= 8 and <= 10 normalizes the URL when assigned to the anchor node similar to the other

\* browsers. However, the parsed components will not be set if the URL assigned did not specify

\* them. (e.g. if you assign a.href = "foo", then a.protocol, a.host, etc. will be empty.) We

\* work around that by performing the parsing in a 2nd step by taking a previously normalized

\* URL (e.g. by assigning to a.href) and assigning it a.href again. This correctly populates the

\* properties such as protocol, hostname, port, etc.

\*

\* IE7 does not normalize the URL when assigned to an anchor node. (Apparently, it does, if one

\* uses the inner HTML approach to assign the URL as part of an HTML snippet -

\* http://stackoverflow.com/a/472729) However, setting img[src] does normalize the URL.

\* Unfortunately, setting img[src] to something like "javascript:foo" on IE throws an exception.

\* Since the primary usage for normalizing URLs is to sanitize such URLs, we can't use that

\* method and IE < 8 is unsupported.

\*

\* References:

\* http://developer.mozilla.org/en-US/docs/Web/API/HTMLAnchorElement

\* http://www.aptana.com/reference/html/api/HTMLAnchorElement.html

\* http://url.spec.whatwg.org/#urlutils

\* https://github.com/angular/angular.js/pull/2902

\* http://james.padolsey.com/javascript/parsing-urls-with-the-dom/

\*

\* @function

\* @param {string} url The URL to be parsed.

\* @description Normalizes and parses a URL.

\* @returns {object} Returns the normalized URL as a dictionary.

\*

\* | member name | Description |

\* |---------------|----------------|

\* | href | A normalized version of the provided URL if it was not an absolute URL |

\* | protocol | The protocol including the trailing colon |

\* | host | The host and port (if the port is non-default) of the normalizedUrl |

\* | search | The search params, minus the question mark |

\* | hash | The hash string, minus the hash symbol

\* | hostname | The hostname

\* | port | The port, without ":"

\* | pathname | The pathname, beginning with "/"

\*

\*/

function urlResolve(url, base) {

var href = url;

if (msie) {

// Normalize before parse. Refer Implementation Notes on why this is

// done in two steps on IE.

urlParsingNode.setAttribute("href", href);

href = urlParsingNode.href;

}

urlParsingNode.setAttribute('href', href);

// urlParsingNode provides the UrlUtils interface - http://url.spec.whatwg.org/#urlutils

return {

href: urlParsingNode.href,

protocol: urlParsingNode.protocol ? urlParsingNode.protocol.replace(/:$/, '') : '',

host: urlParsingNode.host,

search: urlParsingNode.search ? urlParsingNode.search.replace(/^\?/, '') : '',

hash: urlParsingNode.hash ? urlParsingNode.hash.replace(/^#/, '') : '',

hostname: urlParsingNode.hostname,

port: urlParsingNode.port,

pathname: (urlParsingNode.pathname.charAt(0) === '/')

? urlParsingNode.pathname

: '/' + urlParsingNode.pathname

};

}

/\*\*

\* Parse a request URL and determine whether this is a same-origin request as the application document.

\*

\* @param {string|object} requestUrl The url of the request as a string that will be resolved

\* or a parsed URL object.

\* @returns {boolean} Whether the request is for the same origin as the application document.

\*/

function urlIsSameOrigin(requestUrl) {

var parsed = (isString(requestUrl)) ? urlResolve(requestUrl) : requestUrl;

return (parsed.protocol === originUrl.protocol &&

parsed.host === originUrl.host);

}

/\*\*

\* @ngdoc service

\* @name $window

\*

\* @description

\* A reference to the browser's `window` object. While `window`

\* is globally available in JavaScript, it causes testability problems, because

\* it is a global variable. In angular we always refer to it through the

\* `$window` service, so it may be overridden, removed or mocked for testing.

\*

\* Expressions, like the one defined for the `ngClick` directive in the example

\* below, are evaluated with respect to the current scope. Therefore, there is

\* no risk of inadvertently coding in a dependency on a global value in such an

\* expression.

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope, $window) {

$scope.greeting = 'Hello, World!';

$scope.doGreeting = function(greeting) {

$window.alert(greeting);

};

}

</script>

<div ng-controller="Ctrl">

<input type="text" ng-model="greeting" />

<button ng-click="doGreeting(greeting)">ALERT</button>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should display the greeting in the input box', function() {

element(by.model('greeting')).sendKeys('Hello, E2E Tests');

// If we click the button it will block the test runner

// element(':button').click();

});

</file>

</example>

\*/

function $WindowProvider(){

this.$get = valueFn(window);

}

/\*\*

\* @ngdoc provider

\* @name $filterProvider

\* @description

\*

\* Filters are just functions which transform input to an output. However filters need to be

\* Dependency Injected. To achieve this a filter definition consists of a factory function which is

\* annotated with dependencies and is responsible for creating a filter function.

\*

\* ```js

\* // Filter registration

\* function MyModule($provide, $filterProvider) {

\* // create a service to demonstrate injection (not always needed)

\* $provide.value('greet', function(name){

\* return 'Hello ' + name + '!';

\* });

\*

\* // register a filter factory which uses the

\* // greet service to demonstrate DI.

\* $filterProvider.register('greet', function(greet){

\* // return the filter function which uses the greet service

\* // to generate salutation

\* return function(text) {

\* // filters need to be forgiving so check input validity

\* return text && greet(text) || text;

\* };

\* });

\* }

\* ```

\*

\* The filter function is registered with the `$injector` under the filter name suffix with

\* `Filter`.

\*

\* ```js

\* it('should be the same instance', inject(

\* function($filterProvider) {

\* $filterProvider.register('reverse', function(){

\* return ...;

\* });

\* },

\* function($filter, reverseFilter) {

\* expect($filter('reverse')).toBe(reverseFilter);

\* });

\* ```

\*

\*

\* For more information about how angular filters work, and how to create your own filters, see

\* {@link guide/filter Filters} in the Angular Developer Guide.

\*/

/\*\*

\* @ngdoc method

\* @name $filterProvider#register

\* @description

\* Register filter factory function.

\*

\* @param {String} name Name of the filter.

\* @param {Function} fn The filter factory function which is injectable.

\*/

/\*\*

\* @ngdoc service

\* @name $filter

\* @function

\* @description

\* Filters are used for formatting data displayed to the user.

\*

\* The general syntax in templates is as follows:

\*

\* {{ expression [| filter\_name[:parameter\_value] ... ] }}

\*

\* @param {String} name Name of the filter function to retrieve

\* @return {Function} the filter function

\*/

$FilterProvider.$inject = ['$provide'];

function $FilterProvider($provide) {

var suffix = 'Filter';

/\*\*

\* @ngdoc method

\* @name $controllerProvider#register

\* @param {string|Object} name Name of the filter function, or an object map of filters where

\* the keys are the filter names and the values are the filter factories.

\* @returns {Object} Registered filter instance, or if a map of filters was provided then a map

\* of the registered filter instances.

\*/

function register(name, factory) {

if(isObject(name)) {

var filters = {};

forEach(name, function(filter, key) {

filters[key] = register(key, filter);

});

return filters;

} else {

return $provide.factory(name + suffix, factory);

}

}

this.register = register;

this.$get = ['$injector', function($injector) {

return function(name) {

return $injector.get(name + suffix);

};

}];

////////////////////////////////////////

/\* global

currencyFilter: false,

dateFilter: false,

filterFilter: false,

jsonFilter: false,

limitToFilter: false,

lowercaseFilter: false,

numberFilter: false,

orderByFilter: false,

uppercaseFilter: false,

\*/

register('currency', currencyFilter);

register('date', dateFilter);

register('filter', filterFilter);

register('json', jsonFilter);

register('limitTo', limitToFilter);

register('lowercase', lowercaseFilter);

register('number', numberFilter);

register('orderBy', orderByFilter);

register('uppercase', uppercaseFilter);

}

/\*\*

\* @ngdoc filter

\* @name filter

\* @function

\*

\* @description

\* Selects a subset of items from `array` and returns it as a new array.

\*

\* @param {Array} array The source array.

\* @param {string|Object|function()} expression The predicate to be used for selecting items from

\* `array`.

\*

\* Can be one of:

\*

\* - `string`: The string is evaluated as an expression and the resulting value is used for substring match against

\* the contents of the `array`. All strings or objects with string properties in `array` that contain this string

\* will be returned. The predicate can be negated by prefixing the string with `!`.

\*

\* - `Object`: A pattern object can be used to filter specific properties on objects contained

\* by `array`. For example `{name:"M", phone:"1"}` predicate will return an array of items

\* which have property `name` containing "M" and property `phone` containing "1". A special

\* property name `$` can be used (as in `{$:"text"}`) to accept a match against any

\* property of the object. That's equivalent to the simple substring match with a `string`

\* as described above.

\*

\* - `function(value)`: A predicate function can be used to write arbitrary filters. The function is

\* called for each element of `array`. The final result is an array of those elements that

\* the predicate returned true for.

\*

\* @param {function(actual, expected)|true|undefined} comparator Comparator which is used in

\* determining if the expected value (from the filter expression) and actual value (from

\* the object in the array) should be considered a match.

\*

\* Can be one of:

\*

\* - `function(actual, expected)`:

\* The function will be given the object value and the predicate value to compare and

\* should return true if the item should be included in filtered result.

\*

\* - `true`: A shorthand for `function(actual, expected) { return angular.equals(expected, actual)}`.

\* this is essentially strict comparison of expected and actual.

\*

\* - `false|undefined`: A short hand for a function which will look for a substring match in case

\* insensitive way.

\*

\* @example

<example>

<file name="index.html">

<div ng-init="friends = [{name:'John', phone:'555-1276'},

{name:'Mary', phone:'800-BIG-MARY'},

{name:'Mike', phone:'555-4321'},

{name:'Adam', phone:'555-5678'},

{name:'Julie', phone:'555-8765'},

{name:'Juliette', phone:'555-5678'}]"></div>

Search: <input ng-model="searchText">

<table id="searchTextResults">

<tr><th>Name</th><th>Phone</th></tr>

<tr ng-repeat="friend in friends | filter:searchText">

<td>{{friend.name}}</td>

<td>{{friend.phone}}</td>

</tr>

</table>

<hr>

Any: <input ng-model="search.$"> <br>

Name only <input ng-model="search.name"><br>

Phone only <input ng-model="search.phone"><br>

Equality <input type="checkbox" ng-model="strict"><br>

<table id="searchObjResults">

<tr><th>Name</th><th>Phone</th></tr>

<tr ng-repeat="friendObj in friends | filter:search:strict">

<td>{{friendObj.name}}</td>

<td>{{friendObj.phone}}</td>

</tr>

</table>

</file>

<file name="protractor.js" type="protractor">

var expectFriendNames = function(expectedNames, key) {

element.all(by.repeater(key + ' in friends').column(key + '.name')).then(function(arr) {

arr.forEach(function(wd, i) {

expect(wd.getText()).toMatch(expectedNames[i]);

});

});

};

it('should search across all fields when filtering with a string', function() {

var searchText = element(by.model('searchText'));

searchText.clear();

searchText.sendKeys('m');

expectFriendNames(['Mary', 'Mike', 'Adam'], 'friend');

searchText.clear();

searchText.sendKeys('76');

expectFriendNames(['John', 'Julie'], 'friend');

});

it('should search in specific fields when filtering with a predicate object', function() {

var searchAny = element(by.model('search.$'));

searchAny.clear();

searchAny.sendKeys('i');

expectFriendNames(['Mary', 'Mike', 'Julie', 'Juliette'], 'friendObj');

});

it('should use a equal comparison when comparator is true', function() {

var searchName = element(by.model('search.name'));

var strict = element(by.model('strict'));

searchName.clear();

searchName.sendKeys('Julie');

strict.click();

expectFriendNames(['Julie'], 'friendObj');

});

</file>

</example>

\*/

function filterFilter() {

return function(array, expression, comparator) {

if (!isArray(array)) return array;

var comparatorType = typeof(comparator),

predicates = [];

predicates.check = function(value) {

for (var j = 0; j < predicates.length; j++) {

if(!predicates[j](value)) {

return false;

}

}

return true;

};

if (comparatorType !== 'function') {

if (comparatorType === 'boolean' && comparator) {

comparator = function(obj, text) {

return angular.equals(obj, text);

};

} else {

comparator = function(obj, text) {

if (obj && text && typeof obj === 'object' && typeof text === 'object') {

for (var objKey in obj) {

if (objKey.charAt(0) !== '$' && hasOwnProperty.call(obj, objKey) &&

comparator(obj[objKey], text[objKey])) {

return true;

}

}

return false;

}

text = (''+text).toLowerCase();

return (''+obj).toLowerCase().indexOf(text) > -1;

};

}

}

var search = function(obj, text){

if (typeof text == 'string' && text.charAt(0) === '!') {

return !search(obj, text.substr(1));

}

switch (typeof obj) {

case "boolean":

case "number":

case "string":

return comparator(obj, text);

case "object":

switch (typeof text) {

case "object":

return comparator(obj, text);

default:

for ( var objKey in obj) {

if (objKey.charAt(0) !== '$' && search(obj[objKey], text)) {

return true;

}

}

break;

}

return false;

case "array":

for ( var i = 0; i < obj.length; i++) {

if (search(obj[i], text)) {

return true;

}

}

return false;

default:

return false;

}

};

switch (typeof expression) {

case "boolean":

case "number":

case "string":

// Set up expression object and fall through

expression = {$:expression};

// jshint -W086

case "object":

// jshint +W086

for (var key in expression) {

(function(path) {

if (typeof expression[path] == 'undefined') return;

predicates.push(function(value) {

return search(path == '$' ? value : (value && value[path]), expression[path]);

});

})(key);

}

break;

case 'function':

predicates.push(expression);

break;

default:

return array;

}

var filtered = [];

for ( var j = 0; j < array.length; j++) {

var value = array[j];

if (predicates.check(value)) {

filtered.push(value);

}

}

return filtered;

};

}

/\*\*

\* @ngdoc filter

\* @name currency

\* @function

\*

\* @description

\* Formats a number as a currency (ie $1,234.56). When no currency symbol is provided, default

\* symbol for current locale is used.

\*

\* @param {number} amount Input to filter.

\* @param {string=} symbol Currency symbol or identifier to be displayed.

\* @returns {string} Formatted number.

\*

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.amount = 1234.56;

}

</script>

<div ng-controller="Ctrl">

<input type="number" ng-model="amount"> <br>

default currency symbol ($): <span id="currency-default">{{amount | currency}}</span><br>

custom currency identifier (USD$): <span>{{amount | currency:"USD$"}}</span>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should init with 1234.56', function() {

expect(element(by.id('currency-default')).getText()).toBe('$1,234.56');

expect(element(by.binding('amount | currency:"USD$"')).getText()).toBe('USD$1,234.56');

});

it('should update', function() {

if (browser.params.browser == 'safari') {

// Safari does not understand the minus key. See

// https://github.com/angular/protractor/issues/481

return;

}

element(by.model('amount')).clear();

element(by.model('amount')).sendKeys('-1234');

expect(element(by.id('currency-default')).getText()).toBe('($1,234.00)');

expect(element(by.binding('amount | currency:"USD$"')).getText()).toBe('(USD$1,234.00)');

});

</file>

</example>

\*/

currencyFilter.$inject = ['$locale'];

function currencyFilter($locale) {

var formats = $locale.NUMBER\_FORMATS;

return function(amount, currencySymbol){

if (isUndefined(currencySymbol)) currencySymbol = formats.CURRENCY\_SYM;

return formatNumber(amount, formats.PATTERNS[1], formats.GROUP\_SEP, formats.DECIMAL\_SEP, 2).

replace(/\u00A4/g, currencySymbol);

};

}

/\*\*

\* @ngdoc filter

\* @name number

\* @function

\*

\* @description

\* Formats a number as text.

\*

\* If the input is not a number an empty string is returned.

\*

\* @param {number|string} number Number to format.

\* @param {(number|string)=} fractionSize Number of decimal places to round the number to.

\* If this is not provided then the fraction size is computed from the current locale's number

\* formatting pattern. In the case of the default locale, it will be 3.

\* @returns {string} Number rounded to decimalPlaces and places a “,” after each third digit.

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.val = 1234.56789;

}

</script>

<div ng-controller="Ctrl">

Enter number: <input ng-model='val'><br>

Default formatting: <span id='number-default'>{{val | number}}</span><br>

No fractions: <span>{{val | number:0}}</span><br>

Negative number: <span>{{-val | number:4}}</span>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should format numbers', function() {

expect(element(by.id('number-default')).getText()).toBe('1,234.568');

expect(element(by.binding('val | number:0')).getText()).toBe('1,235');

expect(element(by.binding('-val | number:4')).getText()).toBe('-1,234.5679');

});

it('should update', function() {

element(by.model('val')).clear();

element(by.model('val')).sendKeys('3374.333');

expect(element(by.id('number-default')).getText()).toBe('3,374.333');

expect(element(by.binding('val | number:0')).getText()).toBe('3,374');

expect(element(by.binding('-val | number:4')).getText()).toBe('-3,374.3330');

});

</file>

</example>

\*/

numberFilter.$inject = ['$locale'];

function numberFilter($locale) {

var formats = $locale.NUMBER\_FORMATS;

return function(number, fractionSize) {

return formatNumber(number, formats.PATTERNS[0], formats.GROUP\_SEP, formats.DECIMAL\_SEP,

fractionSize);

};

}

var DECIMAL\_SEP = '.';

function formatNumber(number, pattern, groupSep, decimalSep, fractionSize) {

if (number == null || !isFinite(number) || isObject(number)) return '';

var isNegative = number < 0;

number = Math.abs(number);

var numStr = number + '',

formatedText = '',

parts = [];

var hasExponent = false;

if (numStr.indexOf('e') !== -1) {

var match = numStr.match(/([\d\.]+)e(-?)(\d+)/);

if (match && match[2] == '-' && match[3] > fractionSize + 1) {

numStr = '0';

} else {

formatedText = numStr;

hasExponent = true;

}

}

if (!hasExponent) {

var fractionLen = (numStr.split(DECIMAL\_SEP)[1] || '').length;

// determine fractionSize if it is not specified

if (isUndefined(fractionSize)) {

fractionSize = Math.min(Math.max(pattern.minFrac, fractionLen), pattern.maxFrac);

}

var pow = Math.pow(10, fractionSize);

number = Math.round(number \* pow) / pow;

var fraction = ('' + number).split(DECIMAL\_SEP);

var whole = fraction[0];

fraction = fraction[1] || '';

var i, pos = 0,

lgroup = pattern.lgSize,

group = pattern.gSize;

if (whole.length >= (lgroup + group)) {

pos = whole.length - lgroup;

for (i = 0; i < pos; i++) {

if ((pos - i)%group === 0 && i !== 0) {

formatedText += groupSep;

}

formatedText += whole.charAt(i);

}

}

for (i = pos; i < whole.length; i++) {

if ((whole.length - i)%lgroup === 0 && i !== 0) {

formatedText += groupSep;

}

formatedText += whole.charAt(i);

}

// format fraction part.

while(fraction.length < fractionSize) {

fraction += '0';

}

if (fractionSize && fractionSize !== "0") formatedText += decimalSep + fraction.substr(0, fractionSize);

} else {

if (fractionSize > 0 && number > -1 && number < 1) {

formatedText = number.toFixed(fractionSize);

}

}

parts.push(isNegative ? pattern.negPre : pattern.posPre);

parts.push(formatedText);

parts.push(isNegative ? pattern.negSuf : pattern.posSuf);

return parts.join('');

}

function padNumber(num, digits, trim) {

var neg = '';

if (num < 0) {

neg = '-';

num = -num;

}

num = '' + num;

while(num.length < digits) num = '0' + num;

if (trim)

num = num.substr(num.length - digits);

return neg + num;

}

function dateGetter(name, size, offset, trim) {

offset = offset || 0;

return function(date) {

var value = date['get' + name]();

if (offset > 0 || value > -offset)

value += offset;

if (value === 0 && offset == -12 ) value = 12;

return padNumber(value, size, trim);

};

}

function dateStrGetter(name, shortForm) {

return function(date, formats) {

var value = date['get' + name]();

var get = uppercase(shortForm ? ('SHORT' + name) : name);

return formats[get][value];

};

}

function timeZoneGetter(date) {

var zone = -1 \* date.getTimezoneOffset();

var paddedZone = (zone >= 0) ? "+" : "";

paddedZone += padNumber(Math[zone > 0 ? 'floor' : 'ceil'](zone / 60), 2) +

padNumber(Math.abs(zone % 60), 2);

return paddedZone;

}

function ampmGetter(date, formats) {

return date.getHours() < 12 ? formats.AMPMS[0] : formats.AMPMS[1];

}

var DATE\_FORMATS = {

yyyy: dateGetter('FullYear', 4),

yy: dateGetter('FullYear', 2, 0, true),

y: dateGetter('FullYear', 1),

MMMM: dateStrGetter('Month'),

MMM: dateStrGetter('Month', true),

MM: dateGetter('Month', 2, 1),

M: dateGetter('Month', 1, 1),

dd: dateGetter('Date', 2),

d: dateGetter('Date', 1),

HH: dateGetter('Hours', 2),

H: dateGetter('Hours', 1),

hh: dateGetter('Hours', 2, -12),

h: dateGetter('Hours', 1, -12),

mm: dateGetter('Minutes', 2),

m: dateGetter('Minutes', 1),

ss: dateGetter('Seconds', 2),

s: dateGetter('Seconds', 1),

// while ISO 8601 requires fractions to be prefixed with `.` or `,`

// we can be just safely rely on using `sss` since we currently don't support single or two digit fractions

sss: dateGetter('Milliseconds', 3),

EEEE: dateStrGetter('Day'),

EEE: dateStrGetter('Day', true),

a: ampmGetter,

Z: timeZoneGetter

};

var DATE\_FORMATS\_SPLIT = /((?:[^yMdHhmsaZE']+)|(?:'(?:[^']|'')\*')|(?:E+|y+|M+|d+|H+|h+|m+|s+|a|Z))(.\*)/,

NUMBER\_STRING = /^\-?\d+$/;

/\*\*

\* @ngdoc filter

\* @name date

\* @function

\*

\* @description

\* Formats `date` to a string based on the requested `format`.

\*

\* `format` string can be composed of the following elements:

\*

\* \* `'yyyy'`: 4 digit representation of year (e.g. AD 1 => 0001, AD 2010 => 2010)

\* \* `'yy'`: 2 digit representation of year, padded (00-99). (e.g. AD 2001 => 01, AD 2010 => 10)

\* \* `'y'`: 1 digit representation of year, e.g. (AD 1 => 1, AD 199 => 199)

\* \* `'MMMM'`: Month in year (January-December)

\* \* `'MMM'`: Month in year (Jan-Dec)

\* \* `'MM'`: Month in year, padded (01-12)

\* \* `'M'`: Month in year (1-12)

\* \* `'dd'`: Day in month, padded (01-31)

\* \* `'d'`: Day in month (1-31)

\* \* `'EEEE'`: Day in Week,(Sunday-Saturday)

\* \* `'EEE'`: Day in Week, (Sun-Sat)

\* \* `'HH'`: Hour in day, padded (00-23)

\* \* `'H'`: Hour in day (0-23)

\* \* `'hh'`: Hour in am/pm, padded (01-12)

\* \* `'h'`: Hour in am/pm, (1-12)

\* \* `'mm'`: Minute in hour, padded (00-59)

\* \* `'m'`: Minute in hour (0-59)

\* \* `'ss'`: Second in minute, padded (00-59)

\* \* `'s'`: Second in minute (0-59)

\* \* `'.sss' or ',sss'`: Millisecond in second, padded (000-999)

\* \* `'a'`: am/pm marker

\* \* `'Z'`: 4 digit (+sign) representation of the timezone offset (-1200-+1200)

\*

\* `format` string can also be one of the following predefined

\* {@link guide/i18n localizable formats}:

\*

\* \* `'medium'`: equivalent to `'MMM d, y h:mm:ss a'` for en\_US locale

\* (e.g. Sep 3, 2010 12:05:08 pm)

\* \* `'short'`: equivalent to `'M/d/yy h:mm a'` for en\_US locale (e.g. 9/3/10 12:05 pm)

\* \* `'fullDate'`: equivalent to `'EEEE, MMMM d,y'` for en\_US locale

\* (e.g. Friday, September 3, 2010)

\* \* `'longDate'`: equivalent to `'MMMM d, y'` for en\_US locale (e.g. September 3, 2010)

\* \* `'mediumDate'`: equivalent to `'MMM d, y'` for en\_US locale (e.g. Sep 3, 2010)

\* \* `'shortDate'`: equivalent to `'M/d/yy'` for en\_US locale (e.g. 9/3/10)

\* \* `'mediumTime'`: equivalent to `'h:mm:ss a'` for en\_US locale (e.g. 12:05:08 pm)

\* \* `'shortTime'`: equivalent to `'h:mm a'` for en\_US locale (e.g. 12:05 pm)

\*

\* `format` string can contain literal values. These need to be quoted with single quotes (e.g.

\* `"h 'in the morning'"`). In order to output single quote, use two single quotes in a sequence

\* (e.g. `"h 'o''clock'"`).

\*

\* @param {(Date|number|string)} date Date to format either as Date object, milliseconds (string or

\* number) or various ISO 8601 datetime string formats (e.g. yyyy-MM-ddTHH:mm:ss.SSSZ and its

\* shorter versions like yyyy-MM-ddTHH:mmZ, yyyy-MM-dd or yyyyMMddTHHmmssZ). If no timezone is

\* specified in the string input, the time is considered to be in the local timezone.

\* @param {string=} format Formatting rules (see Description). If not specified,

\* `mediumDate` is used.

\* @returns {string} Formatted string or the input if input is not recognized as date/millis.

\*

\* @example

<example>

<file name="index.html">

<span ng-non-bindable>{{1288323623006 | date:'medium'}}</span>:

<span>{{1288323623006 | date:'medium'}}</span><br>

<span ng-non-bindable>{{1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'}}</span>:

<span>{{1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'}}</span><br>

<span ng-non-bindable>{{1288323623006 | date:'MM/dd/yyyy @ h:mma'}}</span>:

<span>{{'1288323623006' | date:'MM/dd/yyyy @ h:mma'}}</span><br>

</file>

<file name="protractor.js" type="protractor">

it('should format date', function() {

expect(element(by.binding("1288323623006 | date:'medium'")).getText()).

toMatch(/Oct 2\d, 2010 \d{1,2}:\d{2}:\d{2} (AM|PM)/);

expect(element(by.binding("1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'")).getText()).

toMatch(/2010\-10\-2\d \d{2}:\d{2}:\d{2} (\-|\+)?\d{4}/);

expect(element(by.binding("'1288323623006' | date:'MM/dd/yyyy @ h:mma'")).getText()).

toMatch(/10\/2\d\/2010 @ \d{1,2}:\d{2}(AM|PM)/);

});

</file>

</example>

\*/

dateFilter.$inject = ['$locale'];

function dateFilter($locale) {

var R\_ISO8601\_STR = /^(\d{4})-?(\d\d)-?(\d\d)(?:T(\d\d)(?::?(\d\d)(?::?(\d\d)(?:\.(\d+))?)?)?(Z|([+-])(\d\d):?(\d\d))?)?$/;

// 1 2 3 4 5 6 7 8 9 10 11

function jsonStringToDate(string) {

var match;

if (match = string.match(R\_ISO8601\_STR)) {

var date = new Date(0),

tzHour = 0,

tzMin = 0,

dateSetter = match[8] ? date.setUTCFullYear : date.setFullYear,

timeSetter = match[8] ? date.setUTCHours : date.setHours;

if (match[9]) {

tzHour = int(match[9] + match[10]);

tzMin = int(match[9] + match[11]);

}

dateSetter.call(date, int(match[1]), int(match[2]) - 1, int(match[3]));

var h = int(match[4]||0) - tzHour;

var m = int(match[5]||0) - tzMin;

var s = int(match[6]||0);

var ms = Math.round(parseFloat('0.' + (match[7]||0)) \* 1000);

timeSetter.call(date, h, m, s, ms);

return date;

}

return string;

}

return function(date, format) {

var text = '',

parts = [],

fn, match;

format = format || 'mediumDate';

format = $locale.DATETIME\_FORMATS[format] || format;

if (isString(date)) {

if (NUMBER\_STRING.test(date)) {

date = int(date);

} else {

date = jsonStringToDate(date);

}

}

if (isNumber(date)) {

date = new Date(date);

}

if (!isDate(date)) {

return date;

}

while(format) {

match = DATE\_FORMATS\_SPLIT.exec(format);

if (match) {

parts = concat(parts, match, 1);

format = parts.pop();

} else {

parts.push(format);

format = null;

}

}

forEach(parts, function(value){

fn = DATE\_FORMATS[value];

text += fn ? fn(date, $locale.DATETIME\_FORMATS)

: value.replace(/(^'|'$)/g, '').replace(/''/g, "'");

});

return text;

};

}

/\*\*

\* @ngdoc filter

\* @name json

\* @function

\*

\* @description

\* Allows you to convert a JavaScript object into JSON string.

\*

\* This filter is mostly useful for debugging. When using the double curly {{value}} notation

\* the binding is automatically converted to JSON.

\*

\* @param {\*} object Any JavaScript object (including arrays and primitive types) to filter.

\* @returns {string} JSON string.

\*

\*

\* @example

<example>

<file name="index.html">

<pre>{{ {'name':'value'} | json }}</pre>

</file>

<file name="protractor.js" type="protractor">

it('should jsonify filtered objects', function() {

expect(element(by.binding("{'name':'value'}")).getText()).toMatch(/\{\n "name": ?"value"\n}/);

});

</file>

</example>

\*

\*/

function jsonFilter() {

return function(object) {

return toJson(object, true);

};

}

/\*\*

\* @ngdoc filter

\* @name lowercase

\* @function

\* @description

\* Converts string to lowercase.

