CFGS DAW

Mòdul 6: Desenvolupament web en entorn client

INS Joan d'Àustria

Jquery UI

Una extensión de Jquery es Jquery UI. Es una librería que incorpora widgets y otros elementos para facilitar la interacción con el usuario.

Podemos descargar la última versión en http://jqueryui.com

En el proceso de descarga podemos guardar la librería completa o sólo los módulos que nos interesen

Jquery UI

Desde la propia página se nos ofrece un themeroller para personalizar los estilos, de la misma manera que teníamos en la librería jquery Mobile:

http://jqueryui.com/themeroller/

La compatibilidad ofrecida, al igual que con jquery es bastante amplia:

- IE 6+
- Firefox 2+
- Opera 9+
- Safari 3+
- Chrome 1+

Jquery UI

Entre lo que nos ofrece jquery UI destaca:

1. Widgets
2. Efectos
3. Interacciones

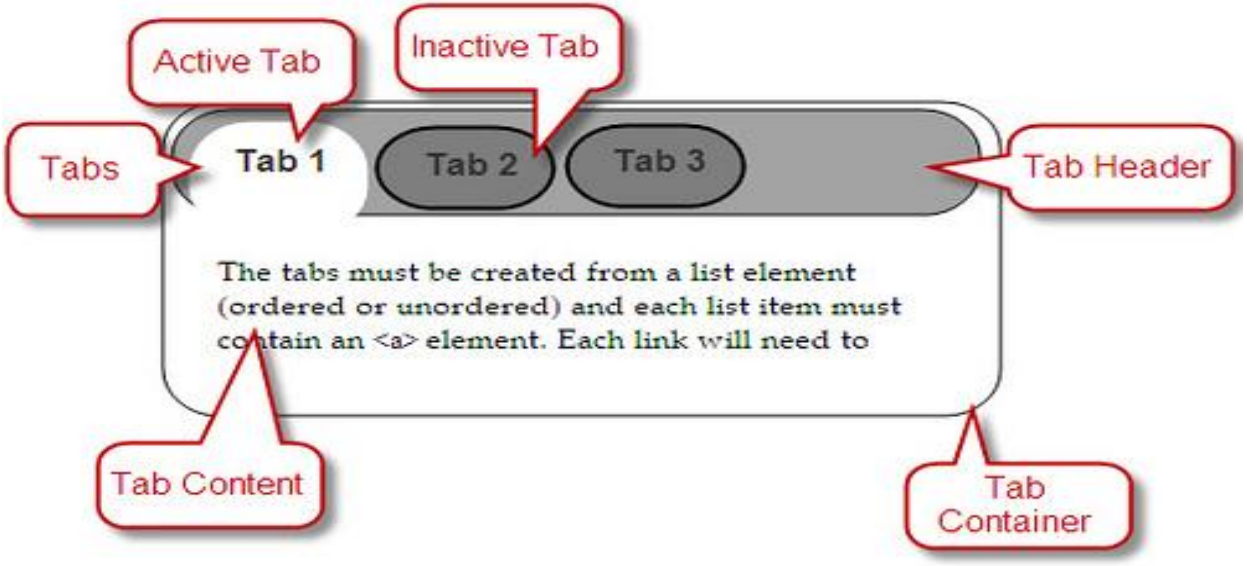
Jquery UI

Widgets >

1. Widgets:

1.1 Tab Widget

Los elementos básicos de este widget son:



Ejemplo: en http://www.fnac.es al entrar en un producto nos recomiendan accesorios

Jquery UI

Widgets > | Tab Widget

Para crearlo necesitamos una lista de elementos y cada elemento de lista debe contener un enlace. También necesitaremos enlazar los archivos jquery UI(se incluyen los generales)

```
<link href="css/smoothness/jquery-ui-1.10.1.custom.css" rel="stylesheet">
<script src="js/jquery-1.9.1.js"></script>
<script src="js/jquery-ui-1.10.1.custom.js"></script>
```

```
<div id="myTabs">
        <ul>
        <li><a href="#a">Tab 1</a></li>
        <li><a href="#b">Tab 2</a></li>
        </ul>
        <div id="a">Contenido del Tab 1.</div>
        <div id="b">Contenido del Tab 2</div>
</div>
```

