1 Introdución

jQuery UI (**jQuery User Interface**) é un complemento que permite usar métodos de jQuery para xerar interfaces de usuario avanzadas na parte do cliente nas páxinas web dinámicas. Podemos atopar máis información no portal web oficial de jQuery (http://www.jquery.org) e de jQquery UI (http://www.jqueryui.com).

O primeiro que podemos facer para comprobar a versatilidade deste complemento é visitar a súa páxina de demos (http://www.jqueryui.com/demos/). Nela podemos observar as funcionalidades (**interactions**, **effects** ou **utilities**) listas para usar, así como os compoñentes (**widgets**) preparados para fornecer interfaces de usuario. Todas elas presentan unha serie de opcións (veñen sendo como unha especie de atributos), métodos e eventos relacionados que nos permiten axustar a funcionalidade aos parámetros requiridos.

2 Escoller os compoñentes na páxina de descarga

Para usar jQuery UI necesitamos descargar a libraría, pero debido a súa amplitude é probable que só precisemos parte dela. A páxina de descargas (http://www.jqueryui.com/download) proporciona un sistema para seleccionar só os compoñentes que necesitamos utilizar no noso portal, de forma que podemos optimizar a libraría para que conteña unicamente os elementos precisos, o que permitirá unha descarga máis rápida por parte dos clientes das nosas páxinas.

As partes que se poden seleccionar son:

- **Versión**: escollémola dependendo da versión de jQuery coa que imos traballar.
- **Compoñentes**: podemos marcar ou desmarcar cada un dos compoñentes; algúns deles teñen dependencias con outros, polo que, automaticamente, se seleccionan estes últimos.
- **Aspecto**: podemos seleccionar o aspecto que van ter as interfaces de usuario ou usar o xerador de aspectos (http://www.jqueryui.com/themeroller/) para crealo nós.

Unha vez que descargamos o arquivo, ao descomprimilo teremos o contido nos seguintes cartafoles:

- **css**: folla de estilos e imaxes para xerar o aspecto escollido.
- **development-bundle**: exemplos e páxinas de documentación; material válido para aprender e profundar no tema, pero non necesario para o funcionamento do complemento.
- **js**: scripts necesarios para que funcionen os compoñentes.

3 Interaccións

As interaccións engaden comportamentos básicos baseados no movemento e posición do rato a outros elementos. Podemos crear listas ordenables, cambiar as dimensións dun elemento ou arrastralo e soltalo e outras moitas con poucas e sinxelas liñas de código. Tamén nos permitirán crear bloques de código ou funcións que poderemos usar noutras aplicacións ou compoñentes máis complexos.

3.1 Arrastrar

Para poder arrastrar un elemento co rato, jQuery UI fornece unha clase que aplicamos aos elementos que se poden arrastrar: **draggable**. O seu uso é moi sinxelo e o único que temos que facer é usar o método do mesmo nome.

```
<!DOCTYPE html>
<html>
<head>
   <title>Exemplo de elemento que se pode arrastrar</title>
   <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
   <link type='text/css' href='css/vader/jquery-ui-1.10.0.custom.css' rel='Stylesheet' />
   <style type="text/css">
      body {
         background-color: pink;
      }
      .arrastrable {
         background-color: greenYellow;
         border: 1px solid black;
         width: 200px;
         height: 50px;
         text-align: center;
         line-height: 50px;
      }
   </style>
   <script type='text/javascript' src='js/jquery-1.9.0.js'></script>
   <script type='text/javascript' src='js/jquery-ui-1.10.0.custom.min.js'></script>
```

```
    <script type='text/javascript'>
        $(document).ready(function() {
            $(".arrastrable").draggable();
        });
    </script>
</head>
<body>
    <div class="arrastrable"><span>Este texto pódese arrastrar</span></div>
    <div class="noArrastrable"><span>Este outro non se pode arrastrar</span></div>
</body>
</html>
```

Opcións

Entre o resto das funcionalidades que podemos usar con esta clase atopamos as barras de desprazamento automáticas que se lle aplican á xanela que contén o elemento que imos arrastrar. Aplícanse cando acercamos o elemento arrastrado aos bordos da xanela. As opcións que modelan este efecto son as seguintes:

Barras de desprazamento automáticas		
scroll	Activa as barras de desprazamento; se o seu valor é *true*, o elemento receptor posuirá un desprazamento automático mentres esteamos a arrastrar un elemento.	**Valor por defecto** `true`
scrollSensitivity	A distancia en píxeles dende o bordo da xanela contedora despois de que a xanela adquira a función de desprazamento. A distancia é relativa ao punteiro do rato non ao elemento que se está a arrastrar. Non ten efecto se non se activa a opción *scroll*.	**Valor por defecto** `20`
scrollSpeed	A velocidade á que a xanela fai o desprazamento unha vez que o punteiro do rato achégase á distancia que lle indica a opción *scrollSensitivity*. Non ten efecto se non se activa a opción *scroll*.	**Valor por defecto** `20`

```
$(".arrastrable").draggable({ scroll: true, scrollSensitivity: 100, scrollSpeed: 100 });
```

Outra posibilidade que podemos contemplar é a definición de fronteiras en canto ao recinto no que se poden desprazar cada un dos elementos que arrastramos coa intención de constrinxir o seu movemento:

Constrición do movemento		
axis	Impide a funcionalidade nun dos dous eixes. Os posibles valores son *"x"* ou *"y"*.	**Valor por defecto** `false`
containment	Impide que o elemento que posúe a funcionalidade saia dos límites do elementos ou zona especificados. Os tipos soportados son: selectores, elementos, cadeas de caracteres (*"parent"*, *"document"* ou *"window"*) e vectores definindo rectángulos da forma *[x1, y1, x2, y2]*.	**Valor por defecto** `False`

```
$(".arrastrable").draggable({ axis: "y", containment: "parent" });
```

Podemos tamén definir a posición relativa do cursor en relación ao elemento que imos arrastrar mentres se produce o arrastre. Por defecto, o cursor aparece centrado no obxecto que arrastramos; podemos cambiar a posición relativa do cursor e pór ao noso gusto a aparencia do cursor mediante a folla de estilos.

Posición do cursor		
cursor	O cursor (debe estar declarado na folla de estilos) que amosa durante a operación de arrastre.	**Valor por defecto** `"auto"`
cursorAt	Inicia o desprazamento relativo do rato, de forma que as coordenadas pódense dar como unha combinación dunha ou dúas destas claves: *top*, *left*, *right*, *bottom*.	**Valor por defecto** `False`
iframeFix	Prevén marcos á hora de capturar o movemento do rato durante o arrastre. Úsase en combinación coa opción *cursorAt* ou nalgún caso onde o cursor do rato non debe estar sobor do manexador. Os tipos que soporta son: *Boolean* (cando o establecemos a *true* todos os iframes da páxina se recobren cunha capa transparente) e *Selector* (se coincide algún iframe o selector se recobre cunha capa transparente).	**Valor por defecto** `False`

```
$(".arrastrable").draggable({ cursor: "crosshair", cursorAt: { top: -5, left: -5 } });
```

Podemos retardar tanto no tempo como na distancia o movemento do elemento a arrastrar para previr un arrastre incorrecto. Para ilo daremos un valor en milisegundos ou en número de píxeles dependendo da opción empregada.

Retardo do inicio		
delay	Tempo en milisegundos dende que facemos clic nun elemento que posúe esta funcionalidade ata que a toma en conta; está pensado para evitar arrastres cando facemos clic en elementos que non queriamos mover.	**Valor por defecto** `0`
distance	Distancia en píxeles que debemos arrastrar o rato antes de que se active a funcionalidade; está pensado para evitar arrastres cando facemos clic en elementos que non queriamos mover.	**Valor por defecto** `1`

```
$(".arrastrable").draggable({ delay: 500, distance: 10 });
```

Podemos permitir o arrastre só cando o cursor está enriba dunha parte específica do elemento que queremos arrastrar. Tamén é válido para especificar o selector que usaremos para mover un elemento ou grupo de elementos. Outra funcionalidade é previr o arrastre dun elemento cando queremos que se mova en conxunto con outros; podémola conseguir cancelando esta funcionalidade para un elemento determinado.