\* @see angular.lowercase

\*/

var lowercaseFilter = valueFn(lowercase);

/\*\*

\* @ngdoc filter

\* @name uppercase

\* @function

\* @description

\* Converts string to uppercase.

\* @see angular.uppercase

\*/

var uppercaseFilter = valueFn(uppercase);

/\*\*

\* @ngdoc filter

\* @name limitTo

\* @function

\*

\* @description

\* Creates a new array or string containing only a specified number of elements. The elements

\* are taken from either the beginning or the end of the source array or string, as specified by

\* the value and sign (positive or negative) of `limit`.

\*

\* @param {Array|string} input Source array or string to be limited.

\* @param {string|number} limit The length of the returned array or string. If the `limit` number

\* is positive, `limit` number of items from the beginning of the source array/string are copied.

\* If the number is negative, `limit` number of items from the end of the source array/string

\* are copied. The `limit` will be trimmed if it exceeds `array.length`

\* @returns {Array|string} A new sub-array or substring of length `limit` or less if input array

\* had less than `limit` elements.

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.numbers = [1,2,3,4,5,6,7,8,9];

$scope.letters = "abcdefghi";

$scope.numLimit = 3;

$scope.letterLimit = 3;

}

</script>

<div ng-controller="Ctrl">

Limit {{numbers}} to: <input type="integer" ng-model="numLimit">

<p>Output numbers: {{ numbers | limitTo:numLimit }}</p>

Limit {{letters}} to: <input type="integer" ng-model="letterLimit">

<p>Output letters: {{ letters | limitTo:letterLimit }}</p>

</div>

</file>

<file name="protractor.js" type="protractor">

var numLimitInput = element(by.model('numLimit'));

var letterLimitInput = element(by.model('letterLimit'));

var limitedNumbers = element(by.binding('numbers | limitTo:numLimit'));

var limitedLetters = element(by.binding('letters | limitTo:letterLimit'));

it('should limit the number array to first three items', function() {

expect(numLimitInput.getAttribute('value')).toBe('3');

expect(letterLimitInput.getAttribute('value')).toBe('3');

expect(limitedNumbers.getText()).toEqual('Output numbers: [1,2,3]');

expect(limitedLetters.getText()).toEqual('Output letters: abc');

});

it('should update the output when -3 is entered', function() {

numLimitInput.clear();

numLimitInput.sendKeys('-3');

letterLimitInput.clear();

letterLimitInput.sendKeys('-3');

expect(limitedNumbers.getText()).toEqual('Output numbers: [7,8,9]');

expect(limitedLetters.getText()).toEqual('Output letters: ghi');

});

it('should not exceed the maximum size of input array', function() {

numLimitInput.clear();

numLimitInput.sendKeys('100');

letterLimitInput.clear();

letterLimitInput.sendKeys('100');

expect(limitedNumbers.getText()).toEqual('Output numbers: [1,2,3,4,5,6,7,8,9]');

expect(limitedLetters.getText()).toEqual('Output letters: abcdefghi');

});

</file>

</example>

\*/

function limitToFilter(){

return function(input, limit) {

if (!isArray(input) && !isString(input)) return input;

limit = int(limit);

if (isString(input)) {

//NaN check on limit

if (limit) {

return limit >= 0 ? input.slice(0, limit) : input.slice(limit, input.length);

} else {

return "";

}

}

var out = [],

i, n;

// if abs(limit) exceeds maximum length, trim it

if (limit > input.length)

limit = input.length;

else if (limit < -input.length)

limit = -input.length;

if (limit > 0) {

i = 0;

n = limit;

} else {

i = input.length + limit;

n = input.length;

}

for (; i<n; i++) {

out.push(input[i]);

}

return out;

};

}

/\*\*

\* @ngdoc filter

\* @name orderBy

\* @function

\*

\* @description

\* Orders a specified `array` by the `expression` predicate.

\*

\* @param {Array} array The array to sort.

\* @param {function(\*)|string|Array.<(function(\*)|string)>} expression A predicate to be

\* used by the comparator to determine the order of elements.

\*

\* Can be one of:

\*

\* - `function`: Getter function. The result of this function will be sorted using the

\* `<`, `=`, `>` operator.

\* - `string`: An Angular expression which evaluates to an object to order by, such as 'name'

\* to sort by a property called 'name'. Optionally prefixed with `+` or `-` to control

\* ascending or descending sort order (for example, +name or -name).

\* - `Array`: An array of function or string predicates. The first predicate in the array

\* is used for sorting, but when two items are equivalent, the next predicate is used.

\*

\* @param {boolean=} reverse Reverse the order of the array.

\* @returns {Array} Sorted copy of the source array.

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.friends =

[{name:'John', phone:'555-1212', age:10},

{name:'Mary', phone:'555-9876', age:19},

{name:'Mike', phone:'555-4321', age:21},

{name:'Adam', phone:'555-5678', age:35},

{name:'Julie', phone:'555-8765', age:29}]

$scope.predicate = '-age';

}

</script>

<div ng-controller="Ctrl">

<pre>Sorting predicate = {{predicate}}; reverse = {{reverse}}</pre>

<hr/>

[ <a href="" ng-click="predicate=''">unsorted</a> ]

<table class="friend">

<tr>

<th><a href="" ng-click="predicate = 'name'; reverse=false">Name</a>

(<a href="" ng-click="predicate = '-name'; reverse=false">^</a>)</th>

<th><a href="" ng-click="predicate = 'phone'; reverse=!reverse">Phone Number</a></th>

<th><a href="" ng-click="predicate = 'age'; reverse=!reverse">Age</a></th>

</tr>

<tr ng-repeat="friend in friends | orderBy:predicate:reverse">

<td>{{friend.name}}</td>

<td>{{friend.phone}}</td>

<td>{{friend.age}}</td>

</tr>

</table>

</div>

</file>

</example>

\*/

orderByFilter.$inject = ['$parse'];

function orderByFilter($parse){

return function(array, sortPredicate, reverseOrder) {

if (!isArray(array)) return array;

if (!sortPredicate) return array;

sortPredicate = isArray(sortPredicate) ? sortPredicate: [sortPredicate];

sortPredicate = map(sortPredicate, function(predicate){

var descending = false, get = predicate || identity;

if (isString(predicate)) {

if ((predicate.charAt(0) == '+' || predicate.charAt(0) == '-')) {

descending = predicate.charAt(0) == '-';

predicate = predicate.substring(1);

}

get = $parse(predicate);

if (get.constant) {

var key = get();

return reverseComparator(function(a,b) {

return compare(a[key], b[key]);

}, descending);

}

}

return reverseComparator(function(a,b){

return compare(get(a),get(b));

}, descending);

});

var arrayCopy = [];

for ( var i = 0; i < array.length; i++) { arrayCopy.push(array[i]); }

return arrayCopy.sort(reverseComparator(comparator, reverseOrder));

function comparator(o1, o2){

for ( var i = 0; i < sortPredicate.length; i++) {

var comp = sortPredicate[i](o1, o2);

if (comp !== 0) return comp;

}

return 0;

}

function reverseComparator(comp, descending) {

return toBoolean(descending)

? function(a,b){return comp(b,a);}

: comp;

}

function compare(v1, v2){

var t1 = typeof v1;

var t2 = typeof v2;

if (t1 == t2) {

if (t1 == "string") {

v1 = v1.toLowerCase();

v2 = v2.toLowerCase();

}

if (v1 === v2) return 0;

return v1 < v2 ? -1 : 1;

} else {

return t1 < t2 ? -1 : 1;

}

}

};

}

function ngDirective(directive) {

if (isFunction(directive)) {

directive = {

link: directive

};

}

directive.restrict = directive.restrict || 'AC';

return valueFn(directive);

}

/\*\*

\* @ngdoc directive

\* @name a

\* @restrict E

\*

\* @description

\* Modifies the default behavior of the html A tag so that the default action is prevented when

\* the href attribute is empty.

\*

\* This change permits the easy creation of action links with the `ngClick` directive

\* without changing the location or causing page reloads, e.g.:

\* `<a href="" ng-click="list.addItem()">Add Item</a>`

\*/

var htmlAnchorDirective = valueFn({

restrict: 'E',

compile: function(element, attr) {

if (msie <= 8) {

// turn <a href ng-click="..">link</a> into a stylable link in IE

// but only if it doesn't have name attribute, in which case it's an anchor

if (!attr.href && !attr.name) {

attr.$set('href', '');

}

// add a comment node to anchors to workaround IE bug that causes element content to be reset

// to new attribute content if attribute is updated with value containing @ and element also

// contains value with @

// see issue #1949

element.append(document.createComment('IE fix'));

}

if (!attr.href && !attr.xlinkHref && !attr.name) {

return function(scope, element) {

// SVGAElement does not use the href attribute, but rather the 'xlinkHref' attribute.

var href = toString.call(element.prop('href')) === '[object SVGAnimatedString]' ?

'xlink:href' : 'href';

element.on('click', function(event){

// if we have no href url, then don't navigate anywhere.

if (!element.attr(href)) {

event.preventDefault();

}

});

};

}

}

});

/\*\*

\* @ngdoc directive

\* @name ngHref

\* @restrict A

\* @priority 99

\*

\* @description

\* Using Angular markup like `{{hash}}` in an href attribute will

\* make the link go to the wrong URL if the user clicks it before

\* Angular has a chance to replace the `{{hash}}` markup with its

\* value. Until Angular replaces the markup the link will be broken

\* and will most likely return a 404 error.

\*

\* The `ngHref` directive solves this problem.

\*

\* The wrong way to write it:

\* ```html

\* <a href="http://www.gravatar.com/avatar/{{hash}}"/>

\* ```

\*

\* The correct way to write it:

\* ```html

\* <a ng-href="http://www.gravatar.com/avatar/{{hash}}"/>

\* ```

\*

\* @element A

\* @param {template} ngHref any string which can contain `{{}}` markup.

\*

\* @example

\* This example shows various combinations of `href`, `ng-href` and `ng-click` attributes

\* in links and their different behaviors:

<example>

<file name="index.html">

<input ng-model="value" /><br />

<a id="link-1" href ng-click="value = 1">link 1</a> (link, don't reload)<br />

<a id="link-2" href="" ng-click="value = 2">link 2</a> (link, don't reload)<br />

<a id="link-3" ng-href="/{{'123'}}">link 3</a> (link, reload!)<br />

<a id="link-4" href="" name="xx" ng-click="value = 4">anchor</a> (link, don't reload)<br />

<a id="link-5" name="xxx" ng-click="value = 5">anchor</a> (no link)<br />

<a id="link-6" ng-href="{{value}}">link</a> (link, change location)

</file>

<file name="protractor.js" type="protractor">

it('should execute ng-click but not reload when href without value', function() {

element(by.id('link-1')).click();

expect(element(by.model('value')).getAttribute('value')).toEqual('1');

expect(element(by.id('link-1')).getAttribute('href')).toBe('');

});

it('should execute ng-click but not reload when href empty string', function() {

element(by.id('link-2')).click();

expect(element(by.model('value')).getAttribute('value')).toEqual('2');

expect(element(by.id('link-2')).getAttribute('href')).toBe('');

});

it('should execute ng-click and change url when ng-href specified', function() {

expect(element(by.id('link-3')).getAttribute('href')).toMatch(/\/123$/);

element(by.id('link-3')).click();

// At this point, we navigate away from an Angular page, so we need

// to use browser.driver to get the base webdriver.

browser.wait(function() {

return browser.driver.getCurrentUrl().then(function(url) {

return url.match(/\/123$/);

});

}, 1000, 'page should navigate to /123');

});

xit('should execute ng-click but not reload when href empty string and name specified', function() {

element(by.id('link-4')).click();

expect(element(by.model('value')).getAttribute('value')).toEqual('4');

expect(element(by.id('link-4')).getAttribute('href')).toBe('');

});

it('should execute ng-click but not reload when no href but name specified', function() {

element(by.id('link-5')).click();

expect(element(by.model('value')).getAttribute('value')).toEqual('5');

expect(element(by.id('link-5')).getAttribute('href')).toBe(null);

});

it('should only change url when only ng-href', function() {

element(by.model('value')).clear();

element(by.model('value')).sendKeys('6');

expect(element(by.id('link-6')).getAttribute('href')).toMatch(/\/6$/);

element(by.id('link-6')).click();

// At this point, we navigate away from an Angular page, so we need

// to use browser.driver to get the base webdriver.

browser.wait(function() {

return browser.driver.getCurrentUrl().then(function(url) {

return url.match(/\/6$/);

});

}, 1000, 'page should navigate to /6');

});

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngSrc

\* @restrict A

\* @priority 99

\*

\* @description

\* Using Angular markup like `{{hash}}` in a `src` attribute doesn't

\* work right: The browser will fetch from the URL with the literal

\* text `{{hash}}` until Angular replaces the expression inside

\* `{{hash}}`. The `ngSrc` directive solves this problem.

\*

\* The buggy way to write it:

\* ```html

\* <img src="http://www.gravatar.com/avatar/{{hash}}"/>

\* ```

\*

\* The correct way to write it:

\* ```html

\* <img ng-src="http://www.gravatar.com/avatar/{{hash}}"/>

\* ```

\*

\* @element IMG

\* @param {template} ngSrc any string which can contain `{{}}` markup.

\*/

/\*\*

\* @ngdoc directive

\* @name ngSrcset

\* @restrict A

\* @priority 99

\*

\* @description

\* Using Angular markup like `{{hash}}` in a `srcset` attribute doesn't

\* work right: The browser will fetch from the URL with the literal

\* text `{{hash}}` until Angular replaces the expression inside

\* `{{hash}}`. The `ngSrcset` directive solves this problem.

\*

\* The buggy way to write it:

\* ```html

\* <img srcset="http://www.gravatar.com/avatar/{{hash}} 2x"/>

\* ```

\*

\* The correct way to write it:

\* ```html

\* <img ng-srcset="http://www.gravatar.com/avatar/{{hash}} 2x"/>

\* ```

\*

\* @element IMG

\* @param {template} ngSrcset any string which can contain `{{}}` markup.

\*/

/\*\*

\* @ngdoc directive

\* @name ngDisabled

\* @restrict A

\* @priority 100

\*

\* @description

\*

\* The following markup will make the button enabled on Chrome/Firefox but not on IE8 and older IEs:

\* ```html

\* <div ng-init="scope = { isDisabled: false }">

\* <button disabled="{{scope.isDisabled}}">Disabled</button>

\* </div>

\* ```

\*

\* The HTML specification does not require browsers to preserve the values of boolean attributes

\* such as disabled. (Their presence means true and their absence means false.)

\* If we put an Angular interpolation expression into such an attribute then the

\* binding information would be lost when the browser removes the attribute.

\* The `ngDisabled` directive solves this problem for the `disabled` attribute.

\* This complementary directive is not removed by the browser and so provides

\* a permanent reliable place to store the binding information.

\*

\* @example

<example>

<file name="index.html">

Click me to toggle: <input type="checkbox" ng-model="checked"><br/>

<button ng-model="button" ng-disabled="checked">Button</button>

</file>

<file name="protractor.js" type="protractor">

it('should toggle button', function() {

expect(element(by.css('button')).getAttribute('disabled')).toBeFalsy();

element(by.model('checked')).click();

expect(element(by.css('button')).getAttribute('disabled')).toBeTruthy();

});

</file>

</example>

\*

\* @element INPUT

\* @param {expression} ngDisabled If the {@link guide/expression expression} is truthy,

\* then special attribute "disabled" will be set on the element

\*/

/\*\*

\* @ngdoc directive

\* @name ngChecked

\* @restrict A

\* @priority 100

\*

\* @description

\* The HTML specification does not require browsers to preserve the values of boolean attributes

\* such as checked. (Their presence means true and their absence means false.)

\* If we put an Angular interpolation expression into such an attribute then the

\* binding information would be lost when the browser removes the attribute.

\* The `ngChecked` directive solves this problem for the `checked` attribute.

\* This complementary directive is not removed by the browser and so provides

\* a permanent reliable place to store the binding information.

\* @example

<example>

<file name="index.html">

Check me to check both: <input type="checkbox" ng-model="master"><br/>

<input id="checkSlave" type="checkbox" ng-checked="master">

</file>

<file name="protractor.js" type="protractor">

it('should check both checkBoxes', function() {

expect(element(by.id('checkSlave')).getAttribute('checked')).toBeFalsy();

element(by.model('master')).click();

expect(element(by.id('checkSlave')).getAttribute('checked')).toBeTruthy();

});

</file>

</example>

\*

\* @element INPUT

\* @param {expression} ngChecked If the {@link guide/expression expression} is truthy,

\* then special attribute "checked" will be set on the element

\*/

/\*\*

\* @ngdoc directive

\* @name ngReadonly

\* @restrict A

\* @priority 100

\*

\* @description

\* The HTML specification does not require browsers to preserve the values of boolean attributes

\* such as readonly. (Their presence means true and their absence means false.)

\* If we put an Angular interpolation expression into such an attribute then the

\* binding information would be lost when the browser removes the attribute.

\* The `ngReadonly` directive solves this problem for the `readonly` attribute.

\* This complementary directive is not removed by the browser and so provides

\* a permanent reliable place to store the binding information.

\* @example

<example>

<file name="index.html">

Check me to make text readonly: <input type="checkbox" ng-model="checked"><br/>

<input type="text" ng-readonly="checked" value="I'm Angular"/>

</file>

<file name="protractor.js" type="protractor">

it('should toggle readonly attr', function() {

expect(element(by.css('[type="text"]')).getAttribute('readonly')).toBeFalsy();

element(by.model('checked')).click();

expect(element(by.css('[type="text"]')).getAttribute('readonly')).toBeTruthy();

});

</file>

</example>

\*

\* @element INPUT

\* @param {expression} ngReadonly If the {@link guide/expression expression} is truthy,

\* then special attribute "readonly" will be set on the element

\*/

/\*\*

\* @ngdoc directive

\* @name ngSelected

\* @restrict A

\* @priority 100

\*

\* @description

\* The HTML specification does not require browsers to preserve the values of boolean attributes

\* such as selected. (Their presence means true and their absence means false.)

\* If we put an Angular interpolation expression into such an attribute then the

\* binding information would be lost when the browser removes the attribute.

\* The `ngSelected` directive solves this problem for the `selected` attribute.

\* This complementary directive is not removed by the browser and so provides

\* a permanent reliable place to store the binding information.

\*

\* @example

<example>

<file name="index.html">

Check me to select: <input type="checkbox" ng-model="selected"><br/>

<select>

<option>Hello!</option>

<option id="greet" ng-selected="selected">Greetings!</option>

</select>

</file>

<file name="protractor.js" type="protractor">

it('should select Greetings!', function() {

expect(element(by.id('greet')).getAttribute('selected')).toBeFalsy();

element(by.model('selected')).click();

expect(element(by.id('greet')).getAttribute('selected')).toBeTruthy();

});

</file>

</example>

\*

\* @element OPTION

\* @param {expression} ngSelected If the {@link guide/expression expression} is truthy,

\* then special attribute "selected" will be set on the element

\*/

/\*\*

\* @ngdoc directive

\* @name ngOpen

\* @restrict A

\* @priority 100

\*

\* @description

\* The HTML specification does not require browsers to preserve the values of boolean attributes

\* such as open. (Their presence means true and their absence means false.)

\* If we put an Angular interpolation expression into such an attribute then the

\* binding information would be lost when the browser removes the attribute.

\* The `ngOpen` directive solves this problem for the `open` attribute.

\* This complementary directive is not removed by the browser and so provides

\* a permanent reliable place to store the binding information.

\* @example

<example>

<file name="index.html">

Check me check multiple: <input type="checkbox" ng-model="open"><br/>

<details id="details" ng-open="open">

<summary>Show/Hide me</summary>

</details>

</file>

<file name="protractor.js" type="protractor">

it('should toggle open', function() {

expect(element(by.id('details')).getAttribute('open')).toBeFalsy();

element(by.model('open')).click();

expect(element(by.id('details')).getAttribute('open')).toBeTruthy();

});

</file>

</example>

\*

\* @element DETAILS

\* @param {expression} ngOpen If the {@link guide/expression expression} is truthy,

\* then special attribute "open" will be set on the element

\*/

var ngAttributeAliasDirectives = {};

// boolean attrs are evaluated

forEach(BOOLEAN\_ATTR, function(propName, attrName) {

// binding to multiple is not supported

if (propName == "multiple") return;

var normalized = directiveNormalize('ng-' + attrName);

ngAttributeAliasDirectives[normalized] = function() {

return {

priority: 100,

link: function(scope, element, attr) {

scope.$watch(attr[normalized], function ngBooleanAttrWatchAction(value) {

attr.$set(attrName, !!value);

});

}

};

};

});

// ng-src, ng-srcset, ng-href are interpolated

forEach(['src', 'srcset', 'href'], function(attrName) {

var normalized = directiveNormalize('ng-' + attrName);

ngAttributeAliasDirectives[normalized] = function() {

return {

priority: 99, // it needs to run after the attributes are interpolated

link: function(scope, element, attr) {

var propName = attrName,

name = attrName;

if (attrName === 'href' &&

toString.call(element.prop('href')) === '[object SVGAnimatedString]') {

name = 'xlinkHref';

attr.$attr[name] = 'xlink:href';

propName = null;

}

attr.$observe(normalized, function(value) {

if (!value)

return;

attr.$set(name, value);

// on IE, if "ng:src" directive declaration is used and "src" attribute doesn't exist

// then calling element.setAttribute('src', 'foo') doesn't do anything, so we need

// to set the property as well to achieve the desired effect.

// we use attr[attrName] value since $set can sanitize the url.

if (msie && propName) element.prop(propName, attr[name]);

});

}

};

};

});

/\* global -nullFormCtrl \*/

var nullFormCtrl = {

$addControl: noop,

$removeControl: noop,

$setValidity: noop,

$setDirty: noop,

$setPristine: noop

};

/\*\*

\* @ngdoc type

\* @name form.FormController

\*

\* @property {boolean} $pristine True if user has not interacted with the form yet.

\* @property {boolean} $dirty True if user has already interacted with the form.

\* @property {boolean} $valid True if all of the containing forms and controls are valid.

\* @property {boolean} $invalid True if at least one containing control or form is invalid.

\*

\* @property {Object} $error Is an object hash, containing references to all invalid controls or

\* forms, where:

\*

\* - keys are validation tokens (error names),

\* - values are arrays of controls or forms that are invalid for given error name.

\*

\*

\* Built-in validation tokens:

\*

\* - `email`

\* - `max`

\* - `maxlength`

\* - `min`

\* - `minlength`

\* - `number`

\* - `pattern`

\* - `required`

\* - `url`

\*

\* @description

\* `FormController` keeps track of all its controls and nested forms as well as state of them,

\* such as being valid/invalid or dirty/pristine.

\*

\* Each {@link ng.directive:form form} directive creates an instance

\* of `FormController`.

\*

\*/

//asks for $scope to fool the BC controller module

FormController.$inject = ['$element', '$attrs', '$scope', '$animate'];

function FormController(element, attrs, $scope, $animate) {

var form = this,

parentForm = element.parent().controller('form') || nullFormCtrl,

invalidCount = 0, // used to easily determine if we are valid

errors = form.$error = {},

controls = [];

// init state

form.$name = attrs.name || attrs.ngForm;

form.$dirty = false;

form.$pristine = true;

form.$valid = true;

form.$invalid = false;

parentForm.$addControl(form);

// Setup initial state of the control

element.addClass(PRISTINE\_CLASS);

toggleValidCss(true);

// convenience method for easy toggling of classes

function toggleValidCss(isValid, validationErrorKey) {

validationErrorKey = validationErrorKey ? '-' + snake\_case(validationErrorKey, '-') : '';

$animate.removeClass(element, (isValid ? INVALID\_CLASS : VALID\_CLASS) + validationErrorKey);

$animate.addClass(element, (isValid ? VALID\_CLASS : INVALID\_CLASS) + validationErrorKey);

}

/\*\*

\* @ngdoc method

\* @name form.FormController#$addControl

\*

\* @description

\* Register a control with the form.

\*

\* Input elements using ngModelController do this automatically when they are linked.

\*/

form.$addControl = function(control) {

// Breaking change - before, inputs whose name was "hasOwnProperty" were quietly ignored

// and not added to the scope. Now we throw an error.

assertNotHasOwnProperty(control.$name, 'input');

controls.push(control);

if (control.$name) {

form[control.$name] = control;

}

};

/\*\*

\* @ngdoc method

\* @name form.FormController#$removeControl

\*

\* @description

\* Deregister a control from the form.

\*

\* Input elements using ngModelController do this automatically when they are destroyed.

\*/

form.$removeControl = function(control) {

if (control.$name && form[control.$name] === control) {

delete form[control.$name];

}

forEach(errors, function(queue, validationToken) {

form.$setValidity(validationToken, true, control);

});

arrayRemove(controls, control);

};

/\*\*

\* @ngdoc method

\* @name form.FormController#$setValidity

\*

\* @description

\* Sets the validity of a form control.

\*

\* This method will also propagate to parent forms.

\*/

form.$setValidity = function(validationToken, isValid, control) {

var queue = errors[validationToken];

if (isValid) {

if (queue) {

arrayRemove(queue, control);

if (!queue.length) {

invalidCount--;

if (!invalidCount) {

toggleValidCss(isValid);

form.$valid = true;

form.$invalid = false;

}

errors[validationToken] = false;

toggleValidCss(true, validationToken);

parentForm.$setValidity(validationToken, true, form);

}

}

} else {

if (!invalidCount) {

toggleValidCss(isValid);

}

if (queue) {

if (includes(queue, control)) return;

} else {

errors[validationToken] = queue = [];

invalidCount++;

toggleValidCss(false, validationToken);

parentForm.$setValidity(validationToken, false, form);

}

queue.push(control);

form.$valid = false;

form.$invalid = true;

}

};

/\*\*

\* @ngdoc method

\* @name form.FormController#$setDirty

\*

\* @description

\* Sets the form to a dirty state.

\*

\* This method can be called to add the 'ng-dirty' class and set the form to a dirty

\* state (ng-dirty class). This method will also propagate to parent forms.

\*/

form.$setDirty = function() {

$animate.removeClass(element, PRISTINE\_CLASS);

$animate.addClass(element, DIRTY\_CLASS);

form.$dirty = true;

form.$pristine = false;

parentForm.$setDirty();

};

/\*\*

\* @ngdoc method

\* @name form.FormController#$setPristine

\*

\* @description

\* Sets the form to its pristine state.

\*

\* This method can be called to remove the 'ng-dirty' class and set the form to its pristine

\* state (ng-pristine class). This method will also propagate to all the controls contained

\* in this form.

\*

\* Setting a form back to a pristine state is often useful when we want to 'reuse' a form after

\* saving or resetting it.

\*/

form.$setPristine = function () {

$animate.removeClass(element, DIRTY\_CLASS);

$animate.addClass(element, PRISTINE\_CLASS);

form.$dirty = false;

form.$pristine = true;

forEach(controls, function(control) {

control.$setPristine();

});

};

}

/\*\*

\* @ngdoc directive

\* @name ngForm

\* @restrict EAC

\*

\* @description

\* Nestable alias of {@link ng.directive:form `form`} directive. HTML

\* does not allow nesting of form elements. It is useful to nest forms, for example if the validity of a

\* sub-group of controls needs to be determined.

\*

\* Note: the purpose of `ngForm` is to group controls,

\* but not to be a replacement for the `<form>` tag with all of its capabilities

\* (e.g. posting to the server, ...).

\*

\* @param {string=} ngForm|name Name of the form. If specified, the form controller will be published into

\* related scope, under this name.

\*

\*/

/\*\*

\* @ngdoc directive

\* @name form

\* @restrict E

\*

\* @description

\* Directive that instantiates

\* {@link form.FormController FormController}.

\*

\* If the `name` attribute is specified, the form controller is published onto the current scope under

\* this name.

\*

\* # Alias: {@link ng.directive:ngForm `ngForm`}

\*

\* In Angular forms can be nested. This means that the outer form is valid when all of the child

\* forms are valid as well. However, browsers do not allow nesting of `<form>` elements, so

\* Angular provides the {@link ng.directive:ngForm `ngForm`} directive which behaves identically to

\* `<form>` but can be nested. This allows you to have nested forms, which is very useful when

\* using Angular validation directives in forms that are dynamically generated using the

\* {@link ng.directive:ngRepeat `ngRepeat`} directive. Since you cannot dynamically generate the `name`

\* attribute of input elements using interpolation, you have to wrap each set of repeated inputs in an

\* `ngForm` directive and nest these in an outer `form` element.

\*

\*

\* # CSS classes

\* - `ng-valid` is set if the form is valid.

\* - `ng-invalid` is set if the form is invalid.

\* - `ng-pristine` is set if the form is pristine.

\* - `ng-dirty` is set if the form is dirty.

\*

\* Keep in mind that ngAnimate can detect each of these classes when added and removed.

\*

\*

\* # Submitting a form and preventing the default action

\*

\* Since the role of forms in client-side Angular applications is different than in classical

\* roundtrip apps, it is desirable for the browser not to translate the form submission into a full

\* page reload that sends the data to the server. Instead some javascript logic should be triggered

\* to handle the form submission in an application-specific way.

\*

\* For this reason, Angular prevents the default action (form submission to the server) unless the

\* `<form>` element has an `action` attribute specified.

\*

\* You can use one of the following two ways to specify what javascript method should be called when

\* a form is submitted:

\*

\* - {@link ng.directive:ngSubmit ngSubmit} directive on the form element

\* - {@link ng.directive:ngClick ngClick} directive on the first

\* button or input field of type submit (input[type=submit])

\*

\* To prevent double execution of the handler, use only one of the {@link ng.directive:ngSubmit ngSubmit}

\* or {@link ng.directive:ngClick ngClick} directives.