Para crear la estructura de tab y cargar los estilos CSS simplemente hay que añadir:

```
<script type="text/javascript">
    $(function(){
        $("#myTabs").tabs();
    });
</script>
```

Jquery UI

Widgets > Tab Widget

Si queremos personalizar el estilo CSS sin usar el themeRoller, podemos añadir un archivo o estilo con las siguientes clases:

Contenedor

```
#myTabs {
        width:400px;
        padding:5px;
        border:1px solid #636363;
        background:#c2c2c2 none;
}
```

Cabecera

```
.ui-widget-header {
        border:0;
        background:#c2c2c2 none;
        font-family:Georgia;
}
```

Contenido

```
#myTabs .ui-widget-content {
        border:1px solid #aaa;
        background:#fff none;
        font-size:80%;
}
```

Pestañas activas y no activas

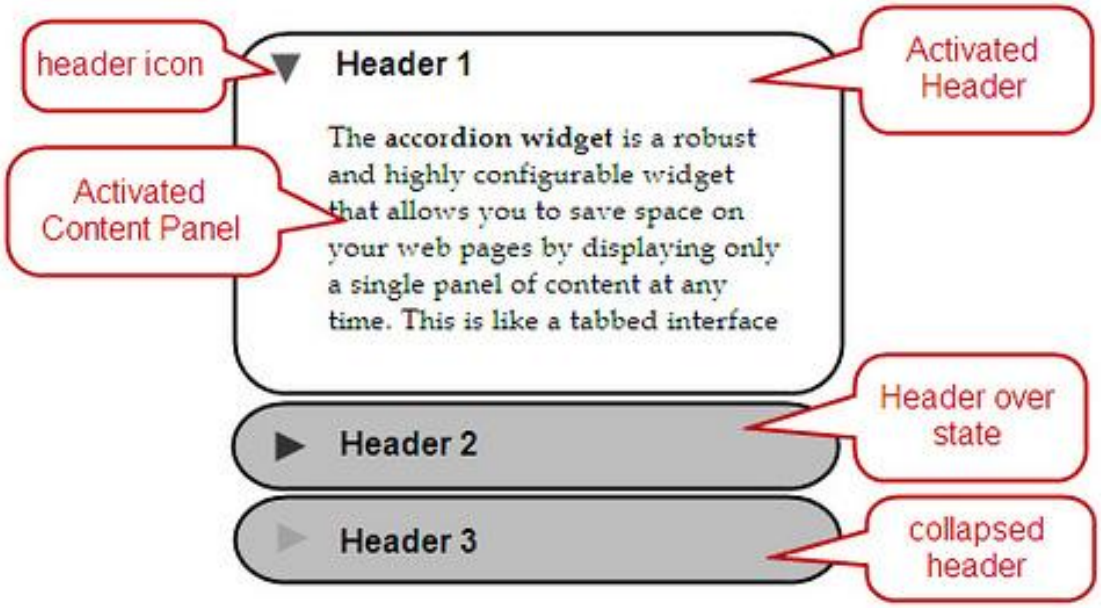
```
.ui-state-default, .ui-widget-content .ui-state-default {
        border:1px solid #636363;
        background:#a2a2a2 none;
}
.ui-state-active, .ui-widget-content .ui-state-active {
        border:1px solid #aaa;
        background:#fff none;
}
```

Jquery UI

Widgets >

1.2 Accordion

Los elementos básicos de este widget son:



Jquery UI

Widgets > Accordion

Para crearlo necesitamos una combinación h2 + div para cada elemento que deben estar incluidas en un contenedor

```
<div id="myAccordion">
	<h2><a href="#">Cabecera 1</a></h2>
	<div>Contenido 1</div>
	<h2><a href="#">Cabecera 2</a></h2>
	<div>Contenido 2</div>
	<h2><a href="#">Cabecera 3</a></h2>
	<div>Contenido 3</div>
</div>
```

```
<script type="text/javascript">
	$(function() {
		$("#myAccordion").accordion();
	});
</script>
```

