Seahorse

Manual de usuario

v1.2 José Facundo Maldonado

Seahorse
Manual de usuario

por José Facundo Maldonado

Índice

1 - Introducción

Seahorse es una librería de JavaScript, licenciada como software libre, creada para simplificar el uso de formularios, particularmente la validación de los mismos. Provee funciones de validación, conversión y comportamiento para el manejo de fechas, horas, números, textos y direcciones de e-mail o URLs.

Todas las funciones son altamente configurables, permitiendo especificar el rango de valores válidos, caracteres no permitidos, formatos y, en el caso de los comportamientos, las respuestas a los eventos de pérdida de foco y de presionado de una tecla.

Seahorse puede ser utilizada junto a cualquier framework de JavaScript, sin embargo, posee un plugin para ser usada junto con jQuery.

2 - Validación

Las funciones de validación se encargan de verificar si una cadena de texto representa o no un determinado tipo de dato. Además del texto, reciben como parámetros algunos valores que determinan los criterios con que se realiza la validación.

2.1 - isNumber()

La función `isNumber()` verifica si una cadena representa un número. Recibe como parámetros la cadena, el caracter usado como separador decimal y el caracter usado como separador de grupos (también conocido como separador de miles).

```
Seahorse.isNumber("1,000,000.00", '.', ',')  ➡ true
Seahorse.isNumber("1 000 000.00", ',', '.')  ➡ false
Seahorse.isNumber("1 000 000.00", ',', ' ')  ➡ true
Seahorse.isNumber("1.000.000,00", ',', '.')  ➡ true
```

2.2 - isInteger()

La función isInteger() verifica si una cadena representa un número entero. Recibe como parámetros la cadena y el caracter usado como separador de grupos (también conocido como separador de miles).

```
Seahorse.isInteger("1,000,000", ',')  ➡ true
Seahorse.isInteger("1 000 000", ',')  ➡ false
Seahorse.isInteger("1 000 000", ' ')  ➡ false
Seahorse.isInteger("1.000.000", '.')  ➡ true
```

2.3 - isNumeric()

La función `isNumeric()` verifica si una cadena contiene solo números.

```
Seahorse.isNumeric("1234")  ➡ true
Seahorse.isNumeric("1,234")  ➡ false
```

2.4 - isAlphabetical()

La función `isAlphabetical()` verifica si una cadena posee solo letras.

```
Seahorse.isAlphabetical("abcdñ")  ➡ true
Seahorse.isAlphabetical("abcñ123")  ➡ false
```

2.5 - isAlphanumeric()

La función `isAlphanumeric()` verifica si una cadena posee solo letras y números.

```
Seahorse.isAlphanumeric("abcdñ123")  ➡ true
Seahorse.isAlphanumeric("abcñ-123")  ➡ false
```

2.6 - isAlphabeticalAscii()

La función `isAlphabeticalAscii()` verifica si una cadena posee solo letras del latín básico.

```
Seahorse.isAlphabeticalAscii("abcd")  ➡ true
Seahorse.isAlphabeticalAscii("abcdñ")  ➡ false
```

2.7 - isAlphanumericAscii()

La función `isAlphanumericAscii()` verifica si una cadena posee solo letras del latín básico y números.

```
Seahorse.isAlphanumericAscii("abc123")   ➡ true
Seahorse.isAlphanumericAscii("abcñ123")  ➡ false
```

2.8 - isAsciiText()

La función `isAsciiText()` verifica si una cadena posee solo caracteres representables por el código ASCII.

```
Seahorse.isAsciiText("abcd (123) @#")    ➡ true
Seahorse.isAsciiText("abcd (123) @#ñá")  ➡ false
```

2.9 - isIPv4()

La función `isIPv4()` verifica si una cadena representa una dirección IP de la versión 4.

```
Seahorse.isIPv4("192.168.0.1")    ➡ false
Seahorse.isIPv4("192.168.0.256")  ➡ false
```

2.10 - isIPv6()

La función `isIPv6()` verifica si una cadena representa una dirección IP de la versión 6.

```
Seahorse.isIPv6("11:22:33:44:55:66:77:88")  ➡ true
Seahorse.isIPv6("11:22:33:44:55:::")        ➡ true
```

2.11 - isEmail()

La función `isEmail()` verifica si una cadena representa una dirección de e-mail.

```
Seahorse.isEmail("test@test.com")  ➡ true
Seahorse.isEmail("test@test")      ➡ false
```

2.12 - isHttp()

La función `isHttp()` verifica si una cadena representa una dirección HTTP.

```
Seahorse.isHttp("http://www.test.com")  ➡ true
Seahorse.isHttp("www.test.com")         ➡ false
```

2.13 - isFtp()

La función `isFtp()` verifica si una cadena representa una dirección FTP.

```
Seahorse.isFtp("ftp://ftp.test.com")  ➡ true
Seahorse.isFtp("ftp.test.com")        ➡ false
```

2.14 - isDate()

La función `isDate()` verifica si una cadena representa una fecha. Recibe como parámetros la cadena y

el formato de la fecha, compuesto mediante la combinación de:

- `d` - día del mes (sin cero a la izquierda)
- `dd` - día del mes (dos dígitos)
- `m` - mes del año (sin cero a la izquierda)
- `mm` - mes del año (dos dígitos)
- `yy` - año (dos dígitos)
- `yyyy` - año (cuatro dígitos)

El formato sirve solo para determinar el orden de las cifras, por lo cual es equivalente, por ejemplo, usar como formato `dd/mm/yyyy` que `dd-mm-yyyy`.

```
Seahorse.isDate("31/07/2010","dd/mm/yyyy")  ➡ true
Seahorse.isDate("31-07-2010","dd/mm/yyyy")  ➡ true
Seahorse.isDate("07/31/2010","mm/dd/yyyy")  ➡ true
Seahorse.isDate("07 31 2010","mm/dd/yyyy")  ➡ true
Seahorse.isDate("07/32/2010","mm/dd/yyyy")  ➡ false
Seahorse.isDate("06/31/2010","mm/dd/yyyy")  ➡ false
```

2.15 - isTime()

La función `isTime()` verifica si una cadena representa un hora del día. Recibe como parámetros la cadena y el formato de la hora, compuesto mediante la combinación de:

- `s` - segundos (sin cero a la izquierda)
- `ss` - segundos (dos dígitos)
- `m` - minutos (sin cero a la izquierda)
- `mm` - minutos (dos dígitos)
- `h` - hora (sin cero a la izquierda)
- `hh` - hora (dos dígitos)

El formato sirve solo para determinar el orden de las cifras, por lo cual es equivalente, por ejemplo, usar como formato `hh:mm:ss` que `hh-mm-ss`.

```
Seahorse.isTime("06:20:22","hh:mm:ss")  ➡ true
Seahorse.isTime("06-20-22","hh:mm:ss")  ➡ true
Seahorse.isTime("22:20:06","s:m:h")     ➡ true
Seahorse.isTime("22 20 06","s:m:h")     ➡ true
Seahorse.isTime("22:20:60","hh:mm:ss")  ➡ false
Seahorse.isTime("24:20:22","hh:mm:ss")  ➡ false
```

3 - Conversión

Las funciones de conversión se encargan de analizar una cadena de texto para convertirla en otro tipo de dato o darle un formato en particular. Además del texto, las funciones reciben, como parámetros, algunos valores que determinan los criterios con que se realiza el análisis y la conversión.

3.1 - parseNumber()

La función `parseNumber()` convierte una cadena en un número. Recibe como parámetros la cadena, el carácter usado como separador decimal y el carácter usado como separador de grupos (también conocido como separador de miles).

```
Seahorse.parseNumber("1,234.5", '.', ',')      ➡ 1234.5
Seahorse.parseNumber("1,234.5", ',', '.')      ➡ 1.2345
Seahorse.parseNumber("1234 5", ',', '.')       ➡ 1234
Seahorse.parseNumber("1234sdfg5", ',', '.')    ➡ 1234
```

3.2 - parseInteger()

La función `parseInteger()` convierte una cadena en un entero. Recibe como parámetros la cadena y el caracter usado como separador de grupos (también conocido como separador de miles).

```
Seahorse.parseInteger("1,234", ',')     ➡ 1234
Seahorse.parseInteger("1.234.5", '.')   ➡ 12345
Seahorse.parseInteger("1 234", ' ')     ➡ 1234
Seahorse.parseInteger("1 234", ',')     ➡ 1
```

3.3 - parseIPv4()

La función `parseIPv4()` convierte una cadena que representa una dirección IP versión 4 en un array con cuatro números.

```
Seahorse.parseIPv4("192.168.0.1")      ➡ 192,168,0,1
Seahorse.parseIPv4("192.168.0.256")    ➡ null
Seahorse.parseIPv4("192 168 0 1")      ➡ null
```

3.4 - parseIPv6()

La función `parseIPv6()` convierte una cadena que representa una dirección IP versión 6 en un array con ocho números hexadecimales.

```
Seahorse.parseIPv6("11:22:33:44:55:66:77:88")      ➡ 0x0011,0x0022,0x0033,0x0044,
                                                     0x0055,0x0066,0x0077,0x0088
Seahorse.parseIPv6("11:22:33:44:55:66:77:FFFG")    ➡ null
Seahorse.parseIPv6("11 22 33 44 55 66 77 88")      ➡ null
```

3.5 - parseDate()

La función `parseDate()` recibe una cadena que representa una fecha y la transforma de acuerdo a un formato determinado. Recibe como parámetros la cadena, un boolean que indica si los campos dejados en blanco deben completarse con la fecha actual y el formato de la fecha, compuesto mediante la combinación de:

- `d` - día del mes (sin cero a la izquierda)
- `dd` - día del mes (dos dígitos)
- `m` - mes del año (sin cero a la izquierda)

`mm` - mes del año (dos dígitos)
`yy` - año (dos dígitos)
`yyyy` - año (cuatro dígitos)

```
Seahorse.parseDate("31/07/2010","dd/mm/yyyy")      ➡ 31/07/2010
Seahorse.parseDate("31-07-2010","dd/mm/yyyy")      ➡ 31/07/2010
Seahorse.parseDate(" 07/31 (2010)","mm/dd/yyyy")   ➡ 07/31/2010
Seahorse.parseDate("07/32/2010","mm/dd/yyyy")      ➡ null
Seahorse.parseDate("07/31","mm/dd/yyyy")           ➡ null
Seahorse.parseDate("07/31","mm/dd/yyyy",true)      ➡ 07/31/2010
Seahorse.parseDate("07","mm/dd/yyyy",false)        ➡ null
Seahorse.parseDate("07","mm/dd/yyyy",true)         ➡ 07/20/2010
```

3.6 - parseTime()

La función `parseTime()` recibe una cadena que representa una hora y la transforma de acuerdo a un formato determinado. Recibe como parámetros la cadena, un boolean que indica si los campos dejados en blanco deben completarse con la hora actual y el formato de la hora, compuesto mediante la combinación de:

`s` - segundos (sin cero a la izquierda)
`ss` - segundos (dos dígitos)
`m` - minutos (sin cero a la izquierda)
`mm` - minutos (dos dígitos)
`h` - hora (sin cero a la izquierda)
`hh` - hora (dos dígitos)

```
Seahorse.parseTime("02:05:33","hh:mm:ss")      ➡ 02:05:33
Seahorse.parseTime("02-05-33","hh:mm:ss")      ➡ 02:05:33
Seahorse.parseTime(" 02/05 (33)","hh:mm:ss")   ➡ 02:05:33
Seahorse.parseTime("02:60:33","hh:mm:ss")      ➡ null
Seahorse.parseTime("02:05","hh:mm:ss")         ➡ null
Seahorse.parseTime("02:05","hh:mm:ss",true)    ➡ 02:05:12
Seahorse.parseTime("07","hh:mm:ss",false)      ➡ null
Seahorse.parseTime("07","hh:mm:ss",true)       ➡ 07:50:12
```

4 - Comportamiento

Los comportamientos permiten realizar la validación en tiempo real de los campos de un formulario mediante los eventos de pérdida de foco y de presionado de una tecla. Los comportamientos son asignados mediante diferentes funciones que reciben como parámetro el id o la referencia del campo, las opciones de validación y las opciones de respuesta.