\* This is because of the following form submission rules in the HTML specification:

\*

\* - If a form has only one input field then hitting enter in this field triggers form submit

\* (`ngSubmit`)

\* - if a form has 2+ input fields and no buttons or input[type=submit] then hitting enter

\* doesn't trigger submit

\* - if a form has one or more input fields and one or more buttons or input[type=submit] then

\* hitting enter in any of the input fields will trigger the click handler on the \*first\* button or

\* input[type=submit] (`ngClick`) \*and\* a submit handler on the enclosing form (`ngSubmit`)

\*

\* @param {string=} name Name of the form. If specified, the form controller will be published into

\* related scope, under this name.

\*

\* ## Animation Hooks

\*

\* Animations in ngForm are triggered when any of the associated CSS classes are added and removed.

\* These classes are: `.ng-pristine`, `.ng-dirty`, `.ng-invalid` and `.ng-valid` as well as any

\* other validations that are performed within the form. Animations in ngForm are similar to how

\* they work in ngClass and animations can be hooked into using CSS transitions, keyframes as well

\* as JS animations.

\*

\* The following example shows a simple way to utilize CSS transitions to style a form element

\* that has been rendered as invalid after it has been validated:

\*

\* <pre>

\* //be sure to include ngAnimate as a module to hook into more

\* //advanced animations

\* .my-form {

\* transition:0.5s linear all;

\* background: white;

\* }

\* .my-form.ng-invalid {

\* background: red;

\* color:white;

\* }

\* </pre>

\*

\* @example

<example deps="angular-animate.js" animations="true" fixBase="true">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.userType = 'guest';

}

</script>

<style>

.my-form {

-webkit-transition:all linear 0.5s;

transition:all linear 0.5s;

background: transparent;

}

.my-form.ng-invalid {

background: red;

}

</style>

<form name="myForm" ng-controller="Ctrl" class="my-form">

userType: <input name="input" ng-model="userType" required>

<span class="error" ng-show="myForm.input.$error.required">Required!</span><br>

<tt>userType = {{userType}}</tt><br>

<tt>myForm.input.$valid = {{myForm.input.$valid}}</tt><br>

<tt>myForm.input.$error = {{myForm.input.$error}}</tt><br>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br>

</form>

</file>

<file name="protractor.js" type="protractor">

it('should initialize to model', function() {

var userType = element(by.binding('userType'));

var valid = element(by.binding('myForm.input.$valid'));

expect(userType.getText()).toContain('guest');

expect(valid.getText()).toContain('true');

});

it('should be invalid if empty', function() {

var userType = element(by.binding('userType'));

var valid = element(by.binding('myForm.input.$valid'));

var userInput = element(by.model('userType'));

userInput.clear();

userInput.sendKeys('');

expect(userType.getText()).toEqual('userType =');

expect(valid.getText()).toContain('false');

});

</file>

</example>

\*

\*/

var formDirectiveFactory = function(isNgForm) {

return ['$timeout', function($timeout) {

var formDirective = {

name: 'form',

restrict: isNgForm ? 'EAC' : 'E',

controller: FormController,

compile: function() {

return {

pre: function(scope, formElement, attr, controller) {

if (!attr.action) {

// we can't use jq events because if a form is destroyed during submission the default

// action is not prevented. see #1238

//

// IE 9 is not affected because it doesn't fire a submit event and try to do a full

// page reload if the form was destroyed by submission of the form via a click handler

// on a button in the form. Looks like an IE9 specific bug.

var preventDefaultListener = function(event) {

event.preventDefault

? event.preventDefault()

: event.returnValue = false; // IE

};

addEventListenerFn(formElement[0], 'submit', preventDefaultListener);

// unregister the preventDefault listener so that we don't not leak memory but in a

// way that will achieve the prevention of the default action.

formElement.on('$destroy', function() {

$timeout(function() {

removeEventListenerFn(formElement[0], 'submit', preventDefaultListener);

}, 0, false);

});

}

var parentFormCtrl = formElement.parent().controller('form'),

alias = attr.name || attr.ngForm;

if (alias) {

setter(scope, alias, controller, alias);

}

if (parentFormCtrl) {

formElement.on('$destroy', function() {

parentFormCtrl.$removeControl(controller);

if (alias) {

setter(scope, alias, undefined, alias);

}

extend(controller, nullFormCtrl); //stop propagating child destruction handlers upwards

});

}

}

};

}

};

return formDirective;

}];

};

var formDirective = formDirectiveFactory();

var ngFormDirective = formDirectiveFactory(true);

/\* global

-VALID\_CLASS,

-INVALID\_CLASS,

-PRISTINE\_CLASS,

-DIRTY\_CLASS

\*/

var URL\_REGEXP = /^(ftp|http|https):\/\/(\w+:{0,1}\w\*@)?(\S+)(:[0-9]+)?(\/|\/([\w#!:.?+=&%@!\-\/]))?$/;

var EMAIL\_REGEXP = /^[a-z0-9!#$%&'\*+/=?^\_`{|}~.-]+@[a-z0-9-]+(\.[a-z0-9-]+)\*$/i;

var NUMBER\_REGEXP = /^\s\*(\-|\+)?(\d+|(\d\*(\.\d\*)))\s\*$/;

var inputType = {

/\*\*

\* @ngdoc input

\* @name input[text]

\*

\* @description

\* Standard HTML text input with angular data binding.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} required Adds `required` validation error key if the value is not entered.

\* @param {string=} ngRequired Adds `required` attribute and `required` validation constraint to

\* the element when the ngRequired expression evaluates to true. Use `ngRequired` instead of

\* `required` when you want to data-bind to the `required` attribute.

\* @param {number=} ngMinlength Sets `minlength` validation error key if the value is shorter than

\* minlength.

\* @param {number=} ngMaxlength Sets `maxlength` validation error key if the value is longer than

\* maxlength.

\* @param {string=} ngPattern Sets `pattern` validation error key if the value does not match the

\* RegExp pattern expression. Expected value is `/regexp/` for inline patterns or `regexp` for

\* patterns defined as scope expressions.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\* @param {boolean=} [ngTrim=true] If set to false Angular will not automatically trim the input.

\*

\* @example

<example name="text-input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.text = 'guest';

$scope.word = /^\s\*\w\*\s\*$/;

}

</script>

<form name="myForm" ng-controller="Ctrl">

Single word: <input type="text" name="input" ng-model="text"

ng-pattern="word" required ng-trim="false">

<span class="error" ng-show="myForm.input.$error.required">

Required!</span>

<span class="error" ng-show="myForm.input.$error.pattern">

Single word only!</span>

<tt>text = {{text}}</tt><br/>

<tt>myForm.input.$valid = {{myForm.input.$valid}}</tt><br/>

<tt>myForm.input.$error = {{myForm.input.$error}}</tt><br/>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br/>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br/>

</form>

</file>

<file name="protractor.js" type="protractor">

var text = element(by.binding('text'));

var valid = element(by.binding('myForm.input.$valid'));

var input = element(by.model('text'));

it('should initialize to model', function() {

expect(text.getText()).toContain('guest');

expect(valid.getText()).toContain('true');

});

it('should be invalid if empty', function() {

input.clear();

input.sendKeys('');

expect(text.getText()).toEqual('text =');

expect(valid.getText()).toContain('false');

});

it('should be invalid if multi word', function() {

input.clear();

input.sendKeys('hello world');

expect(valid.getText()).toContain('false');

});

</file>

</example>

\*/

'text': textInputType,

/\*\*

\* @ngdoc input

\* @name input[number]

\*

\* @description

\* Text input with number validation and transformation. Sets the `number` validation

\* error if not a valid number.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} min Sets the `min` validation error key if the value entered is less than `min`.

\* @param {string=} max Sets the `max` validation error key if the value entered is greater than `max`.

\* @param {string=} required Sets `required` validation error key if the value is not entered.

\* @param {string=} ngRequired Adds `required` attribute and `required` validation constraint to

\* the element when the ngRequired expression evaluates to true. Use `ngRequired` instead of

\* `required` when you want to data-bind to the `required` attribute.

\* @param {number=} ngMinlength Sets `minlength` validation error key if the value is shorter than

\* minlength.

\* @param {number=} ngMaxlength Sets `maxlength` validation error key if the value is longer than

\* maxlength.

\* @param {string=} ngPattern Sets `pattern` validation error key if the value does not match the

\* RegExp pattern expression. Expected value is `/regexp/` for inline patterns or `regexp` for

\* patterns defined as scope expressions.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\*

\* @example

<example name="number-input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.value = 12;

}

</script>

<form name="myForm" ng-controller="Ctrl">

Number: <input type="number" name="input" ng-model="value"

min="0" max="99" required>

<span class="error" ng-show="myForm.input.$error.required">

Required!</span>

<span class="error" ng-show="myForm.input.$error.number">

Not valid number!</span>

<tt>value = {{value}}</tt><br/>

<tt>myForm.input.$valid = {{myForm.input.$valid}}</tt><br/>

<tt>myForm.input.$error = {{myForm.input.$error}}</tt><br/>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br/>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br/>

</form>

</file>

<file name="protractor.js" type="protractor">

var value = element(by.binding('value'));

var valid = element(by.binding('myForm.input.$valid'));

var input = element(by.model('value'));

it('should initialize to model', function() {

expect(value.getText()).toContain('12');

expect(valid.getText()).toContain('true');

});

it('should be invalid if empty', function() {

input.clear();

input.sendKeys('');

expect(value.getText()).toEqual('value =');

expect(valid.getText()).toContain('false');

});

it('should be invalid if over max', function() {

input.clear();

input.sendKeys('123');

expect(value.getText()).toEqual('value =');

expect(valid.getText()).toContain('false');

});

</file>

</example>

\*/

'number': numberInputType,

/\*\*

\* @ngdoc input

\* @name input[url]

\*

\* @description

\* Text input with URL validation. Sets the `url` validation error key if the content is not a

\* valid URL.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} required Sets `required` validation error key if the value is not entered.

\* @param {string=} ngRequired Adds `required` attribute and `required` validation constraint to

\* the element when the ngRequired expression evaluates to true. Use `ngRequired` instead of

\* `required` when you want to data-bind to the `required` attribute.

\* @param {number=} ngMinlength Sets `minlength` validation error key if the value is shorter than

\* minlength.

\* @param {number=} ngMaxlength Sets `maxlength` validation error key if the value is longer than

\* maxlength.

\* @param {string=} ngPattern Sets `pattern` validation error key if the value does not match the

\* RegExp pattern expression. Expected value is `/regexp/` for inline patterns or `regexp` for

\* patterns defined as scope expressions.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\*

\* @example

<example name="url-input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.text = 'http://google.com';

}

</script>

<form name="myForm" ng-controller="Ctrl">

URL: <input type="url" name="input" ng-model="text" required>

<span class="error" ng-show="myForm.input.$error.required">

Required!</span>

<span class="error" ng-show="myForm.input.$error.url">

Not valid url!</span>

<tt>text = {{text}}</tt><br/>

<tt>myForm.input.$valid = {{myForm.input.$valid}}</tt><br/>

<tt>myForm.input.$error = {{myForm.input.$error}}</tt><br/>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br/>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br/>

<tt>myForm.$error.url = {{!!myForm.$error.url}}</tt><br/>

</form>

</file>

<file name="protractor.js" type="protractor">

var text = element(by.binding('text'));

var valid = element(by.binding('myForm.input.$valid'));

var input = element(by.model('text'));

it('should initialize to model', function() {

expect(text.getText()).toContain('http://google.com');

expect(valid.getText()).toContain('true');

});

it('should be invalid if empty', function() {

input.clear();

input.sendKeys('');

expect(text.getText()).toEqual('text =');

expect(valid.getText()).toContain('false');

});

it('should be invalid if not url', function() {

input.clear();

input.sendKeys('box');

expect(valid.getText()).toContain('false');

});

</file>

</example>

\*/

'url': urlInputType,

/\*\*

\* @ngdoc input

\* @name input[email]

\*

\* @description

\* Text input with email validation. Sets the `email` validation error key if not a valid email

\* address.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} required Sets `required` validation error key if the value is not entered.

\* @param {string=} ngRequired Adds `required` attribute and `required` validation constraint to

\* the element when the ngRequired expression evaluates to true. Use `ngRequired` instead of

\* `required` when you want to data-bind to the `required` attribute.

\* @param {number=} ngMinlength Sets `minlength` validation error key if the value is shorter than

\* minlength.

\* @param {number=} ngMaxlength Sets `maxlength` validation error key if the value is longer than

\* maxlength.

\* @param {string=} ngPattern Sets `pattern` validation error key if the value does not match the

\* RegExp pattern expression. Expected value is `/regexp/` for inline patterns or `regexp` for

\* patterns defined as scope expressions.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\*

\* @example

<example name="email-input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.text = 'me@example.com';

}

</script>

<form name="myForm" ng-controller="Ctrl">

Email: <input type="email" name="input" ng-model="text" required>

<span class="error" ng-show="myForm.input.$error.required">

Required!</span>

<span class="error" ng-show="myForm.input.$error.email">

Not valid email!</span>

<tt>text = {{text}}</tt><br/>

<tt>myForm.input.$valid = {{myForm.input.$valid}}</tt><br/>

<tt>myForm.input.$error = {{myForm.input.$error}}</tt><br/>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br/>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br/>

<tt>myForm.$error.email = {{!!myForm.$error.email}}</tt><br/>

</form>

</file>

<file name="protractor.js" type="protractor">

var text = element(by.binding('text'));

var valid = element(by.binding('myForm.input.$valid'));

var input = element(by.model('text'));

it('should initialize to model', function() {

expect(text.getText()).toContain('me@example.com');

expect(valid.getText()).toContain('true');

});

it('should be invalid if empty', function() {

input.clear();

input.sendKeys('');

expect(text.getText()).toEqual('text =');

expect(valid.getText()).toContain('false');

});

it('should be invalid if not email', function() {

input.clear();

input.sendKeys('xxx');

expect(valid.getText()).toContain('false');

});

</file>

</example>

\*/

'email': emailInputType,

/\*\*

\* @ngdoc input

\* @name input[radio]

\*

\* @description

\* HTML radio button.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string} value The value to which the expression should be set when selected.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\* @param {string} ngValue Angular expression which sets the value to which the expression should

\* be set when selected.

\*

\* @example

<example name="radio-input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.color = 'blue';

$scope.specialValue = {

"id": "12345",

"value": "green"

};

}

</script>

<form name="myForm" ng-controller="Ctrl">

<input type="radio" ng-model="color" value="red"> Red <br/>

<input type="radio" ng-model="color" ng-value="specialValue"> Green <br/>

<input type="radio" ng-model="color" value="blue"> Blue <br/>

<tt>color = {{color | json}}</tt><br/>

</form>

Note that `ng-value="specialValue"` sets radio item's value to be the value of `$scope.specialValue`.

</file>

<file name="protractor.js" type="protractor">

it('should change state', function() {

var color = element(by.binding('color'));

expect(color.getText()).toContain('blue');

element.all(by.model('color')).get(0).click();

expect(color.getText()).toContain('red');

});

</file>

</example>

\*/

'radio': radioInputType,

/\*\*

\* @ngdoc input

\* @name input[checkbox]

\*

\* @description

\* HTML checkbox.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} ngTrueValue The value to which the expression should be set when selected.

\* @param {string=} ngFalseValue The value to which the expression should be set when not selected.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\*

\* @example

<example name="checkbox-input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.value1 = true;

$scope.value2 = 'YES'

}

</script>

<form name="myForm" ng-controller="Ctrl">

Value1: <input type="checkbox" ng-model="value1"> <br/>

Value2: <input type="checkbox" ng-model="value2"

ng-true-value="YES" ng-false-value="NO"> <br/>

<tt>value1 = {{value1}}</tt><br/>

<tt>value2 = {{value2}}</tt><br/>

</form>

</file>

<file name="protractor.js" type="protractor">

it('should change state', function() {

var value1 = element(by.binding('value1'));

var value2 = element(by.binding('value2'));

expect(value1.getText()).toContain('true');

expect(value2.getText()).toContain('YES');

element(by.model('value1')).click();

element(by.model('value2')).click();

expect(value1.getText()).toContain('false');

expect(value2.getText()).toContain('NO');

});

</file>

</example>

\*/

'checkbox': checkboxInputType,

'hidden': noop,

'button': noop,

'submit': noop,

'reset': noop,

'file': noop

};

// A helper function to call $setValidity and return the value / undefined,

// a pattern that is repeated a lot in the input validation logic.

function validate(ctrl, validatorName, validity, value){

ctrl.$setValidity(validatorName, validity);

return validity ? value : undefined;

}

function addNativeHtml5Validators(ctrl, validatorName, element) {

var validity = element.prop('validity');

if (isObject(validity)) {

var validator = function(value) {

// Don't overwrite previous validation, don't consider valueMissing to apply (ng-required can

// perform the required validation)

if (!ctrl.$error[validatorName] && (validity.badInput || validity.customError ||

validity.typeMismatch) && !validity.valueMissing) {

ctrl.$setValidity(validatorName, false);

return;

}

return value;

};

ctrl.$parsers.push(validator);

}

}

function textInputType(scope, element, attr, ctrl, $sniffer, $browser) {

var validity = element.prop('validity');

// In composition mode, users are still inputing intermediate text buffer,

// hold the listener until composition is done.

// More about composition events: https://developer.mozilla.org/en-US/docs/Web/API/CompositionEvent

if (!$sniffer.android) {

var composing = false;

element.on('compositionstart', function(data) {

composing = true;

});

element.on('compositionend', function() {

composing = false;

listener();

});

}

var listener = function() {

if (composing) return;

var value = element.val();

// By default we will trim the value

// If the attribute ng-trim exists we will avoid trimming

// e.g. <input ng-model="foo" ng-trim="false">

if (toBoolean(attr.ngTrim || 'T')) {

value = trim(value);

}

if (ctrl.$viewValue !== value ||

// If the value is still empty/falsy, and there is no `required` error, run validators

// again. This enables HTML5 constraint validation errors to affect Angular validation

// even when the first character entered causes an error.

(validity && value === '' && !validity.valueMissing)) {

if (scope.$$phase) {

ctrl.$setViewValue(value);

} else {

scope.$apply(function() {

ctrl.$setViewValue(value);

});

}

}

};

// if the browser does support "input" event, we are fine - except on IE9 which doesn't fire the

// input event on backspace, delete or cut

if ($sniffer.hasEvent('input')) {

element.on('input', listener);

} else {

var timeout;

var deferListener = function() {

if (!timeout) {

timeout = $browser.defer(function() {

listener();

timeout = null;

});

}

};

element.on('keydown', function(event) {

var key = event.keyCode;

// ignore

// command modifiers arrows

if (key === 91 || (15 < key && key < 19) || (37 <= key && key <= 40)) return;

deferListener();

});

// if user modifies input value using context menu in IE, we need "paste" and "cut" events to catch it

if ($sniffer.hasEvent('paste')) {

element.on('paste cut', deferListener);

}

}

// if user paste into input using mouse on older browser

// or form autocomplete on newer browser, we need "change" event to catch it

element.on('change', listener);

ctrl.$render = function() {

element.val(ctrl.$isEmpty(ctrl.$viewValue) ? '' : ctrl.$viewValue);

};

// pattern validator

var pattern = attr.ngPattern,

patternValidator,

match;

if (pattern) {

var validateRegex = function(regexp, value) {

return validate(ctrl, 'pattern', ctrl.$isEmpty(value) || regexp.test(value), value);

};

match = pattern.match(/^\/(.\*)\/([gim]\*)$/);

if (match) {

pattern = new RegExp(match[1], match[2]);

patternValidator = function(value) {

return validateRegex(pattern, value);

};

} else {

patternValidator = function(value) {

var patternObj = scope.$eval(pattern);

if (!patternObj || !patternObj.test) {

throw minErr('ngPattern')('noregexp',

'Expected {0} to be a RegExp but was {1}. Element: {2}', pattern,

patternObj, startingTag(element));

}

return validateRegex(patternObj, value);

};

}

ctrl.$formatters.push(patternValidator);

ctrl.$parsers.push(patternValidator);

}

// min length validator

if (attr.ngMinlength) {

var minlength = int(attr.ngMinlength);

var minLengthValidator = function(value) {

return validate(ctrl, 'minlength', ctrl.$isEmpty(value) || value.length >= minlength, value);

};

ctrl.$parsers.push(minLengthValidator);

ctrl.$formatters.push(minLengthValidator);

}

// max length validator

if (attr.ngMaxlength) {

var maxlength = int(attr.ngMaxlength);

var maxLengthValidator = function(value) {

return validate(ctrl, 'maxlength', ctrl.$isEmpty(value) || value.length <= maxlength, value);

};

ctrl.$parsers.push(maxLengthValidator);

ctrl.$formatters.push(maxLengthValidator);

}

}

function numberInputType(scope, element, attr, ctrl, $sniffer, $browser) {

textInputType(scope, element, attr, ctrl, $sniffer, $browser);

ctrl.$parsers.push(function(value) {

var empty = ctrl.$isEmpty(value);

if (empty || NUMBER\_REGEXP.test(value)) {

ctrl.$setValidity('number', true);

return value === '' ? null : (empty ? value : parseFloat(value));

} else {

ctrl.$setValidity('number', false);

return undefined;

}

});

addNativeHtml5Validators(ctrl, 'number', element);

ctrl.$formatters.push(function(value) {

return ctrl.$isEmpty(value) ? '' : '' + value;

});

if (attr.min) {

var minValidator = function(value) {

var min = parseFloat(attr.min);

return validate(ctrl, 'min', ctrl.$isEmpty(value) || value >= min, value);

};

ctrl.$parsers.push(minValidator);

ctrl.$formatters.push(minValidator);

}

if (attr.max) {

var maxValidator = function(value) {

var max = parseFloat(attr.max);

return validate(ctrl, 'max', ctrl.$isEmpty(value) || value <= max, value);

};

ctrl.$parsers.push(maxValidator);

ctrl.$formatters.push(maxValidator);

}

ctrl.$formatters.push(function(value) {

return validate(ctrl, 'number', ctrl.$isEmpty(value) || isNumber(value), value);

});

}

function urlInputType(scope, element, attr, ctrl, $sniffer, $browser) {

textInputType(scope, element, attr, ctrl, $sniffer, $browser);

var urlValidator = function(value) {

return validate(ctrl, 'url', ctrl.$isEmpty(value) || URL\_REGEXP.test(value), value);

};

ctrl.$formatters.push(urlValidator);

ctrl.$parsers.push(urlValidator);

}

function emailInputType(scope, element, attr, ctrl, $sniffer, $browser) {

textInputType(scope, element, attr, ctrl, $sniffer, $browser);

var emailValidator = function(value) {

return validate(ctrl, 'email', ctrl.$isEmpty(value) || EMAIL\_REGEXP.test(value), value);

};

ctrl.$formatters.push(emailValidator);

ctrl.$parsers.push(emailValidator);

}

function radioInputType(scope, element, attr, ctrl) {

// make the name unique, if not defined

if (isUndefined(attr.name)) {

element.attr('name', nextUid());

}

element.on('click', function() {

if (element[0].checked) {

scope.$apply(function() {

ctrl.$setViewValue(attr.value);

});

}

});

ctrl.$render = function() {

var value = attr.value;

element[0].checked = (value == ctrl.$viewValue);

};

attr.$observe('value', ctrl.$render);

}

function checkboxInputType(scope, element, attr, ctrl) {

var trueValue = attr.ngTrueValue,

falseValue = attr.ngFalseValue;

if (!isString(trueValue)) trueValue = true;

if (!isString(falseValue)) falseValue = false;

element.on('click', function() {

scope.$apply(function() {

ctrl.$setViewValue(element[0].checked);

});

});

ctrl.$render = function() {

element[0].checked = ctrl.$viewValue;

};

// Override the standard `$isEmpty` because a value of `false` means empty in a checkbox.

ctrl.$isEmpty = function(value) {

return value !== trueValue;

};

ctrl.$formatters.push(function(value) {

return value === trueValue;

});

ctrl.$parsers.push(function(value) {

return value ? trueValue : falseValue;

});

}

/\*\*

\* @ngdoc directive

\* @name textarea

\* @restrict E

\*

\* @description

\* HTML textarea element control with angular data-binding. The data-binding and validation

\* properties of this element are exactly the same as those of the

\* {@link ng.directive:input input element}.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} required Sets `required` validation error key if the value is not entered.

\* @param {string=} ngRequired Adds `required` attribute and `required` validation constraint to

\* the element when the ngRequired expression evaluates to true. Use `ngRequired` instead of

\* `required` when you want to data-bind to the `required` attribute.

\* @param {number=} ngMinlength Sets `minlength` validation error key if the value is shorter than

\* minlength.

\* @param {number=} ngMaxlength Sets `maxlength` validation error key if the value is longer than

\* maxlength.

\* @param {string=} ngPattern Sets `pattern` validation error key if the value does not match the

\* RegExp pattern expression. Expected value is `/regexp/` for inline patterns or `regexp` for

\* patterns defined as scope expressions.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\*/

/\*\*

\* @ngdoc directive

\* @name input

\* @restrict E

\*

\* @description

\* HTML input element control with angular data-binding. Input control follows HTML5 input types

\* and polyfills the HTML5 validation behavior for older browsers.

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} required Sets `required` validation error key if the value is not entered.

\* @param {boolean=} ngRequired Sets `required` attribute if set to true

\* @param {number=} ngMinlength Sets `minlength` validation error key if the value is shorter than

\* minlength.

\* @param {number=} ngMaxlength Sets `maxlength` validation error key if the value is longer than

\* maxlength.

\* @param {string=} ngPattern Sets `pattern` validation error key if the value does not match the

\* RegExp pattern expression. Expected value is `/regexp/` for inline patterns or `regexp` for

\* patterns defined as scope expressions.

\* @param {string=} ngChange Angular expression to be executed when input changes due to user

\* interaction with the input element.

\*

\* @example

<example name="input-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.user = {name: 'guest', last: 'visitor'};

}

</script>

<div ng-controller="Ctrl">

<form name="myForm">

User name: <input type="text" name="userName" ng-model="user.name" required>

<span class="error" ng-show="myForm.userName.$error.required">

Required!</span><br>

Last name: <input type="text" name="lastName" ng-model="user.last"

ng-minlength="3" ng-maxlength="10">

<span class="error" ng-show="myForm.lastName.$error.minlength">

Too short!</span>

<span class="error" ng-show="myForm.lastName.$error.maxlength">

Too long!</span><br>

</form>

<hr>

<tt>user = {{user}}</tt><br/>

<tt>myForm.userName.$valid = {{myForm.userName.$valid}}</tt><br>

<tt>myForm.userName.$error = {{myForm.userName.$error}}</tt><br>

<tt>myForm.lastName.$valid = {{myForm.lastName.$valid}}</tt><br>

<tt>myForm.lastName.$error = {{myForm.lastName.$error}}</tt><br>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br>

<tt>myForm.$error.minlength = {{!!myForm.$error.minlength}}</tt><br>

<tt>myForm.$error.maxlength = {{!!myForm.$error.maxlength}}</tt><br>

</div>

</file>

<file name="protractor.js" type="protractor">

var user = element(by.binding('{{user}}'));

var userNameValid = element(by.binding('myForm.userName.$valid'));

var lastNameValid = element(by.binding('myForm.lastName.$valid'));

var lastNameError = element(by.binding('myForm.lastName.$error'));

var formValid = element(by.binding('myForm.$valid'));

var userNameInput = element(by.model('user.name'));

var userLastInput = element(by.model('user.last'));

it('should initialize to model', function() {

expect(user.getText()).toContain('{"name":"guest","last":"visitor"}');

expect(userNameValid.getText()).toContain('true');

expect(formValid.getText()).toContain('true');

});

it('should be invalid if empty when required', function() {

userNameInput.clear();

userNameInput.sendKeys('');

expect(user.getText()).toContain('{"last":"visitor"}');

expect(userNameValid.getText()).toContain('false');

expect(formValid.getText()).toContain('false');

});

it('should be valid if empty when min length is set', function() {

userLastInput.clear();

userLastInput.sendKeys('');

expect(user.getText()).toContain('{"name":"guest","last":""}');

expect(lastNameValid.getText()).toContain('true');

expect(formValid.getText()).toContain('true');

});

it('should be invalid if less than required min length', function() {

userLastInput.clear();

userLastInput.sendKeys('xx');

expect(user.getText()).toContain('{"name":"guest"}');

expect(lastNameValid.getText()).toContain('false');

expect(lastNameError.getText()).toContain('minlength');

expect(formValid.getText()).toContain('false');

});

it('should be invalid if longer than max length', function() {

userLastInput.clear();

userLastInput.sendKeys('some ridiculously long name');

expect(user.getText()).toContain('{"name":"guest"}');

expect(lastNameValid.getText()).toContain('false');

expect(lastNameError.getText()).toContain('maxlength');

expect(formValid.getText()).toContain('false');

});

</file>

</example>

\*/

var inputDirective = ['$browser', '$sniffer', function($browser, $sniffer) {

return {

restrict: 'E',

require: '?ngModel',

link: function(scope, element, attr, ctrl) {

if (ctrl) {

(inputType[lowercase(attr.type)] || inputType.text)(scope, element, attr, ctrl, $sniffer,

$browser);

}

}

};

}];

var VALID\_CLASS = 'ng-valid',

INVALID\_CLASS = 'ng-invalid',

PRISTINE\_CLASS = 'ng-pristine',

DIRTY\_CLASS = 'ng-dirty';

/\*\*

\* @ngdoc type

\* @name ngModel.NgModelController

\*

\* @property {string} $viewValue Actual string value in the view.