Jquery UI

Widgets > Accordion

Si queremos personalizar el estilo CSS sin usar el themeRoller, podemos añadir un archivo o estilo con las siguientes clases:

Contenedor

```
#myAccordion {
        width:400px;
        border:1px solid #636363;
        padding-bottom:1px;
}
```

Cabecera

```
.ui-accordion-header {
        font-family:Georgia;
        background:#e2e2e2 none;
        border:1px solid #ffffff;
}
```

Contenido

```
.ui-widget-content {
        font-size:70%;
        border:none;
}
```

Pestañas activas y no activas

```
.ui-accordion .ui-accordion-header {
        margin:0 0 -1px;
}
#myAccordion .ui-state-active, #myAccordion
                .ui-widget-content .ui-state-active {
        background:#ffffff;
}
.ui-state-hover, .ui-widget-content .ui-state-hover {
        background:#d2d2d2;
}
```

Jquery UI

Widgets >

1.3 Dialog

A veces, tenemos que preguntar o informar al usuario sobre un determinado aspecto. La forma tradicional de hacer esto ha sido mediante diálogos, ventanas popup o pantallas, aunque ninguna de estas soluciones se adapta a las necesidades reales de un programador.

Jquery UI

Widgets > Dialog

Para crear diálogos simplemente tenemos que tener un div con el atributo title

```
<div id="myDialog" title="Título">
Contenidos
</div>
```

```
<script type="text/javascript">
        $(function(){
                $("#myDialog").dialog();
        });
</script>
```

Obtenemos un diálogo que se puede mover por la pantalla y que se presenta por encima del resto de contenidos.

Ejemplo: En http://www.mediamarkt.es/ mostrar un producto y hacer click en su imagen

Jquery UI

Widgets > Dialog

El problema es que este diálogo ya está abierto. Debería estar cerrado y abrirlo nosotros cuando quisiéramos. Tenemos que cambiar la declaración del diálogo y añadir, por ejemplo, un botón que se encargue de abrirlo

```
<div id="myDialog" title="Título">
Contenidos
</div>
<input type="button" onclick="muestra()">
```

```
<script>
$(function(){
        var dialogOpts = {autoOpen: false};
        $("#myDialog").dialog(dialogOpts);
});
function muestra(){
        if(!$("#myDialog").dialog("isOpen")) {
                $("#myDialog").dialog("open");
        }
}
</script>
```

Entre llaves separando por comas podemos indicar las diferentes opciones de los diálogos

Jquery UI

Widgets > Dialog

En las opciones podremos indicar todo lo necesario para configurar el diálogo:

```
var dialogOpts = {
width: 500,
height: 300,
minWidth: 150,
minHeight: 150,
maxWidth: 600,
maxHeight: 450
};
```

```
var dialogOpts = {
modal: true
};
```

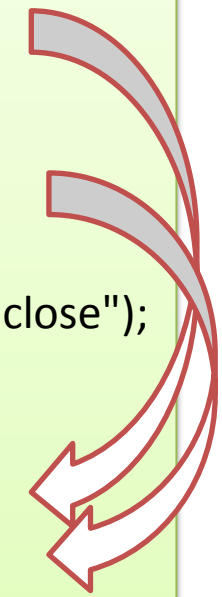
```
var dialogOpts = {
autoOpen: false
};
```

Jquery UI

Widgets > Dialog

Si queremos añadir botones, deberemos definir una función de respuesta para cada uno de ellos, y configurar el diálogo para que llame a estas funciones:

```
<script>
$(function(){
var aceptar = function() {
        alert('ok');
};
var cancelar = function() {
        alert('ooops');
        $("#myDialog").dialog("close");
};
var dialogOpts = {
        buttons: {
                "Ok": aceptar,
                "Cancel": cancelar
        },
        autoOpen: false
};
$("#myDialog").dialog(dialogOpts);
});
```