Las opciones de validación determinan como es validado el campo y son similares a los parámetros recibidos por las funciones de validación. Las opciones de respuesta permiten determinar las acciones que se realizarán cuando se detecte que un campo posee un valor válido o inválido.

4.1 - Textos

Para definir entradas que aceptan datos tipo texto, existen los métodos `text()`, que por sí solo no impone ninguna restricción, `alphanumeric()`, que solo acepta letras y números (no permite símbolos ni signos de puntuación), `alphabetical()`, que solo acepta letras, y `numeric()`, que solo acepta números.

Como opciones de validación, todas las funciones aceptan los parámetros: `notEmpty`, `minLength`, `maxLength`, `asciiCharacters`, `requiredCharacters`, `forbiddenCharacters`, `allowedCharacters` y `aditionalValidation`.

`notEmpty` permite indicar si una entrada puede ser dejada o no en blanco. Por defecto su valor es falso (las entradas se pueden dejar en blanco).

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.text("textNe1", {}, {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textNe1").seahorse.verify();

      Seahorse.text("textNe2", {"notEmpty": false}, {"okClass": "ok", "errorClass":
"error"});
      document.getElementById("textNe2").seahorse.verify();

      Seahorse.text("textNe3", {"notEmpty": true}, {"okClass": "ok", "errorClass":
"error"});
      document.getElementById("textNe3").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table>
      <tr>
        <td>text()</td>
        <td></td>
        <td><input type="text" id="textNe1" value=""/></td>
      </tr>
      <tr>
        <td>text()</td>
        <td>notEmpty = true</td>
        <td><input type="text" id="textNe2" value=""/></td>
      </tr>
      <tr>
        <td>text()</td>
```

```
        <td>notEmpty = false</td>
        <td><input type="text" id="textNe3" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`minLength` y `maxLength` permiten definir la cantidad mínima y máxima de caracteres que puede tener una entrada.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
     {
     Seahorse.text("textMl1", {"notEmpty": true, "minLength" : 3},
               {"okClass": "ok", "errorClass": "error"});
     document.getElementById("textMl1").seahorse.verify();

     Seahorse.text("textMl2", {"maxLength" : 6}, {"okClass": "ok", "errorClass":
"error"});
     document.getElementById("textMl2").seahorse.verify();
     }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
     <tr>
       <td>text()</td>
       <td>notEmpty = true, minLength = 3</td>
       <td><input type="text" id="textMl1" value=""/></td>
     </tr>
     <tr>
       <td>text()</td>
       <td>maxLengt = 6</td>
       <td><input type="text" id="textMl2" value=""/></td>
     </tr>
   </table></center>
   </form>

</body>
</html>
```

`asciiCharacters` permite indicar si la entrada permitirá solo el ingreso de caracteres ASCII, en vez de permitir todos los caracteres Unicode. Por defecto, se permiten todos los caracteres.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
```

```
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.text("textAc1", {}, {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textAc1").seahorse.verify();

      Seahorse.text("textAc2", {"asciiCharacters": false}, {"okClass": "ok", "errorClass":
"error"});
      document.getElementById("textAc2").seahorse.verify();

      Seahorse.text("textAc3", {"asciiCharacters": true}, {"okClass": "ok", "errorClass":
"error"});
      document.getElementById("textAc3").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo codigo">
      <tr>
        <td>text()</td>
        <td></td>
        <td><input type="text" id="textAc1" value=""/></td>
      </tr>
      <tr>
        <td>text()</td>
        <td>asciiCharacters = false</td>
        <td><input type="text" id="textAc2" value=""/></td>
      </tr>
      <tr>
        <td>text()</td>
        <td>asciiCharacters = true</td>
        <td><input type="text" id="textAc3" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`requiredCharacters` permite indicar los caracteres que deben ser ingresados para que la entrada sea considerada válida.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.text("textRc1", {"requiredCharacters": "_"},
                {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textRc1").seahorse.verify();

      Seahorse.alphanumeric("textRc2", {"requiredCharacters": "a1"},
```

```
                {"okClass": "ok", "errorClass": "error"});
     document.getElementById("textRc2").seahorse.verify();

     Seahorse.alphabetical("textRc3", {"requiredCharacters": ["a".charCodeAt(0),
"b".charCodeAt(0)]},
                {"okClass": "ok", "errorClass": "error"});
     document.getElementById("textRc3").seahorse.verify();
     }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo codigo">
     <tr>
       <td>text()</td>
       <td>requiredCharacters = "_"</td>
       <td><input type="text" id="textRc1" value=""/></td>
     </tr>
     <tr>
       <td>alphanumeric()</td>
       <td>requiredCharacters = "a1"</td>
       <td><input type="text" id="textRc2" value=""/></td>
     </tr>
     <tr>
       <td>alphabetical()</td>
       <td>requiredCharacters = ["a", "b"]</td>
       <td><input type="text" id="textRc3" value=""/></td>
     </tr>
   </table></center>
   </form>

</body>
</html>
```

`forbiddenCharacters` permite indicar los caracteres que no deben ser ingresados para que la entrada sea considerada válida.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
     {
     Seahorse.text("textFc1", {"forbiddenCharacters": "_"},
             {"okClass": "ok", "errorClass": "error"});
     document.getElementById("textFc1").seahorse.verify();

     Seahorse.alphanumeric("textFc2", {"forbiddenCharacters": "a1"},
             {"okClass": "ok", "errorClass": "error"});
     document.getElementById("textFc2").seahorse.verify();

     Seahorse.alphabetical("textFc3", {"forbiddenCharacters": ["a".charCodeAt(0),
"b".charCodeAt(0)]},
             {"okClass": "ok", "errorClass": "error"});
     document.getElementById("textFc3").seahorse.verify();
     }
</script>
</head>
```

```
<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo codigo">
      <tr>
        <td>text()</td>
        <td>forbiddenCharacters = "_"</td>
        <td><input type="text" id="textFc1" value=""/></td>
      </tr>
      <tr>
        <td>alphanumeric()</td>
        <td>forbiddenCharacters = "a1"</td>
        <td><input type="text" id="textFc2" value=""/></td>
      </tr>
      <tr>
        <td>alphabetical()</td>
        <td>forbiddenCharacters = ["a", "b"]</td>
        <td><input type="text" id="textFc3" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`allowedCharacters` permite indicar un grupo de caracteres que una entrada puede tener y aún así ser considerada válida. Este parámetro permite, por ejemplo, que una entrada de tipo alphabetical acepte algunos signos de puntación.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.alphanumeric("textAllc1", {"allowedCharacters": " "},
              {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textAllc1").seahorse.verify();

      Seahorse.alphabetical("textAllc2", {"allowedCharacters": "123"},
              {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textAllc2").seahorse.verify();

      Seahorse.numeric("textAllc3", {"allowedCharacters": ["a".charCodeAt(0),
"b".charCodeAt(0)]},
              {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textAllc3").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo codigo">
      <tr>
        <td>alphanumeric()</td>
        <td>allowedCharacters = " "</td>
        <td><input type="text" id="textAllc1" value=""/></td>
```

```
      </tr>
      <tr>
        <td>alphabetical()</td>
        <td>allowedCharacters = "123"</td>
        <td><input type="text" id="textAllc2" value=""/></td>
      </tr>
      <tr>
        <td>numeric()</td>
        <td>allowedCharacters = ["a", "b"]</td>
        <td><input type="text" id="textAllc3" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`aditionalValidation` permite especificar una función que realice una verificación adicional para determinar si la entrada es válida o no.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.text("textAv1", {"aditionalValidation": oddLength},
        {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textAv1").seahorse.verify();

      Seahorse.text("textAv2", {"aditionalValidation": evenLength},
        {"okClass": "ok", "errorClass": "error"});
      document.getElementById("textAv2").seahorse.verify();
      }

   function oddLength(element)
      { return element.value.length%2 == 1;}

   function evenLength(element)
      { return element.value.length%2 == 0;}
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo codigo">
      <tr>
        <td>text()</td>
        <td>aditionalValidation = oddLength()</td>
        <td><input type="text" id="textAv1" value=""/></td>
      </tr>
      <tr>
        <td>text()</td>
        <td>aditionalValidation = evenLenght()</td>
        <td><input type="text" id="textAv2" value=""/></td>
      </tr>
   </table></center>
   </form>
```

```
</body>
</html>
```

4.2 - Números

Para definir entradas numéricas se debe utilizar las funciones `number()`, para números de todo tipo, e `integer()`, para números enteros.

Como opciones de validación, las funciones aceptan los parámetros: `groupingCharacter`, `decimalCharacter`, `minValue`, `maxValue`, `notEmpty` y `aditionalValidation`.

`decimalCharacter` y `groupingCharacter` permiten definir que caracteres se utilizarán, respectivamente, como separador decimal y como separador de miles o carácter de agrupamiento.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.number( "number21", {"decimalCharacter": '.', "groupingCharacter" : ','},
              {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("number21").seahorse.verify();

      Seahorse.number( "number22", {"decimalCharacter": ',', "groupingCharacter" : '.'},
              {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("number22").seahorse.verify();

      Seahorse.number( "number23", {"decimalCharacter": '.', "groupingCharacter" : ' '},
              {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("number23").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>number()</td>
        <td>decimalCharacter = '.', groupingCharacter = ','</td>
        <td><input type="text" id="number21" value=""/></td>
      </tr>
      <tr>
        <td>number()</td>
        <td>decimalCharacter = ',', groupingCharacter = '.'</td>
        <td><input type="text" id="number22" value=""/></td>
      </tr>
      <tr>
        <td>number()</td>
        <td>decimalCharacter = '.', groupingCharacter = ' '</td>
        <td><input type="text" id="number23" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`minValue` y `maxValue` permiten definir los valores mínimos y máximos que puede tomar una entrada.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.number( "numberMv1",
        {"decimalCharacter": '.', "groupingCharacter" : ',',
        "minValue" : -10.5, "maxValue" : 10.5},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("numberMv1").seahorse.verify();

      Seahorse.number( "numberMv2",
        {"decimalCharacter": '.', "groupingCharacter" : ',',
        "minValue" : 0, "maxValue" : 10.5},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("numberMv2").seahorse.verify();

      Seahorse.number( "numberMv3",
        {"decimalCharacter": '.', "groupingCharacter" : ',',
        "minValue" : -10.5, "maxValue" : 0},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("numberMv3").seahorse.verify();

      Seahorse.integer( "numberMv4",
        {"groupingCharacter" : ',', "minValue" : -10, "maxValue" : 10},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("numberMv4").seahorse.verify();

      Seahorse.integer( "numberMv5",
        {"groupingCharacter" : ',', "minValue" : 0, "maxValue" : 10},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("numberMv5").seahorse.verify();

      Seahorse.integer( "numberMv6",
        {"groupingCharacter" : ',', "minValue" : -10, "maxValue" : 0},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("numberMv6").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>number()</td>
        <td>
           decimalCharacter = '.', groupingCharacter = ','<br/>
           minValue = -10.5, maxValue = 10.5
        </td>
        <td><input type="text" id="numberMv1" value=""/></td>
      </tr>
      <tr>
        <td>number()</td>
        <td>
           decimalCharacter = '.', groupingCharacter = ','<br/>
```

```
            minValue = 0, maxValue = 10.5
          </td>
          <td><input type="text" id="numberMv2" value=""/></td>
        </tr>
        <tr>
          <td>number()</td>
          <td>
            decimalCharacter = '.', groupingCharacter = ','<br/>
            minValue = -10.5, maxValue = 0
          </td>
          <td><input type="text" id="numberMv3" value=""/></td>
        </tr>
        <tr>
          <td>integer()</td>
          <td>
            groupingCharacter = ','<br/>
            minValue = -10, maxValue = 10
          </td>
          <td><input type="text" id="numberMv4" value=""/></td>
        </tr>
        <tr>
          <td>integer()</td>
          <td>
            groupingCharacter = ','<br/>
            minValue = 0, maxValue = 10
          </td>
          <td><input type="text" id="numberMv5" value=""/></td>
        </tr>
        <tr>
          <td>integer()</td>
          <td>
            groupingCharacter = ','<br/>
            minValue = -10, maxValue = 0
          </td>
          <td><input type="text" id="numberMv6" value=""/></td>
        </tr>
    </table></center>
    </form>

</body>
</html>
```

`notEmpty` y `aditionalValidation` funcionan de manera idéntica que en el caso de textos.