\* @property {\*} $modelValue The value in the model, that the control is bound to.

\* @property {Array.<Function>} $parsers Array of functions to execute, as a pipeline, whenever

the control reads value from the DOM. Each function is called, in turn, passing the value

through to the next. The last return value is used to populate the model.

Used to sanitize / convert the value as well as validation. For validation,

the parsers should update the validity state using

{@link ngModel.NgModelController#$setValidity $setValidity()},

and return `undefined` for invalid values.

\*

\* @property {Array.<Function>} $formatters Array of functions to execute, as a pipeline, whenever

the model value changes. Each function is called, in turn, passing the value through to the

next. Used to format / convert values for display in the control and validation.

\* ```js

\* function formatter(value) {

\* if (value) {

\* return value.toUpperCase();

\* }

\* }

\* ngModel.$formatters.push(formatter);

\* ```

\*

\* @property {Array.<Function>} $viewChangeListeners Array of functions to execute whenever the

\* view value has changed. It is called with no arguments, and its return value is ignored.

\* This can be used in place of additional $watches against the model value.

\*

\* @property {Object} $error An object hash with all errors as keys.

\*

\* @property {boolean} $pristine True if user has not interacted with the control yet.

\* @property {boolean} $dirty True if user has already interacted with the control.

\* @property {boolean} $valid True if there is no error.

\* @property {boolean} $invalid True if at least one error on the control.

\*

\* @description

\*

\* `NgModelController` provides API for the `ng-model` directive. The controller contains

\* services for data-binding, validation, CSS updates, and value formatting and parsing. It

\* purposefully does not contain any logic which deals with DOM rendering or listening to

\* DOM events. Such DOM related logic should be provided by other directives which make use of

\* `NgModelController` for data-binding.

\*

\* ## Custom Control Example

\* This example shows how to use `NgModelController` with a custom control to achieve

\* data-binding. Notice how different directives (`contenteditable`, `ng-model`, and `required`)

\* collaborate together to achieve the desired result.

\*

\* Note that `contenteditable` is an HTML5 attribute, which tells the browser to let the element

\* contents be edited in place by the user. This will not work on older browsers.

\*

\* <example name="NgModelController" module="customControl">

<file name="style.css">

[contenteditable] {

border: 1px solid black;

background-color: white;

min-height: 20px;

}

.ng-invalid {

border: 1px solid red;

}

</file>

<file name="script.js">

angular.module('customControl', []).

directive('contenteditable', function() {

return {

restrict: 'A', // only activate on element attribute

require: '?ngModel', // get a hold of NgModelController

link: function(scope, element, attrs, ngModel) {

if(!ngModel) return; // do nothing if no ng-model

// Specify how UI should be updated

ngModel.$render = function() {

element.html(ngModel.$viewValue || '');

};

// Listen for change events to enable binding

element.on('blur keyup change', function() {

scope.$apply(read);

});

read(); // initialize

// Write data to the model

function read() {

var html = element.html();

// When we clear the content editable the browser leaves a <br> behind

// If strip-br attribute is provided then we strip this out

if( attrs.stripBr && html == '<br>' ) {

html = '';

}

ngModel.$setViewValue(html);

}

}

};

});

</file>

<file name="index.html">

<form name="myForm">

<div contenteditable

name="myWidget" ng-model="userContent"

strip-br="true"

required>Change me!</div>

<span ng-show="myForm.myWidget.$error.required">Required!</span>

<hr>

<textarea ng-model="userContent"></textarea>

</form>

</file>

<file name="protractor.js" type="protractor">

it('should data-bind and become invalid', function() {

if (browser.params.browser == 'safari' || browser.params.browser == 'firefox') {

// SafariDriver can't handle contenteditable

// and Firefox driver can't clear contenteditables very well

return;

}

var contentEditable = element(by.css('[contenteditable]'));

var content = 'Change me!';

expect(contentEditable.getText()).toEqual(content);

contentEditable.clear();

contentEditable.sendKeys(protractor.Key.BACK\_SPACE);

expect(contentEditable.getText()).toEqual('');

expect(contentEditable.getAttribute('class')).toMatch(/ng-invalid-required/);

});

</file>

\* </example>

\*

\*

\*/

var NgModelController = ['$scope', '$exceptionHandler', '$attrs', '$element', '$parse', '$animate',

function($scope, $exceptionHandler, $attr, $element, $parse, $animate) {

this.$viewValue = Number.NaN;

this.$modelValue = Number.NaN;

this.$parsers = [];

this.$formatters = [];

this.$viewChangeListeners = [];

this.$pristine = true;

this.$dirty = false;

this.$valid = true;

this.$invalid = false;

this.$name = $attr.name;

var ngModelGet = $parse($attr.ngModel),

ngModelSet = ngModelGet.assign;

if (!ngModelSet) {

throw minErr('ngModel')('nonassign', "Expression '{0}' is non-assignable. Element: {1}",

$attr.ngModel, startingTag($element));

}

/\*\*

\* @ngdoc method

\* @name ngModel.NgModelController#$render

\*

\* @description

\* Called when the view needs to be updated. It is expected that the user of the ng-model

\* directive will implement this method.

\*/

this.$render = noop;

/\*\*

\* @ngdoc method

\* @name ngModel.NgModelController#$isEmpty

\*

\* @description

\* This is called when we need to determine if the value of the input is empty.

\*

\* For instance, the required directive does this to work out if the input has data or not.

\* The default `$isEmpty` function checks whether the value is `undefined`, `''`, `null` or `NaN`.

\*

\* You can override this for input directives whose concept of being empty is different to the

\* default. The `checkboxInputType` directive does this because in its case a value of `false`

\* implies empty.

\*

\* @param {\*} value Reference to check.

\* @returns {boolean} True if `value` is empty.

\*/

this.$isEmpty = function(value) {

return isUndefined(value) || value === '' || value === null || value !== value;

};

var parentForm = $element.inheritedData('$formController') || nullFormCtrl,

invalidCount = 0, // used to easily determine if we are valid

$error = this.$error = {}; // keep invalid keys here

// Setup initial state of the control

$element.addClass(PRISTINE\_CLASS);

toggleValidCss(true);

// convenience method for easy toggling of classes

function toggleValidCss(isValid, validationErrorKey) {

validationErrorKey = validationErrorKey ? '-' + snake\_case(validationErrorKey, '-') : '';

$animate.removeClass($element, (isValid ? INVALID\_CLASS : VALID\_CLASS) + validationErrorKey);

$animate.addClass($element, (isValid ? VALID\_CLASS : INVALID\_CLASS) + validationErrorKey);

}

/\*\*

\* @ngdoc method

\* @name ngModel.NgModelController#$setValidity

\*

\* @description

\* Change the validity state, and notifies the form when the control changes validity. (i.e. it

\* does not notify form if given validator is already marked as invalid).

\*

\* This method should be called by validators - i.e. the parser or formatter functions.

\*

\* @param {string} validationErrorKey Name of the validator. the `validationErrorKey` will assign

\* to `$error[validationErrorKey]=isValid` so that it is available for data-binding.

\* The `validationErrorKey` should be in camelCase and will get converted into dash-case

\* for class name. Example: `myError` will result in `ng-valid-my-error` and `ng-invalid-my-error`

\* class and can be bound to as `{{someForm.someControl.$error.myError}}` .

\* @param {boolean} isValid Whether the current state is valid (true) or invalid (false).

\*/

this.$setValidity = function(validationErrorKey, isValid) {

// Purposeful use of ! here to cast isValid to boolean in case it is undefined

// jshint -W018

if ($error[validationErrorKey] === !isValid) return;

// jshint +W018

if (isValid) {

if ($error[validationErrorKey]) invalidCount--;

if (!invalidCount) {

toggleValidCss(true);

this.$valid = true;

this.$invalid = false;

}

} else {

toggleValidCss(false);

this.$invalid = true;

this.$valid = false;

invalidCount++;

}

$error[validationErrorKey] = !isValid;

toggleValidCss(isValid, validationErrorKey);

parentForm.$setValidity(validationErrorKey, isValid, this);

};

/\*\*

\* @ngdoc method

\* @name ngModel.NgModelController#$setPristine

\*

\* @description

\* Sets the control to its pristine state.

\*

\* This method can be called to remove the 'ng-dirty' class and set the control to its pristine

\* state (ng-pristine class).

\*/

this.$setPristine = function () {

this.$dirty = false;

this.$pristine = true;

$animate.removeClass($element, DIRTY\_CLASS);

$animate.addClass($element, PRISTINE\_CLASS);

};

/\*\*

\* @ngdoc method

\* @name ngModel.NgModelController#$setViewValue

\*

\* @description

\* Update the view value.

\*

\* This method should be called when the view value changes, typically from within a DOM event handler.

\* For example {@link ng.directive:input input} and

\* {@link ng.directive:select select} directives call it.

\*

\* It will update the $viewValue, then pass this value through each of the functions in `$parsers`,

\* which includes any validators. The value that comes out of this `$parsers` pipeline, be applied to

\* `$modelValue` and the \*\*expression\*\* specified in the `ng-model` attribute.

\*

\* Lastly, all the registered change listeners, in the `$viewChangeListeners` list, are called.

\*

\* Note that calling this function does not trigger a `$digest`.

\*

\* @param {string} value Value from the view.

\*/

this.$setViewValue = function(value) {

this.$viewValue = value;

// change to dirty

if (this.$pristine) {

this.$dirty = true;

this.$pristine = false;

$animate.removeClass($element, PRISTINE\_CLASS);

$animate.addClass($element, DIRTY\_CLASS);

parentForm.$setDirty();

}

forEach(this.$parsers, function(fn) {

value = fn(value);

});

if (this.$modelValue !== value) {

this.$modelValue = value;

ngModelSet($scope, value);

forEach(this.$viewChangeListeners, function(listener) {

try {

listener();

} catch(e) {

$exceptionHandler(e);

}

});

}

};

// model -> value

var ctrl = this;

$scope.$watch(function ngModelWatch() {

var value = ngModelGet($scope);

// if scope model value and ngModel value are out of sync

if (ctrl.$modelValue !== value) {

var formatters = ctrl.$formatters,

idx = formatters.length;

ctrl.$modelValue = value;

while(idx--) {

value = formatters[idx](value);

}

if (ctrl.$viewValue !== value) {

ctrl.$viewValue = value;

ctrl.$render();

}

}

return value;

});

}];

/\*\*

\* @ngdoc directive

\* @name ngModel

\*

\* @element input

\*

\* @description

\* The `ngModel` directive binds an `input`,`select`, `textarea` (or custom form control) to a

\* property on the scope using {@link ngModel.NgModelController NgModelController},

\* which is created and exposed by this directive.

\*

\* `ngModel` is responsible for:

\*

\* - Binding the view into the model, which other directives such as `input`, `textarea` or `select`

\* require.

\* - Providing validation behavior (i.e. required, number, email, url).

\* - Keeping the state of the control (valid/invalid, dirty/pristine, validation errors).

\* - Setting related css classes on the element (`ng-valid`, `ng-invalid`, `ng-dirty`, `ng-pristine`) including animations.

\* - Registering the control with its parent {@link ng.directive:form form}.

\*

\* Note: `ngModel` will try to bind to the property given by evaluating the expression on the

\* current scope. If the property doesn't already exist on this scope, it will be created

\* implicitly and added to the scope.

\*

\* For best practices on using `ngModel`, see:

\*

\* - [https://github.com/angular/angular.js/wiki/Understanding-Scopes]

\*

\* For basic examples, how to use `ngModel`, see:

\*

\* - {@link ng.directive:input input}

\* - {@link input[text] text}

\* - {@link input[checkbox] checkbox}

\* - {@link input[radio] radio}

\* - {@link input[number] number}

\* - {@link input[email] email}

\* - {@link input[url] url}

\* - {@link ng.directive:select select}

\* - {@link ng.directive:textarea textarea}

\*

\* # CSS classes

\* The following CSS classes are added and removed on the associated input/select/textarea element

\* depending on the validity of the model.

\*

\* - `ng-valid` is set if the model is valid.

\* - `ng-invalid` is set if the model is invalid.

\* - `ng-pristine` is set if the model is pristine.

\* - `ng-dirty` is set if the model is dirty.

\*

\* Keep in mind that ngAnimate can detect each of these classes when added and removed.

\*

\* ## Animation Hooks

\*

\* Animations within models are triggered when any of the associated CSS classes are added and removed

\* on the input element which is attached to the model. These classes are: `.ng-pristine`, `.ng-dirty`,

\* `.ng-invalid` and `.ng-valid` as well as any other validations that are performed on the model itself.

\* The animations that are triggered within ngModel are similar to how they work in ngClass and

\* animations can be hooked into using CSS transitions, keyframes as well as JS animations.

\*

\* The following example shows a simple way to utilize CSS transitions to style an input element

\* that has been rendered as invalid after it has been validated:

\*

\* <pre>

\* //be sure to include ngAnimate as a module to hook into more

\* //advanced animations

\* .my-input {

\* transition:0.5s linear all;

\* background: white;

\* }

\* .my-input.ng-invalid {

\* background: red;

\* color:white;

\* }

\* </pre>

\*

\* @example

\* <example deps="angular-animate.js" animations="true" fixBase="true">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.val = '1';

}

</script>

<style>

.my-input {

-webkit-transition:all linear 0.5s;

transition:all linear 0.5s;

background: transparent;

}

.my-input.ng-invalid {

color:white;

background: red;

}

</style>

Update input to see transitions when valid/invalid.

Integer is a valid value.

<form name="testForm" ng-controller="Ctrl">

<input ng-model="val" ng-pattern="/^\d+$/" name="anim" class="my-input" />

</form>

</file>

\* </example>

\*/

var ngModelDirective = function() {

return {

require: ['ngModel', '^?form'],

controller: NgModelController,

link: function(scope, element, attr, ctrls) {

// notify others, especially parent forms

var modelCtrl = ctrls[0],

formCtrl = ctrls[1] || nullFormCtrl;

formCtrl.$addControl(modelCtrl);

scope.$on('$destroy', function() {

formCtrl.$removeControl(modelCtrl);

});

}

};

};

/\*\*

\* @ngdoc directive

\* @name ngChange

\*

\* @description

\* Evaluate the given expression when the user changes the input.

\* The expression is evaluated immediately, unlike the JavaScript onchange event

\* which only triggers at the end of a change (usually, when the user leaves the

\* form element or presses the return key).

\* The expression is not evaluated when the value change is coming from the model.

\*

\* Note, this directive requires `ngModel` to be present.

\*

\* @element input

\* @param {expression} ngChange {@link guide/expression Expression} to evaluate upon change

\* in input value.

\*

\* @example

\* <example name="ngChange-directive">

\* <file name="index.html">

\* <script>

\* function Controller($scope) {

\* $scope.counter = 0;

\* $scope.change = function() {

\* $scope.counter++;

\* };

\* }

\* </script>

\* <div ng-controller="Controller">

\* <input type="checkbox" ng-model="confirmed" ng-change="change()" id="ng-change-example1" />

\* <input type="checkbox" ng-model="confirmed" id="ng-change-example2" />

\* <label for="ng-change-example2">Confirmed</label><br />

\* <tt>debug = {{confirmed}}</tt><br/>

\* <tt>counter = {{counter}}</tt><br/>

\* </div>

\* </file>

\* <file name="protractor.js" type="protractor">

\* var counter = element(by.binding('counter'));

\* var debug = element(by.binding('confirmed'));

\*

\* it('should evaluate the expression if changing from view', function() {

\* expect(counter.getText()).toContain('0');

\*

\* element(by.id('ng-change-example1')).click();

\*

\* expect(counter.getText()).toContain('1');

\* expect(debug.getText()).toContain('true');

\* });

\*

\* it('should not evaluate the expression if changing from model', function() {

\* element(by.id('ng-change-example2')).click();

\* expect(counter.getText()).toContain('0');

\* expect(debug.getText()).toContain('true');

\* });

\* </file>

\* </example>

\*/

var ngChangeDirective = valueFn({

require: 'ngModel',

link: function(scope, element, attr, ctrl) {

ctrl.$viewChangeListeners.push(function() {

scope.$eval(attr.ngChange);

});

}

});

var requiredDirective = function() {

return {

require: '?ngModel',

link: function(scope, elm, attr, ctrl) {

if (!ctrl) return;

attr.required = true; // force truthy in case we are on non input element

var validator = function(value) {

if (attr.required && ctrl.$isEmpty(value)) {

ctrl.$setValidity('required', false);

return;

} else {

ctrl.$setValidity('required', true);

return value;

}

};

ctrl.$formatters.push(validator);

ctrl.$parsers.unshift(validator);

attr.$observe('required', function() {

validator(ctrl.$viewValue);

});

}

};

};

/\*\*

\* @ngdoc directive

\* @name ngList

\*

\* @description

\* Text input that converts between a delimited string and an array of strings. The delimiter

\* can be a fixed string (by default a comma) or a regular expression.

\*

\* @element input

\* @param {string=} ngList optional delimiter that should be used to split the value. If

\* specified in form `/something/` then the value will be converted into a regular expression.

\*

\* @example

<example name="ngList-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.names = ['igor', 'misko', 'vojta'];

}

</script>

<form name="myForm" ng-controller="Ctrl">

List: <input name="namesInput" ng-model="names" ng-list required>

<span class="error" ng-show="myForm.namesInput.$error.required">

Required!</span>

<br>

<tt>names = {{names}}</tt><br/>

<tt>myForm.namesInput.$valid = {{myForm.namesInput.$valid}}</tt><br/>

<tt>myForm.namesInput.$error = {{myForm.namesInput.$error}}</tt><br/>

<tt>myForm.$valid = {{myForm.$valid}}</tt><br/>

<tt>myForm.$error.required = {{!!myForm.$error.required}}</tt><br/>

</form>

</file>

<file name="protractor.js" type="protractor">

var listInput = element(by.model('names'));

var names = element(by.binding('{{names}}'));

var valid = element(by.binding('myForm.namesInput.$valid'));

var error = element(by.css('span.error'));

it('should initialize to model', function() {

expect(names.getText()).toContain('["igor","misko","vojta"]');

expect(valid.getText()).toContain('true');

expect(error.getCssValue('display')).toBe('none');

});

it('should be invalid if empty', function() {

listInput.clear();

listInput.sendKeys('');

expect(names.getText()).toContain('');

expect(valid.getText()).toContain('false');

expect(error.getCssValue('display')).not.toBe('none'); });

</file>

</example>

\*/

var ngListDirective = function() {

return {

require: 'ngModel',

link: function(scope, element, attr, ctrl) {

var match = /\/(.\*)\//.exec(attr.ngList),

separator = match && new RegExp(match[1]) || attr.ngList || ',';

var parse = function(viewValue) {

// If the viewValue is invalid (say required but empty) it will be `undefined`

if (isUndefined(viewValue)) return;

var list = [];

if (viewValue) {

forEach(viewValue.split(separator), function(value) {

if (value) list.push(trim(value));

});

}

return list;

};

ctrl.$parsers.push(parse);

ctrl.$formatters.push(function(value) {

if (isArray(value)) {

return value.join(', ');

}

return undefined;

});

// Override the standard $isEmpty because an empty array means the input is empty.

ctrl.$isEmpty = function(value) {

return !value || !value.length;

};

}

};

};

var CONSTANT\_VALUE\_REGEXP = /^(true|false|\d+)$/;

/\*\*

\* @ngdoc directive

\* @name ngValue

\*

\* @description

\* Binds the given expression to the value of `input[select]` or `input[radio]`, so

\* that when the element is selected, the `ngModel` of that element is set to the

\* bound value.

\*

\* `ngValue` is useful when dynamically generating lists of radio buttons using `ng-repeat`, as

\* shown below.

\*

\* @element input

\* @param {string=} ngValue angular expression, whose value will be bound to the `value` attribute

\* of the `input` element

\*

\* @example

<example name="ngValue-directive">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.names = ['pizza', 'unicorns', 'robots'];

$scope.my = { favorite: 'unicorns' };

}

</script>

<form ng-controller="Ctrl">

<h2>Which is your favorite?</h2>

<label ng-repeat="name in names" for="{{name}}">

{{name}}

<input type="radio"

ng-model="my.favorite"

ng-value="name"

id="{{name}}"

name="favorite">

</label>

<div>You chose {{my.favorite}}</div>

</form>

</file>

<file name="protractor.js" type="protractor">

var favorite = element(by.binding('my.favorite'));

it('should initialize to model', function() {

expect(favorite.getText()).toContain('unicorns');

});

it('should bind the values to the inputs', function() {

element.all(by.model('my.favorite')).get(0).click();

expect(favorite.getText()).toContain('pizza');

});

</file>

</example>

\*/

var ngValueDirective = function() {

return {

priority: 100,

compile: function(tpl, tplAttr) {

if (CONSTANT\_VALUE\_REGEXP.test(tplAttr.ngValue)) {

return function ngValueConstantLink(scope, elm, attr) {

attr.$set('value', scope.$eval(attr.ngValue));

};

} else {

return function ngValueLink(scope, elm, attr) {

scope.$watch(attr.ngValue, function valueWatchAction(value) {

attr.$set('value', value);

});

};

}

}

};

};

/\*\*

\* @ngdoc directive

\* @name ngBind

\* @restrict AC

\*

\* @description

\* The `ngBind` attribute tells Angular to replace the text content of the specified HTML element

\* with the value of a given expression, and to update the text content when the value of that

\* expression changes.

\*

\* Typically, you don't use `ngBind` directly, but instead you use the double curly markup like

\* `{{ expression }}` which is similar but less verbose.

\*

\* It is preferable to use `ngBind` instead of `{{ expression }}` when a template is momentarily

\* displayed by the browser in its raw state before Angular compiles it. Since `ngBind` is an

\* element attribute, it makes the bindings invisible to the user while the page is loading.

\*

\* An alternative solution to this problem would be using the

\* {@link ng.directive:ngCloak ngCloak} directive.

\*

\*

\* @element ANY

\* @param {expression} ngBind {@link guide/expression Expression} to evaluate.

\*

\* @example

\* Enter a name in the Live Preview text box; the greeting below the text box changes instantly.

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.name = 'Whirled';

}

</script>

<div ng-controller="Ctrl">

Enter name: <input type="text" ng-model="name"><br>

Hello <span ng-bind="name"></span>!

</div>

</file>

<file name="protractor.js" type="protractor">

it('should check ng-bind', function() {

var nameInput = element(by.model('name'));

expect(element(by.binding('name')).getText()).toBe('Whirled');

nameInput.clear();

nameInput.sendKeys('world');

expect(element(by.binding('name')).getText()).toBe('world');

});

</file>

</example>

\*/

var ngBindDirective = ngDirective(function(scope, element, attr) {

element.addClass('ng-binding').data('$binding', attr.ngBind);

scope.$watch(attr.ngBind, function ngBindWatchAction(value) {

// We are purposefully using == here rather than === because we want to

// catch when value is "null or undefined"

// jshint -W041

element.text(value == undefined ? '' : value);

});

});

/\*\*

\* @ngdoc directive

\* @name ngBindTemplate

\*

\* @description

\* The `ngBindTemplate` directive specifies that the element

\* text content should be replaced with the interpolation of the template

\* in the `ngBindTemplate` attribute.

\* Unlike `ngBind`, the `ngBindTemplate` can contain multiple `{{` `}}`

\* expressions. This directive is needed since some HTML elements

\* (such as TITLE and OPTION) cannot contain SPAN elements.

\*

\* @element ANY

\* @param {string} ngBindTemplate template of form

\* <tt>{{</tt> <tt>expression</tt> <tt>}}</tt> to eval.

\*

\* @example

\* Try it here: enter text in text box and watch the greeting change.

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.salutation = 'Hello';

$scope.name = 'World';

}

</script>

<div ng-controller="Ctrl">

Salutation: <input type="text" ng-model="salutation"><br>

Name: <input type="text" ng-model="name"><br>

<pre ng-bind-template="{{salutation}} {{name}}!"></pre>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should check ng-bind', function() {

var salutationElem = element(by.binding('salutation'));

var salutationInput = element(by.model('salutation'));

var nameInput = element(by.model('name'));

expect(salutationElem.getText()).toBe('Hello World!');

salutationInput.clear();

salutationInput.sendKeys('Greetings');

nameInput.clear();

nameInput.sendKeys('user');

expect(salutationElem.getText()).toBe('Greetings user!');

});

</file>

</example>

\*/

var ngBindTemplateDirective = ['$interpolate', function($interpolate) {

return function(scope, element, attr) {

// TODO: move this to scenario runner

var interpolateFn = $interpolate(element.attr(attr.$attr.ngBindTemplate));

element.addClass('ng-binding').data('$binding', interpolateFn);

attr.$observe('ngBindTemplate', function(value) {

element.text(value);

});

};

}];

/\*\*

\* @ngdoc directive

\* @name ngBindHtml

\*

\* @description

\* Creates a binding that will innerHTML the result of evaluating the `expression` into the current

\* element in a secure way. By default, the innerHTML-ed content will be sanitized using the {@link

\* ngSanitize.$sanitize $sanitize} service. To utilize this functionality, ensure that `$sanitize`

\* is available, for example, by including {@link ngSanitize} in your module's dependencies (not in

\* core Angular.) You may also bypass sanitization for values you know are safe. To do so, bind to

\* an explicitly trusted value via {@link ng.$sce#trustAsHtml $sce.trustAsHtml}. See the example

\* under {@link ng.$sce#Example Strict Contextual Escaping (SCE)}.

\*

\* Note: If a `$sanitize` service is unavailable and the bound value isn't explicitly trusted, you

\* will have an exception (instead of an exploit.)

\*

\* @element ANY

\* @param {expression} ngBindHtml {@link guide/expression Expression} to evaluate.

\*

\* @example

Try it here: enter text in text box and watch the greeting change.

<example module="ngBindHtmlExample" deps="angular-sanitize.js">

<file name="index.html">

<div ng-controller="ngBindHtmlCtrl">

<p ng-bind-html="myHTML"></p>

</div>

</file>

<file name="script.js">

angular.module('ngBindHtmlExample', ['ngSanitize'])

.controller('ngBindHtmlCtrl', ['$scope', function ngBindHtmlCtrl($scope) {

$scope.myHTML =

'I am an <code>HTML</code>string with <a href="#">links!</a> and other <em>stuff</em>';

}]);

</file>

<file name="protractor.js" type="protractor">

it('should check ng-bind-html', function() {

expect(element(by.binding('myHTML')).getText()).toBe(

'I am an HTMLstring with links! and other stuff');

});

</file>

</example>

\*/

var ngBindHtmlDirective = ['$sce', '$parse', function($sce, $parse) {

return function(scope, element, attr) {

element.addClass('ng-binding').data('$binding', attr.ngBindHtml);

var parsed = $parse(attr.ngBindHtml);

function getStringValue() { return (parsed(scope) || '').toString(); }

scope.$watch(getStringValue, function ngBindHtmlWatchAction(value) {

element.html($sce.getTrustedHtml(parsed(scope)) || '');

});

};

}];

function classDirective(name, selector) {

name = 'ngClass' + name;

return ['$animate', function($animate) {

return {

restrict: 'AC',

link: function(scope, element, attr) {

var oldVal;

scope.$watch(attr[name], ngClassWatchAction, true);

attr.$observe('class', function(value) {

ngClassWatchAction(scope.$eval(attr[name]));

});

if (name !== 'ngClass') {

scope.$watch('$index', function($index, old$index) {

// jshint bitwise: false

var mod = $index & 1;

if (mod !== old$index & 1) {

var classes = arrayClasses(scope.$eval(attr[name]));

mod === selector ?

addClasses(classes) :

removeClasses(classes);

}

});

}

function addClasses(classes) {

var newClasses = digestClassCounts(classes, 1);

attr.$addClass(newClasses);

}

function removeClasses(classes) {

var newClasses = digestClassCounts(classes, -1);

attr.$removeClass(newClasses);

}

function digestClassCounts (classes, count) {

var classCounts = element.data('$classCounts') || {};

var classesToUpdate = [];

forEach(classes, function (className) {

if (count > 0 || classCounts[className]) {

classCounts[className] = (classCounts[className] || 0) + count;

if (classCounts[className] === +(count > 0)) {

classesToUpdate.push(className);

}

}

});

element.data('$classCounts', classCounts);

return classesToUpdate.join(' ');

}

function updateClasses (oldClasses, newClasses) {

var toAdd = arrayDifference(newClasses, oldClasses);

var toRemove = arrayDifference(oldClasses, newClasses);

toRemove = digestClassCounts(toRemove, -1);

toAdd = digestClassCounts(toAdd, 1);

if (toAdd.length === 0) {

$animate.removeClass(element, toRemove);

} else if (toRemove.length === 0) {

$animate.addClass(element, toAdd);

} else {

$animate.setClass(element, toAdd, toRemove);

}

}

function ngClassWatchAction(newVal) {

if (selector === true || scope.$index % 2 === selector) {

var newClasses = arrayClasses(newVal || []);

if (!oldVal) {

addClasses(newClasses);

} else if (!equals(newVal,oldVal)) {

var oldClasses = arrayClasses(oldVal);

updateClasses(oldClasses, newClasses);

}

}

oldVal = copy(newVal);

}

}

};

function arrayDifference(tokens1, tokens2) {

var values = [];

outer:

for(var i = 0; i < tokens1.length; i++) {

var token = tokens1[i];

for(var j = 0; j < tokens2.length; j++) {

if(token == tokens2[j]) continue outer;

}

values.push(token);

}

return values;

}

function arrayClasses (classVal) {

if (isArray(classVal)) {

return classVal;

} else if (isString(classVal)) {

return classVal.split(' ');

} else if (isObject(classVal)) {

var classes = [], i = 0;

forEach(classVal, function(v, k) {

if (v) {

classes.push(k);

}

});

return classes;

}

return classVal;

}

}];

}

/\*\*

\* @ngdoc directive

\* @name ngClass

\* @restrict AC

\*

\* @description

\* The `ngClass` directive allows you to dynamically set CSS classes on an HTML element by databinding

\* an expression that represents all classes to be added.