Manexadores		
cancel	Prevén o arrastre dende os elementos que se especifican.	**Valor por defecto** `"input, textarea, button, select, option"`
handle	Restrinxe a posición dende a que se pode arrastrar un elemento, empregando para ilo descendentes do elemento que podiamos arrastrar inicialmente.	**Valor por defecto** `False`

```
$(".arrastrable").draggable({ cancel: ".title", handle: "h2" });
```

Podemos devolver o elemento arrastrado á súa posición orixinal cando o deixamos de arrastrar:

Volver á posición inicial		
revert	Volve o elemento á súa posición inicial cando o paramos de arrastrar. Soporta varios tipos: *Boolean* (se o pomos a *true* sempre volve ao lugar de partida) e *String* (*valid* volve á posición de partida se non o soltamos nun elemento receptor e *invalid* ao revés).	**Valor por defecto** `False`
revertDuration	Os milisegundos que dura o efecto de volver á posición inicial.	**Valor por defecto** `500`

```
$(".arrastrable").draggable({ revert: true, revertDuration: 200  });
```

Podemos colocar o elemento que arrastramos no interior ou exterior dun elemento do DOM e podemos indicar a distancia en píxeles que debe gardar en relación ao elemento que o recolle. Tamén o podemos introducir nunha grella e establecer de antemán as dimensións das celas que a compoñen.

Colocar o elemento		
grid	Só permite colocar o elemento que se pode arrastrar nunha grella imaxinaria de axuda, cada *x* e *y* píxeles; o vector debe estar no formato *[x, y]*.	**Valor por defecto** `false`
snap	Indica se o elemento pode colocarse sobre outros elementos. Soporta varios tipos de datos: *Boolean* (cando está a *true* o elemento pódese colocar noutros elementos que se poden arrastrar) e *Selector* (un selector que especifica a que elementos se pode anexar).	**Valor por defecto** `false`
snapMode	Determina a que borde se anexa. Se o valor é *false* ignórase. Os posibles valores son: *inner*, *outer* e *both*.	**Valor por defecto** `"both"`
snapTolerance	Distancia en píxeles dende o borde do elemento anexado ata o borde do elemento ao que o anexamos. Ignórase se o valor da opción é *false*.	**Valor por defecto** `20`

```
$(".arrastrable").draggable({ grid: [ 50, 20 ], snap: true, snapMode: "inner", snapTolerance: 30 });
```

Podemos fornecer de retroalimentación aos usuarios cando arrastran un obxecto mediante manexadores. Podemos mover ou duplicar o obxecto orixinal usando o cursor ou usar unha función que devolva un elemento do DOM que se amosará preto do cursor mentres dure o arrastre. Tamén podemos controlar a transparencia do elemento manexado.

Para que o usuario teña claro cal é o elemento que arrastramos visualízase diante do resto e debemos asegurarnos que cando se coloque apareza por riba dos demais se é o que queremos facer mediante as opcións existentes para ilo.

Retroalimentación visual		
helper	Permite usar un elemento manexador como elemento para arrastrar. Soporta dous tipos: *String* (se o establecemos a *clone* entón arrastramos un clon do elemento orixinal) ou *Function* (unha función que devolverá un elemento do DOM para usar mentres se arrastra).	**Valor por defecto** `"original"`
opacity	Opacidade do elemento mentres está a ser arrastrado.	**Valor por defecto** `false`
stack	Controla a propiedade *z-index* do conxunto de elementos, de forma que sempre coloca por riba o elemento que está a ser arrastrado.	**Valor por defecto** `false`
zIndex	Propiedade *z-index* do elemento mentres se arrastra.	**Valor por defecto** `False`

```
$(".arrastrable").draggable({ helper: "clone", opacity: 0.35, stack: ".products", zIndex: 100 });
```

Podemos combinar con outras interaccións o efecto buscado.

Arrastrar e ordenar		
connectToSortable	Permite que o elemento arrastrado poida ser colocado no medio doutros elementos especificados como ordenables; desta maneira formará parte da lista de elementos ordenables (para isto o valor de *helper* debe ser *clone*). Temos que incluír o plugin correspondente a *Sortable* para que funcione a interacción.	**Valor por defecto** `False`
`$(".arrastrable").draggable({ connectToSortable: "#elementoAOrdenar" });`		

Arrastrar e soltar		
scope	Agrupa conxuntos de elementos que se poden arrastrar cando se usa en conxunto coa opción *accept* dos elementos receptores, de forma que só acepte aqueles elementos co mesmo valor que el posúe.	**Valor por defecto** `"default"`
`$(".arrastrable").draggable({ scope: "tarefa" });`		