4.3 - Fechas y horarios

Para definir campos de fechas y horarios se debe utilizar las funciones `date()`, para fechas, y `time()`, para horarios.

Como opciones de validación, las funciones aceptan los parámetros: `format`, `autofill`, `minValue`, `maxValue`, `notEmpty` y `aditionalValidation`.

`format` permite definir el formato de la fecha u hora. El formato para las fechas se constituye mediante la combinación de 'd' o 'dd' (día de uno o dos dígitos), 'm' o 'mm' (mes de uno o dos dígitos) y 'yy' o 'yyyy' (año de dos o cuatro dígitos). El formato para las horas se constituye mediante la combinación de 's' o 'ss' (segundos de uno o dos dígitos), 'm' o 'mm' (minutos de uno o dos dígitos) y 'h' o 'hh' (horas de uno o dos dígitos).

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
    .ok { background-color:#f0fff0; }
    .error { background-color:#fff0f0; }
</style>
```

```
<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.date( "date21", {"format" : 'mm/dd/yyyy'}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("date21").seahorse.verify();

      Seahorse.date( "date22", {"format" : 'dd/mm/yyyy'}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("date22").seahorse.verify();

      Seahorse.date( "date23", {"format" : 'yyyy-mm-dd'}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("date23").seahorse.verify();

      Seahorse.time( "time21", {"format" : 'hh:mm:ss'}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("time21").seahorse.verify();

      Seahorse.time( "time22", {"format" : 'hh-mm-ss'}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("time22").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>date()</td>
        <td>format = mm/dd/yyyy</td>
        <td><input type="text" id="date21" value=""/></td>
      </tr>
      <tr>
        <td>date()</td>
        <td>format = dd/mm/yyyy</td>
        <td><input type="text" id="date22" value=""/></td>
      </tr>
      <tr>
        <td>date()</td>
        <td>format = yyyy-mm-dd</td>
        <td><input type="text" id="date23" value=""/></td>
      </tr>
      <tr>
        <td>time()</td>
        <td>format = hh:mm:ss</td>
        <td><input type="text" id="time21" value=""/></td>
      </tr>
      <tr>
        <td>time()</td>
        <td>format = hh-mm-ss</td>
        <td><input type="text" id="time22" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`autofill` indica si los campos incompletos de una fecha u hora deben ser completados con la fecha u hora actual.

```
<html>

<head>
```

```
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.date( "date31", {"format" : 'yyyy-mm-dd'},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("date31").seahorse.verify();

      Seahorse.date( "date32", {"format" : 'yyyy-mm-dd', 'autofill': true},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("date32").seahorse.verify();

      Seahorse.date( "date33", {"format" : 'yyyy-mm-dd', 'autofill': false},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("date33").seahorse.verify();

      Seahorse.time( "time31", {"format" : 'hh:mm:ss'},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("time31").seahorse.verify();

      Seahorse.time( "time32", {"format" : 'hh:mm:ss', 'autofill': true},
        {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("time32").seahorse.verify();

      Seahorse.time( "time33", {"format" : 'hh:mm:ss', 'autofill': false},
{"okClass": "ok", "errorClass": "error"} );
      document.getElementById("time33").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>date()</td>
        <td>format = yyyy-mm-dd</td>
        <td><input type="text" id="date31" value=""/></td>
      </tr>
      <tr>
        <td>date()</td>
        <td>format = yyyy-mm-dd, autofill = true</td>
        <td><input type="text" id="date32" value=""/></td>
      </tr>
      <tr>
        <td>date()</td>
        <td>format = yyyy-mm-dd, autofill = false</td>
        <td><input type="text" id="date33" value=""/></td>
      </tr>
      <tr>
        <td>time()</td>
        <td>format = hh:mm:ss</td>
        <td><input type="text" id="time31" value=""/></td>
      </tr>
      <tr>
        <td>time()</td>
        <td>format = hh:mm:ss, autofill = true</td>
        <td><input type="text" id="time32" value=""/></td>
      </tr>
      <tr>
```

```
      <td>time()</td>
      <td>format = hh:mm:ss, autofill = false</td>
      <td><input type="text" id="time33" value=""/></td>
    </tr>
  </table></center>
  </form>

</body>
</html>
```

`minValue` y `maxValue` permiten definir los valores mínimos y máximos que puede tomar una entrada.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
  .ok { background-color:#f0fff0; }
  .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
  function init()
    {
    Seahorse.date( "date41", {"format" : 'dd/mm/yyyy',
      "minValue" : "01/01/1996", "maxValue" : "07/07/2010"},
      {"okClass": "ok", "errorClass": "error"} );
    document.getElementById("date41").seahorse.verify();

    Seahorse.date( "date42", {"format" : 'dd/mm/yyyy',
      "minValue" : "01/01/2010", "maxValue" : "31/12/2010"},
      {"okClass": "ok", "errorClass": "error"} );
    document.getElementById("date42").seahorse.verify();

    Seahorse.time( "time41", {"format" : 'hh:mm:ss',
      "minValue" : "00:00:00", "maxValue" : "12:00:00"},
      {"okClass": "ok", "errorClass": "error"} );
    document.getElementById("time41").seahorse.verify();

    Seahorse.time( "time42", {"format" : 'hh:mm:ss',
      "minValue" : "12:34:56", "maxValue" : "23:59:59"},
      {"okClass": "ok", "errorClass": "error"} );
    document.getElementById("time42").seahorse.verify();
    }
</script>
</head>

<body onload="javascript:init();">

  <form action="" method="GET">
  <center><table class="cuadro_ejemplo">
    <tr>
      <td>date()</td>
      <td>format = dd/mm/yyyy, minValue = "01/01/1996", maxValue = "07/07/2010"</td>
      <td><input type="text" id="date41" value=""/></td>
    </tr>
    <tr>
      <td>date()</td>
      <td>format = dd/mm/yyyy, minValue = "01/01/2010", maxValue = "31/12/2010"</td>
      <td><input type="text" id="date42" value=""/></td>
    </tr>
    <tr>
      <td>time()</td>
      <td>format = hh:mm:ss, minValue = "00:00:00", maxValue = "12:00:00"</td>
      <td><input type="text" id="time41" value=""/></td>
    </tr>
```

```
      <tr>
        <td>time()</td>
        <td>format = hh:mm:ss, minValue = "12:34:56", maxValue = "23:59:59"</td>
        <td><input type="text" id="time42" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

`notEmpty` y `aditionalValidation` funcionan de manera idéntica que en el caso de textos.

4.4 - Direcciones de Internet

Las funciones `ipAddress()`, `email()`, `http()` y `ftp()` permiten, respectivamente, definir campos para direcciones IP, direcciones de e-mail y direcciones HTTP o FTP.

Como opciones de validación, las funciones aceptan solo los parámetros `notEmpty` y `aditionalValidation`, salvo por `ipAddress()` que acepta, además, el parámetro `version` que puede tomar los valores 4 o 6.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.ipAddress( "ipv42", {"version" : 4}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("ipv42").seahorse.verify();

      Seahorse.ipAddress( "ipv62", {"version" : 6}, {"okClass": "ok", "errorClass":
"error"} );
      document.getElementById("ipv62").seahorse.verify();

      Seahorse.email( "email2", {}, {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("email2").seahorse.verify();

      Seahorse.http( "http2", {}, {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("http2").seahorse.verify();

      Seahorse.ftp( "ftp2", {}, {"okClass": "ok", "errorClass": "error"} );
      document.getElementById("ftp2").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>ipAddress()</td>
        <td>version = 4</td>
        <td><input type="text" id="ipv42" value=""/></td>
      </tr>
      <tr>
        <td>ipAddress()</td>
        <td>version = 6</td>
```

```
      <td><input type="text" id="ipv62" value=""/></td>
    </tr>
    <tr>
      <td>email()</td>
      <td></td>
      <td><input type="text" id="email2" value=""/></td>
    </tr>
    <tr>
      <td>http()</td>
      <td></td>
      <td><input type="text" id="http2" value=""/></td>
    </tr>
    <tr>
      <td>ftp()</td>
      <td></td>
      <td><input type="text" id="ftp2" value=""/></td>
    </tr>
  </table></center>
  </form>

</body>
</html>
```

4.5 - Opciones de respuesta

Las opciones de respuesta permiten determinar las acciones se realizan cuando se detecta que un campo posee un valor válido o inválido. A diferencia de las opciones de validación, sus parámetros son comunes a todas las funciones de comportamiento: `okClass`, `errorClass`, `targetErrorClass`, `targetOkClass`, `targetId`, `hiddenElementId`, `callbackFunction`, `forbidEntrance`, `autoparse` y `errorMessage`.

Los parámetros `okClass` y `errorClass` determinan las clases CSS que se agregan al campo cuando su valor es válido o inválido.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
  .colorGreen { color:green; }
  .colorRed { color:red; }
  .colorBlue { color:blue; }
  .colorYellow { color:yellow; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
  function init()
    {
    Seahorse.alphabetical( "resOpts1a", {}, {"okClass": "colorGreen", "errorClass":
"colorRed",
      "forbidEntrance": false} );
    document.getElementById("resOpts1a").seahorse.verify();

    Seahorse.alphabetical( "resOpts1b", {}, {"okClass": "colorBlue", "errorClass":
"colorYellow",
      "forbidEntrance": false} );
    document.getElementById("resOpts1b").seahorse.verify();
    }
</script>
</head>

<body onload="javascript:init();">

  <form action="" method="GET">
  <center><table class="cuadro_ejemplo">
    <tr>
```

```
      <td>alphabetical()</td>
      <td>okClass = 'colorGreen', errorClass = 'colorRed'</td>
      <td><input type="text" id="resOpts1a" value=""/></td>
    </tr>
    <tr>
      <td>alphabetical()</td>
      <td>okClass = 'colorBlue', errorClass = 'colorYellow'</td>
      <td><input type="text" id="resOpts1b" value=""/></td>
    </tr>
  </table></center>
  </form>

</body>
</html>
```

Los parámetros `targetOkClass` y `targetErrorClass` determinan las clases CSS que se agregan, luego de que el campo pierde el foco, a un elemento HTML cuando el valor del campo es válido o inválido. El parámetro `targetId` indica el id del elemento al que se agregan esas clases.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
  .colorGreen { color:green; }
  .colorRed { color:red; }
  .colorBlue { color:blue; }
  .colorYellow { color:yellow; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
  function init()
    {
    Seahorse.alphabetical( "resOpts2a", {},
      {"targetOkClass": "colorGreen", "targetErrorClass": "colorRed",
      "forbidEntrance": false, "targetId": "resOpts-targetA"} );

    Seahorse.alphabetical( "resOpts2b", {},
      {"targetOkClass": "colorBlue", "targetErrorClass": "colorYellow",
      "forbidEntrance": false, "targetId": "resOpts-targetB"} );
    }
</script>
</head>

<body onload="javascript:init();">

  <form action="" method="GET">
  <center><table class="cuadro_ejemplo">
    <tr>
      <td>alphabetical()</td>
      <td>
        targetOkClass = 'colorGreen', targetErrorClass = 'colorRed',<br/>
        targetId = 'resOpts-targetA'
      </td>
      <td><input type="text" id="resOpts2a" value=""/></td>
      <td id="resOpts-targetA">target</td>
    </tr>
    <tr>
      <td>alphabetical()</td>
      <td>
        targetOkClass = 'colorBlue', targetErrorClass = 'colorYellow',<br/>
        targetId = 'resOpts-targetB'
      </td>
      <td><input type="text" id="resOpts2b" value=""/></td>
      <td id="resOpts-targetB">target</td>
    </tr>
```

```
   </table></center>
   </form>

</body>
</html>
```

El parámetro `hiddenElementId` indica el id de un elemento HTML que se ocultará o mostrará, según el valor del campo sea válido o inválido.