\*

\* The directive operates in three different ways, depending on which of three types the expression

\* evaluates to:

\*

\* 1. If the expression evaluates to a string, the string should be one or more space-delimited class

\* names.

\*

\* 2. If the expression evaluates to an array, each element of the array should be a string that is

\* one or more space-delimited class names.

\*

\* 3. If the expression evaluates to an object, then for each key-value pair of the

\* object with a truthy value the corresponding key is used as a class name.

\*

\* The directive won't add duplicate classes if a particular class was already set.

\*

\* When the expression changes, the previously added classes are removed and only then the

\* new classes are added.

\*

\* @animations

\* add - happens just before the class is applied to the element

\* remove - happens just before the class is removed from the element

\*

\* @element ANY

\* @param {expression} ngClass {@link guide/expression Expression} to eval. The result

\* of the evaluation can be a string representing space delimited class

\* names, an array, or a map of class names to boolean values. In the case of a map, the

\* names of the properties whose values are truthy will be added as css classes to the

\* element.

\*

\* @example Example that demonstrates basic bindings via ngClass directive.

<example>

<file name="index.html">

<p ng-class="{strike: deleted, bold: important, red: error}">Map Syntax Example</p>

<input type="checkbox" ng-model="deleted"> deleted (apply "strike" class)<br>

<input type="checkbox" ng-model="important"> important (apply "bold" class)<br>

<input type="checkbox" ng-model="error"> error (apply "red" class)

<hr>

<p ng-class="style">Using String Syntax</p>

<input type="text" ng-model="style" placeholder="Type: bold strike red">

<hr>

<p ng-class="[style1, style2, style3]">Using Array Syntax</p>

<input ng-model="style1" placeholder="Type: bold, strike or red"><br>

<input ng-model="style2" placeholder="Type: bold, strike or red"><br>

<input ng-model="style3" placeholder="Type: bold, strike or red"><br>

</file>

<file name="style.css">

.strike {

text-decoration: line-through;

}

.bold {

font-weight: bold;

}

.red {

color: red;

}

</file>

<file name="protractor.js" type="protractor">

var ps = element.all(by.css('p'));

it('should let you toggle the class', function() {

expect(ps.first().getAttribute('class')).not.toMatch(/bold/);

expect(ps.first().getAttribute('class')).not.toMatch(/red/);

element(by.model('important')).click();

expect(ps.first().getAttribute('class')).toMatch(/bold/);

element(by.model('error')).click();

expect(ps.first().getAttribute('class')).toMatch(/red/);

});

it('should let you toggle string example', function() {

expect(ps.get(1).getAttribute('class')).toBe('');

element(by.model('style')).clear();

element(by.model('style')).sendKeys('red');

expect(ps.get(1).getAttribute('class')).toBe('red');

});

it('array example should have 3 classes', function() {

expect(ps.last().getAttribute('class')).toBe('');

element(by.model('style1')).sendKeys('bold');

element(by.model('style2')).sendKeys('strike');

element(by.model('style3')).sendKeys('red');

expect(ps.last().getAttribute('class')).toBe('bold strike red');

});

</file>

</example>

## Animations

The example below demonstrates how to perform animations using ngClass.

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

<input id="setbtn" type="button" value="set" ng-click="myVar='my-class'">

<input id="clearbtn" type="button" value="clear" ng-click="myVar=''">

<br>

<span class="base-class" ng-class="myVar">Sample Text</span>

</file>

<file name="style.css">

.base-class {

-webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

}

.base-class.my-class {

color: red;

font-size:3em;

}

</file>

<file name="protractor.js" type="protractor">

it('should check ng-class', function() {

expect(element(by.css('.base-class')).getAttribute('class')).not.

toMatch(/my-class/);

element(by.id('setbtn')).click();

expect(element(by.css('.base-class')).getAttribute('class')).

toMatch(/my-class/);

element(by.id('clearbtn')).click();

expect(element(by.css('.base-class')).getAttribute('class')).not.

toMatch(/my-class/);

});

</file>

</example>

## ngClass and pre-existing CSS3 Transitions/Animations

The ngClass directive still supports CSS3 Transitions/Animations even if they do not follow the ngAnimate CSS naming structure.

Upon animation ngAnimate will apply supplementary CSS classes to track the start and end of an animation, but this will not hinder

any pre-existing CSS transitions already on the element. To get an idea of what happens during a class-based animation, be sure

to view the step by step details of {@link ngAnimate.$animate#addclass $animate.addClass} and

{@link ngAnimate.$animate#removeclass $animate.removeClass}.

\*/

var ngClassDirective = classDirective('', true);

/\*\*

\* @ngdoc directive

\* @name ngClassOdd

\* @restrict AC

\*

\* @description

\* The `ngClassOdd` and `ngClassEven` directives work exactly as

\* {@link ng.directive:ngClass ngClass}, except they work in

\* conjunction with `ngRepeat` and take effect only on odd (even) rows.

\*

\* This directive can be applied only within the scope of an

\* {@link ng.directive:ngRepeat ngRepeat}.

\*

\* @element ANY

\* @param {expression} ngClassOdd {@link guide/expression Expression} to eval. The result

\* of the evaluation can be a string representing space delimited class names or an array.

\*

\* @example

<example>

<file name="index.html">

<ol ng-init="names=['John', 'Mary', 'Cate', 'Suz']">

<li ng-repeat="name in names">

<span ng-class-odd="'odd'" ng-class-even="'even'">

{{name}}

</span>

</li>

</ol>

</file>

<file name="style.css">

.odd {

color: red;

}

.even {

color: blue;

}

</file>

<file name="protractor.js" type="protractor">

it('should check ng-class-odd and ng-class-even', function() {

expect(element(by.repeater('name in names').row(0).column('name')).getAttribute('class')).

toMatch(/odd/);

expect(element(by.repeater('name in names').row(1).column('name')).getAttribute('class')).

toMatch(/even/);

});

</file>

</example>

\*/

var ngClassOddDirective = classDirective('Odd', 0);

/\*\*

\* @ngdoc directive

\* @name ngClassEven

\* @restrict AC

\*

\* @description

\* The `ngClassOdd` and `ngClassEven` directives work exactly as

\* {@link ng.directive:ngClass ngClass}, except they work in

\* conjunction with `ngRepeat` and take effect only on odd (even) rows.

\*

\* This directive can be applied only within the scope of an

\* {@link ng.directive:ngRepeat ngRepeat}.

\*

\* @element ANY

\* @param {expression} ngClassEven {@link guide/expression Expression} to eval. The

\* result of the evaluation can be a string representing space delimited class names or an array.

\*

\* @example

<example>

<file name="index.html">

<ol ng-init="names=['John', 'Mary', 'Cate', 'Suz']">

<li ng-repeat="name in names">

<span ng-class-odd="'odd'" ng-class-even="'even'">

{{name}} &nbsp; &nbsp; &nbsp;

</span>

</li>

</ol>

</file>

<file name="style.css">

.odd {

color: red;

}

.even {

color: blue;

}

</file>

<file name="protractor.js" type="protractor">

it('should check ng-class-odd and ng-class-even', function() {

expect(element(by.repeater('name in names').row(0).column('name')).getAttribute('class')).

toMatch(/odd/);

expect(element(by.repeater('name in names').row(1).column('name')).getAttribute('class')).

toMatch(/even/);

});

</file>

</example>

\*/

var ngClassEvenDirective = classDirective('Even', 1);

/\*\*

\* @ngdoc directive

\* @name ngCloak

\* @restrict AC

\*

\* @description

\* The `ngCloak` directive is used to prevent the Angular html template from being briefly

\* displayed by the browser in its raw (uncompiled) form while your application is loading. Use this

\* directive to avoid the undesirable flicker effect caused by the html template display.

\*

\* The directive can be applied to the `<body>` element, but the preferred usage is to apply

\* multiple `ngCloak` directives to small portions of the page to permit progressive rendering

\* of the browser view.

\*

\* `ngCloak` works in cooperation with the following css rule embedded within `angular.js` and

\* `angular.min.js`.

\* For CSP mode please add `angular-csp.css` to your html file (see {@link ng.directive:ngCsp ngCsp}).

\*

\* ```css

\* [ng\:cloak], [ng-cloak], [data-ng-cloak], [x-ng-cloak], .ng-cloak, .x-ng-cloak {

\* display: none !important;

\* }

\* ```

\*

\* When this css rule is loaded by the browser, all html elements (including their children) that

\* are tagged with the `ngCloak` directive are hidden. When Angular encounters this directive

\* during the compilation of the template it deletes the `ngCloak` element attribute, making

\* the compiled element visible.

\*

\* For the best result, the `angular.js` script must be loaded in the head section of the html

\* document; alternatively, the css rule above must be included in the external stylesheet of the

\* application.

\*

\* Legacy browsers, like IE7, do not provide attribute selector support (added in CSS 2.1) so they

\* cannot match the `[ng\:cloak]` selector. To work around this limitation, you must add the css

\* class `ng-cloak` in addition to the `ngCloak` directive as shown in the example below.

\*

\* @element ANY

\*

\* @example

<example>

<file name="index.html">

<div id="template1" ng-cloak>{{ 'hello' }}</div>

<div id="template2" ng-cloak class="ng-cloak">{{ 'hello IE7' }}</div>

</file>

<file name="protractor.js" type="protractor">

it('should remove the template directive and css class', function() {

expect($('#template1').getAttribute('ng-cloak')).

toBeNull();

expect($('#template2').getAttribute('ng-cloak')).

toBeNull();

});

</file>

</example>

\*

\*/

var ngCloakDirective = ngDirective({

compile: function(element, attr) {

attr.$set('ngCloak', undefined);

element.removeClass('ng-cloak');

}

});

/\*\*

\* @ngdoc directive

\* @name ngController

\*

\* @description

\* The `ngController` directive attaches a controller class to the view. This is a key aspect of how angular

\* supports the principles behind the Model-View-Controller design pattern.

\*

\* MVC components in angular:

\*

\* \* Model — The Model is scope properties; scopes are attached to the DOM where scope properties

\* are accessed through bindings.

\* \* View — The template (HTML with data bindings) that is rendered into the View.

\* \* Controller — The `ngController` directive specifies a Controller class; the class contains business

\* logic behind the application to decorate the scope with functions and values

\*

\* Note that you can also attach controllers to the DOM by declaring it in a route definition

\* via the {@link ngRoute.$route $route} service. A common mistake is to declare the controller

\* again using `ng-controller` in the template itself. This will cause the controller to be attached

\* and executed twice.

\*

\* @element ANY

\* @scope

\* @param {expression} ngController Name of a globally accessible constructor function or an

\* {@link guide/expression expression} that on the current scope evaluates to a

\* constructor function. The controller instance can be published into a scope property

\* by specifying `as propertyName`.

\*

\* @example

\* Here is a simple form for editing user contact information. Adding, removing, clearing, and

\* greeting are methods declared on the controller (see source tab). These methods can

\* easily be called from the angular markup. Notice that the scope becomes the `this` for the

\* controller's instance. This allows for easy access to the view data from the controller. Also

\* notice that any changes to the data are automatically reflected in the View without the need

\* for a manual update. The example is shown in two different declaration styles you may use

\* according to preference.

<example>

<file name="index.html">

<script>

function SettingsController1() {

this.name = "John Smith";

this.contacts = [

{type: 'phone', value: '408 555 1212'},

{type: 'email', value: 'john.smith@example.org'} ];

};

SettingsController1.prototype.greet = function() {

alert(this.name);

};

SettingsController1.prototype.addContact = function() {

this.contacts.push({type: 'email', value: 'yourname@example.org'});

};

SettingsController1.prototype.removeContact = function(contactToRemove) {

var index = this.contacts.indexOf(contactToRemove);

this.contacts.splice(index, 1);

};

SettingsController1.prototype.clearContact = function(contact) {

contact.type = 'phone';

contact.value = '';

};

</script>

<div id="ctrl-as-exmpl" ng-controller="SettingsController1 as settings">

Name: <input type="text" ng-model="settings.name"/>

[ <a href="" ng-click="settings.greet()">greet</a> ]<br/>

Contact:

<ul>

<li ng-repeat="contact in settings.contacts">

<select ng-model="contact.type">

<option>phone</option>

<option>email</option>

</select>

<input type="text" ng-model="contact.value"/>

[ <a href="" ng-click="settings.clearContact(contact)">clear</a>

| <a href="" ng-click="settings.removeContact(contact)">X</a> ]

</li>

<li>[ <a href="" ng-click="settings.addContact()">add</a> ]</li>

</ul>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should check controller as', function() {

var container = element(by.id('ctrl-as-exmpl'));

expect(container.findElement(by.model('settings.name'))

.getAttribute('value')).toBe('John Smith');

var firstRepeat =

container.findElement(by.repeater('contact in settings.contacts').row(0));

var secondRepeat =

container.findElement(by.repeater('contact in settings.contacts').row(1));

expect(firstRepeat.findElement(by.model('contact.value')).getAttribute('value'))

.toBe('408 555 1212');

expect(secondRepeat.findElement(by.model('contact.value')).getAttribute('value'))

.toBe('john.smith@example.org');

firstRepeat.findElement(by.linkText('clear')).click();

expect(firstRepeat.findElement(by.model('contact.value')).getAttribute('value'))

.toBe('');

container.findElement(by.linkText('add')).click();

expect(container.findElement(by.repeater('contact in settings.contacts').row(2))

.findElement(by.model('contact.value'))

.getAttribute('value'))

.toBe('yourname@example.org');

});

</file>

</example>

<example>

<file name="index.html">

<script>

function SettingsController2($scope) {

$scope.name = "John Smith";

$scope.contacts = [

{type:'phone', value:'408 555 1212'},

{type:'email', value:'john.smith@example.org'} ];

$scope.greet = function() {

alert(this.name);

};

$scope.addContact = function() {

this.contacts.push({type:'email', value:'yourname@example.org'});

};

$scope.removeContact = function(contactToRemove) {

var index = this.contacts.indexOf(contactToRemove);

this.contacts.splice(index, 1);

};

$scope.clearContact = function(contact) {

contact.type = 'phone';

contact.value = '';

};

}

</script>

<div id="ctrl-exmpl" ng-controller="SettingsController2">

Name: <input type="text" ng-model="name"/>

[ <a href="" ng-click="greet()">greet</a> ]<br/>

Contact:

<ul>

<li ng-repeat="contact in contacts">

<select ng-model="contact.type">

<option>phone</option>

<option>email</option>

</select>

<input type="text" ng-model="contact.value"/>

[ <a href="" ng-click="clearContact(contact)">clear</a>

| <a href="" ng-click="removeContact(contact)">X</a> ]

</li>

<li>[ <a href="" ng-click="addContact()">add</a> ]</li>

</ul>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should check controller', function() {

var container = element(by.id('ctrl-exmpl'));

expect(container.findElement(by.model('name'))

.getAttribute('value')).toBe('John Smith');

var firstRepeat =

container.findElement(by.repeater('contact in contacts').row(0));

var secondRepeat =

container.findElement(by.repeater('contact in contacts').row(1));

expect(firstRepeat.findElement(by.model('contact.value')).getAttribute('value'))

.toBe('408 555 1212');

expect(secondRepeat.findElement(by.model('contact.value')).getAttribute('value'))

.toBe('john.smith@example.org');

firstRepeat.findElement(by.linkText('clear')).click();

expect(firstRepeat.findElement(by.model('contact.value')).getAttribute('value'))

.toBe('');

container.findElement(by.linkText('add')).click();

expect(container.findElement(by.repeater('contact in contacts').row(2))

.findElement(by.model('contact.value'))

.getAttribute('value'))

.toBe('yourname@example.org');

});

</file>

</example>

\*/

var ngControllerDirective = [function() {

return {

scope: true,

controller: '@',

priority: 500

};

}];

/\*\*

\* @ngdoc directive

\* @name ngCsp

\*

\* @element html

\* @description

\* Enables [CSP (Content Security Policy)](https://developer.mozilla.org/en/Security/CSP) support.

\*

\* This is necessary when developing things like Google Chrome Extensions.

\*

\* CSP forbids apps to use `eval` or `Function(string)` generated functions (among other things).

\* For us to be compatible, we just need to implement the "getterFn" in $parse without violating

\* any of these restrictions.

\*

\* AngularJS uses `Function(string)` generated functions as a speed optimization. Applying the `ngCsp`

\* directive will cause Angular to use CSP compatibility mode. When this mode is on AngularJS will

\* evaluate all expressions up to 30% slower than in non-CSP mode, but no security violations will

\* be raised.

\*

\* CSP forbids JavaScript to inline stylesheet rules. In non CSP mode Angular automatically

\* includes some CSS rules (e.g. {@link ng.directive:ngCloak ngCloak}).

\* To make those directives work in CSP mode, include the `angular-csp.css` manually.

\*

\* In order to use this feature put the `ngCsp` directive on the root element of the application.

\*

\* \*Note: This directive is only available in the `ng-csp` and `data-ng-csp` attribute form.\*

\*

\* @example

\* This example shows how to apply the `ngCsp` directive to the `html` tag.

```html

<!doctype html>

<html ng-app ng-csp>

...

...

</html>

```

\*/

// ngCsp is not implemented as a proper directive any more, because we need it be processed while we bootstrap

// the system (before $parse is instantiated), for this reason we just have a csp() fn that looks for ng-csp attribute

// anywhere in the current doc

/\*\*

\* @ngdoc directive

\* @name ngClick

\*

\* @description

\* The ngClick directive allows you to specify custom behavior when

\* an element is clicked.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngClick {@link guide/expression Expression} to evaluate upon

\* click. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-click="count = count + 1" ng-init="count=0">

Increment

</button>

count: {{count}}

</file>

<file name="protractor.js" type="protractor">

it('should check ng-click', function() {

expect(element(by.binding('count')).getText()).toMatch('0');

element(by.css('button')).click();

expect(element(by.binding('count')).getText()).toMatch('1');

});

</file>

</example>

\*/

/\*

\* A directive that allows creation of custom onclick handlers that are defined as angular

\* expressions and are compiled and executed within the current scope.

\*

\* Events that are handled via these handler are always configured not to propagate further.

\*/

var ngEventDirectives = {};

forEach(

'click dblclick mousedown mouseup mouseover mouseout mousemove mouseenter mouseleave keydown keyup keypress submit focus blur copy cut paste'.split(' '),

function(name) {

var directiveName = directiveNormalize('ng-' + name);

ngEventDirectives[directiveName] = ['$parse', function($parse) {

return {

compile: function($element, attr) {

var fn = $parse(attr[directiveName]);

return function(scope, element, attr) {

element.on(lowercase(name), function(event) {

scope.$apply(function() {

fn(scope, {$event:event});

});

});

};

}

};

}];

}

);

/\*\*

\* @ngdoc directive

\* @name ngDblclick

\*

\* @description

\* The `ngDblclick` directive allows you to specify custom behavior on a dblclick event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngDblclick {@link guide/expression Expression} to evaluate upon

\* a dblclick. (The Event object is available as `$event`)

\*

\* @example

<example>

<file name="index.html">

<button ng-dblclick="count = count + 1" ng-init="count=0">

Increment (on double click)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngMousedown

\*

\* @description

\* The ngMousedown directive allows you to specify custom behavior on mousedown event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngMousedown {@link guide/expression Expression} to evaluate upon

\* mousedown. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-mousedown="count = count + 1" ng-init="count=0">

Increment (on mouse down)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngMouseup

\*

\* @description

\* Specify custom behavior on mouseup event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngMouseup {@link guide/expression Expression} to evaluate upon

\* mouseup. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-mouseup="count = count + 1" ng-init="count=0">

Increment (on mouse up)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngMouseover

\*

\* @description

\* Specify custom behavior on mouseover event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngMouseover {@link guide/expression Expression} to evaluate upon

\* mouseover. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-mouseover="count = count + 1" ng-init="count=0">

Increment (when mouse is over)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngMouseenter

\*

\* @description

\* Specify custom behavior on mouseenter event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngMouseenter {@link guide/expression Expression} to evaluate upon

\* mouseenter. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-mouseenter="count = count + 1" ng-init="count=0">

Increment (when mouse enters)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngMouseleave

\*

\* @description

\* Specify custom behavior on mouseleave event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngMouseleave {@link guide/expression Expression} to evaluate upon

\* mouseleave. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-mouseleave="count = count + 1" ng-init="count=0">

Increment (when mouse leaves)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngMousemove

\*

\* @description

\* Specify custom behavior on mousemove event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngMousemove {@link guide/expression Expression} to evaluate upon

\* mousemove. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<button ng-mousemove="count = count + 1" ng-init="count=0">

Increment (when mouse moves)

</button>

count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngKeydown

\*

\* @description

\* Specify custom behavior on keydown event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngKeydown {@link guide/expression Expression} to evaluate upon

\* keydown. (Event object is available as `$event` and can be interrogated for keyCode, altKey, etc.)

\*

\* @example

<example>

<file name="index.html">

<input ng-keydown="count = count + 1" ng-init="count=0">

key down count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngKeyup

\*

\* @description

\* Specify custom behavior on keyup event.

\*

\* @element ANY

\* @priority 0

\* @param {expression} ngKeyup {@link guide/expression Expression} to evaluate upon

\* keyup. (Event object is available as `$event` and can be interrogated for keyCode, altKey, etc.)

\*

\* @example

<example>

<file name="index.html">

<input ng-keyup="count = count + 1" ng-init="count=0">

key up count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngKeypress

\*

\* @description

\* Specify custom behavior on keypress event.

\*

\* @element ANY

\* @param {expression} ngKeypress {@link guide/expression Expression} to evaluate upon

\* keypress. ({@link guide/expression#-event- Event object is available as `$event`}

\* and can be interrogated for keyCode, altKey, etc.)

\*

\* @example

<example>

<file name="index.html">

<input ng-keypress="count = count + 1" ng-init="count=0">

key press count: {{count}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngSubmit

\*

\* @description

\* Enables binding angular expressions to onsubmit events.

\*

\* Additionally it prevents the default action (which for form means sending the request to the

\* server and reloading the current page), but only if the form does not contain `action`,

\* `data-action`, or `x-action` attributes.

\*

\* @element form

\* @priority 0

\* @param {expression} ngSubmit {@link guide/expression Expression} to eval.

\* ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.list = [];

$scope.text = 'hello';

$scope.submit = function() {

if ($scope.text) {

$scope.list.push(this.text);

$scope.text = '';

}

};

}

</script>

<form ng-submit="submit()" ng-controller="Ctrl">

Enter text and hit enter:

<input type="text" ng-model="text" name="text" />

<input type="submit" id="submit" value="Submit" />

<pre>list={{list}}</pre>

</form>

</file>

<file name="protractor.js" type="protractor">

it('should check ng-submit', function() {

expect(element(by.binding('list')).getText()).toBe('list=[]');

element(by.css('#submit')).click();

expect(element(by.binding('list')).getText()).toContain('hello');

expect(element(by.input('text')).getAttribute('value')).toBe('');

});

it('should ignore empty strings', function() {

expect(element(by.binding('list')).getText()).toBe('list=[]');

element(by.css('#submit')).click();

element(by.css('#submit')).click();

expect(element(by.binding('list')).getText()).toContain('hello');

});

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngFocus

\*

\* @description

\* Specify custom behavior on focus event.

\*

\* @element window, input, select, textarea, a

\* @priority 0

\* @param {expression} ngFocus {@link guide/expression Expression} to evaluate upon

\* focus. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

\* See {@link ng.directive:ngClick ngClick}

\*/

/\*\*

\* @ngdoc directive

\* @name ngBlur

\*

\* @description

\* Specify custom behavior on blur event.

\*

\* @element window, input, select, textarea, a

\* @priority 0

\* @param {expression} ngBlur {@link guide/expression Expression} to evaluate upon

\* blur. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

\* See {@link ng.directive:ngClick ngClick}

\*/

/\*\*

\* @ngdoc directive

\* @name ngCopy

\*

\* @description

\* Specify custom behavior on copy event.

\*

\* @element window, input, select, textarea, a

\* @priority 0

\* @param {expression} ngCopy {@link guide/expression Expression} to evaluate upon

\* copy. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<input ng-copy="copied=true" ng-init="copied=false; value='copy me'" ng-model="value">

copied: {{copied}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngCut

\*

\* @description

\* Specify custom behavior on cut event.

\*

\* @element window, input, select, textarea, a

\* @priority 0

\* @param {expression} ngCut {@link guide/expression Expression} to evaluate upon

\* cut. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<input ng-cut="cut=true" ng-init="cut=false; value='cut me'" ng-model="value">

cut: {{cut}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngPaste

\*

\* @description

\* Specify custom behavior on paste event.

\*

\* @element window, input, select, textarea, a

\* @priority 0

\* @param {expression} ngPaste {@link guide/expression Expression} to evaluate upon

\* paste. ({@link guide/expression#-event- Event object is available as `$event`})

\*

\* @example

<example>

<file name="index.html">

<input ng-paste="paste=true" ng-init="paste=false" placeholder='paste here'>

pasted: {{paste}}

</file>

</example>

\*/

/\*\*

\* @ngdoc directive

\* @name ngIf

\* @restrict A

\*

\* @description

\* The `ngIf` directive removes or recreates a portion of the DOM tree based on an

\* {expression}. If the expression assigned to `ngIf` evaluates to a false

\* value then the element is removed from the DOM, otherwise a clone of the

\* element is reinserted into the DOM.

\*

\* `ngIf` differs from `ngShow` and `ngHide` in that `ngIf` completely removes and recreates the

\* element in the DOM rather than changing its visibility via the `display` css property. A common

\* case when this difference is significant is when using css selectors that rely on an element's

\* position within the DOM, such as the `:first-child` or `:last-child` pseudo-classes.

\*

\* Note that when an element is removed using `ngIf` its scope is destroyed and a new scope

\* is created when the element is restored. The scope created within `ngIf` inherits from

\* its parent scope using

\* [prototypal inheritance](https://github.com/angular/angular.js/wiki/The-Nuances-of-Scope-Prototypal-Inheritance).

\* An important implication of this is if `ngModel` is used within `ngIf` to bind to

\* a javascript primitive defined in the parent scope. In this case any modifications made to the

\* variable within the child scope will override (hide) the value in the parent scope.

\*

\* Also, `ngIf` recreates elements using their compiled state. An example of this behavior

\* is if an element's class attribute is directly modified after it's compiled, using something like

\* jQuery's `.addClass()` method, and the element is later removed. When `ngIf` recreates the element

\* the added class will be lost because the original compiled state is used to regenerate the element.

\*

\* Additionally, you can provide animations via the `ngAnimate` module to animate the `enter`

\* and `leave` effects.

\*

\* @animations

\* enter - happens just after the ngIf contents change and a new DOM element is created and injected into the ngIf container

\* leave - happens just before the ngIf contents are removed from the DOM

\*

\* @element ANY

\* @scope

\* @priority 600

\* @param {expression} ngIf If the {@link guide/expression expression} is falsy then

\* the element is removed from the DOM tree. If it is truthy a copy of the compiled

\* element is added to the DOM tree.

\*

\* @example

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

Click me: <input type="checkbox" ng-model="checked" ng-init="checked=true" /><br/>

Show when checked:

<span ng-if="checked" class="animate-if">

I'm removed when the checkbox is unchecked.

</span>

</file>

<file name="animations.css">

.animate-if {

background:white;

border:1px solid black;

padding:10px;

}

.animate-if.ng-enter, .animate-if.ng-leave {

-webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

}

.animate-if.ng-enter,

.animate-if.ng-leave.ng-leave-active {

opacity:0;

}

.animate-if.ng-leave,

.animate-if.ng-enter.ng-enter-active {

opacity:1;

}

</file>

</example>

\*/

var ngIfDirective = ['$animate', function($animate) {

return {

transclude: 'element',

priority: 600,

terminal: true,

restrict: 'A',

$$tlb: true,

link: function ($scope, $element, $attr, ctrl, $transclude) {

var block, childScope, previousElements;

$scope.$watch($attr.ngIf, function ngIfWatchAction(value) {

if (toBoolean(value)) {

if (!childScope) {

childScope = $scope.$new();

$transclude(childScope, function (clone) {

clone[clone.length++] = document.createComment(' end ngIf: ' + $attr.ngIf + ' ');

// Note: We only need the first/last node of the cloned nodes.

// However, we need to keep the reference to the jqlite wrapper as it might be changed later

// by a directive with templateUrl when it's template arrives.

block = {

clone: clone

};

$animate.enter(clone, $element.parent(), $element);

});

}

} else {

if(previousElements) {

previousElements.remove();

previousElements = null;

}

if(childScope) {

childScope.$destroy();

childScope = null;

}

if(block) {

previousElements = getBlockElements(block.clone);

$animate.leave(previousElements, function() {

previousElements = null;

});

block = null;

}

}

});

}

};

}];

/\*\*

\* @ngdoc directive

\* @name ngInclude

\* @restrict ECA

\*

\* @description

\* Fetches, compiles and includes an external HTML fragment.

\*

\* By default, the template URL is restricted to the same domain and protocol as the

\* application document. This is done by calling {@link ng.$sce#getTrustedResourceUrl

\* $sce.getTrustedResourceUrl} on it. To load templates from other domains or protocols

\* you may either {@link ng.$sceDelegateProvider#resourceUrlWhitelist whitelist them} or

\* [wrap them](ng.$sce#trustAsResourceUrl) as trusted values. Refer to Angular's {@link

\* ng.$sce Strict Contextual Escaping}.