Miscelánea		
addClasses	Se o establecemos a *false* prevemos se poida engadir unha clase que o transforme en elemento a arrastrar. Úsase para optimizar as chamadas a este tipo de elementos cando manexamos centos deles.	**Valor por defecto** `true`
appendTo	Indica que elemento do manexador se engade mentres dura o arrastre. Soporta varios tipos: *jQuery* (un obxecto jQuery contendo o elemento a engadir), *Element* (o elemento a engadir), *Selector* (un selector que especifique que elemento vai engadir), *String* (o texto *parent* busca o pai do elemento).	**Valor por defecto** `"parent"`
disabled	Desactiva a funcionalidade cando pomos esta opción a *true*.	**Valor por defecto** `false`
refreshPositions	Cando se establece a *true*, todas as posicións ás que podemos arrastrar o elemento se calculan con cada movemento do rato. Hai que ter coidado se usamos esta opción con páxinas construídas dinamicamente pois pode facer que a usabilidade se vexa decrementada.	**Valor por defecto** `False`
`$(".arrastrable").draggable({ disabled: true });`		

Métodos

Entre os métodos que podemos usar atopamos:

draggable		
destroy	Elimina a funcionalidade volvendo o elemento ao seu estado inicial.	`$(".arrastrable").draggable("destroy");`
disable	Desactiva a funcionalidade.	`$(".arrastrable").draggable("disable");`
enable	Activa a funcionalidade.	`$(".arrastrable").draggable("enable");`
option	**`option(nomeDaOpcion)`** Devolve o valor actual asociado á opción especificada.	`var nome = $(".arrastrable").draggable("option","disabled");`
	`option()` Devolve un obxecto contendo os pares chave/valor asociados ás opcións actuais.	`var opcions = $(".arrastrable").draggable("option");`
	`option(nomeDaOpcion,valor)` Asigna o valor á opción especificada.	`$(".arrastrable").draggable("option","disabled",true);`
	`option(opcions)` Asigna unha ou máis opcións mediante pares opción/valor.	`$(".arrastrable").draggable("option",{disabled:true});`
widget	Devolve un obxecto que contén o elemento.	`var contedor = $(".arrastrable").draggable("widget");`

Eventos

Entre os eventos aos que responde atopamos:

draggable		
create	Ao crear o elemento.	`$(".arrastrable").on("dragcreate",function(evento,ui) {} );`
drag	Mentres o rato arrastra o elemento.	`$(".arrastrable").on("drag",function(evento,ui) {} );`
start	Ao comezar o arrastre.	`$(".arrastrable").on("dragstart",function(evento,ui) {} );`
stop	Ao finalizar o arrastre.	`$(".arrastrable").on("dragstop",function(evento,ui) {} );`

3.2 Soltar

Na operación de soltar participan dous elementos: o elemento que se arrastra e o elemento onde se deixa caer o que se arrastra. Por ilo, jQueryUI fornece dúas clases distintas:

- **draggable**: que se aplica sobre os elementos que se poden arrastrar.
- **droppable**: que se aplica sobre os elementos que teñen que saber que se deixa caer algo sobre eles.

A detección de que se soltou nun lugar determinado vai ser tan sinxela como a característica do arrastre.

```
$(".receptor").droppable();
```

Con este código conseguimos que os elementos que pertencen á clase **receptor** detecten cando algún elemento que se pode arrastrar se solta enriba deles. De todos modos, algo máis teremos que facer cando se deixe caer o elemento que se arrastrou sobre o elemento que o pode recibir, polo que teremos que capturar este momento mediante un evento e a partires da captura facer algunha outra acción.

```
$(".receptor").droppable({
   drop: function(evento, ui) { $(this).html("Bruto, deixáchelo caer sobre min!!!"); }
});
```

Neste caso, o evento **drop** provocaría que, cando soltamos un elemento que se pode arrastrar enriba do elemento receptor, cambie o texto que hai no elemento receptor.