```
<html>

<head>
<title>Seahorse example</title>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.alphabetical( "resOpts3a", {},
        {"hiddenElementId": "resOpts-hiddenTargetA", "forbidEntrance": false} );
      document.getElementById("resOpts3a").seahorse.verify();

      Seahorse.alphabetical( "resOpts3b", {},
        {"hiddenElementId": "resOpts-hiddenTargetB", "forbidEntrance": false} );
      document.getElementById("resOpts3b").seahorse.verify();
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>alphabetical()</td>
        <td>hiddenElementId = 'resOpts-hiddenTargetA'</td>
        <td><input type="text" id="resOpts3a" value=""/></td>
        <td><span id="resOpts-hiddenTargetA">target</span></td>
      </tr>
      <tr>
        <td>alphabetical()</td>
        <td>hiddenElementId = 'resOpts-hiddenTargetB'</td>
        <td><input type="text" id="resOpts3b" value=""/></td>
        <td><span id="resOpts-hiddenTargetB">target</span></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

El parámetro `callbackFunction` permite definir una función que es invocada cuando se pierde el foco del campo y a la cual se le indica si el mismo tiene un valor válido o inválido.

```
<html>

<head>
<title>Seahorse example</title>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.alphabetical( "resOpts4a", {},
        {"callbackFunction": function(elem, valid) {alert(valid);}, "forbidEntrance":
false} );
```

```
     Seahorse.numeric( "resOpts4b", {},
       {"callbackFunction": function(elem, valid) {alert(valid);}, "forbidEntrance":
false} );
     }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
     <tr>
       <td>alphabetical()</td>
       <td>callbackFunction = function(elem, valid) {alert(valid);}</td>
       <td><input type="text" id="resOpts4a" value=""/></td>
     </tr>
     <tr>
       <td>numeric()</td>
       <td>callbackFunction = function(elem, valid) {alert(valid);}</td>
       <td><input type="text" id="resOpts4b" value=""/></td>
     </tr>
   </table></center>
   </form>

</body>
</html>
```

El parámetro `forbidEntrance` permite indicar si se debe evitar, mediante el evento `onKeyPress`, el ingreso de caracteres que provocarían que el valor del campo sea inválido. Por defecto, se impide el ingreso de dichos caracteres.

```
<html>

<head>
<title>Seahorse example</title>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
     {
     Seahorse.numeric( "resOpts5a", {}, {"forbidEntrance": false} );

     Seahorse.numeric( "resOpts5b", {}, {"forbidEntrance": true} );
     }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
     <tr>
       <td>numeric()</td>
       <td>forbidEntrance = 'false'</td>
       <td><input type="text" id="resOpts5a" value=""/></td>
     </tr>
     <tr>
       <td>numeric()</td>
       <td>forbidEntrance = 'true'</td>
       <td><input type="text" id="resOpts5b" value=""/></td>
     </tr>
   </table></center>
   </form>

</body>
</html>
```

El parámetro `autoparse` permite indicar si se debe convertir el contenido del campo, luego de que se pierde el foco del mismo, a fin de evitar la aparición de caracteres innecesarios.

```
<html>

<head>
<title>Seahorse example</title>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.integer( "resOpts6a", {"groupingCharacter" : ','}, {"autoparse": false} );

      Seahorse.integer( "resOpts6b", {"groupingCharacter" : ','}, {"autoparse": true} );
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET">
   <center><table class="cuadro_ejemplo">
      <tr>
        <td>integer()</td>
        <td>groupingCharacter : ',', autparse = 'false'</td>
        <td><input type="text" id="resOpts6a" value=""/></td>
      </tr>
      <tr>
        <td>integer()</td>
        <td>groupingCharacter : ',', autparse = 'true'</td>
        <td><input type="text" id="resOpts6b" value=""/></td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

El parámetro `errorMessage` define un mensaje que explica las condiciones que debe cumplir un valor ingresado al campo para ser considerado válido. Este parámetro solo tiene sentido definirlo cuando al formulario del campo se le asignó un comportamiento a través de una de las funciones de esta librería.

4.6 - Formularios

La función `form()` permite agregar un comportamiento a los formularios, según el cual, antes de enviarse un formulario, se verifica si todos sus campos poseen valores válidos. Esta función recibe, como parámetros, el id del formulario, la función a la que se invoca cuando se intenta enviar el formulario pero uno o más de sus campos poseen valores inválidos y un mensaje de error.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.text("form0_username", {"notEmpty": true},
          {"okClass": "ok", "errorClass": "error", "errorMessage": "Username required"});
```

```
      document.getElementById("form0_username").seahorse.verify();

      Seahorse.email("form0_email", {"notEmpty": true},
          {"okClass": "ok", "errorClass": "error", "errorMessage": "E-mail address
required"});
      document.getElementById("form0_email").seahorse.verify();

      Seahorse.form("form0", function(msjs, array) {alert(msjs);}, "Submit error" );
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET" id="form0">
   <center><table class="cuadro_formulario">
      <tr>
        <td align="right">Username</td>
        <td align="left"><input id="form0_username" type="text" name="username" value=""/
></td>
      </tr>
      <tr>
        <td align="right">Password</td>
        <td align="left"><input type="password" name="password" value="1234"/></td>
      </tr>
      <tr>
        <td align="right" valign="top">Gender</td>
        <td align="left">
          <input type="radio" name="gender" value="M" checked/><br/>
          <input type="radio" name="gender" value="F"/>
        </td>
      </tr>
      <tr>
        <td align="right">E-mail</td>
        <td align="left"><input id="form0_email" type="text" name="email" value=""/></td>
      </tr>
      <tr>
        <td align="right">Receive news</td>
        <td align="left"><input type="checkbox" name="news" value="true"/><br/></td>
      </tr>
      <tr>
        <td align="right" valign="top">Preferences</td>
        <td align="left"><select name="preferences" multiple="multiple">
          <option>Option A</option>
          <option>Option B</option>
           <option>Option C</option>
           <option>Option D</option>
        </select></td>
      </tr>
      <tr>
        <td colspan="2" align="right">
          <button id="form_0_btn">Send</button>
        </td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

4.7 - El objeto 'seahorse'

Cuando se asigna un comportamiento a un elemento, se crea un objeto que es guardado, como un atributo, en el elemento. Este objeto almacena las opciones de validación y de respuesta del comportamiento del elemento, así también como la definición de las funciones `validate()`, `parse()` y `verify()`.

`validate()` y `verify()` verifican si el campo posee un valor válido o no, con la diferencia de que

`verify()`, además, ejecuta las acciones definidas por las opciones de respuesta, por lo cual a veces puede resultar conveniente invocar a esta función luego de que se carga la página.

`parse()`, si el campo es un número, un entero, una fecha, un horario o una dirección IP, devuelve el valor convertido, según su tipo, del campo.

Dado que los comportamientos de Seahorse hacen uso de los eventos onKeyUp, onBlur y onKeyPress para implementar la validación en tiempo real, en el objeto 'seahorse' se almacenan también las funciones, relacionadas a estos eventos, que el elemento poseía antes de asignársele el comportamiento . Estas funciones son invocadas después que las funciones relacionadas a la validación en tiempo real.

5 - Otras funciones

5.1 - Comparaciones

Las funciones `compareDate()` y `compareTime()` reciben dos fechas o dos horas, respectivamente, y verifican cual de ellas es mayor que la otra. Reciben como parámetros las fechas o las horas y el formato de las mismas, y devuelven un valor negativo si la primer fecha u hora es menor que la segunda, cero si son iguales y un valor positivo si la primer fecha u hora es mayor que la segunda.

5.2 - Serialización

La función `serialize()` codifica todos los valores de los elementos de un formulario en una cadena de texto, utilizando las codificaciones URL o JSON.

6 - jQuery

El plugin de Seahorse para jQuery permite utilizar los selectores de jQuery para asignar comportamientos o para verificar si los campos poseen valores válidos, reduciendo la cantidad de líneas de código y facilitando su mantenimiento.

Proporciona las funciones `seaBehavior()`, que asigna comportamientos a los campos seleccionados, `seaForm()`, que asigna comportamientos a los formularios seleccionados, `seaValidate()`, que verifica si todos los elementos poseen valores válidos, y `seaVerify()` que invoca al método `verify()` de todos los elementos.

```
<html>

<head>
<title>Seahorse example</title>

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript" src="js/seahorse.jquery-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      jQuery(".integer").seaBehavior( "integer",
         {"groupingCharacter" : ','},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Integer error"} );

      jQuery(".number").seaBehavior( "number",
         {"decimalCharacter": '.', "groupingCharacter" : ','},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Number error"} );

      jQuery(".date").seaBehavior( "date",
         {"format" : 'dd/mm/yyyy'},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Date error"} );

      jQuery(".time").seaBehavior( "time",
         {"format" : 'hh-mm-ss'},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Time error"} );

      jQuery(".alphabetical").seaBehavior( "alphabetical", {},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Alphabetical error"} );

      jQuery(".alphanumeric").seaBehavior( "alphanumeric", {},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Alphanumeric error"} );

      jQuery(".numeric").seaBehavior( "numeric", {},
         {"okClass": "ok", "errorClass": "error", "errorMessage": "Numeric error"} );

      jQuery(":input").seaVerify();
      jQuery("#formulario").seaForm(function(msjs, array) {alert(msjs);}, "Submit
error");
      }
</script>
</head>

<body onload="javascript:init();">

   <form action="" method="GET" id="formulario">
   <center><table class="cuadro_ejemplo">
      <tr>
         <td>number()</td>
         <td><input type="text" class="number" value="1,000.00"/></td>
      </tr>
      <tr>
```

```
        <td>integer()</td>
        <td><input type="text" class="integer" value="1,000"/></td>
      </tr>
      <tr>
        <td>date()</td>
        <td><input type="text" class="date" value="31/12/2000"/></td>
      </tr>
      <tr>
        <td>time()</td>
        <td><input type="text" class="time" value="23:59:59"/></td>
      </tr>
      <tr>
        <td>alphanumeric()</td>
        <td><input type="text" class="alphanumeric" value="Alphanumeric123"/></td>
      </tr>
      <tr>
        <td>alphabetical()</td>
        <td><input type="text" class="alphabetical" value="Alphabetical"/></td>
      </tr>
      <tr>
        <td>numeric()</td>
        <td><input type="text" class="numeric" value="1234"/></td>
      </tr>
      <tr>
        <td colspan="2" align="right">
           <input type="submit" value="Submit"/>
        </td>
      </tr>
   </table></center>
   </form>