\*

\* In addition, the browser's

\* [Same Origin Policy](https://code.google.com/p/browsersec/wiki/Part2#Same-origin\_policy\_for\_XMLHttpRequest)

\* and [Cross-Origin Resource Sharing (CORS)](http://www.w3.org/TR/cors/)

\* policy may further restrict whether the template is successfully loaded.

\* For example, `ngInclude` won't work for cross-domain requests on all browsers and for `file://`

\* access on some browsers.

\*

\* @animations

\* enter - animation is used to bring new content into the browser.

\* leave - animation is used to animate existing content away.

\*

\* The enter and leave animation occur concurrently.

\*

\* @scope

\* @priority 400

\*

\* @param {string} ngInclude|src angular expression evaluating to URL. If the source is a string constant,

\* make sure you wrap it in \*\*single\*\* quotes, e.g. `src="'myPartialTemplate.html'"`.

\* @param {string=} onload Expression to evaluate when a new partial is loaded.

\*

\* @param {string=} autoscroll Whether `ngInclude` should call {@link ng.$anchorScroll

\* $anchorScroll} to scroll the viewport after the content is loaded.

\*

\* - If the attribute is not set, disable scrolling.

\* - If the attribute is set without value, enable scrolling.

\* - Otherwise enable scrolling only if the expression evaluates to truthy value.

\*

\* @example

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

<div ng-controller="Ctrl">

<select ng-model="template" ng-options="t.name for t in templates">

<option value="">(blank)</option>

</select>

url of the template: <tt>{{template.url}}</tt>

<hr/>

<div class="slide-animate-container">

<div class="slide-animate" ng-include="template.url"></div>

</div>

</div>

</file>

<file name="script.js">

function Ctrl($scope) {

$scope.templates =

[ { name: 'template1.html', url: 'template1.html'},

{ name: 'template2.html', url: 'template2.html'} ];

$scope.template = $scope.templates[0];

}

</file>

<file name="template1.html">

Content of template1.html

</file>

<file name="template2.html">

Content of template2.html

</file>

<file name="animations.css">

.slide-animate-container {

position:relative;

background:white;

border:1px solid black;

height:40px;

overflow:hidden;

}

.slide-animate {

padding:10px;

}

.slide-animate.ng-enter, .slide-animate.ng-leave {

-webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

position:absolute;

top:0;

left:0;

right:0;

bottom:0;

display:block;

padding:10px;

}

.slide-animate.ng-enter {

top:-50px;

}

.slide-animate.ng-enter.ng-enter-active {

top:0;

}

.slide-animate.ng-leave {

top:0;

}

.slide-animate.ng-leave.ng-leave-active {

top:50px;

}

</file>

<file name="protractor.js" type="protractor">

var templateSelect = element(by.model('template'));

var includeElem = element(by.css('[ng-include]'));

it('should load template1.html', function() {

expect(includeElem.getText()).toMatch(/Content of template1.html/);

});

it('should load template2.html', function() {

if (browser.params.browser == 'firefox') {

// Firefox can't handle using selects

// See https://github.com/angular/protractor/issues/480

return;

}

templateSelect.click();

templateSelect.element.all(by.css('option')).get(2).click();

expect(includeElem.getText()).toMatch(/Content of template2.html/);

});

it('should change to blank', function() {

if (browser.params.browser == 'firefox') {

// Firefox can't handle using selects

return;

}

templateSelect.click();

templateSelect.element.all(by.css('option')).get(0).click();

expect(includeElem.isPresent()).toBe(false);

});

</file>

</example>

\*/

/\*\*

\* @ngdoc event

\* @name ngInclude#$includeContentRequested

\* @eventType emit on the scope ngInclude was declared in

\* @description

\* Emitted every time the ngInclude content is requested.

\*/

/\*\*

\* @ngdoc event

\* @name ngInclude#$includeContentLoaded

\* @eventType emit on the current ngInclude scope

\* @description

\* Emitted every time the ngInclude content is reloaded.

\*/

var ngIncludeDirective = ['$http', '$templateCache', '$anchorScroll', '$animate', '$sce',

function($http, $templateCache, $anchorScroll, $animate, $sce) {

return {

restrict: 'ECA',

priority: 400,

terminal: true,

transclude: 'element',

controller: angular.noop,

compile: function(element, attr) {

var srcExp = attr.ngInclude || attr.src,

onloadExp = attr.onload || '',

autoScrollExp = attr.autoscroll;

return function(scope, $element, $attr, ctrl, $transclude) {

var changeCounter = 0,

currentScope,

previousElement,

currentElement;

var cleanupLastIncludeContent = function() {

if(previousElement) {

previousElement.remove();

previousElement = null;

}

if(currentScope) {

currentScope.$destroy();

currentScope = null;

}

if(currentElement) {

$animate.leave(currentElement, function() {

previousElement = null;

});

previousElement = currentElement;

currentElement = null;

}

};

scope.$watch($sce.parseAsResourceUrl(srcExp), function ngIncludeWatchAction(src) {

var afterAnimation = function() {

if (isDefined(autoScrollExp) && (!autoScrollExp || scope.$eval(autoScrollExp))) {

$anchorScroll();

}

};

var thisChangeId = ++changeCounter;

if (src) {

$http.get(src, {cache: $templateCache}).success(function(response) {

if (thisChangeId !== changeCounter) return;

var newScope = scope.$new();

ctrl.template = response;

// Note: This will also link all children of ng-include that were contained in the original

// html. If that content contains controllers, ... they could pollute/change the scope.

// However, using ng-include on an element with additional content does not make sense...

// Note: We can't remove them in the cloneAttchFn of $transclude as that

// function is called before linking the content, which would apply child

// directives to non existing elements.

var clone = $transclude(newScope, function(clone) {

cleanupLastIncludeContent();

$animate.enter(clone, null, $element, afterAnimation);

});

currentScope = newScope;

currentElement = clone;

currentScope.$emit('$includeContentLoaded');

scope.$eval(onloadExp);

}).error(function() {

if (thisChangeId === changeCounter) cleanupLastIncludeContent();

});

scope.$emit('$includeContentRequested');

} else {

cleanupLastIncludeContent();

ctrl.template = null;

}

});

};

}

};

}];

// This directive is called during the $transclude call of the first `ngInclude` directive.

// It will replace and compile the content of the element with the loaded template.

// We need this directive so that the element content is already filled when

// the link function of another directive on the same element as ngInclude

// is called.

var ngIncludeFillContentDirective = ['$compile',

function($compile) {

return {

restrict: 'ECA',

priority: -400,

require: 'ngInclude',

link: function(scope, $element, $attr, ctrl) {

$element.html(ctrl.template);

$compile($element.contents())(scope);

}

};

}];

/\*\*

\* @ngdoc directive

\* @name ngInit

\* @restrict AC

\*

\* @description

\* The `ngInit` directive allows you to evaluate an expression in the

\* current scope.

\*

\* <div class="alert alert-error">

\* The only appropriate use of `ngInit` is for aliasing special properties of

\* {@link ng.directive:ngRepeat `ngRepeat`}, as seen in the demo below. Besides this case, you

\* should use {@link guide/controller controllers} rather than `ngInit`

\* to initialize values on a scope.

\* </div>

\* <div class="alert alert-warning">

\* \*\*Note\*\*: If you have assignment in `ngInit` along with {@link ng.$filter `$filter`}, make

\* sure you have parenthesis for correct precedence:

\* <pre class="prettyprint">

\* <div ng-init="test1 = (data | orderBy:'name')"></div>

\* </pre>

\* </div>

\*

\* @priority 450

\*

\* @element ANY

\* @param {expression} ngInit {@link guide/expression Expression} to eval.

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.list = [['a', 'b'], ['c', 'd']];

}

</script>

<div ng-controller="Ctrl">

<div ng-repeat="innerList in list" ng-init="outerIndex = $index">

<div ng-repeat="value in innerList" ng-init="innerIndex = $index">

<span class="example-init">list[ {{outerIndex}} ][ {{innerIndex}} ] = {{value}};</span>

</div>

</div>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should alias index positions', function() {

var elements = element.all(by.css('.example-init'));

expect(elements.get(0).getText()).toBe('list[ 0 ][ 0 ] = a;');

expect(elements.get(1).getText()).toBe('list[ 0 ][ 1 ] = b;');

expect(elements.get(2).getText()).toBe('list[ 1 ][ 0 ] = c;');

expect(elements.get(3).getText()).toBe('list[ 1 ][ 1 ] = d;');

});

</file>

</example>

\*/

var ngInitDirective = ngDirective({

priority: 450,

compile: function() {

return {

pre: function(scope, element, attrs) {

scope.$eval(attrs.ngInit);

}

};

}

});

/\*\*

\* @ngdoc directive

\* @name ngNonBindable

\* @restrict AC

\* @priority 1000

\*

\* @description

\* The `ngNonBindable` directive tells Angular not to compile or bind the contents of the current

\* DOM element. This is useful if the element contains what appears to be Angular directives and

\* bindings but which should be ignored by Angular. This could be the case if you have a site that

\* displays snippets of code, for instance.

\*

\* @element ANY

\*

\* @example

\* In this example there are two locations where a simple interpolation binding (`{{}}`) is present,

\* but the one wrapped in `ngNonBindable` is left alone.

\*

\* @example

<example>

<file name="index.html">

<div>Normal: {{1 + 2}}</div>

<div ng-non-bindable>Ignored: {{1 + 2}}</div>

</file>

<file name="protractor.js" type="protractor">

it('should check ng-non-bindable', function() {

expect(element(by.binding('1 + 2')).getText()).toContain('3');

expect(element.all(by.css('div')).last().getText()).toMatch(/1 \+ 2/);

});

</file>

</example>

\*/

var ngNonBindableDirective = ngDirective({ terminal: true, priority: 1000 });

/\*\*

\* @ngdoc directive

\* @name ngPluralize

\* @restrict EA

\*

\* @description

\* `ngPluralize` is a directive that displays messages according to en-US localization rules.

\* These rules are bundled with angular.js, but can be overridden

\* (see {@link guide/i18n Angular i18n} dev guide). You configure ngPluralize directive

\* by specifying the mappings between

\* [plural categories](http://unicode.org/repos/cldr-tmp/trunk/diff/supplemental/language\_plural\_rules.html)

\* and the strings to be displayed.

\*

\* # Plural categories and explicit number rules

\* There are two

\* [plural categories](http://unicode.org/repos/cldr-tmp/trunk/diff/supplemental/language\_plural\_rules.html)

\* in Angular's default en-US locale: "one" and "other".

\*

\* While a plural category may match many numbers (for example, in en-US locale, "other" can match

\* any number that is not 1), an explicit number rule can only match one number. For example, the

\* explicit number rule for "3" matches the number 3. There are examples of plural categories

\* and explicit number rules throughout the rest of this documentation.

\*

\* # Configuring ngPluralize

\* You configure ngPluralize by providing 2 attributes: `count` and `when`.

\* You can also provide an optional attribute, `offset`.

\*

\* The value of the `count` attribute can be either a string or an {@link guide/expression

\* Angular expression}; these are evaluated on the current scope for its bound value.

\*

\* The `when` attribute specifies the mappings between plural categories and the actual

\* string to be displayed. The value of the attribute should be a JSON object.

\*

\* The following example shows how to configure ngPluralize:

\*

\* ```html

\* <ng-pluralize count="personCount"

when="{'0': 'Nobody is viewing.',

\* 'one': '1 person is viewing.',

\* 'other': '{} people are viewing.'}">

\* </ng-pluralize>

\*```

\*

\* In the example, `"0: Nobody is viewing."` is an explicit number rule. If you did not

\* specify this rule, 0 would be matched to the "other" category and "0 people are viewing"

\* would be shown instead of "Nobody is viewing". You can specify an explicit number rule for

\* other numbers, for example 12, so that instead of showing "12 people are viewing", you can

\* show "a dozen people are viewing".

\*

\* You can use a set of closed braces (`{}`) as a placeholder for the number that you want substituted

\* into pluralized strings. In the previous example, Angular will replace `{}` with

\* <span ng-non-bindable>`{{personCount}}`</span>. The closed braces `{}` is a placeholder

\* for <span ng-non-bindable>{{numberExpression}}</span>.

\*

\* # Configuring ngPluralize with offset

\* The `offset` attribute allows further customization of pluralized text, which can result in

\* a better user experience. For example, instead of the message "4 people are viewing this document",

\* you might display "John, Kate and 2 others are viewing this document".

\* The offset attribute allows you to offset a number by any desired value.

\* Let's take a look at an example:

\*

\* ```html

\* <ng-pluralize count="personCount" offset=2

\* when="{'0': 'Nobody is viewing.',

\* '1': '{{person1}} is viewing.',

\* '2': '{{person1}} and {{person2}} are viewing.',

\* 'one': '{{person1}}, {{person2}} and one other person are viewing.',

\* 'other': '{{person1}}, {{person2}} and {} other people are viewing.'}">

\* </ng-pluralize>

\* ```

\*

\* Notice that we are still using two plural categories(one, other), but we added

\* three explicit number rules 0, 1 and 2.

\* When one person, perhaps John, views the document, "John is viewing" will be shown.

\* When three people view the document, no explicit number rule is found, so

\* an offset of 2 is taken off 3, and Angular uses 1 to decide the plural category.

\* In this case, plural category 'one' is matched and "John, Marry and one other person are viewing"

\* is shown.

\*

\* Note that when you specify offsets, you must provide explicit number rules for

\* numbers from 0 up to and including the offset. If you use an offset of 3, for example,

\* you must provide explicit number rules for 0, 1, 2 and 3. You must also provide plural strings for

\* plural categories "one" and "other".

\*

\* @param {string|expression} count The variable to be bound to.

\* @param {string} when The mapping between plural category to its corresponding strings.

\* @param {number=} offset Offset to deduct from the total number.

\*

\* @example

<example>

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.person1 = 'Igor';

$scope.person2 = 'Misko';

$scope.personCount = 1;

}

</script>

<div ng-controller="Ctrl">

Person 1:<input type="text" ng-model="person1" value="Igor" /><br/>

Person 2:<input type="text" ng-model="person2" value="Misko" /><br/>

Number of People:<input type="text" ng-model="personCount" value="1" /><br/>

<!--- Example with simple pluralization rules for en locale --->

Without Offset:

<ng-pluralize count="personCount"

when="{'0': 'Nobody is viewing.',

'one': '1 person is viewing.',

'other': '{} people are viewing.'}">

</ng-pluralize><br>

<!--- Example with offset --->

With Offset(2):

<ng-pluralize count="personCount" offset=2

when="{'0': 'Nobody is viewing.',

'1': '{{person1}} is viewing.',

'2': '{{person1}} and {{person2}} are viewing.',

'one': '{{person1}}, {{person2}} and one other person are viewing.',

'other': '{{person1}}, {{person2}} and {} other people are viewing.'}">

</ng-pluralize>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should show correct pluralized string', function() {

var withoutOffset = element.all(by.css('ng-pluralize')).get(0);

var withOffset = element.all(by.css('ng-pluralize')).get(1);

var countInput = element(by.model('personCount'));

expect(withoutOffset.getText()).toEqual('1 person is viewing.');

expect(withOffset.getText()).toEqual('Igor is viewing.');

countInput.clear();

countInput.sendKeys('0');

expect(withoutOffset.getText()).toEqual('Nobody is viewing.');

expect(withOffset.getText()).toEqual('Nobody is viewing.');

countInput.clear();

countInput.sendKeys('2');

expect(withoutOffset.getText()).toEqual('2 people are viewing.');

expect(withOffset.getText()).toEqual('Igor and Misko are viewing.');

countInput.clear();

countInput.sendKeys('3');

expect(withoutOffset.getText()).toEqual('3 people are viewing.');

expect(withOffset.getText()).toEqual('Igor, Misko and one other person are viewing.');

countInput.clear();

countInput.sendKeys('4');

expect(withoutOffset.getText()).toEqual('4 people are viewing.');

expect(withOffset.getText()).toEqual('Igor, Misko and 2 other people are viewing.');

});

it('should show data-bound names', function() {

var withOffset = element.all(by.css('ng-pluralize')).get(1);

var personCount = element(by.model('personCount'));

var person1 = element(by.model('person1'));

var person2 = element(by.model('person2'));

personCount.clear();

personCount.sendKeys('4');

person1.clear();

person1.sendKeys('Di');

person2.clear();

person2.sendKeys('Vojta');

expect(withOffset.getText()).toEqual('Di, Vojta and 2 other people are viewing.');

});

</file>

</example>

\*/

var ngPluralizeDirective = ['$locale', '$interpolate', function($locale, $interpolate) {

var BRACE = /{}/g;

return {

restrict: 'EA',

link: function(scope, element, attr) {

var numberExp = attr.count,

whenExp = attr.$attr.when && element.attr(attr.$attr.when), // we have {{}} in attrs

offset = attr.offset || 0,

whens = scope.$eval(whenExp) || {},

whensExpFns = {},

startSymbol = $interpolate.startSymbol(),

endSymbol = $interpolate.endSymbol(),

isWhen = /^when(Minus)?(.+)$/;

forEach(attr, function(expression, attributeName) {

if (isWhen.test(attributeName)) {

whens[lowercase(attributeName.replace('when', '').replace('Minus', '-'))] =

element.attr(attr.$attr[attributeName]);

}

});

forEach(whens, function(expression, key) {

whensExpFns[key] =

$interpolate(expression.replace(BRACE, startSymbol + numberExp + '-' +

offset + endSymbol));

});

scope.$watch(function ngPluralizeWatch() {

var value = parseFloat(scope.$eval(numberExp));

if (!isNaN(value)) {

//if explicit number rule such as 1, 2, 3... is defined, just use it. Otherwise,

//check it against pluralization rules in $locale service

if (!(value in whens)) value = $locale.pluralCat(value - offset);

return whensExpFns[value](scope, element, true);

} else {

return '';

}

}, function ngPluralizeWatchAction(newVal) {

element.text(newVal);

});

}

};

}];

/\*\*

\* @ngdoc directive

\* @name ngRepeat

\*

\* @description

\* The `ngRepeat` directive instantiates a template once per item from a collection. Each template

\* instance gets its own scope, where the given loop variable is set to the current collection item,

\* and `$index` is set to the item index or key.

\*

\* Special properties are exposed on the local scope of each template instance, including:

\*

\* | Variable | Type | Details |

\* |-----------|-----------------|-----------------------------------------------------------------------------|

\* | `$index` | {@type number} | iterator offset of the repeated element (0..length-1) |

\* | `$first` | {@type boolean} | true if the repeated element is first in the iterator. |

\* | `$middle` | {@type boolean} | true if the repeated element is between the first and last in the iterator. |

\* | `$last` | {@type boolean} | true if the repeated element is last in the iterator. |

\* | `$even` | {@type boolean} | true if the iterator position `$index` is even (otherwise false). |

\* | `$odd` | {@type boolean} | true if the iterator position `$index` is odd (otherwise false). |

\*

\* Creating aliases for these properties is possible with {@link ng.directive:ngInit `ngInit`}.

\* This may be useful when, for instance, nesting ngRepeats.

\*

\* # Special repeat start and end points

\* To repeat a series of elements instead of just one parent element, ngRepeat (as well as other ng directives) supports extending

\* the range of the repeater by defining explicit start and end points by using \*\*ng-repeat-start\*\* and \*\*ng-repeat-end\*\* respectively.

\* The \*\*ng-repeat-start\*\* directive works the same as \*\*ng-repeat\*\*, but will repeat all the HTML code (including the tag it's defined on)

\* up to and including the ending HTML tag where \*\*ng-repeat-end\*\* is placed.

\*

\* The example below makes use of this feature:

\* ```html

\* <header ng-repeat-start="item in items">

\* Header {{ item }}

\* </header>

\* <div class="body">

\* Body {{ item }}

\* </div>

\* <footer ng-repeat-end>

\* Footer {{ item }}

\* </footer>

\* ```

\*

\* And with an input of {@type ['A','B']} for the items variable in the example above, the output will evaluate to:

\* ```html

\* <header>

\* Header A

\* </header>

\* <div class="body">

\* Body A

\* </div>

\* <footer>

\* Footer A

\* </footer>

\* <header>

\* Header B

\* </header>

\* <div class="body">

\* Body B

\* </div>

\* <footer>

\* Footer B

\* </footer>

\* ```

\*

\* The custom start and end points for ngRepeat also support all other HTML directive syntax flavors provided in AngularJS (such

\* as \*\*data-ng-repeat-start\*\*, \*\*x-ng-repeat-start\*\* and \*\*ng:repeat-start\*\*).

\*

\* @animations

\* \*\*.enter\*\* - when a new item is added to the list or when an item is revealed after a filter

\*

\* \*\*.leave\*\* - when an item is removed from the list or when an item is filtered out

\*

\* \*\*.move\*\* - when an adjacent item is filtered out causing a reorder or when the item contents are reordered

\*

\* @element ANY

\* @scope

\* @priority 1000

\* @param {repeat\_expression} ngRepeat The expression indicating how to enumerate a collection. These

\* formats are currently supported:

\*

\* \* `variable in expression` – where variable is the user defined loop variable and `expression`

\* is a scope expression giving the collection to enumerate.

\*

\* For example: `album in artist.albums`.

\*

\* \* `(key, value) in expression` – where `key` and `value` can be any user defined identifiers,

\* and `expression` is the scope expression giving the collection to enumerate.

\*

\* For example: `(name, age) in {'adam':10, 'amalie':12}`.

\*

\* \* `variable in expression track by tracking\_expression` – You can also provide an optional tracking function

\* which can be used to associate the objects in the collection with the DOM elements. If no tracking function

\* is specified the ng-repeat associates elements by identity in the collection. It is an error to have

\* more than one tracking function to resolve to the same key. (This would mean that two distinct objects are

\* mapped to the same DOM element, which is not possible.) Filters should be applied to the expression,

\* before specifying a tracking expression.

\*

\* For example: `item in items` is equivalent to `item in items track by $id(item)'. This implies that the DOM elements

\* will be associated by item identity in the array.

\*

\* For example: `item in items track by $id(item)`. A built in `$id()` function can be used to assign a unique

\* `$$hashKey` property to each item in the array. This property is then used as a key to associated DOM elements

\* with the corresponding item in the array by identity. Moving the same object in array would move the DOM

\* element in the same way in the DOM.

\*

\* For example: `item in items track by item.id` is a typical pattern when the items come from the database. In this

\* case the object identity does not matter. Two objects are considered equivalent as long as their `id`

\* property is same.

\*

\* For example: `item in items | filter:searchText track by item.id` is a pattern that might be used to apply a filter

\* to items in conjunction with a tracking expression.

\*

\* @example

\* This example initializes the scope to a list of names and

\* then uses `ngRepeat` to display every person:

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

<div ng-init="friends = [

{name:'John', age:25, gender:'boy'},

{name:'Jessie', age:30, gender:'girl'},

{name:'Johanna', age:28, gender:'girl'},

{name:'Joy', age:15, gender:'girl'},

{name:'Mary', age:28, gender:'girl'},

{name:'Peter', age:95, gender:'boy'},

{name:'Sebastian', age:50, gender:'boy'},

{name:'Erika', age:27, gender:'girl'},

{name:'Patrick', age:40, gender:'boy'},

{name:'Samantha', age:60, gender:'girl'}

]">

I have {{friends.length}} friends. They are:

<input type="search" ng-model="q" placeholder="filter friends..." />

<ul class="example-animate-container">

<li class="animate-repeat" ng-repeat="friend in friends | filter:q">

[{{$index + 1}}] {{friend.name}} who is {{friend.age}} years old.

</li>

</ul>

</div>

</file>

<file name="animations.css">

.example-animate-container {

background:white;

border:1px solid black;

list-style:none;

margin:0;

padding:0 10px;

}

.animate-repeat {

line-height:40px;

list-style:none;

box-sizing:border-box;

}

.animate-repeat.ng-move,

.animate-repeat.ng-enter,

.animate-repeat.ng-leave {

-webkit-transition:all linear 0.5s;

transition:all linear 0.5s;

}

.animate-repeat.ng-leave.ng-leave-active,

.animate-repeat.ng-move,

.animate-repeat.ng-enter {

opacity:0;

max-height:0;

}

.animate-repeat.ng-leave,

.animate-repeat.ng-move.ng-move-active,

.animate-repeat.ng-enter.ng-enter-active {

opacity:1;

max-height:40px;

}

</file>

<file name="protractor.js" type="protractor">

var friends = element.all(by.repeater('friend in friends'));

it('should render initial data set', function() {

expect(friends.count()).toBe(10);

expect(friends.get(0).getText()).toEqual('[1] John who is 25 years old.');

expect(friends.get(1).getText()).toEqual('[2] Jessie who is 30 years old.');

expect(friends.last().getText()).toEqual('[10] Samantha who is 60 years old.');

expect(element(by.binding('friends.length')).getText())

.toMatch("I have 10 friends. They are:");

});

it('should update repeater when filter predicate changes', function() {

expect(friends.count()).toBe(10);

element(by.model('q')).sendKeys('ma');

expect(friends.count()).toBe(2);

expect(friends.get(0).getText()).toEqual('[1] Mary who is 28 years old.');

expect(friends.last().getText()).toEqual('[2] Samantha who is 60 years old.');

});

</file>

</example>

\*/

var ngRepeatDirective = ['$parse', '$animate', function($parse, $animate) {

var NG\_REMOVED = '$$NG\_REMOVED';

var ngRepeatMinErr = minErr('ngRepeat');

return {

transclude: 'element',

priority: 1000,

terminal: true,

$$tlb: true,

link: function($scope, $element, $attr, ctrl, $transclude){

var expression = $attr.ngRepeat;

var match = expression.match(/^\s\*([\s\S]+?)\s+in\s+([\s\S]+?)(?:\s+track\s+by\s+([\s\S]+?))?\s\*$/),

trackByExp, trackByExpGetter, trackByIdExpFn, trackByIdArrayFn, trackByIdObjFn,

lhs, rhs, valueIdentifier, keyIdentifier,

hashFnLocals = {$id: hashKey};

if (!match) {

throw ngRepeatMinErr('iexp', "Expected expression in form of '\_item\_ in \_collection\_[ track by \_id\_]' but got '{0}'.",

expression);

}

lhs = match[1];

rhs = match[2];

trackByExp = match[3];

if (trackByExp) {

trackByExpGetter = $parse(trackByExp);

trackByIdExpFn = function(key, value, index) {

// assign key, value, and $index to the locals so that they can be used in hash functions

if (keyIdentifier) hashFnLocals[keyIdentifier] = key;

hashFnLocals[valueIdentifier] = value;

hashFnLocals.$index = index;

return trackByExpGetter($scope, hashFnLocals);

};

} else {

trackByIdArrayFn = function(key, value) {

return hashKey(value);

};

trackByIdObjFn = function(key) {

return key;

};

}

match = lhs.match(/^(?:([\$\w]+)|\(([\$\w]+)\s\*,\s\*([\$\w]+)\))$/);

if (!match) {

throw ngRepeatMinErr('iidexp', "'\_item\_' in '\_item\_ in \_collection\_' should be an identifier or '(\_key\_, \_value\_)' expression, but got '{0}'.",

lhs);

}

valueIdentifier = match[3] || match[1];

keyIdentifier = match[2];

// Store a list of elements from previous run. This is a hash where key is the item from the

// iterator, and the value is objects with following properties.

// - scope: bound scope

// - element: previous element.

// - index: position

var lastBlockMap = {};

//watch props

$scope.$watchCollection(rhs, function ngRepeatAction(collection){

var index, length,

previousNode = $element[0], // current position of the node

nextNode,

// Same as lastBlockMap but it has the current state. It will become the

// lastBlockMap on the next iteration.

nextBlockMap = {},

arrayLength,

childScope,

key, value, // key/value of iteration

trackById,

trackByIdFn,

collectionKeys,

block, // last object information {scope, element, id}

nextBlockOrder = [],

elementsToRemove;

if (isArrayLike(collection)) {

collectionKeys = collection;

trackByIdFn = trackByIdExpFn || trackByIdArrayFn;

} else {

trackByIdFn = trackByIdExpFn || trackByIdObjFn;

// if object, extract keys, sort them and use to determine order of iteration over obj props

collectionKeys = [];

for (key in collection) {

if (collection.hasOwnProperty(key) && key.charAt(0) != '$') {

collectionKeys.push(key);

}

}

collectionKeys.sort();

}

arrayLength = collectionKeys.length;

// locate existing items

length = nextBlockOrder.length = collectionKeys.length;

for(index = 0; index < length; index++) {

key = (collection === collectionKeys) ? index : collectionKeys[index];

value = collection[key];

trackById = trackByIdFn(key, value, index);

assertNotHasOwnProperty(trackById, '`track by` id');

if(lastBlockMap.hasOwnProperty(trackById)) {

block = lastBlockMap[trackById];

delete lastBlockMap[trackById];

nextBlockMap[trackById] = block;

nextBlockOrder[index] = block;

} else if (nextBlockMap.hasOwnProperty(trackById)) {

// restore lastBlockMap

forEach(nextBlockOrder, function(block) {

if (block && block.scope) lastBlockMap[block.id] = block;

});

// This is a duplicate and we need to throw an error

throw ngRepeatMinErr('dupes', "Duplicates in a repeater are not allowed. Use 'track by' expression to specify unique keys. Repeater: {0}, Duplicate key: {1}",

expression, trackById);

} else {

// new never before seen block

nextBlockOrder[index] = { id: trackById };

nextBlockMap[trackById] = false;

}

}

// remove existing items

for (key in lastBlockMap) {

// lastBlockMap is our own object so we don't need to use special hasOwnPropertyFn

if (lastBlockMap.hasOwnProperty(key)) {

block = lastBlockMap[key];

elementsToRemove = getBlockElements(block.clone);

$animate.leave(elementsToRemove);

forEach(elementsToRemove, function(element) { element[NG\_REMOVED] = true; });

block.scope.$destroy();

}

}

// we are not using forEach for perf reasons (trying to avoid #call)

for (index = 0, length = collectionKeys.length; index < length; index++) {

key = (collection === collectionKeys) ? index : collectionKeys[index];

value = collection[key];

block = nextBlockOrder[index];

if (nextBlockOrder[index - 1]) previousNode = getBlockEnd(nextBlockOrder[index - 1]);

if (block.scope) {

// if we have already seen this object, then we need to reuse the

// associated scope/element

childScope = block.scope;

nextNode = previousNode;

do {

nextNode = nextNode.nextSibling;

} while(nextNode && nextNode[NG\_REMOVED]);

if (getBlockStart(block) != nextNode) {

// existing item which got moved

$animate.move(getBlockElements(block.clone), null, jqLite(previousNode));

}

previousNode = getBlockEnd(block);

} else {

// new item which we don't know about

childScope = $scope.$new();

}

childScope[valueIdentifier] = value;

if (keyIdentifier) childScope[keyIdentifier] = key;

childScope.$index = index;

childScope.$first = (index === 0);

childScope.$last = (index === (arrayLength - 1));

childScope.$middle = !(childScope.$first || childScope.$last);

// jshint bitwise: false

childScope.$odd = !(childScope.$even = (index&1) === 0);

// jshint bitwise: true

if (!block.scope) {

$transclude(childScope, function(clone) {

clone[clone.length++] = document.createComment(' end ngRepeat: ' + expression + ' ');

$animate.enter(clone, null, jqLite(previousNode));

previousNode = clone;

block.scope = childScope;

// Note: We only need the first/last node of the cloned nodes.