Se queremos, podemos combinar ambos no código seguinte.

```
<!DOCTYPE html>
<html>
<head>
   <title>Exemplo de elemento que se pode arrastrar</title>
   <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
   <link type='text/css' href='css/vader/jquery-ui-1.10.0.custom.css' rel='Stylesheet' />
   <style type="text/css">
      body { background-color: pink; }
      .arrastrable {
         background-color: greenYellow;
         border: 1px solid black;
         width: 200px;
         height: 50px;
         text-align: center;
         line-height: 50px;
      }
      .receptor {
         background-color: yellow;
         border: 1px solid white;
         width: 300px;
         height: 300px;
         text-align: center;
      }
   </style>
   <script type='text/javascript' src='js/jquery-1.9.0.js'></script>
   <script type='text/javascript' src='js/jquery-ui-1.10.0.custom.min.js'></script>
   <script type='text/javascript'>
      $(document).ready(function() {
         $(".arrastrable").draggable();
         $(".receptor").droppable({
            drop: function(evento, ui) { $(this).html("Bruto, deixáchelo caer sobre min!!!"); }
         });
      });
   </script>
</head>
<body>
   <div class="arrastrable"><span>Este texto pódese arrastrar</span></div>
   <div class="noArrastrable receptor"><span>Este outro non se pode arrastrar</span></div>
</body>
</html>
```

Opcións

Tamén podemos configurar as opcións ou os eventos relacionados con estes elementos receptores no momento de crealos ou, despois de facelo, usando o método **option** noutro momento. Por exemplo, podemos configurar no momento que creamos o receptor que aceptamos un elemento cando, polo menos, o 50% do seu tamaño estea sobor do receptor:

```
$("#receptorGrande").droppable({
   tolerance: "intersect"
});
```

Entre as opcións que podemos usar atopamos:

Opcións

accept	Controla que os elementos arrastrados sexan aceptados; os tipos soportados son selectores ou funcións.	**Valor por defecto** `"*"`
activeClass	Engádese a clase ao receptor mentres está sendo arrastrado un elemento que pode aceptar.	**Valor por defecto** `false`
addClasses	Establécese a *false* para prever que un elemento mude de clase mentres está sendo engadido; precisa dunha optimización axeitada cando pretendemos iniciar centos de elementos con clase *.droppable()*.	**Valor por defecto** `true`
disabled	Desactiva a funcionalidade cando pomos esta opción a *true*.	**Valor por defecto** **`false`**
greedy	Por defecto, cando un elemento se solta en receptores aniñados, cada receptor recibirá o elemento. Nembargante, se mudamos o valor desta opción a *true*, os antecesores, aínda que sexan receptores, non recibirán o elemento arrastrado.	**Valor por defecto** **`false`**
hoverClass	A clase engádese ao receptor mentres un elemento que pode aceptar está enriba do receptor.	**Valor por defecto** **`false`**
scope	Agrupa conxuntos de elementos receptores cando se usa en conxunto coa opción *accept* dos elementos que se poden arrastrar, de forma que só acepte aqueles elementos co mesmo valor que el posúe.	**Valor por defecto** `"default"`
tolerance	Especifica como sabemos se o elemento arrastrado está xa sobre o elemento receptor. Os posibles valores son: *fit* (está totalmente sobre o receptor), *intersect* (polo menos o 50%), *pointer* (o punteiro do rato está sobre o receptor) ou *touch* (ten algunha parte sobre o receptor).	**Valor por defecto** `"intersect"`

```
$(".receptor").droppable({
    accept: ".especial",
    activeClass: ".unhaClase",
    addClasses: false,
    disabled: true,
    greedy: true,
    hoverClass: "drop-hover"
    scope: "tarefa",
    tolerance: "pointer"  });
```

Métodos

Entre os métodos que podemos usar atopamos:

droppable		
destroy	Elimina a funcionalidade volvendo o elemento ao seu estado inicial.	`$(".receptor").droppable("destroy");`
disable	Desactiva a funcionalidade.	`$(".receptor").droppable("disable");`
enable	Activa a funcionalidade.	`$(".selector").droppable("enable");`
option	**`option(nomeDaOpcion)`** Devolve o valor actual asociado á opción especificada.	`var nome = $(".receptor").droppable("option","disabled");`
	`option()` Devolve un obxecto contendo os pares chave/valor asociados ás opcións actuais.	`var opcions = $(".receptor").droppable("option");`
	`option(nomeDaOpcion,valor)` Asigna o valor á opción especificada.	`$(".receptor").droppable("option","disabled",true);`
	`option(opcions)` Asigna unha ou máis opcións mediante pares opción/valor.	`$(".receptor").droppable("option",{disabled:true});`
widget	Devolve un obxecto que contén o elemento.	`var contedor = $(".receptor").droppable("widget");`