</body>
</html>
```

7 - Anexos

7.1 - Especificación API

7.1.1 - Sumary

About Seahorse

Introduction	Seahorse is an open source JavaScript library created for simplify the use of forms, particularly the form validation.
License	This library is licensed under the LGPL Version 3 license.

Validation behaviors

Validation options	The validation options are parameters that restrict the values, or the format, that the data of an input can have to be considered valid.
Response options	The response options are parameters that determines the courses of actions when an user enter a valid or invalid input.

Functions

Validation behaviors

text()	Gives to an element a text input behavior.
alphabetical()	Gives to an element an alphabetical input behavior.
alphanumeric()	Gives to an element an alphanumeric input behavior.
numeric()	Gives to an element a numeric input behavior.
number()	Gives to an element a float input behavior.
integer()	Gives to an element an integer input behavior.
date()	Gives to an element a date input behavior.
time()	Gives to an element a time input behavior.
ipAddress()	Gives to an element an IP address input behavior.
email()	Gives to an element an e-mail address input behavior.
http()	Gives to an element a HTTP address input behavior.
ftp()	Gives to an element a FTP address input behavior.
form()	Gives to an element a form behavior.
removeBehavior()	Removes, from a element, any behavior asigned by a function of the Seahorse's library.

Validation functions

isNumber()	Checks if a string represents a number.
isInteger()	Checks if a string represents an integer.
isNumeric()	Checks if a string contains only numbers.
isAlphabetical()	Checks if a string contains only alphabetical characters (including accents and another not Ascii characters).
isAlphanumeric()	Checks if a string contains only alphanumeric characters.
isAlphabeticalAscii()	Checks if a string contains only alphabetical ASCII characters.
isAlphanumericAscii()	Checks if a string contains only alphanumeric ASCII characters.
isAsciiText()	Checks if a string contains only ASCII characters.
isIPv4()	Checks if a string represents an IP version 4 address.
isIPv6()	Checks if a string represents an IP version 6 address.
isEmail()	Checks if a string represents an e-mail address.
isHttp()	Checks if a string represents a HTTP address.
isFtp()	Checks if a string represents a FTP address.
isMonth()	Checks if a string represents a month.
isDay()	Checks if a string represents a day of a particular month.
isDate()	Checks if string represents a date.
isTime()	Checks if string represents an instant of time.
isDateFormat()	Checks if string is a valid date format.
isTimeFormat()	Checks if string is a valid time format.
validateForm()	Checks if all the elements of a form have valid values.
passFilter()	Acording to the value of 'contains', checks if all the characters of 'cad' are contained in 'filter' or if neither of the characters are contained.

Parsing functions

parseNumber()	Parses a string and returns an number.
parseInteger()	Parses a string and returns a integer.
parseIPv4()	Parses a string that represents an IPv4 address and returns a array with four numbers.
parseIPv6()	Parses a string that represents an IPv6 address and returns a array with eight hexadecimal numbers.
parseDate()	Receives a string representing a date and transforms it according to the format passed as parameter.
parseTime()	Receives a string representing an instant of time and transforms it according to the format passed as parameter.
filterString()	Returns all the characters of 'cad' that are contained, or that aren't contained, in 'filter'.

Others

compareDate()	Compare two strings that represents dates.
compareTime()	Compare two strings that represents times.
serialize()	Encode the elements of a form as a string.

7.1.2 - About Seahorse

Introduction

Seahorse is an open source JavaScript library created for simplify the use of forms, particularly the form validation.

It provides functions to validate, parse and serialize data and to add real-time validation to inputs.

Seahorse can be used alone or it can be used with jQuery, using the plugin designed for that purpose.

License

This library is licensed under the LGPL Version 3 license.

That means you can use Seahorse to develop commercial or open source projects, but, in both cases, you must publish, under a licence compatible with LGPL v3, any changes you make to the library.

7.1.3 - Validation behaviors

One of the main features of this library are the functions to assign validation behaviors to the inputs.

An input with a validation behavior can avoid the input of invalid characters or perform diferent actions (like add a CSS class to a element or invoke a JavaScript function) when a user inserts valid or invalid data.

The validation behaviors functions receives as parameter the id of the element to be validated, a JSON element with the validation options and a JSON element with the response options.

Validation options

The validation options are parameters that restrict the values, or the format, that the data of an input can have to be considered valid.

The variables that can be defined in the JSON element of validation options are:

notEmpty	A boolean indicating if the field can be left in blank.
minLength	The minimum length of a text input (by default, 0).
maxLength	The maximun length of a text input (by default, infinity).
minValue	The minimum value of a input (by default, minus infinity for numbers and null for dates and times).
maxValue	The maximun value of a input (by default, infinity for numbers and null for dates and times).
format	The format of a input (by default 'yyyy-mm-dd' for dates and 'hh:mm:ss' for times).
autofill	A boolean indicating if the incomplete fields of dates or times must be completed with the actual date or time.
version	The version of an IP address input (by default, 4).
requiredCharacters	A string or a array of unicode values representing the group of characters that a text input must have to be considered valid.
forbiddenCharacters	A string or a array of unicode values representing the group of characters that a text input must not have to be considered valid.
allowedCharacters	A string or a array of unicode values representing the group of characters that a text input can have and be considered valid, no matter the restrictions of his type.
asciiCharacters	A boolean indicating if a text input must have only ASCII characters (by default 'false').
decimalCharacter	The character used as decimal character in a numeric input (by default '.').
groupingCharacter	The character used as grouping character in a numeric input (by default ',').

aditionalValidation	A user's defined function that test if a field has a valid value. This function is called only if the field has been validated by the standard Seahorse's validation function.

This variables are optionals and they are not used by all the functions, each function uses only a few variables, for example, for restrict the lenght, text() uses minLength and maxLength while number() uses minValue and maxValue.

Response options

The response options are parameters that determines the courses of actions when an user enter a valid or invalid input.

The variables that can be defined in the JSON element of response options are:

errorClass	The class added to the input if the data entered is invalid.
okClass	The class added to the input if the data entered is valid.
targetErrorClass	The class added to a given element if the data entered is invalid.
targetOkClass	The class added to a given element if the data entered is valid.
targetId	The id of the element to which add the classes 'targetErrorClass' or 'targetOkClass'.
hiddenElementId	The id of the element to hide or show, depending if the data entered is valid or invalid.
callbackFunction	A function to invoke after that the input was validated. The function must receive two parameters: 'element' (the object that represents the input) and 'valid' (a boolean indicating if the data entered is valid or not).
forbidEntrance	A boolean indicating if the entrance of invalid characters must be forbidden (by default, 'true').
autoparse	A boolean indicating if the data in the fields must be parsed in order to eliminate unnecessary characters.
errorMessage	A message explaining why the value of the input is invalid.

This variables are optionals and can be specified for all the validation functions.

7.1.4 - Functions

Sumary

Validation behaviors

text()	Gives to an element a text input behavior.
alphabetical()	Gives to an element an alphabetical input behavior.
alphanumeric()	Gives to an element an alphanumeric input behavior.
numeric()	Gives to an element a numeric input behavior.
number()	Gives to an element a float input behavior.
integer()	Gives to an element an integer input behavior.
date()	Gives to an element a date input behavior.
time()	Gives to an element a time input behavior.
ipAddress()	Gives to an element an IP address input behavior.
email()	Gives to an element an e-mail address input behavior.
http()	Gives to an element a HTTP address input behavior.
ftp()	Gives to an element a FTP address input behavior.
form()	Gives to an element a form behavior.
removeBehavior()	Removes, from a element, any behavior asigned by a function of the Seahorse's library.

Validation functions

isNumber()	Checks if a string represents a number.
isInteger()	Checks if a string represents an integer.
isNumeric()	Checks if a string contains only numbers.
isAlphabetical()	Checks if a string contains only alphabetical characters (including accents and another not Ascii characters).
isAlphanumeric()	Checks if a string contains only alphanumeric characters.
isAlphabeticalAscii()	Checks if a string contains only alphabetical ASCII characters.
isAlphanumericAscii()	Checks if a string contains only alphanumeric ASCII characters.
isAsciiText()	Checks if a string contains only ASCII characters.
isIPv4()	Checks if a string represents an IP version 4 address.
isIPv6()	Checks if a string represents an IP version 6 address.
isEmail()	Checks if a string represents an e-mail address.
isHttp()	Checks if a string represents a HTTP address.

isFtp()	Checks if a string represents a FTP address.
isMonth()	Checks if a string represents a month.
isDay()	Checks if a string represents a day of a particular month.
isDate()	Checks if string represents a date.
isTime()	Checks if string represents an instant of time.
isDateFormat()	Checks if string is a valid date format.
isTimeFormat()	Checks if string is a valid time format.
validateForm()	Checks if all the elements of a form have valid values.
passFilter()	Acording to the value of 'contains', checks if all the characters of 'cad' are contained in 'filter' or if neither of the characters are contained.

Parsing functions

parseNumber()	Parses a string and returns an number.
parseInteger()	Parses a string and returns a integer.
parseIPv4()	Parses a string that represents an IPv4 address and returns a array with four numbers.
parseIPv6()	Parses a string that represents an IPv6 address and returns a array with eight hexadecimal numbers.
parseDate()	Receives a string representing a date and transforms it according to the format passed as parameter.
parseTime()	Receives a string representing an instant of time and transforms it according to the format passed as parameter.
filterString()	Returns all the characters of 'cad' that are contained, or that aren't contained, in 'filter'.

Others

compareDate()	Compare two strings that represents dates.
compareTime()	Compare two strings that represents times.
serialize()	Encode the elements of a form as a string.

Validation behaviors

text()

```
text : function(  element,
                  validationOptions,
                  responseOptions    )
```

Gives to an element a text input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minLength, maxLength, asciiCharacters, allowedCharacters, requiredCharacters and forbiddenCharacters).
responseOptions	A JSON element with the response options.

alphabetical()

```
alphabetical : function(  element,
                          validationOptions,
                          responseOptions    )
```

Gives to an element an alphabetical input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minLength, maxLength, asciiCharacters, allowedCharacters, requiredCharacters and forbiddenCharacters).
responseOptions	A JSON element with the response options.

alphanumeric()

```
alphanumeric : function(  element,
                          validationOptions,
                          responseOptions    )
```

Gives to an element an alphanumeric input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minLength, maxLength, asciiCharacters, allowedCharacters, requiredCharacters and forbiddenCharacters).
responseOptions	A JSON element with the response options.

numeric()

```
numeric : function(  element,
                     validationOptions,
                     responseOptions    )
```

Gives to an element a numeric input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minLength, maxLength, asciiCharacters, allowedCharacters, requiredCharacters and forbiddenCharacters).
responseOptions	A JSON element with the response options.

number()

```
number : function(  element,
                    validationOptions,
                    responseOptions    )
```

Gives to an element a float input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minValue, maxValue, groupingCharacter and decimalCharacter).
responseOptions	A JSON element with the response options.

integer()

```
integer : function(  element,
                     validationOptions,
                     responseOptions    )
```

Gives to an element an integer input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minValue, maxValue and groupingCharacter).

responseOptions	A JSON element with the response options.

date()

```
date : function(  element,
                  validationOptions,
                  responseOptions     )
```

Gives to an element a date input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minValue, maxValue, autofill and format).
responseOptions	A JSON element with the response options.

time()

```
time : function(  element,
                  validationOptions,
                  responseOptions     )
```

Gives to an element a time input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation, minValue, maxValue, autofill and format).
responseOptions	A JSON element with the response options.

ipAddress()

```
ipAddress : function(  element,
                       validationOptions,
                       responseOptions     )
```

Gives to an element an IP address input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty, aditionalValidation and version).
responseOptions	A JSON element with the response options.

email()

```
email : function(  element,
                   validationOptions,
                   responseOptions     )
```

Gives to an element an e-mail address input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty and aditionalValidation).
responseOptions	A JSON element with the response options.

http()

```
http : function(  element,
                  validationOptions,
                  responseOptions    )
```

Gives to an element a HTTP address input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty and aditionalValidation).
responseOptions	A JSON element with the response options.

ftp()