// However, we need to keep the reference to the jqlite wrapper as it might be changed later

// by a directive with templateUrl when it's template arrives.

block.clone = clone;

nextBlockMap[block.id] = block;

});

}

}

lastBlockMap = nextBlockMap;

});

}

};

function getBlockStart(block) {

return block.clone[0];

}

function getBlockEnd(block) {

return block.clone[block.clone.length - 1];

}

}];

/\*\*

\* @ngdoc directive

\* @name ngShow

\*

\* @description

\* The `ngShow` directive shows or hides the given HTML element based on the expression

\* provided to the ngShow attribute. The element is shown or hidden by removing or adding

\* the `ng-hide` CSS class onto the element. The `.ng-hide` CSS class is predefined

\* in AngularJS and sets the display style to none (using an !important flag).

\* For CSP mode please add `angular-csp.css` to your html file (see {@link ng.directive:ngCsp ngCsp}).

\*

\* ```html

\* <!-- when $scope.myValue is truthy (element is visible) -->

\* <div ng-show="myValue"></div>

\*

\* <!-- when $scope.myValue is falsy (element is hidden) -->

\* <div ng-show="myValue" class="ng-hide"></div>

\* ```

\*

\* When the ngShow expression evaluates to false then the ng-hide CSS class is added to the class attribute

\* on the element causing it to become hidden. When true, the ng-hide CSS class is removed

\* from the element causing the element not to appear hidden.

\*

\* ## Why is !important used?

\*

\* You may be wondering why !important is used for the .ng-hide CSS class. This is because the `.ng-hide` selector

\* can be easily overridden by heavier selectors. For example, something as simple

\* as changing the display style on a HTML list item would make hidden elements appear visible.

\* This also becomes a bigger issue when dealing with CSS frameworks.

\*

\* By using !important, the show and hide behavior will work as expected despite any clash between CSS selector

\* specificity (when !important isn't used with any conflicting styles). If a developer chooses to override the

\* styling to change how to hide an element then it is just a matter of using !important in their own CSS code.

\*

\* ### Overriding .ng-hide

\*

\* If you wish to change the hide behavior with ngShow/ngHide then this can be achieved by

\* restating the styles for the .ng-hide class in CSS:

\* ```css

\* .ng-hide {

\* //!annotate CSS Specificity|Not to worry, this will override the AngularJS default...

\* display:block!important;

\*

\* //this is just another form of hiding an element

\* position:absolute;

\* top:-9999px;

\* left:-9999px;

\* }

\* ```

\*

\* Just remember to include the important flag so the CSS override will function.

\*

\* <div class="alert alert-warning">

\* \*\*Note:\*\* Here is a list of values that ngShow will consider as a falsy value (case insensitive):<br />

\* "f" / "0" / "false" / "no" / "n" / "[]"

\* </div>

\*

\* ## A note about animations with ngShow

\*

\* Animations in ngShow/ngHide work with the show and hide events that are triggered when the directive expression

\* is true and false. This system works like the animation system present with ngClass except that

\* you must also include the !important flag to override the display property

\* so that you can perform an animation when the element is hidden during the time of the animation.

\*

\* ```css

\* //

\* //a working example can be found at the bottom of this page

\* //

\* .my-element.ng-hide-add, .my-element.ng-hide-remove {

\* transition:0.5s linear all;

\* display:block!important;

\* }

\*

\* .my-element.ng-hide-add { ... }

\* .my-element.ng-hide-add.ng-hide-add-active { ... }

\* .my-element.ng-hide-remove { ... }

\* .my-element.ng-hide-remove.ng-hide-remove-active { ... }

\* ```

\*

\* @animations

\* addClass: .ng-hide - happens after the ngShow expression evaluates to a truthy value and the just before contents are set to visible

\* removeClass: .ng-hide - happens after the ngShow expression evaluates to a non truthy value and just before the contents are set to hidden

\*

\* @element ANY

\* @param {expression} ngShow If the {@link guide/expression expression} is truthy

\* then the element is shown or hidden respectively.

\*

\* @example

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

Click me: <input type="checkbox" ng-model="checked"><br/>

<div>

Show:

<div class="check-element animate-show" ng-show="checked">

<span class="glyphicon glyphicon-thumbs-up"></span> I show up when your checkbox is checked.

</div>

</div>

<div>

Hide:

<div class="check-element animate-show" ng-hide="checked">

<span class="glyphicon glyphicon-thumbs-down"></span> I hide when your checkbox is checked.

</div>

</div>

</file>

<file name="glyphicons.css">

@import url(//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap-glyphicons.css);

</file>

<file name="animations.css">

.animate-show {

-webkit-transition:all linear 0.5s;

transition:all linear 0.5s;

line-height:20px;

opacity:1;

padding:10px;

border:1px solid black;

background:white;

}

.animate-show.ng-hide-add,

.animate-show.ng-hide-remove {

display:block!important;

}

.animate-show.ng-hide {

line-height:0;

opacity:0;

padding:0 10px;

}

.check-element {

padding:10px;

border:1px solid black;

background:white;

}

</file>

<file name="protractor.js" type="protractor">

var thumbsUp = element(by.css('span.glyphicon-thumbs-up'));

var thumbsDown = element(by.css('span.glyphicon-thumbs-down'));

it('should check ng-show / ng-hide', function() {

expect(thumbsUp.isDisplayed()).toBeFalsy();

expect(thumbsDown.isDisplayed()).toBeTruthy();

element(by.model('checked')).click();

expect(thumbsUp.isDisplayed()).toBeTruthy();

expect(thumbsDown.isDisplayed()).toBeFalsy();

});

</file>

</example>

\*/

var ngShowDirective = ['$animate', function($animate) {

return function(scope, element, attr) {

scope.$watch(attr.ngShow, function ngShowWatchAction(value){

$animate[toBoolean(value) ? 'removeClass' : 'addClass'](element, 'ng-hide');

});

};

}];

/\*\*

\* @ngdoc directive

\* @name ngHide

\*

\* @description

\* The `ngHide` directive shows or hides the given HTML element based on the expression

\* provided to the ngHide attribute. The element is shown or hidden by removing or adding

\* the `ng-hide` CSS class onto the element. The `.ng-hide` CSS class is predefined

\* in AngularJS and sets the display style to none (using an !important flag).

\* For CSP mode please add `angular-csp.css` to your html file (see {@link ng.directive:ngCsp ngCsp}).

\*

\* ```html

\* <!-- when $scope.myValue is truthy (element is hidden) -->

\* <div ng-hide="myValue"></div>

\*

\* <!-- when $scope.myValue is falsy (element is visible) -->

\* <div ng-hide="myValue" class="ng-hide"></div>

\* ```

\*

\* When the ngHide expression evaluates to true then the .ng-hide CSS class is added to the class attribute

\* on the element causing it to become hidden. When false, the ng-hide CSS class is removed

\* from the element causing the element not to appear hidden.

\*

\* ## Why is !important used?

\*

\* You may be wondering why !important is used for the .ng-hide CSS class. This is because the `.ng-hide` selector

\* can be easily overridden by heavier selectors. For example, something as simple

\* as changing the display style on a HTML list item would make hidden elements appear visible.

\* This also becomes a bigger issue when dealing with CSS frameworks.

\*

\* By using !important, the show and hide behavior will work as expected despite any clash between CSS selector

\* specificity (when !important isn't used with any conflicting styles). If a developer chooses to override the

\* styling to change how to hide an element then it is just a matter of using !important in their own CSS code.

\*

\* ### Overriding .ng-hide

\*

\* If you wish to change the hide behavior with ngShow/ngHide then this can be achieved by

\* restating the styles for the .ng-hide class in CSS:

\* ```css

\* .ng-hide {

\* //!annotate CSS Specificity|Not to worry, this will override the AngularJS default...

\* display:block!important;

\*

\* //this is just another form of hiding an element

\* position:absolute;

\* top:-9999px;

\* left:-9999px;

\* }

\* ```

\*

\* Just remember to include the important flag so the CSS override will function.

\*

\* <div class="alert alert-warning">

\* \*\*Note:\*\* Here is a list of values that ngHide will consider as a falsy value (case insensitive):<br />

\* "f" / "0" / "false" / "no" / "n" / "[]"

\* </div>

\*

\* ## A note about animations with ngHide

\*

\* Animations in ngShow/ngHide work with the show and hide events that are triggered when the directive expression

\* is true and false. This system works like the animation system present with ngClass, except that

\* you must also include the !important flag to override the display property so

\* that you can perform an animation when the element is hidden during the time of the animation.

\*

\* ```css

\* //

\* //a working example can be found at the bottom of this page

\* //

\* .my-element.ng-hide-add, .my-element.ng-hide-remove {

\* transition:0.5s linear all;

\* display:block!important;

\* }

\*

\* .my-element.ng-hide-add { ... }

\* .my-element.ng-hide-add.ng-hide-add-active { ... }

\* .my-element.ng-hide-remove { ... }

\* .my-element.ng-hide-remove.ng-hide-remove-active { ... }

\* ```

\*

\* @animations

\* removeClass: .ng-hide - happens after the ngHide expression evaluates to a truthy value and just before the contents are set to hidden

\* addClass: .ng-hide - happens after the ngHide expression evaluates to a non truthy value and just before the contents are set to visible

\*

\* @element ANY

\* @param {expression} ngHide If the {@link guide/expression expression} is truthy then

\* the element is shown or hidden respectively.

\*

\* @example

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

Click me: <input type="checkbox" ng-model="checked"><br/>

<div>

Show:

<div class="check-element animate-hide" ng-show="checked">

<span class="glyphicon glyphicon-thumbs-up"></span> I show up when your checkbox is checked.

</div>

</div>

<div>

Hide:

<div class="check-element animate-hide" ng-hide="checked">

<span class="glyphicon glyphicon-thumbs-down"></span> I hide when your checkbox is checked.

</div>

</div>

</file>

<file name="glyphicons.css">

@import url(//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap-glyphicons.css);

</file>

<file name="animations.css">

.animate-hide {

-webkit-transition:all linear 0.5s;

transition:all linear 0.5s;

line-height:20px;

opacity:1;

padding:10px;

border:1px solid black;

background:white;

}

.animate-hide.ng-hide-add,

.animate-hide.ng-hide-remove {

display:block!important;

}

.animate-hide.ng-hide {

line-height:0;

opacity:0;

padding:0 10px;

}

.check-element {

padding:10px;

border:1px solid black;

background:white;

}

</file>

<file name="protractor.js" type="protractor">

var thumbsUp = element(by.css('span.glyphicon-thumbs-up'));

var thumbsDown = element(by.css('span.glyphicon-thumbs-down'));

it('should check ng-show / ng-hide', function() {

expect(thumbsUp.isDisplayed()).toBeFalsy();

expect(thumbsDown.isDisplayed()).toBeTruthy();

element(by.model('checked')).click();

expect(thumbsUp.isDisplayed()).toBeTruthy();

expect(thumbsDown.isDisplayed()).toBeFalsy();

});

</file>

</example>

\*/

var ngHideDirective = ['$animate', function($animate) {

return function(scope, element, attr) {

scope.$watch(attr.ngHide, function ngHideWatchAction(value){

$animate[toBoolean(value) ? 'addClass' : 'removeClass'](element, 'ng-hide');

});

};

}];

/\*\*

\* @ngdoc directive

\* @name ngStyle

\* @restrict AC

\*

\* @description

\* The `ngStyle` directive allows you to set CSS style on an HTML element conditionally.

\*

\* @element ANY

\* @param {expression} ngStyle {@link guide/expression Expression} which evals to an

\* object whose keys are CSS style names and values are corresponding values for those CSS

\* keys.

\*

\* @example

<example>

<file name="index.html">

<input type="button" value="set" ng-click="myStyle={color:'red'}">

<input type="button" value="clear" ng-click="myStyle={}">

<br/>

<span ng-style="myStyle">Sample Text</span>

<pre>myStyle={{myStyle}}</pre>

</file>

<file name="style.css">

span {

color: black;

}

</file>

<file name="protractor.js" type="protractor">

var colorSpan = element(by.css('span'));

it('should check ng-style', function() {

expect(colorSpan.getCssValue('color')).toBe('rgba(0, 0, 0, 1)');

element(by.css('input[value=set]')).click();

expect(colorSpan.getCssValue('color')).toBe('rgba(255, 0, 0, 1)');

element(by.css('input[value=clear]')).click();

expect(colorSpan.getCssValue('color')).toBe('rgba(0, 0, 0, 1)');

});

</file>

</example>

\*/

var ngStyleDirective = ngDirective(function(scope, element, attr) {

scope.$watch(attr.ngStyle, function ngStyleWatchAction(newStyles, oldStyles) {

if (oldStyles && (newStyles !== oldStyles)) {

forEach(oldStyles, function(val, style) { element.css(style, '');});

}

if (newStyles) element.css(newStyles);

}, true);

});

/\*\*

\* @ngdoc directive

\* @name ngSwitch

\* @restrict EA

\*

\* @description

\* The `ngSwitch` directive is used to conditionally swap DOM structure on your template based on a scope expression.

\* Elements within `ngSwitch` but without `ngSwitchWhen` or `ngSwitchDefault` directives will be preserved at the location

\* as specified in the template.

\*

\* The directive itself works similar to ngInclude, however, instead of downloading template code (or loading it

\* from the template cache), `ngSwitch` simply chooses one of the nested elements and makes it visible based on which element

\* matches the value obtained from the evaluated expression. In other words, you define a container element

\* (where you place the directive), place an expression on the \*\*`on="..."` attribute\*\*

\* (or the \*\*`ng-switch="..."` attribute\*\*), define any inner elements inside of the directive and place

\* a when attribute per element. The when attribute is used to inform ngSwitch which element to display when the on

\* expression is evaluated. If a matching expression is not found via a when attribute then an element with the default

\* attribute is displayed.

\*

\* <div class="alert alert-info">

\* Be aware that the attribute values to match against cannot be expressions. They are interpreted

\* as literal string values to match against.

\* For example, \*\*`ng-switch-when="someVal"`\*\* will match against the string `"someVal"` not against the

\* value of the expression `$scope.someVal`.

\* </div>

\* @animations

\* enter - happens after the ngSwitch contents change and the matched child element is placed inside the container

\* leave - happens just after the ngSwitch contents change and just before the former contents are removed from the DOM

\*

\* @usage

\* <ANY ng-switch="expression">

\* <ANY ng-switch-when="matchValue1">...</ANY>

\* <ANY ng-switch-when="matchValue2">...</ANY>

\* <ANY ng-switch-default>...</ANY>

\* </ANY>

\*

\*

\* @scope

\* @priority 800

\* @param {\*} ngSwitch|on expression to match against <tt>ng-switch-when</tt>.

\* On child elements add:

\*

\* \* `ngSwitchWhen`: the case statement to match against. If match then this

\* case will be displayed. If the same match appears multiple times, all the

\* elements will be displayed.

\* \* `ngSwitchDefault`: the default case when no other case match. If there

\* are multiple default cases, all of them will be displayed when no other

\* case match.

\*

\*

\* @example

<example module="ngAnimate" deps="angular-animate.js" animations="true">

<file name="index.html">

<div ng-controller="Ctrl">

<select ng-model="selection" ng-options="item for item in items">

</select>

<tt>selection={{selection}}</tt>

<hr/>

<div class="animate-switch-container"

ng-switch on="selection">

<div class="animate-switch" ng-switch-when="settings">Settings Div</div>

<div class="animate-switch" ng-switch-when="home">Home Span</div>

<div class="animate-switch" ng-switch-default>default</div>

</div>

</div>

</file>

<file name="script.js">

function Ctrl($scope) {

$scope.items = ['settings', 'home', 'other'];

$scope.selection = $scope.items[0];

}

</file>

<file name="animations.css">

.animate-switch-container {

position:relative;

background:white;

border:1px solid black;

height:40px;

overflow:hidden;

}

.animate-switch {

padding:10px;

}

.animate-switch.ng-animate {

-webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;

position:absolute;

top:0;

left:0;

right:0;

bottom:0;

}

.animate-switch.ng-leave.ng-leave-active,

.animate-switch.ng-enter {

top:-50px;

}

.animate-switch.ng-leave,

.animate-switch.ng-enter.ng-enter-active {

top:0;

}

</file>

<file name="protractor.js" type="protractor">

var switchElem = element(by.css('[ng-switch]'));

var select = element(by.model('selection'));

it('should start in settings', function() {

expect(switchElem.getText()).toMatch(/Settings Div/);

});

it('should change to home', function() {

select.element.all(by.css('option')).get(1).click();

expect(switchElem.getText()).toMatch(/Home Span/);

});

it('should select default', function() {

select.element.all(by.css('option')).get(2).click();

expect(switchElem.getText()).toMatch(/default/);

});

</file>

</example>

\*/

var ngSwitchDirective = ['$animate', function($animate) {

return {

restrict: 'EA',

require: 'ngSwitch',

// asks for $scope to fool the BC controller module

controller: ['$scope', function ngSwitchController() {

this.cases = {};

}],

link: function(scope, element, attr, ngSwitchController) {

var watchExpr = attr.ngSwitch || attr.on,

selectedTranscludes,

selectedElements,

previousElements,

selectedScopes = [];

scope.$watch(watchExpr, function ngSwitchWatchAction(value) {

var i, ii = selectedScopes.length;

if(ii > 0) {

if(previousElements) {

for (i = 0; i < ii; i++) {

previousElements[i].remove();

}

previousElements = null;

}

previousElements = [];

for (i= 0; i<ii; i++) {

var selected = selectedElements[i];

selectedScopes[i].$destroy();

previousElements[i] = selected;

$animate.leave(selected, function() {

previousElements.splice(i, 1);

if(previousElements.length === 0) {

previousElements = null;

}

});

}

}

selectedElements = [];

selectedScopes = [];

if ((selectedTranscludes = ngSwitchController.cases['!' + value] || ngSwitchController.cases['?'])) {

scope.$eval(attr.change);

forEach(selectedTranscludes, function(selectedTransclude) {

var selectedScope = scope.$new();

selectedScopes.push(selectedScope);

selectedTransclude.transclude(selectedScope, function(caseElement) {

var anchor = selectedTransclude.element;

selectedElements.push(caseElement);

$animate.enter(caseElement, anchor.parent(), anchor);

});

});

}

});

}

};

}];

var ngSwitchWhenDirective = ngDirective({

transclude: 'element',

priority: 800,

require: '^ngSwitch',

link: function(scope, element, attrs, ctrl, $transclude) {

ctrl.cases['!' + attrs.ngSwitchWhen] = (ctrl.cases['!' + attrs.ngSwitchWhen] || []);

ctrl.cases['!' + attrs.ngSwitchWhen].push({ transclude: $transclude, element: element });

}

});

var ngSwitchDefaultDirective = ngDirective({

transclude: 'element',

priority: 800,

require: '^ngSwitch',

link: function(scope, element, attr, ctrl, $transclude) {

ctrl.cases['?'] = (ctrl.cases['?'] || []);

ctrl.cases['?'].push({ transclude: $transclude, element: element });

}

});

/\*\*

\* @ngdoc directive

\* @name ngTransclude

\* @restrict AC

\*

\* @description

\* Directive that marks the insertion point for the transcluded DOM of the nearest parent directive that uses transclusion.

\*

\* Any existing content of the element that this directive is placed on will be removed before the transcluded content is inserted.

\*

\* @element ANY

\*

\* @example

<example module="transclude">

<file name="index.html">

<script>

function Ctrl($scope) {

$scope.title = 'Lorem Ipsum';

$scope.text = 'Neque porro quisquam est qui dolorem ipsum quia dolor...';

}

angular.module('transclude', [])

.directive('pane', function(){

return {

restrict: 'E',

transclude: true,

scope: { title:'@' },

template: '<div style="border: 1px solid black;">' +

'<div style="background-color: gray">{{title}}</div>' +

'<div ng-transclude></div>' +

'</div>'

};

});

</script>

<div ng-controller="Ctrl">

<input ng-model="title"><br>

<textarea ng-model="text"></textarea> <br/>

<pane title="{{title}}">{{text}}</pane>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should have transcluded', function() {

var titleElement = element(by.model('title'));

titleElement.clear();

titleElement.sendKeys('TITLE');

var textElement = element(by.model('text'));

textElement.clear();

textElement.sendKeys('TEXT');

expect(element(by.binding('title')).getText()).toEqual('TITLE');

expect(element(by.binding('text')).getText()).toEqual('TEXT');

});

</file>

</example>

\*

\*/

var ngTranscludeDirective = ngDirective({

link: function($scope, $element, $attrs, controller, $transclude) {

if (!$transclude) {

throw minErr('ngTransclude')('orphan',

'Illegal use of ngTransclude directive in the template! ' +

'No parent directive that requires a transclusion found. ' +

'Element: {0}',

startingTag($element));

}

$transclude(function(clone) {

$element.empty();

$element.append(clone);

});

}

});

/\*\*

\* @ngdoc directive

\* @name script

\* @restrict E

\*

\* @description

\* Load the content of a `<script>` element into {@link ng.$templateCache `$templateCache`}, so that the

\* template can be used by {@link ng.directive:ngInclude `ngInclude`},

\* {@link ngRoute.directive:ngView `ngView`}, or {@link guide/directive directives}. The type of the

\* `<script>` element must be specified as `text/ng-template`, and a cache name for the template must be

\* assigned through the element's `id`, which can then be used as a directive's `templateUrl`.

\*

\* @param {string} type Must be set to `'text/ng-template'`.

\* @param {string} id Cache name of the template.

\*

\* @example

<example>

<file name="index.html">

<script type="text/ng-template" id="/tpl.html">

Content of the template.

</script>

<a ng-click="currentTpl='/tpl.html'" id="tpl-link">Load inlined template</a>

<div id="tpl-content" ng-include src="currentTpl"></div>

</file>

<file name="protractor.js" type="protractor">

it('should load template defined inside script tag', function() {

element(by.css('#tpl-link')).click();

expect(element(by.css('#tpl-content')).getText()).toMatch(/Content of the template/);

});

</file>

</example>

\*/

var scriptDirective = ['$templateCache', function($templateCache) {

return {

restrict: 'E',

terminal: true,

compile: function(element, attr) {

if (attr.type == 'text/ng-template') {

var templateUrl = attr.id,

// IE is not consistent, in scripts we have to read .text but in other nodes we have to read .textContent

text = element[0].text;

$templateCache.put(templateUrl, text);

}

}

};

}];

var ngOptionsMinErr = minErr('ngOptions');

/\*\*

\* @ngdoc directive

\* @name select

\* @restrict E

\*

\* @description

\* HTML `SELECT` element with angular data-binding.

\*

\* # `ngOptions`

\*

\* The `ngOptions` attribute can be used to dynamically generate a list of `<option>`

\* elements for the `<select>` element using the array or object obtained by evaluating the

\* `ngOptions` comprehension\_expression.

\*

\* When an item in the `<select>` menu is selected, the array element or object property

\* represented by the selected option will be bound to the model identified by the `ngModel`

\* directive.

\*

\* <div class="alert alert-warning">

\* \*\*Note:\*\* `ngModel` compares by reference, not value. This is important when binding to an

\* array of objects. See an example [in this jsfiddle](http://jsfiddle.net/qWzTb/).

\* </div>

\*

\* Optionally, a single hard-coded `<option>` element, with the value set to an empty string, can

\* be nested into the `<select>` element. This element will then represent the `null` or "not selected"

\* option. See example below for demonstration.

\*

\* <div class="alert alert-warning">

\* \*\*Note:\*\* `ngOptions` provides an iterator facility for the `<option>` element which should be used instead

\* of {@link ng.directive:ngRepeat ngRepeat} when you want the

\* `select` model to be bound to a non-string value. This is because an option element can only

\* be bound to string values at present.

\* </div>

\*

\* @param {string} ngModel Assignable angular expression to data-bind to.

\* @param {string=} name Property name of the form under which the control is published.

\* @param {string=} required The control is considered valid only if value is entered.

\* @param {string=} ngRequired Adds `required` attribute and `required` validation constraint to

\* the element when the ngRequired expression evaluates to true. Use `ngRequired` instead of

\* `required` when you want to data-bind to the `required` attribute.

\* @param {comprehension\_expression=} ngOptions in one of the following forms:

\*

\* \* for array data sources:

\* \* `label` \*\*`for`\*\* `value` \*\*`in`\*\* `array`

\* \* `select` \*\*`as`\*\* `label` \*\*`for`\*\* `value` \*\*`in`\*\* `array`

\* \* `label` \*\*`group by`\*\* `group` \*\*`for`\*\* `value` \*\*`in`\*\* `array`

\* \* `select` \*\*`as`\*\* `label` \*\*`group by`\*\* `group` \*\*`for`\*\* `value` \*\*`in`\*\* `array` \*\*`track by`\*\* `trackexpr`

\* \* for object data sources:

\* \* `label` \*\*`for (`\*\*`key` \*\*`,`\*\* `value`\*\*`) in`\*\* `object`

\* \* `select` \*\*`as`\*\* `label` \*\*`for (`\*\*`key` \*\*`,`\*\* `value`\*\*`) in`\*\* `object`

\* \* `label` \*\*`group by`\*\* `group` \*\*`for (`\*\*`key`\*\*`,`\*\* `value`\*\*`) in`\*\* `object`

\* \* `select` \*\*`as`\*\* `label` \*\*`group by`\*\* `group`

\* \*\*`for` `(`\*\*`key`\*\*`,`\*\* `value`\*\*`) in`\*\* `object`

\*

\* Where:

\*

\* \* `array` / `object`: an expression which evaluates to an array / object to iterate over.

\* \* `value`: local variable which will refer to each item in the `array` or each property value

\* of `object` during iteration.

\* \* `key`: local variable which will refer to a property name in `object` during iteration.

\* \* `label`: The result of this expression will be the label for `<option>` element. The

\* `expression` will most likely refer to the `value` variable (e.g. `value.propertyName`).

\* \* `select`: The result of this expression will be bound to the model of the parent `<select>`

\* element. If not specified, `select` expression will default to `value`.

\* \* `group`: The result of this expression will be used to group options using the `<optgroup>`

\* DOM element.

\* \* `trackexpr`: Used when working with an array of objects. The result of this expression will be

\* used to identify the objects in the array. The `trackexpr` will most likely refer to the

\* `value` variable (e.g. `value.propertyName`).