Eventos

Entre os eventos aos que responde atopamos:

droppable		
activate	Actívase cando un elemento que se pode aceptar empeza a ser arrastrado.	`$(".receptor").on("dropactivate",function( evento, ui ) {} );`
create	Ao crear o elemento.	`$(".receptor").on("dropcreate",function(evento,ui) {} );`
deactivate	Actívase cando un elemento que se pode aceptar deixa de ser arrastrado.	`$(".receptor").on("dropdeactivate",function( evento, ui ) {} );`
drop	Actívase ao recibir un elemento.	`$(".receptor").on("drop",function(evento,ui) {} );`
out	Actívase cando deixa de estar enriba un elemento que se pode deixar caer.	`$(".receptor").on("dropout",function(evento,ui) {} );`
over	Actívase cando pasa por riba un elemento que se pode deixar caer.	`$(".receptor").on("dropover",function(evento,ui) {} );`

Estas funcións que se executan dependendo dos eventos (tanto para os elementos que se poden arrastrar como para os que son receptores) reciben dous parámetros: o primeiro é o obxecto evento do navegador e o segundo é outro obxecto que estamos recibindo co nome **ui** e que ten os seguintes atributos:

- **ui.draggable**: obxecto jQuery equivalente ao elemento arrastrado que está provocando o evento.
- **ui.helper**: asistente para acceder ás propiedades do elemento arrastrado.
- **ui.position**: posición do elemento arrastrado, relativa ao receptor.
- **ui.offset**: posición absoluta do elemento arrastrado.

O evento clave cando queremos xestionar a utilidade de arrastrar e soltar é **drop**. Poderiamos, por exemplo, usando estes atributos modificar tamén propiedades css do elemento arrastrado como pode ser a cor de fondo:

```
$(".receptor").droppable({
   drop: function(evento, ui) { ui.draggable.css("background-color", "red"); }
});
```

Tamén podemos acceder ás propiedades do obxecto cando o arrastramos mediante a propiedade **helper**:

```
$(".arrastrable").draggable({
   drag: function(evento, ui) {
      ui.helper.html("Socorro, estou a ser arrastrado!!!");
      ui.helper.css("width", 500);
   }
});
$(".receptor").droppable({
   drop: function(evento, ui) {
      $(this).html("Bruto, deixáchelo caer sobre min!!!");
      ui.draggable.css("background-color", "red");
      ui.helper.html("Cheguei, eran horas!!!");
   }
});
```

4 Compoñentes

4.1 O calendario

Supoñamos que necesitamos un campo de texto para escribir unha data, polo que precisamos que ao premer nel amose un calendario para podela seleccionar. Este comportamento faino o compoñente *Datepicker*.

```
<!DOCTYPE html>
<html>
<head>
   <title>Exemplo de calendario con jQuery UI</title>
   <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
   <link type='text/css' href='css/vader/jquery-ui-1.10.0.custom.css' rel='Stylesheet' />
   <script type='text/javascript' src='js/jquery-1.9.0.js'></script>
   <script type='text/javascript' src='js/jquery-ui-1.10.0.custom.min.js'></script>
   <script type='text/javascript'>
      $(document).ready(function() {
         $("#data").datepicker();
      });
   </script>
</head>
<body>
   <form>
      <input type='text' id='data' />
   </form>
</body>
</html>
```

O problema que presenta este código é o formato no que aparece a data. Como é lóxico é algo que xa está pensado e podemos modificar unha serie de atributos que nos van permitir amosar o resultado como queremos. A maiores temos tamén métodos que nos permiten traballar estes atributos dunha maneira ordenada.

Así, se queremos que o calendario se presente no formato no que estamos acostumados a velos normalmente, é dicir, empezando a semana no luns en vez de no domingo e amosando a data resultante tras escoller ordenada como día, mes e ano teremos que introducir as seguintes modificacións nos atributos:

```
         $("#data").datepicker({ firstDay: 1 ,  dateFormat: "dd-mm-yy" });
```