```
ftp : function(  element,
                 validationOptions,
                 responseOptions    )
```

Gives to an element a FTP address input behavior.

Parameters

element	The element id or the element object.
validationOptions	A JSON element with the validation options (notEmpty and aditionalValidation).
responseOptions	A JSON element with the response options.

form()

```
form : function(  form,
                  responseFunction,
                  errorMessage       )
```

Gives to an element a form behavior.
That means that it modifies the submit method of the form in order to validate all the inputs before submit. The 'responseFunction' parameter is the function called when the form is submited but one or more inputs, of the form, has invalid values. This function must receive two parameters: a string with the error messages of the form and the inputs and an array with all the input elements that have invalid values.

Parameters

form	The form id or the form object.
responseFunction	The function called if one of more inputs of the form are invalid.
errorMessage	The message to be displayed if one of more inputs of the form are invalid.

removeBehavior()

```
removeBehavior : function(    element )
```

Removes, from a element, any behavior asigned by a function of the Seahorse's library.

Parameters

element The element id or the element object.

isNumber()

```
isNumber : function(  cad,
                      ds,
                      gs   )
```

Checks if a string represents a number.

Parameters
cad A string.
ds The character used as digital separator.
gs The character used as grouping separator.

Returns
`true` if the string represents a number and `false` in the opposite case.

isInteger()

```
isInteger : function(  cad,
                       gs   )
```

Checks if a string represents an integer.

Parameters
cad A string.
gs The character used as grouping separator.

Returns
`true` if the string represents an integer and `false` in the opposite case.

isNumeric()

```
isNumeric : function(  cad )
```

Checks if a string contains only numbers.

Parameters
cad A string.

Returns
`true` if the string contains only numbers and `false` in the opposite case.

isAlphabetical()

```
isAlphabetical : function(  cad )
```

Checks if a string contains only alphabetical characters (including accents and another not Ascii characters).

Parameters
cad A string.

Returns
`true` if the string contains only alphabetical characters and `false` in the opposite case.

isAlphanumeric()

isAlphanumeric : function(cad)

Checks if a string contains only alphanumeric characters.

Parameters
cad A string.

Returns
`true` if the string contains only alphanumeric characters and `false` in the opposite case.

isAlphabeticalAscii()

isAlphabeticalAscii : function(cad)

Checks if a string contains only alphabetical ASCII characters.

Parameters
cad A string.

Returns
`true` if the string contains only alphabetical ASCII characters and `false` in the opposite case.

isAlphanumericAscii()

isAlphanumericAscii : function(cad)

Checks if a string contains only alphanumeric ASCII characters.

Parameters
cad A string.

Returns
`true` if the string contains only alphanumeric ASCII characters and `false` in the opposite case.

isAsciiText()

isAsciiText : function(cad)

Checks if a string contains only ASCII characters.

Parameters
cad A string.

Returns
`true` if the string contains only ASCII characters and `false` in the opposite case.

isIPv4()

isIPv4 : function(ip)

Checks if a string represents an IP version 4 address.

Parameters
ip A string.

Returns
`true` if the string represents an IP version 4 address and `false` in the opposite case.

isIPv6()
isIPv6 : function(ip)

Checks if a string represents an IP version 6 address.

Parameters
ip A string.

Returns
`true` if the string represents an IP version 6 address and `false` in the opposite case.

isEmail()
isEmail : function(mail)

Checks if a string represents an e-mail address.

Parameters
mail A string.

Returns
`true` if the string represents an e-mail address and `false` in the opposite case.

isHttp()
isHttp : function(http)

Checks if a string represents a HTTP address.

Parameters
http A string.

Returns
`true` if the string represents a HTTP address and `false` in the opposite case.

isFtp()
isFtp : function(ftp)

Checks if a string represents a FTP address.

Parameters
ftp A string.

Returns
`true` if the string represents a FTP address and `false` in the opposite case.

isMonth()
isMonth : function(mes)

Checks if a string represents a month. This function considers that a month is a number that must be between 1 and 12 (while the class Date considers a month like a number between 0 and 11).

Parameters
mes A string.

Returns
`true` if the string represents a month and `false` in the opposite case.

isDay()

```
isDay : function(  dia,
                   mes,
                   anio )
```

Checks if a string represents a day of a particular month. This function considers that a month is a number that must be between 1 and 12 (while the class Date considers a month like a number between 0 and 11).

Parameters
dia A string that represents a day.
mes A string that represents a month.
anio A string that represents a year.

Returns
`true` if 'dia' represents a day of the month 'mes' and `false` in the opposite case or if 'mes' it's not a month.

isDate()

```
isDate : function(  cad,
                    format,
                    fill       )
```

Checks if string represents a date.

Parameters
cad A string that represents a date.
format A string that represents a date format.
fill A boolean that indicates if the empty fields have to be completed with the actual date.

Returns
`true` if the string represents a date and `false` in the opposite case.

isTime()

```
isTime : function(  cad,
                    format,
                    fill       )
```

Checks if string represents an instant of time.

Parameters
cad A string that represents an instant of time.
format A string that represents a time format.
fill A boolean that indicates if the empty fields have to be completed with the actual time.

Returns

`true` if the string represents an instant of time and `false` in the opposite case.

isDateFormat()

isDateFormat : function(format)

Checks if string is a valid date format.

Parameters
cad A string that represents a date format.

Returns
`true` if the string is a valid date format and `false` in the opposite case.

isTimeFormat()

isTimeFormat : function(format)

Checks if string is a valid time format.

Parameters
cad A string that represents a time format.

Returns
`true` if the string is a valid time format and `false` in the opposite case.

validateForm()

validateForm : function(idForm)

Checks if all the elements of a form have valid values.

Parameters
idForm The id of the form to be validated.

Returns
`true` if all the elements of the form have valid values. `false` if one or more elements of the form have invalid values.

passFilter()

passFilter : function(cad,
filter,
contains)

Acording to the value of 'contains', checks if all the characters of 'cad' are contained in 'filter' or if neither of the characters are contained.

Parameters
cad A string to filter.
filter A string used as filter.
contains A boolean indicating the mode of operation.

Returns
`true` if all the characters of 'cad' are contained in 'filter' and `false` in the oposite case. `true` if neither of the characters of 'cad' are contained in 'filter' and `false` in the opposite case.

Parsing functions

parseNumber()

```
parseNumber : function(  cad,
                         ds,
                         gs   )
```

Parses a string and returns an number.

Parameters
cad A string.
ds The character used as digital separator.
gs The character used as grouping separator.

Returns
A number if the string is a number or `NaN` in the opposite case.

parseInteger()

```
parseInteger : function(  cad,
                          gs   )
```

Parses a string and returns a integer.

Parameters
cad A string.
gs The character used as grouping separator.

Returns
An integer if the string is a number or `NaN` in the opposite case.

parseIPv4()

```
parseIPv4 : function(  ip )
```

Parses a string that represents an IPv4 address and returns a array with four numbers.

Parameters
ip A string.

Returns
An array of four numbers if the string represents an IPv4 address or `null` in the opposite case.

parseIPv6()

```
parseIPv6 : function(  ip )
```

Parses a string that represents an IPv6 address and returns a array with eight hexadecimal numbers.

Parameters
ip A string.

Returns
An array of eight hexadecimal numbers if the string represents an IPv6 address or `null` in the opposite case.

parseDate()

```
parseDate : function(  cad,
                       format,
                       fill       )
```

Receives a string representing a date and transforms it according to the format passed as parameter.

Parameters
cad A string that represents a date.
format A string that represents a valid date format.
fill A boolean that indicates if the empty fields have to be completed with the actual date.

Returns
A formated string that represents a date if 'cad' represents a date and 'format' is a valid date format or `null` in the opposite case.

parseTime()

```
parseTime : function(  cad,
                       format,
                       fill       )
```

Receives a string representing an instant of time and transforms it according to the format passed as parameter.

Parameters
cad A string that represents an instant of time.
format A string that represents a valid time format.
fill A boolean that indicates if the empty fields have to be completed with the actual time.

Returns
A formated string that represents an instant of time if 'cad' represents an instant of time and 'format' is a valid time format or `null` in the opposite case.

filterString()

```
filterString : function(  cad,
                          filter,
                          contains )
```

Returns all the characters of 'cad' that are contained, or that aren't contained, in 'filter'.

Parameters
cad A string to filter.
filter A string used as filter.
contains A boolean indicating the mode of operation.

Returns
All the characters of 'cad' that are contained in 'filter', if 'contains' is equal to `true`. All the characters of 'cad' that aren't contained in 'filter', if 'contains' is equal to `false`.

Others

compareDate()

```
compareDate : function(  date1,
```

```
                                date2,
                                dateFormat )
```

Compare two strings that represents dates.

Parameters

date1	The first date.
date2	The second date.
dateFormat	The date format of the two dates.

Returns

A negative number if date1 < date2, a positive number if date1 > date2, zero if date1 = date2 or null in case of error.

compareTime()

```
compareTime : function(   time1,
                          time2,
                          timeFormat  )
```

Compare two strings that represents times.

Parameters

time1	The first time.
time2	The second time.
timeFormat	The date format of the two times.

Returns

A negative number if time1 < time2, a positive number if time1 > time2, zero if time1 = time2 or null in case of error.

serialize()

```
serialize : function(   form,
                        notation )
```

Encode the elements of a form as a string.

Parameters

form	The form id or the form object.
codification	The notation to be used (JSON or URL)

Returns

The form serialized.

7.2 - Ejemplos

7.2.1 - Validación

```
<html>

<head>
<title>Seahorse's examples - Validation</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="js/seahorse-1.2.js"></script>

</head>
<body>

   <table>
      <tr>
         <td>Seahorse.isNumber("1.000.000,00", ',', '.')</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isNumber("1.000.000,00", ',', '.'));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.isInteger("1,000,000", ',')</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="=" /></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isInteger("1,000,000", ','));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.isNumeric("1234")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isNumeric("1234"));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.isAlphabetical("abcd&ntilde;")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isAlphabetical("abcd\u00c9"));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.isAlphanumeric("abcd&ntilde;123")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isAlphanumeric("abcd\u00c9123"));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.isAlphabeticalAscii("abcd")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isAlphabeticalAscii("abcd"));
            </script>
         </td>
```

```
</tr>
<tr>
   <td>Seahorse.isAlphanumericAscii("abc123")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isAlphanumericAscii("abcd"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isAsciiText("abcd (123) @#")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isAsciiText("abcd"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isIPv4("192.168.0.1")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isIPv4("192.168.0.1"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isIPv6("11:22:33:44:55:66:77:88")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isIPv6("11:22:33:44:55:66:77:88"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isEmail("test@test.com")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isEmail("test@test.com"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isHttp("http://www.test.com")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isHttp("http://www.test.com"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isFtp("ftp://ftp.test.com")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
      <script type="text/javascript">
         document.write(Seahorse.isFtp("ftp://ftp.test.com"));
      </script>
   </td>
</tr>
<tr>
   <td>Seahorse.isDate("31/07/2010","dd/mm/yyyy")</td>
   <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
   <td>
```

```
            <script type="text/javascript">
               document.write(Seahorse.isDate("31/07/2010","dd/mm/yyyy"));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.isTime("06:20:22","hh:mm:ss")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.isTime("06:20:22","hh:mm:ss"));
            </script>
         </td>
      </tr>
   </table>

</body>
</html>
```

7.2.2 - Conversión

```
<html>

<head>
<title>Seahorse's examples - Parsing</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<script type="text/javascript" src="js/seahorse-1.2.js"></script>

</head>
<body>

   <table>
      <tr>
         <td>Seahorse.parseNumber("1,234.5", '.', ',')</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.parseNumber("1,234.5", '.', ','));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.parseInteger("1,234", ',')</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.parseInteger("1,234", ','));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.parseIPv4("192.168.0.1")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.parseIPv4("192.168.0.1"));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.parseIPv6("11:22:33:44:55:66:77:88")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.parseIPv6("11:22:33:44:55:66:77:88"));
            </script>
         </td>
      </tr>
```