\*

\* @example

<example>

<file name="index.html">

<script>

function MyCntrl($scope) {

$scope.colors = [

{name:'black', shade:'dark'},

{name:'white', shade:'light'},

{name:'red', shade:'dark'},

{name:'blue', shade:'dark'},

{name:'yellow', shade:'light'}

];

$scope.color = $scope.colors[2]; // red

}

</script>

<div ng-controller="MyCntrl">

<ul>

<li ng-repeat="color in colors">

Name: <input ng-model="color.name">

[<a href ng-click="colors.splice($index, 1)">X</a>]

</li>

<li>

[<a href ng-click="colors.push({})">add</a>]

</li>

</ul>

<hr/>

Color (null not allowed):

<select ng-model="color" ng-options="c.name for c in colors"></select><br>

Color (null allowed):

<span class="nullable">

<select ng-model="color" ng-options="c.name for c in colors">

<option value="">-- choose color --</option>

</select>

</span><br/>

Color grouped by shade:

<select ng-model="color" ng-options="c.name group by c.shade for c in colors">

</select><br/>

Select <a href ng-click="color={name:'not in list'}">bogus</a>.<br>

<hr/>

Currently selected: {{ {selected\_color:color} }}

<div style="border:solid 1px black; height:20px"

ng-style="{'background-color':color.name}">

</div>

</div>

</file>

<file name="protractor.js" type="protractor">

it('should check ng-options', function() {

expect(element(by.binding('{selected\_color:color}')).getText()).toMatch('red');

element.all(by.select('color')).first().click();

element.all(by.css('select[ng-model="color"] option')).first().click();

expect(element(by.binding('{selected\_color:color}')).getText()).toMatch('black');

element(by.css('.nullable select[ng-model="color"]')).click();

element.all(by.css('.nullable select[ng-model="color"] option')).first().click();

expect(element(by.binding('{selected\_color:color}')).getText()).toMatch('null');

});

</file>

</example>

\*/

var ngOptionsDirective = valueFn({ terminal: true });

// jshint maxlen: false

var selectDirective = ['$compile', '$parse', function($compile, $parse) {

//000011111111110000000000022222222220000000000000000000003333333333000000000000004444444444444440000000005555555555555550000000666666666666666000000000000000777777777700000000000000000008888888888

var NG\_OPTIONS\_REGEXP = /^\s\*([\s\S]+?)(?:\s+as\s+([\s\S]+?))?(?:\s+group\s+by\s+([\s\S]+?))?\s+for\s+(?:([\$\w][\$\w]\*)|(?:\(\s\*([\$\w][\$\w]\*)\s\*,\s\*([\$\w][\$\w]\*)\s\*\)))\s+in\s+([\s\S]+?)(?:\s+track\s+by\s+([\s\S]+?))?$/,

nullModelCtrl = {$setViewValue: noop};

// jshint maxlen: 100

return {

restrict: 'E',

require: ['select', '?ngModel'],

controller: ['$element', '$scope', '$attrs', function($element, $scope, $attrs) {

var self = this,

optionsMap = {},

ngModelCtrl = nullModelCtrl,

nullOption,

unknownOption;

self.databound = $attrs.ngModel;

self.init = function(ngModelCtrl\_, nullOption\_, unknownOption\_) {

ngModelCtrl = ngModelCtrl\_;

nullOption = nullOption\_;

unknownOption = unknownOption\_;

};

self.addOption = function(value) {

assertNotHasOwnProperty(value, '"option value"');

optionsMap[value] = true;

if (ngModelCtrl.$viewValue == value) {

$element.val(value);

if (unknownOption.parent()) unknownOption.remove();

}

};

self.removeOption = function(value) {

if (this.hasOption(value)) {

delete optionsMap[value];

if (ngModelCtrl.$viewValue == value) {

this.renderUnknownOption(value);

}

}

};

self.renderUnknownOption = function(val) {

var unknownVal = '? ' + hashKey(val) + ' ?';

unknownOption.val(unknownVal);

$element.prepend(unknownOption);

$element.val(unknownVal);

unknownOption.prop('selected', true); // needed for IE

};

self.hasOption = function(value) {

return optionsMap.hasOwnProperty(value);

};

$scope.$on('$destroy', function() {

// disable unknown option so that we don't do work when the whole select is being destroyed

self.renderUnknownOption = noop;

});

}],

link: function(scope, element, attr, ctrls) {

// if ngModel is not defined, we don't need to do anything

if (!ctrls[1]) return;

var selectCtrl = ctrls[0],

ngModelCtrl = ctrls[1],

multiple = attr.multiple,

optionsExp = attr.ngOptions,

nullOption = false, // if false, user will not be able to select it (used by ngOptions)

emptyOption,

// we can't just jqLite('<option>') since jqLite is not smart enough

// to create it in <select> and IE barfs otherwise.

optionTemplate = jqLite(document.createElement('option')),

optGroupTemplate =jqLite(document.createElement('optgroup')),

unknownOption = optionTemplate.clone();

// find "null" option

for(var i = 0, children = element.children(), ii = children.length; i < ii; i++) {

if (children[i].value === '') {

emptyOption = nullOption = children.eq(i);

break;

}

}

selectCtrl.init(ngModelCtrl, nullOption, unknownOption);

// required validator

if (multiple) {

ngModelCtrl.$isEmpty = function(value) {

return !value || value.length === 0;

};

}

if (optionsExp) setupAsOptions(scope, element, ngModelCtrl);

else if (multiple) setupAsMultiple(scope, element, ngModelCtrl);

else setupAsSingle(scope, element, ngModelCtrl, selectCtrl);

////////////////////////////

function setupAsSingle(scope, selectElement, ngModelCtrl, selectCtrl) {

ngModelCtrl.$render = function() {

var viewValue = ngModelCtrl.$viewValue;

if (selectCtrl.hasOption(viewValue)) {

if (unknownOption.parent()) unknownOption.remove();

selectElement.val(viewValue);

if (viewValue === '') emptyOption.prop('selected', true); // to make IE9 happy

} else {

if (isUndefined(viewValue) && emptyOption) {

selectElement.val('');

} else {

selectCtrl.renderUnknownOption(viewValue);

}

}

};

selectElement.on('change', function() {

scope.$apply(function() {

if (unknownOption.parent()) unknownOption.remove();

ngModelCtrl.$setViewValue(selectElement.val());

});

});

}

function setupAsMultiple(scope, selectElement, ctrl) {

var lastView;

ctrl.$render = function() {

var items = new HashMap(ctrl.$viewValue);

forEach(selectElement.find('option'), function(option) {

option.selected = isDefined(items.get(option.value));

});

};

// we have to do it on each watch since ngModel watches reference, but

// we need to work of an array, so we need to see if anything was inserted/removed

scope.$watch(function selectMultipleWatch() {

if (!equals(lastView, ctrl.$viewValue)) {

lastView = copy(ctrl.$viewValue);

ctrl.$render();

}

});

selectElement.on('change', function() {

scope.$apply(function() {

var array = [];

forEach(selectElement.find('option'), function(option) {

if (option.selected) {

array.push(option.value);

}

});

ctrl.$setViewValue(array);

});

});

}

function setupAsOptions(scope, selectElement, ctrl) {

var match;

if (!(match = optionsExp.match(NG\_OPTIONS\_REGEXP))) {

throw ngOptionsMinErr('iexp',

"Expected expression in form of " +

"'\_select\_ (as \_label\_)? for (\_key\_,)?\_value\_ in \_collection\_'" +

" but got '{0}'. Element: {1}",

optionsExp, startingTag(selectElement));

}

var displayFn = $parse(match[2] || match[1]),

valueName = match[4] || match[6],

keyName = match[5],

groupByFn = $parse(match[3] || ''),

valueFn = $parse(match[2] ? match[1] : valueName),

valuesFn = $parse(match[7]),

track = match[8],

trackFn = track ? $parse(match[8]) : null,

// This is an array of array of existing option groups in DOM.

// We try to reuse these if possible

// - optionGroupsCache[0] is the options with no option group

// - optionGroupsCache[?][0] is the parent: either the SELECT or OPTGROUP element

optionGroupsCache = [[{element: selectElement, label:''}]];

if (nullOption) {

// compile the element since there might be bindings in it

$compile(nullOption)(scope);

// remove the class, which is added automatically because we recompile the element and it

// becomes the compilation root

nullOption.removeClass('ng-scope');

// we need to remove it before calling selectElement.empty() because otherwise IE will

// remove the label from the element. wtf?

nullOption.remove();

}

// clear contents, we'll add what's needed based on the model

selectElement.empty();

selectElement.on('change', function() {

scope.$apply(function() {

var optionGroup,

collection = valuesFn(scope) || [],

locals = {},

key, value, optionElement, index, groupIndex, length, groupLength, trackIndex;

if (multiple) {

value = [];

for (groupIndex = 0, groupLength = optionGroupsCache.length;

groupIndex < groupLength;

groupIndex++) {

// list of options for that group. (first item has the parent)

optionGroup = optionGroupsCache[groupIndex];

for(index = 1, length = optionGroup.length; index < length; index++) {

if ((optionElement = optionGroup[index].element)[0].selected) {

key = optionElement.val();

if (keyName) locals[keyName] = key;

if (trackFn) {

for (trackIndex = 0; trackIndex < collection.length; trackIndex++) {

locals[valueName] = collection[trackIndex];

if (trackFn(scope, locals) == key) break;

}

} else {

locals[valueName] = collection[key];

}

value.push(valueFn(scope, locals));

}

}

}

} else {

key = selectElement.val();

if (key == '?') {

value = undefined;

} else if (key === ''){

value = null;

} else {

if (trackFn) {

for (trackIndex = 0; trackIndex < collection.length; trackIndex++) {

locals[valueName] = collection[trackIndex];

if (trackFn(scope, locals) == key) {

value = valueFn(scope, locals);

break;

}

}

} else {

locals[valueName] = collection[key];

if (keyName) locals[keyName] = key;

value = valueFn(scope, locals);

}

}

// Update the null option's selected property here so $render cleans it up correctly

if (optionGroupsCache[0].length > 1) {

if (optionGroupsCache[0][1].id !== key) {

optionGroupsCache[0][1].selected = false;

}

}

}

ctrl.$setViewValue(value);

});

});

ctrl.$render = render;

// TODO(vojta): can't we optimize this ?

scope.$watch(render);

function render() {

// Temporary location for the option groups before we render them

var optionGroups = {'':[]},

optionGroupNames = [''],

optionGroupName,

optionGroup,

option,

existingParent, existingOptions, existingOption,

modelValue = ctrl.$modelValue,

values = valuesFn(scope) || [],

keys = keyName ? sortedKeys(values) : values,

key,

groupLength, length,

groupIndex, index,

locals = {},

selected,

selectedSet = false, // nothing is selected yet

lastElement,

element,

label;

if (multiple) {

if (trackFn && isArray(modelValue)) {

selectedSet = new HashMap([]);

for (var trackIndex = 0; trackIndex < modelValue.length; trackIndex++) {

locals[valueName] = modelValue[trackIndex];

selectedSet.put(trackFn(scope, locals), modelValue[trackIndex]);

}

} else {

selectedSet = new HashMap(modelValue);

}

}

// We now build up the list of options we need (we merge later)

for (index = 0; length = keys.length, index < length; index++) {

key = index;

if (keyName) {

key = keys[index];

if ( key.charAt(0) === '$' ) continue;

locals[keyName] = key;

}

locals[valueName] = values[key];

optionGroupName = groupByFn(scope, locals) || '';

if (!(optionGroup = optionGroups[optionGroupName])) {

optionGroup = optionGroups[optionGroupName] = [];

optionGroupNames.push(optionGroupName);

}

if (multiple) {

selected = isDefined(

selectedSet.remove(trackFn ? trackFn(scope, locals) : valueFn(scope, locals))

);

} else {

if (trackFn) {

var modelCast = {};

modelCast[valueName] = modelValue;

selected = trackFn(scope, modelCast) === trackFn(scope, locals);

} else {

selected = modelValue === valueFn(scope, locals);

}

selectedSet = selectedSet || selected; // see if at least one item is selected

}

label = displayFn(scope, locals); // what will be seen by the user

// doing displayFn(scope, locals) || '' overwrites zero values

label = isDefined(label) ? label : '';

optionGroup.push({

// either the index into array or key from object

id: trackFn ? trackFn(scope, locals) : (keyName ? keys[index] : index),

label: label,

selected: selected // determine if we should be selected

});

}

if (!multiple) {

if (nullOption || modelValue === null) {

// insert null option if we have a placeholder, or the model is null

optionGroups[''].unshift({id:'', label:'', selected:!selectedSet});

} else if (!selectedSet) {

// option could not be found, we have to insert the undefined item

optionGroups[''].unshift({id:'?', label:'', selected:true});

}

}

// Now we need to update the list of DOM nodes to match the optionGroups we computed above

for (groupIndex = 0, groupLength = optionGroupNames.length;

groupIndex < groupLength;

groupIndex++) {

// current option group name or '' if no group

optionGroupName = optionGroupNames[groupIndex];

// list of options for that group. (first item has the parent)

optionGroup = optionGroups[optionGroupName];

if (optionGroupsCache.length <= groupIndex) {

// we need to grow the optionGroups

existingParent = {

element: optGroupTemplate.clone().attr('label', optionGroupName),

label: optionGroup.label

};

existingOptions = [existingParent];

optionGroupsCache.push(existingOptions);

selectElement.append(existingParent.element);

} else {

existingOptions = optionGroupsCache[groupIndex];

existingParent = existingOptions[0]; // either SELECT (no group) or OPTGROUP element

// update the OPTGROUP label if not the same.

if (existingParent.label != optionGroupName) {

existingParent.element.attr('label', existingParent.label = optionGroupName);

}

}

lastElement = null; // start at the beginning

for(index = 0, length = optionGroup.length; index < length; index++) {

option = optionGroup[index];

if ((existingOption = existingOptions[index+1])) {

// reuse elements

lastElement = existingOption.element;

if (existingOption.label !== option.label) {

lastElement.text(existingOption.label = option.label);

}

if (existingOption.id !== option.id) {

lastElement.val(existingOption.id = option.id);

}

// lastElement.prop('selected') provided by jQuery has side-effects

if (existingOption.selected !== option.selected) {

lastElement.prop('selected', (existingOption.selected = option.selected));

}

} else {

// grow elements

// if it's a null option

if (option.id === '' && nullOption) {

// put back the pre-compiled element

element = nullOption;

} else {

// jQuery(v1.4.2) Bug: We should be able to chain the method calls, but

// in this version of jQuery on some browser the .text() returns a string

// rather then the element.

(element = optionTemplate.clone())

.val(option.id)

.attr('selected', option.selected)

.text(option.label);

}

existingOptions.push(existingOption = {

element: element,

label: option.label,

id: option.id,

selected: option.selected

});

if (lastElement) {

lastElement.after(element);

} else {

existingParent.element.append(element);

}

lastElement = element;

}

}

// remove any excessive OPTIONs in a group

index++; // increment since the existingOptions[0] is parent element not OPTION

while(existingOptions.length > index) {

existingOptions.pop().element.remove();

}

}

// remove any excessive OPTGROUPs from select

while(optionGroupsCache.length > groupIndex) {

optionGroupsCache.pop()[0].element.remove();

}

}

}

}

};

}];

var optionDirective = ['$interpolate', function($interpolate) {

var nullSelectCtrl = {

addOption: noop,

removeOption: noop

};

return {

restrict: 'E',

priority: 100,

compile: function(element, attr) {

if (isUndefined(attr.value)) {

var interpolateFn = $interpolate(element.text(), true);

if (!interpolateFn) {

attr.$set('value', element.text());

}

}

return function (scope, element, attr) {

var selectCtrlName = '$selectController',

parent = element.parent(),

selectCtrl = parent.data(selectCtrlName) ||

parent.parent().data(selectCtrlName); // in case we are in optgroup

if (selectCtrl && selectCtrl.databound) {

// For some reason Opera defaults to true and if not overridden this messes up the repeater.

// We don't want the view to drive the initialization of the model anyway.

element.prop('selected', false);

} else {

selectCtrl = nullSelectCtrl;

}

if (interpolateFn) {

scope.$watch(interpolateFn, function interpolateWatchAction(newVal, oldVal) {

attr.$set('value', newVal);

if (newVal !== oldVal) selectCtrl.removeOption(oldVal);

selectCtrl.addOption(newVal);

});

} else {

selectCtrl.addOption(attr.value);

}

element.on('$destroy', function() {

selectCtrl.removeOption(attr.value);

});

};

}

};

}];

var styleDirective = valueFn({

restrict: 'E',

terminal: true

});

if (window.angular.bootstrap) {

//AngularJS is already loaded, so we can return here...

console.log('WARNING: Tried to load angular more than once.');

return;

}

//try to bind to jquery now so that one can write angular.element().read()

//but we will rebind on bootstrap again.

bindJQuery();

publishExternalAPI(angular);

jqLite(document).ready(function() {

angularInit(document, bootstrap);

});

})(window, document);

!angular.$$csp() && angular.element(document).find('head').prepend('<style type="text/css">@charset "UTF-8";[ng\\:cloak],[ng-cloak],[data-ng-cloak],[x-ng-cloak],.ng-cloak,.x-ng-cloak,.ng-hide{display:none !important;}ng\\:form{display:block;}.ng-animate-block-transitions{transition:0s all!important;-webkit-transition:0s all!important;}</style>');ontent-row">

<div class="content-float" style="width:80px; margin-top:10px;"><label>价格</label>：</div>

<div class="content-float" style="width:300px; margin-top:10px;"><% if (ViewData["GPrice"] != null)

{%><%= ViewData["GPrice"].ToString() %><%} %></div>

</div>

<div class="content-row">

<div class="content-float" style="width:80px; margin-top:10px;"><label>物流</label>：</div>

<div class="content-float" style="width:300px; margin-top:10px;"><% if (ViewData["address"] != null)

{%>

<%if (ViewData["AddProvince"].ToString() == ViewData["AddCity"].ToString())

{%>

<%= ViewData["AddProvince"].ToString()%>

<%}

else

{%>

<%= ViewData["address"].ToString()%><%}

} %></div>

</div>

<%-- <div class="content-row">

<div class="content-float" style="width:80px; margin-top:10px;"><label>成交</label>：</div>

<div class="content-float" style="width:300px; margin-top:10px;"><% if (ViewData["GSales"] != null)

{%><%= ViewData["GSales"].ToString() %><%} %>个成交<% if (ViewData["GComments"] != null)

{%><%= ViewData["GComments"].ToString() %><%} %>个评价</div>

</div>--%>

<%--<div class="content-row">

<div class="content-float" style="width:80px; margin-top:10px;"><label>数量</label>：</div>

<div class="content-float" style="width:300px; margin-top:10px;"><input type="text" class="content-yellow-page-detailed-input"></div>

</div>--%>

<%if (Convert.ToInt32(ViewData["CoState"]) >= 5)

{ %>

<div class="content-row">

<button id="InquiryBtn" class="btn btn-primary content-yellow-page-detailed-button">询价</button>

</div>

<%} %>

</div>

</div>

<%--<div class="content-row content-yellow-page-detailed-title">

<div class="test\_box">

<div class="test\_tab">

<input type="radio" id="testTabRadio1" class="test\_radio" name="tab" checked="checked" />

<label class="test\_label" for="testTabRadio1">详细信息</label>

<div class="test\_tab\_content">

<p>加工定制品牌<% if (ViewData["GBrand"] != null)

{%><%= ViewData["GBrand"].ToString() %><%} %>种类<% if (ViewData["Category"] != null)

{%><%= ViewData["Category"].ToString() %><%}%>型号<% if (ViewData["GModel"] != null)

{%><%= ViewData["GModel"].ToString() %><%}%></p>

</div>

</div>

<div class="test\_tab">

<input type="radio" id="testTabRadio2" class="test\_radio" name="tab" />

<label class="test\_label" for="testTabRadio2">性能和图纸</label>

<div class="test\_tab\_content">

<img <%if (ViewData["Draw1"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Draw1"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%> />

<img <%if (ViewData["Draw2"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Draw2"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%> />

</div>

</div>

<div class="test\_tab">

<input type="radio" id="testTabRadio3" class="test\_radio" name="tab" />

<label class="test\_label" for="testTabRadio3">成交</label>

<div class="test\_tab\_content">

<ul>

<li>采购商</li>

<li>商品规格</li>

<li>采购数量</li>

<li>采购与日期</li>

</ul>

<ul>

<li>东北供应商</li>

<li>特大</li>

<li>2000</li>

<li>2016.08.24</li>

</ul>

</div>

</div>

<div class="test\_tab">

<input type="radio" id="testTabRadio4" class="test\_radio" name="tab" />

<label class="test\_label" for="testTabRadio4">评价</label>

<div class="test\_tab\_content">

<ul>

<li>采购商</li>

<li>商品质量</li>

<li>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*</li>

</ul>

<ul>

<li></li>

<li>物流速度</li>

<li>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*</li>

</ul>

<ul>

<li></li>

<li>技术支持</li>

<li>\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*</li>

</ul>

</div>

</div>

</div>

</div>--%>

<div id="yellow-page-box" class="content-yellow-page-tab-margin">

<ul>

<li class="on">详细信息</li>

<li>性能和图纸</li>

<%--<li>成交</li>

<li>评价</li>--%>

</ul>

<div>

<div class="tab" style="display:block">

<div class="content-row">

<div class="content-yellow-page-tab1-title">加工定制是品牌<% if (ViewData["GBrand"] != null)

{%><%= ViewData["GBrand"].ToString() %><%} %></div>

<div class="content-yellow-page-tab1-text">种类<% if (ViewData["Category"] != null)

{%><%= ViewData["Category"].ToString() %><%}%></div>

<div class="content-yellow-page-tab1-text">型号<% if (ViewData["GModel"] != null){%><%= ViewData["GModel"].ToString() %><%}%></div>

<div><img <%if (ViewData["Det1"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Det1"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%> class="content-yellow-page-tab-img"/></div>

<div><img <%if (ViewData["Det2"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Det2"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%>class="content-yellow-page-tab-img"/></div>

<div><img <%if (ViewData["Det3"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Det3"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%> class="content-yellow-page-tab-img"/></div>

</div>

</div>

<div class="tab">

<img <%if (ViewData["Draw1"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Draw1"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%> class="content-yellow-page-tab-img"/>

<img <%if (ViewData["Draw2"] != null)

{%>

src='<%: ViewData["ImgUrl"].ToString()+ViewData["Draw2"].ToString() %>' <%}

else

{ %>

style="display: none" <% }%> class="content-yellow-page-tab-img"/>

</div>

<div class="tab">

<div class="content-row">

<div class="content-yellow-page-tab3-title">采购商</div>

<div class="content-yellow-page-tab3-text">商品规格</div>

<div class="content-yellow-page-tab3-text">采购数量</div>

<div class="content-yellow-page-tab3-text">采购与日期</div>

</div>

<div class="content-row">

<div class="content-yellow-page-tab3-title">东北供应商</div>

<div class="content-yellow-page-tab3-text">特大</div>

<div class="content-yellow-page-tab3-text">1000</div>

<div class="content-yellow-page-tab3-text">2016-10-18</div>

</div>

</div>

<div class="tab">

<div class="content-row">

<div class="content-yellow-page-tab4-title">采购商</div>

<div class="content-yellow-page-tab4-text2">商品质量</div>

<div class="content-yellow-page-tab4-text">商品质量商品质量商品质量</div>

</div>

<div class="content-row">

<div class="content-yellow-page-tab4-title"></div>

<div class="content-yellow-page-tab4-text2">物流速度</div>

<div class="content-yellow-page-tab4-text">物流速度物流速度</div>

</div>

<div class="content-row">

<div class="content-yellow-page-tab4-title"></div>

<div class="content-yellow-page-tab4-text2">技术支持</div>

<div class="content-yellow-page-tab4-text">技术支持技术支持技术支持</div>

</div>

</div>

</div>

</div>

<div class="content-yellow-page-index-bluebg-second">&nbsp;&nbsp;&nbsp;相关产品

</div>

<div id="Prlistview" ></div>

</div>

</div>

</div>

<!-- 模态框（Modal） -->

<form id="OInquiry" name="OInquiry" method="post">

<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">

<div class="modal-tender motai-xiang">

<div class="modal-content">

<div class="modal-header">

<a>

<button type="button" class="close"

data-dismiss="modal" aria-hidden="true">

&times;

</button>

</a>

<h4>请输入内容:</h4>

</div>

<div class="modal-body ">

<div class="content-row">

<div class="detail-input-title">

<label id="Company\_title">产品名称</label>：<span class="color-red">\*</span>

</div>

<div class="personal-inquiry-title">

<input maxlength="50" value="<% if (ViewData["GName"] != null){%><%= ViewData["GName"].ToString()%><%} %>" id="Company" name="Company" type="text" class="search-key-input">

</div>

<div class="detail-input-blank">

</div>

<div id="Company\_Error" class="detail-input-text-error">

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

<label>供货方式</label>：<span class="color-red">\*</span>

</div>

<div class="personal-inquiry-title "><label id="dtStart\_title"><input type="radio" id="LongSupply" checked name="SupplyList">长期供货</label></div>

<label id="dtEnd\_title" style="display:none">长期供货</label>

<div class="personal-inquiry-date-title">

<div id="StartTime" class="input-group date form\_date search-key-time-width" data-link-format="yyyy-mm-dd" data-link-field="dtStart" data-date-format="yyyy-mm-dd" data-date="">

<input id="LStartTime" name="StartTime" class="form-control" readonly value="" size="16" type="text">

<span id="StartTimeDel" class="input-group-addon">

<span class="glyphicon glyphicon-remove"></span>

</span>

<span id="StartTimeSelect" class="input-group-addon">

<span class="glyphicon glyphicon-calendar"></span>

</span>

</div>

<input errorid="dtStart\_Error" id="dtStart" name="dtStart" value="" type="hidden">

<div class="input-group search-key-text">

<span>～</span>

</div>

<div id="EndTime" class="input-group date form\_date search-key-time-width" data-date="" data-date-format="yyyy-mm-dd" data-link-field="dtEnd" data-link-format="yyyy-mm-dd">

<input name="EndTime" id="LEndTime" class="form-control" size="16" type="text" value="" readonly>

<span id="EndTimeDel" class="input-group-addon"><span class="glyphicon glyphicon-remove"></span></span>

<span id="EndTimeSelect" class="input-group-addon"><span class="glyphicon glyphicon-calendar"></span></span>

</div>

<input errorid="dtStart\_Error" type="hidden" id="dtEnd" name="dtEnd" value="" />

</div>

<div class="detail-input-text-error">

<div id="dtStart\_Error"></div>

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

</div>

<div class="personal-inquiry-title personal-inquiry-checkbox-margin">

<label><input id="PLocking" type="checkbox">&nbsp;单价锁定</label>

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

</div>

<div class="personal-inquiry-title">

<label id="PNumber\_title"><input type="radio" id="SingleSupply" name="SupplyList">单笔供货</label>

<label style="display:none" id="Unit\_title">单笔供货</label>

</div>

<div class="detail-input-title">

<input maxlength="8" errorid="PNumber\_Error" id="PNumber" name="PNumber" type="text" placeholder="数量" class="personal-inquiry-input-xs">

</div>

<div class="detail-input-title">

<input maxlength="5" errorid="PNumber\_Error" id="Unit" placeholder="单位" name="Unit" type="text" class="personal-inquiry-input-xs">

</div>

<div class="detail-input-text-error">

<div id="PNumber\_Error"></div>

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

<label id="NContacts\_title">联系人</label>：<span class="color-red">\*</span>

</div>

<div class="personal-inquiry-title">

<input maxlength="10" value="<% if (ViewData["LoginName"] != null){%><%= ViewData["LoginName"].ToString()%><%} %>" id="NContacts" name="NContacts" type="text" class="search-key-input">

</div>

<div class="detail-input-blank">

</div>

<div id="NContacts\_Error" class="detail-input-text-error">

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

<label id="CPhone\_title">联系电话</label>：<span class="color-red">\*</span>

</div>

<div class="personal-inquiry-title">

<input maxlength="11" value="<% if (ViewData["LoginPhone"] != null){%><%= ViewData["LoginPhone"].ToString()%><%} %>" id="CPhone" name="CPhone" type="text" class="search-key-input">

</div>

<div class="detail-input-blank">

</div>

<div id="CPhone\_Error" class="detail-input-text-error">

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

<label>工程号</label>：

</div>

<div class="personal-inquiry-title">

<input id="ENnumber" type="text" class="search-key-input">

</div>

<div class="detail-input-blank">

</div>

</div>

<div class="content-row">

<div class="detail-input-title">

<label>详细说明</label>：

</div>

<div class="personal-inquiry-textarea-title">

<textarea id="PDetail" name="PDetail" maxlength="250" rows="5" class="personal-inquiry-textarea"></textarea>

</div>

<div class="detail-input-text-error">

</div>

</div>

<div class="content-row">

<div id="file" class="personal-inquiry-title-margin">

</div>

<div class=" content-float content-inquiry-text-margin3"><a id="FileText"></a></div>

<div class="detail-title" id="Del\_File" >

<img src="../../../images/shanchu.png" class="content-on-line-delete-left"/>

</div>

<div class="detail-input-blank">

</div>

<div id="IMG1\_Error" class="detail-input-text-error">

</div>

</div>

<div class="content-row">

<div id="file2" class="personal-inquiry-title-margin">

</div>

<div class=" content-float content-inquiry-text-margin3"><a id="FileText2"></a></div>

<div class="detail-title" id="Del\_File2">

<img src="../../../images/shanchu.png" class="content-on-line-delete-left"/>

</div>

<div class="detail-input-blank">

</div>

<div id="IMG2\_Error" class="detail-input-text-error">

</div>

</div>

<div class="content-row">

<div id="file3" class="personal-inquiry-title-margin">

</div>

<div class=" content-float content-inquiry-text-margin3"><a id="FileText3"></a></div>

<div class="detail-title" id="Del\_File3">

<img src="../../../images/shanchu.png" class="content-on-line-delete-left"/>

</div>

<div class="detail-input-blank">

</div>

<div id="IMG3\_Error" class="detail-input-text-error">

</div>

</div>

<%-- <div class="content-row">

<a href="">

<button type="button" class="btn btn-primary search-button">发送询价</button>

</a>

<a href="">

<button type="button" class="btn btn-primary search-button">返回</button>

</a>

</div> --%>

</div>

<div class="modal-footer">

<button type="button" id="CloseMode" class="btn" data-dismiss="modal">

关闭

</button>

<button type="button" id="BusSubmit" class="btn btn-primary number-circular-blue" >提交

</button>

</div>

</div>

</div><!-- /.modal-content -->

</div><!-- /.modal -->

</form>

</asp:Content>

<asp:Content ID="Content3" ContentPlaceHolderID="HeaderScriptsSection" runat="server">

<%: Styles.Render("~/Content/bs/PDet") %>

<%: Styles.Render("~/Content/jqueryval") %>

<%: Styles.Render("~/Content/Upload") %>

<%: Styles.Render("~/Content/CustomFilter") %>

<%: Scripts.Render("~/bundles/SiteMaster") %>

<%: Scripts.Render("~/bundles/ListView") %>

<%: Scripts.Render("~/bundles/Upload") %>

</asp:Content>

<asp:Content ID="Content4" ContentPlaceHolderID="FooterScriptsSection" runat="server">

<%: Scripts.Render("~/bundles/CustomFilter") %>

<%: Scripts.Render("~/bundles/bs/PDetailedJs") %>

<%:Scripts.Render("~/bundles/jqueryval") %>

</asp:Content>