Cambios en la versión 1.3 de AngularJS

A continuación se relacionan los cambios más importantes que han tenido lugar tras la aparición de la versión 1.3

Acompañamos estos contenidos con dos bloques de ejemplos:

- Un bloque adicional que revisa muchos de los ejemplos del curso pero implementados cambiando la librería para que utilice la nueva versión. En casi todos los casos, el cambio fundamental es muy simple: en lugar de declarar como controlador una función separada, se declara mediante una llamada al método ***controller*** del módulo correspondiente. Podrás comprobar que todo lo demás funciona de la misma forma.
- En cuanto a novedades reales de la versión no hay muchas que afecten al desarrollo tradicional, pero hemos optado por incluir 3 ejemplos sencillos que recogen algunas de estas novedades. En este documento, señalamos en los comentarios correspondientes, cuál es el nombre del ejemplo que implementa la funcionalidad comentada.

Cambios respecto a soporte

- No hay soporte de IE8 (o versiones inferiores)

Cambios estructurales

- El más importante: AngularJS ya no busca controladores dentro del objeto global window.
- Esto significa que si declaramos un controlador en una función, éste no va a ser reconocido como tal.
- Debemos usar la sintaxis **angular.module("app",[]).controller("controlador"…)**
- Esto es por razones de modularidad.

Cambios respecto a tipos de datos

- Los tipos ***date***, ***time***, ***datetime-local***, ***month***, ***week*** ahora requieren siempre un objeto ***Date*** como modelo
- Los valores **'f'**, **'0'**, **'false'**, **'no'**, **'n'**, '**[]**' ya no se tratan como ***falsy***. Esto solo se hace con los objetos ***falsy*** de JavaScript.
    - O sea: **false**, **null**, **undefined**, **NaN**, **0** y **""**.

- ***Scope#$id*** es ahora de tipo número (no cadena)

Cambios respecto a métodos

- Ya no se puede invocar ***.bind***, ***.call*** or ***.apply*** sobre una función en expresiones de Angular. Se trata de no permitir cambiar el comportamiento de las funciones existentes de forma imprevista.
- Las llamadas a ***attr.$observe*** ya no devuelven la función ***observer***, sino una función para de-registro.
- No se puede usar ***object*** dentro de expresiones Angular. Si se requiere, debe hacerse accesible en el $***scope***
- **angular.copy** ahora copia el prototipo del objeto en el objeto de destino.

Bindings de tipo "one-time"

- Si ponemos el prefijo **::** en un identificador del modelo, el enlace solo se realizará una vez, y el elemento dejará de ser observable.
- NOTA: esto se demuestra en el ejemplo extra **One-Time-Binding.html**

Cambios respecto a objetos

- Si hay expresión ***ng-pattern*** (tal como ***ng-pattern="exp"***) o un atributo de patrón (algo así como el ***pattern = "{{exp}}"***) y la expresión se evalúa como cadena, el validador no analizará la cadena como un objeto literal sino que se creará la cadena entera como expresión regular y se comprobará su validez.
- ***forEach*** solo iterará por el número inicial de elementos de la matriz.
- Si se agregan más elementos a la matriz durante la iteración, éstos no se iterarán en la llamada inicial ***forEach***
- ***jqLite***: anteriormente era posible asignar datos ***jqLite*** en los nodos de texto o comentarios, pero ahora está permitido sólo en elementos y nodos de documento como en ***jQuery***.
- $***interpolate***: La función devuelta por $***interpolate*** ya no dispone de un ***array .parts*** en su interior. En su lugar tiene dos matrices:
- ***.expressions***, un array de las expresiones en el texto interpolado. Las expresiones son analizadas con $***parse***, con una capa extra para convertirlas a cadenas
- ***.separators***, una matriz de cadenas que representan las separaciones entre las interpolaciones en el texto. Este array es siempre 1 ítem más largo que la matriz de .expressions

Mejoras y cambios en la gestión de formularios

- Probablemente, el aspecto que más modificaciones incluye respecto a la versión anterior.
- Cualquier mecanismo que recoge datos utiliza ***ng-model***
- El modelo funciona en los dos sentidos y refleja cualquier cambio

```
<div class="field">
    <input type="text" ng-model="miNombre" />
    <p>Mi nombre es: <strong></strong></p><pre>{{miNombre}}</pre>
    <button ng-click="miNombre='Valor predeterminado'">Reinicializar a
        'Valor predeterminado'</button>
</div>
```

- Nota: Los ejemplos extra incluyen dos ficheros donde se muestran algunas de estas características, de nombre: **AyudasEdicionForm.html** y **AyudasEdicionForm(retardos).html**. Simplemente, ejemplifican cómo incluir un retardo controlado en el cambio de valor de un campo introducido y cómo hacer que el cambio generado solo se muestre cuando pierde el foco (Evento *Blur*).
- Muchas validaciones de un campo tienen lugar automáticamente;

```
<div class="field">
    <input type="email"
           minlength="5"
           maxlength="100"
           ng-model="myEmail"
           required />
</div>
```

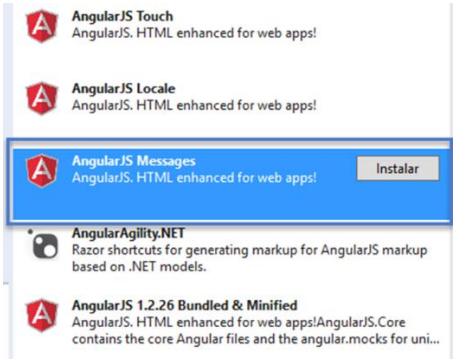
ngAria

- El módulo de ngAria proporciona soporte para agregar atributos ARIA que transmiten información semántica acerca de la aplicación para permitir que las tecnologías asistidas transmitan la información correspondiente a las personas con discapacidades.
- Para permitir la incorporación de las etiquetas ARIA, sólo requiere el módulo en su aplicación y las etiquetas se enlazarán con las directivas **ng-show/ng-hide**, **input**, **textarea**, **button**, **select** y **ng-required** y agregarán los atributos de ARIA adecuados.
- En la actualidad, se aplican los siguientes atributos ARIA:

- aria-hidden
- aria-checked
- aria-disabled
- aria-required
- aria-invalid
- aria-multiline
- aria-valuenow
- aria-valuemin
- aria-valuemax
- tabindex

ngMessages

- El módulo **ngMessages** ofrece soporte mejorado para la visualización de mensajes de validación o notificación
- Normalmente dentro de formularios o al representar objetos que devuelven datos de clave/valor).
- En lugar de depender de código JavaScript o usar sentencias **ng-if** complejas, **ngMessages** y **ngMessage** están diseñadas para manejar la complejidad, la herencia y la secuencia de prioridad basada en el orden de cómo se definen los mensajes de la plantilla
- En la actualidad, el módulo de **ngMessages** sólo contiene el código para las directivas de **ngMessages** y **ngMessage**
- El módulo **ngMessages** requiere la librería ***angular-messages.js***



Marino Posadas

Nov. 2014