```
      <tr>
         <td>Seahorse.parseDate("31/07/2010","dd/mm/yyyy")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.parseDate("31/07/2010","dd/mm/yyyy"));
            </script>
         </td>
      </tr>
      <tr>
         <td>Seahorse.parseTime("02:05:33","hh:mm:ss")</td>
         <td><img src="img/icon-arrow-right.png" title="=" alt="="/></td>
         <td>
            <script type="text/javascript">
               document.write(Seahorse.parseTime("02:05:33","hh:mm:ss"));
            </script>
         </td>
      </tr>
   </table>

</body>
</html>
```

7.2.3 - Comportamientos

```
<html>

<head>
<title>Seahorse's examples - Parsing</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      Seahorse.number( "number1", {"decimalCharacter": '.', "groupingCharacter" : ','},
{ "okClass": "ok", "errorClass": "error" } );
      document.getElementById("number1").seahorse.verify();

      Seahorse.integer( "integer1", {"groupingCharacter" : ','}, { "okClass": "ok",
"errorClass": "error" } );
      document.getElementById("integer1").seahorse.verify();

      Seahorse.date( "date1", {"format" : 'dd/mm/yyyy', 'autofill': true}, { "okClass":
"ok", "errorClass": "error" } );
      document.getElementById("date1").seahorse.verify();

      Seahorse.time( "time1", {"format" : 'hh:mm:ss', 'autofill': true}, { "okClass":
"ok", "errorClass": "error" } );
      document.getElementById("time1").seahorse.verify();

      Seahorse.ipAddress( "ipv41", {"version" : 4 }, { "okClass": "ok", "errorClass":
"error" } );
      document.getElementById("ipv41").seahorse.verify();

      Seahorse.email( "email1", {}, { "okClass": "ok", "errorClass": "error" } );
      document.getElementById("email1").seahorse.verify();

      Seahorse.http( "http1", {}, { "okClass": "ok", "errorClass": "error" } );
      document.getElementById("http1").seahorse.verify();

      Seahorse.ftp( "ftp1", {}, { "okClass": "ok", "errorClass": "error" } );
      document.getElementById("ftp1").seahorse.verify();
```

```
     Seahorse.text( "text1", {}, { "okClass": "ok", "errorClass": "error" } );
     document.getElementById("text1").seahorse.verify();

     Seahorse.alphabetical( "alphabetical1", {}, { "okClass": "ok", "errorClass": "error"
} );
     document.getElementById("alphabetical1").seahorse.verify();

     Seahorse.alphanumeric( "alphanumeric1", {}, { "okClass": "ok", "errorClass": "error"
} );
     document.getElementById("alphanumeric1").seahorse.verify();

     Seahorse.numeric( "numeric1", {}, { "okClass": "ok", "errorClass": "error" } );
     document.getElementById("numeric1").seahorse.verify();
     }
</script>

</head>
<body onload="javascript:init();">

   <table>
     <tr>
        <td>number()</td>
        <td><input type="text" id="number1" value="1,000.00"/></td>
     </tr>
     <tr>
        <td>integer()</td>
        <td><input type="text" id="integer1" value="1,000"/></td>
     </tr>
     <tr>
        <td>date()</td>
        <td><input type="text" id="date1" value="31/12/2000"/></td>
     </tr>
     <tr>
        <td>time()</td>
        <td><input type="text" id="time1" value="23:59:59"/></td>
     </tr>
     <tr>
        <td>ipAddress()</td>
        <td><input type="text" id="ipv41" value="192.168.0.1"/></td>
     </tr>
     <tr>
        <td>email()</td>
        <td><input type="text" id="email1" value="test@server.com"/></td>
     </tr>
     <tr>
        <td>http()</td>
        <td><input type="text" id="http1" value="http://www.test.com"/></td>
     </tr>
     <tr>
        <td>ftp()</td>
        <td><input type="text" id="ftp1" value="ftp://ftp.test.com"/></td>
     </tr>
     <tr>
        <td>text()</td>
        <td><input type="text" id="text1" value="Text"/></td>
     </tr>
     <tr>
        <td>alphanumeric()</td>
        <td><input type="text" id="alphanumeric1" value="Alphanumeric123"/></td>
     </tr>
     <tr>
        <td>alphabetical()</td>
        <td><input type="text" id="alphabetical1" value="Alphabetical"/></td>
     </tr>
     <tr>
        <td>numeric()</td>
        <td><input type="text" id="numeric1" value="1234"/></td>
```

```
      </tr>
   </table>

</body>
</html>
```

7.2.4 - jQuery

```
<html>

<head>
<title>Seahorse's examples - Parsing</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<style type="text/css">
   .ok { background-color:#f0fff0; }
   .error { background-color:#fff0f0; }
</style>

<script type="text/javascript" src="js/jquery-1.4.2.js"></script>
<script type="text/javascript" src="js/seahorse-1.2.js"></script>
<script type="text/javascript" src="js/seahorse.jquery-1.2.js"></script>
<script type="text/javascript">
   function init()
      {
      jQuery(".number").seaBehavior( "number", {"decimalCharacter": '.',
"groupingCharacter" : ','}, { "okClass": "ok", "errorClass": "error" } );
      jQuery(".number").seaVerify();

      jQuery(".integer").seaBehavior("integer", {"groupingCharacter" : ','}, { "okClass":
"ok", "errorClass": "error" } );
      jQuery(".integer").seaVerify();

      jQuery(".date").seaBehavior( "date", {"format" : 'dd/mm/yyyy', 'autofill': true},
{ "okClass": "ok", "errorClass": "error" } );
      jQuery(".date").seaVerify();

      jQuery(".time").seaBehavior( "time", {"format" : 'hh:mm:ss', 'autofill': true},
{ "okClass": "ok", "errorClass": "error" } );
      jQuery(".time").seaVerify();

      jQuery(".ipAddress")
         .seaBehavior( "ipAddress", {"version" : 4 }, { "okClass": "ok", "errorClass":
"error" } )
         .seaVerify();

      jQuery(".email")
         .seaBehavior( "email", {}, { "okClass": "ok", "errorClass": "error" } )
         .seaVerify();

      jQuery(".http")
         .seaBehavior( "http", {}, { "okClass": "ok", "errorClass": "error" } )
         .seaVerify();

      jQuery(".ftp")
         .seaBehavior( "ftp", {}, { "okClass": "ok", "errorClass": "error" } )
         .seaVerify();

      jQuery(".text")
         .seaBehavior( "text", {}, { "okClass": "ok", "errorClass": "error" } )
         .seaVerify();

      jQuery(".alphabetical")
         .seaBehavior( "alphabetical", {}, { "okClass": "ok", "errorClass": "error" } )
         .seaVerify();

      jQuery(".alphanumeric")
         .seaBehavior( "alphanumeric", {}, { "okClass": "ok", "errorClass": "error" } )
```

```
        .seaVerify();

     jQuery(".numeric")
        .seaBehavior( "numeric", {}, { "okClass": "ok", "errorClass": "error" } )
        .seaVerify();
     }
</script>

</head>
<body onload="javascript:init();">

   <table>
     <tr>
        <td>number()</td>
        <td><input type="text" class="number" value="1,000.00"/></td>
     </tr>
     <tr>
        <td>integer()</td>
        <td><input type="text" class="integer" value="1,000"/></td>
     </tr>
     <tr>
        <td>date()</td>
        <td><input type="text" class="date" value="31/12/2000"/></td>
     </tr>
     <tr>
        <td>time()</td>
        <td><input type="text" class="time" value="23:59:59"/></td>
     </tr>
     <tr>
        <td>ipAddress()</td>
        <td><input type="text" class="ipAddress" value="192.168.0.1"/></td>
     </tr>
     <tr>
        <td>email()</td>
        <td><input type="text" class="email" value="test@server.com"/></td>
     </tr>
     <tr>
        <td>http()</td>
        <td><input type="text" class="http" value="http://www.test.com"/></td>
     </tr>
     <tr>
        <td>ftp()</td>
        <td><input type="text" class="ftp" value="ftp://ftp.test.com"/></td>
     </tr>
     <tr>
        <td>text()</td>
        <td><input type="text" class="text" value="Text"/></td>
     </tr>
     <tr>
        <td>alphanumeric()</td>
        <td><input type="text" class="alphanumeric" value="Alphanumeric123"/></td>
     </tr>
     <tr>
        <td>alphabetical()</td>
        <td><input type="text" class="alphabetical" value="Alphabetical"/></td>
     </tr>
     <tr>
        <td>numeric()</td>
        <td><input type="text" class="numeric" value="1234"/></td>
     </tr>
   </table>

</body>
</html>
```

7.3 - Texto legal de la licencia

creative commons

LEGAL CODE

Atribución-NoComercial-SinDerivadas 2.5 (Argentina)

CREATIVE COMMONS CORPORATION NO ES UN ESTUDIO JURÍDICO NI PROVEE SERVICIOS LEGALES. LA DISTRIBUCIÓN DE ESTA LICENCIA NO CREA UNA RELACIÓN ABOGADO-CLIENTE. CREATIVE COMMONS PROVEE ESTA INFORMACIÓN "TAL Y COMO SE LA ENCUENTRA". CREATIVE COMMONS NO DA GARANTÍAS EN RELACIÓN A LA INFORMACIÓN PROPORCIONADA Y SE LIBERA DE RESPONSABILIDAD POR LOS DAÑOS QUE RESULTEN DE SU USO.

Licencia

LA OBRA (TAL COMO SE DEFINE MÁS ABAJO) SE PROVEE BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS ("CCPL" O "LICENCIA"). LA OBRA ESTÁ PROTEGIDA POR EL DERECHO DE AUTOR Y/O POR OTRAS LEYES APLICABLES. ESTÁ PROHIBIDO CUALQUIER USO DE LA OBRA DIFERENTE AL AUTORIZADO BAJO ESTA LICENCIA O POR EL DERECHO DE AUTOR.

MEDIANTE EL EJERCICIO DE CUALQUIERA DE LOS DERECHOS AQUÍ OTORGADOS SOBRE LA OBRA, USTED ACEPTA Y ACUERDA QUEDAR OBLIGADO POR LOS TÉRMINOS DE ESTA LICENCIA. EL LICENCIANTE LE CONCEDE LOS DERECHOS AQUÍ CONTENIDOS CONSIDERANDO QUE USTED ACEPTA SUS TÉRMINOS Y CONDICIONES.

1. Definiciones

a. **"Obra Colectiva"** significa una obra, tal como una edición periódica, antología o enciclopedia, en la cual la Obra, en su integridad y forma inalterada, se ensambla junto a otras contribuciones que en sí mismas también constituyen obras separadas e independientes, dentro de un conjunto colectivo. Una obra que integra una Obra Colectiva no será considerada una Obra Derivada (tal como se define más abajo) a los fines de esta Licencia.

b. **"Obra Derivada"** significa una obra basada sobre la Obra o sobre la Obra y otras obras preexistentes, tales como una traducción, arreglo musical, dramatización, ficcionalización, versión fílmica, grabación sonora, reproducción artística, resumen, condensación, o cualquier otra forma en la cual la Obra puede ser reformulada, transformada o adaptada. Una obra que constituye una Obra Colectiva no será considerada una Obra Derivada a los fines de esta Licencia. Para evitar dudas, cuando la Obra es una composición musical o grabación sonora, la sincronización de la Obra en una relación temporal con una imagen en movimiento ("synching") será considerada una Obra Derivada a los fines de esta Licencia.

c. **"Licenciante"** significa el individuo o entidad que ofrece la Obra bajo los términos de esta Licencia.

d. **"Autor Original"** significa el individuo o entidad que creó la Obra.

e. **"Obra"** significa la obra sujeta al derecho de autor que se ofrece bajo los términos de esta Licencia.

f. **"Usted"** significa un individuo o entidad ejerciendo los derechos bajo esta Licencia quien previamente no ha violado los términos de esta Licencia con respecto a la Obra, o quien, a pesar de una previa violación, ha recibido permiso expreso del Licenciante para ejercer los derechos bajo esta Licencia.

2. Derechos de Uso Libre y Legítimo. Nada en esta licencia tiene por objeto reducir, limitar, o restringir cualquiera de los derechos provenientes del uso libre, legítimo, derecho de cita u otras limitaciones que tienen los derechos exclusivos del titular bajo las leyes del derecho de autor u otras normas que resulten aplicables.

3. Concesión de la Licencia. Sujeto a los términos y condiciones de esta Licencia, el Licenciante por este medio le concede a Usted una licencia de alcance mundial, libre de regalías, no-exclusiva, perpetua (por la duración del derecho de autor aplicable) para ejercer los derechos sobre la Obra como se establece abajo:

a. para reproducir la Obra, para incorporar la Obra dentro de una o más Obras Colectivas, y para reproducir la Obra cuando es incorporada dentro de una Obra Colectiva;

b. para distribuir copias o fonogramas, exhibir públicamente, ejecutar públicamente y ejecutar públicamente por medio de una transmisión de audio digital las Obras, incluyendo las incorporadas en Obras Colectivas;

Los derechos precedentes pueden ejercerse en todos los medios y formatos ahora conocidos o a inventarse. Los derechos precedentes incluyen el derecho de hacer las modificaciones técnicamente necesarias para ejercer los derechos en otros medios y formatos, pero excluyen los derechos para hacer Obras Derivadas. Todos los derechos no concedidos expresamente por el Licenciante son reservados, incluyendo, aunque no sólo limitado a estos, los derechos presentados en las Secciones 4 (d) y 4(e).

4. Restricciones. La licencia concedida arriba en la Sección 3 está expresamente sujeta a, y limitada por, las siguientes restricciones:

a. Usted puede distribuir, exhibir públicamente, ejecutar públicamente o ejecutar públicamente la Obra en forma digital sólo bajo los términos de esta Licencia, y Usted debe incluir una copia de esta Licencia o de su Identificador Uniforme de Recursos (Uniform Resource Identifier) con cada copia o fonograma de la Obra que Usted distribuya, exhiba públicamente, ejecute públicamente, o ejecute públicamente en forma digital. Usted no podrá ofrecer o imponer condición alguna sobre la Obra que altere o restrinja los términos de esta Licencia o el ejercicio de los derechos aquí concedidos a los destinatarios. Usted no puede sublicenciar la Obra. Usted debe mantener intactas todas las notas que se refieren a esta Licencia y a la limitación de garantías. Usted no puede distribuir, exhibir públicamente, ejecutar públicamente o ejecutar públicamente en forma digital la Obra con medida tecnológica alguna que controle el acceso o uso de la Obra de una forma inconsistente con los términos de este Acuerdo de Licencia. Lo antedicho se aplica a la Obra cuando es incorporada en una Obra Colectiva, pero esto no requiere que la Obra Colectiva, con excepción de la Obra en sí misma, quede sujeta a los términos de esta Licencia. Si Usted crea una Obra Colectiva, bajo requerimiento de cualquier Licenciante Usted debe, en la medida de lo posible, quitar de la Obra Colectiva cualquier crédito requerido en la cláusula 4(c), conforme lo solicitado.

b. Usted no puede ejercer ninguno de los derechos a Usted concedidos precedentemente en la Sección 3 de alguna forma que esté primariamente orientada, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas. El intercambio de la Obra por otros materiales protegidos por el derecho de autor mediante el intercambio de archivos digitales (file-sharing) u otras formas, no será considerado con la intención de, o dirigido a, la obtención de ventajas comerciales o compensaciones monetarias privadas, siempre y cuando no haya pago de ninguna compensación monetaria en relación con el intercambio de obras protegidas por el derecho de autor.

c. Si Usted distribuye, exhibe públicamente, ejecuta públicamente o ejecuta públicamente en forma digital la Obra, Usted debe mantener intacta toda la información de derecho de autor de la Obra y proporcionar, de forma razonable según el medio o manera que Usted esté utilizando: (i) el nombre del Autor Original si está provisto (o seudónimo, si fuere aplicable), y/o (ii) el nombre de la parte o las partes que el Autor Original y/o el Licenciante hubieren designado para la atribución (v.g., un instituto patrocinador, editorial, publicación) en la información de los derechos de autor del Licenciante, términos de servicios o de otras formas razonables; el título de la Obra si está provisto; en la medida de lo razonablemente factible y, si está provisto, el Identificador Uniforme de Recursos (Uniform Resource Identifier) que el Licenciante especifica para ser asociado con la Obra, salvo que tal URI no se refiera a la nota sobre los derechos de autor o a la información sobre el licenciamiento de la Obra. Tal crédito puede ser implementado de cualquier forma razonable; en el caso, sin embargo, de Obras Colectivas, tal crédito aparecerá, como mínimo, donde aparece el crédito de cualquier otro autor comparable y de una manera, al menos, tan destacada como el crédito de otro autor comparable.

d. Para evitar dudas, cuando una Obra es una composición musical:

    i. Derechos Económicos y Ejecución bajo estas Licencias. El Licenciante se reserva el derecho exclusivo de colectar, ya sea individualmente o vía una sociedad de gestión colectiva de derechos (v.g., SADAIC, ARGENTORES), los valores (royalties) por la

ejecución pública o por la ejecución pública en forma digital (v.g., webcast) de la Obra si esta ejecución está principalmente orientada a, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas.

ii. Derechos Económicos sobre Fonogramas. El Licenciante se reserva el derecho exclusivo de colectar, ya sea individualmente, vía una sociedad de gestión colectiva de derechos (v.g., SADAIC, AADI-CAPIF), o vía una agencia de derechos musicales o algún agente designado, los valores (royalties) por cualquier fonograma que Usted cree de la Obra ("versión", "cover") y a distribuirlos, conforme a las disposiciones aplicables del derecho de autor, si su distribución de la versión (cover) está principalmente orientada a, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas.

e. Derechos Económicos y Ejecución Digital (Webcasting). Para evitar dudas, cuando la Obra es una grabación sonora, el Licenciante se reserva el derecho exclusivo de colectar, ya sea individualmente o vía una sociedad de gestión colectiva de derechos (v.g., SADAIC, ARGENTORES), los valores (royalties) por la ejecución pública digital de la Obra (v.g., webcast), conforme a las disposiciones aplicables de derecho de autor, si esta ejecución está principalmente orientada a, o dirigida hacia, la obtención de ventajas comerciales o compensaciones monetarias privadas.

5. Representaciones, Garantías y Limitación de Responsabilidad. A MENOS QUE SEA ACORDADO DE OTRA FORMA Y POR ESCRITO ENTRE LAS PARTES, EL LICENCIANTE OFRECE LA OBRA "TAL Y COMO SE LA ENCUENTRA" Y NO OTORGA EN RELACIÓN A LA OBRA NINGÚN TIPO DE REPRESENTACIONES O GARANTÍAS, SEAN EXPRESAS, IMPLÍCITAS O LEGALES; SE EXCLUYEN ENTRE OTRAS, SIN LIMITACIÓN, LAS GARANTÍAS SOBRE LAS CONDICIONES, CUALIDADES, TITULARIDAD O EXACTITUD DE LA OBRA, ASÍ COMO TAMBIÉN, LAS GARANTÍAS SOBRE LA AUSENCIA DE ERRORES U OTROS DEFECTOS, SEAN ESTOS MANIFIESTOS O LATENTES, PUEDAN O NO DESCUBRIRSE. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE GARANTÍAS IMPLÍCITAS, POR TANTO ESTAS EXCLUSIONES PUEDEN NO APLICÁRSELE A USTED.

6. Limitación de Responsabilidad. EXCEPTO EN LA EXTENSIÓN REQUERIDA POR LA LEY APLICABLE, EL LICENCIANTE EN NINGÚN CASO SERÁ REPONSABLE FRENTE A USTED, CUALQUIERA SEA LA TEORÍA LEGAL, POR CUALQUIER DAÑO ESPECIAL, INCIDENTAL, CONSECUENTE, PUNITIVO O EJEMPLAR, PROVENIENTE DE ESTA LICENCIA O DEL USO DE LA OBRA, AUN CUANDO EL LICENCIANTE HAYA SIDO INFORMADO SOBRE LA POSIBILIDAD DE TALES DAÑOS.

7. Finalización

a. Esta Licencia y los derechos aquí concedidos finalizarán automáticamente en caso que Usted viole los términos de la misma. Los individuos o entidades que hayan recibido de Usted Obras Colectivas conforme a esta Licencia, sin embargo, no verán finalizadas sus licencias siempre y cuando permanezcan en un cumplimiento íntegro de esas licencias. Las secciones 1, 2, 5, 6, 7, y 8 subsistirán a cualquier finalización de esta Licencia.

b. Sujeta a los términos y condiciones precedentes, la Licencia concedida aquí es perpetua (por la duración del derecho de autor aplicable a la Obra). A pesar de lo antedicho, el Licenciante se reserva el derecho de difundir la Obra bajo diferentes términos de Licencia o de detener la distribución de la Obra en cualquier momento; sin embargo, ninguna de tales elecciones servirá para revocar esta Licencia (o cualquier otra licencia que haya sido, o sea requerida, para ser concedida bajo los términos de esta Licencia), y esta Licencia continuará con plenos efectos y validez a menos que termine como se indicó precedentemente.

8. Misceláneo

a. Cada vez que Usted distribuye o ejecuta públicamente en forma digital la Obra o una Obra Colectiva, el Licenciante ofrece a los destinatarios una licencia para la Obra en los mismos términos y condiciones que la licencia concedida a Usted bajo esta Licencia.

b. Si alguna disposición de esta Licencia es inválida o no exigible bajo la ley aplicable, esto no afectará la validez o exigibilidad de los restantes términos de esta Licencia, y sin necesidad de más acción de las partes de este acuerdo, tal disposición será reformada en la mínima extensión necesaria para volverla válida y exigible.

c. Ningún término o disposición de esta Licencia se considerará renunciado y ninguna violación se

considerará consentida a no ser que tal renuncia o consentimiento sea por escrito y firmada por las partes que serán afectadas por tal renuncia o consentimiento.

d. Esta Licencia constituye el acuerdo integral entre las partes con respecto a la Obra licenciada aquí. No hay otros entendimientos, acuerdos o representaciones con respecto a la Obra que no estén especificados aquí. El Licenciante no será obligado por ninguna disposición adicional que pueda aparecer en cualquier comunicación proveniente de Usted. Esta Licencia no puede ser modificada sin el mutuo acuerdo por escrito entre el Licenciante y Usted.

Creative Commons no es una parte para esta Licencia y no da garantía alguna en relación con la Obra. Creative Commons no será responsable bajo ninguna teoría legal, frente a Usted o cualquier parte, de ningún daño en absoluto, excluyendo también, sin limitación, cualquier daño general, especial, incidental o consecuente, originado en relación con esta licencia. No obstante las dos (2) oraciones precedentes, si en virtud de este documento Creative Commons se ha identificado expresamente como el Licenciante, tendrá todos los derechos y obligaciones del Licenciante.

Excepto con el propósito limitado de indicar al público que la Obra está licenciada bajo la CCPL, ninguna parte usará la marca "Creative Commons", o ninguna marca o logotipo relacionado a Creative Commons, sin el previo consentimiento por escrito de Creative Commons. Cualquier uso permitido será de conformidad con la guía de uso de la marca Creative Commons entonces vigente, según sea publicada periódicamente en su sitio web o, de otro modo, esté disponible mediante solicitud.

Creative Commons puede ser contactada en http://creativecommons.org/.