Jquery UI

Widgets > Dialog

Ahora que sabemos definir las funciones, podemos introducir en el contenido del diálogo elementos de formulario y recoger su valor en las funciones

```
<div id="myDialog" title="Introduce tus datos">
<p>Introduce los datos a  continuación:</p>
email: <input type="text" id="respuesta" />
</div>
```

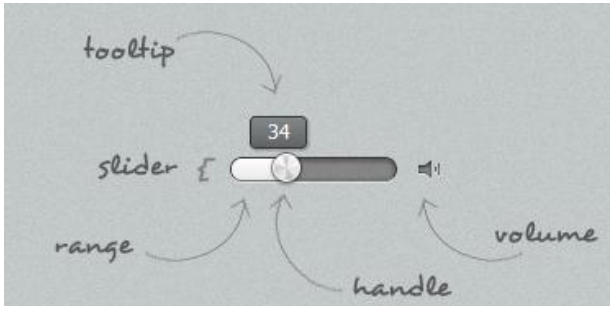
```
<script>
$(function(){
var aceptar = function() {
        var answer = $("#respuesta").val();
};
var cancelar = function() {
        $("#myDialog").dialog("close");
};
var dialogOpts = {
        buttons: {
                "Ok": aceptar,
                "Cancel": cancelar
        }
};
$("#myDialog").dialog(dialogOpts);
});
```

Jquery UI

Widgets >

1.4 Slider

Este componente nos sirve para que el usuario seleccione un valor de un rango posible de forma visual y sencilla.



Ejemplo: En http://shop.mango.com/ES/mango podemos encontrar el slider para seleccionar el precio del producto que buscamos

Jquery UI

Widgets > Slider

Para crear un slider simplemente tenemos que crear un div

```
<div id="mySlider">
</div>
```

```
<script type="text/javascript">
        $(function(){
                $("#mySlider").slider();
        });
</script>
```

Si queremos estilos personalizados:

```
. background-div {
}
#mySlider {
}
#mySlider .ui-slider-handle {
}
```

Jquery UI

Widgets > Slider

Nuevamente disponemos de opciones para configurarlo

```
<script type="text/javascript">
        $(function(){
                var sliderOpts = {
                ...
                };
                $("#mySlider").slider(sliderOpts);
        });
</script>
```

```
var sliderOpts = {
orientation: "vertical"
};
```

```
var sliderOpts = {
min: 0,
max: 100
};
```

```
var sliderOpts = {
step: 25
};
```

```
var sliderOpts = {
value: 50
}
```

```
var sliderOpts = {
values: [1, 100],
range:true
};
```

Jquery UI

Widgets > Slider

Para acceder al valor, necesitamos crear un función que capture los cambios producidos

```
<div id="mySlider"></div>
<p id="valor"></p>
```

```
$(function(){
	sliderOpts = {
		value: 5,
		min: 0,
		max: 10,
		change: function(event, ui) {
			var x = $(this).slider("value");
			$("#valor").html(x);
		}
	};
	$("#mySlider").slider(sliderOpts);
});
</script>
```

Jquery UI

Widgets > Slider

En caso de trabajar con multiples valores, la mejor forma para acceder al array es:

```
<div id="mySlider">
</div>
<p id="valor">
</p>
```

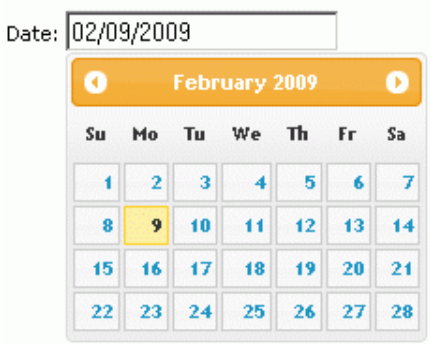
```
$(function(){
	sliderOpts = {
		values: [0, 10],
		range:true,
		min: 0,
		max: 10,
		change: function(event, ui) {
			var min = ui.values[0];
			var max = ui.values[1];
			$("#valor").html("max: " + max + ", min: "+ min);
		}
	};
$("#mySlider").slider(sliderOpts);});
```

Jquery UI

Widgets >

1.5 DatePicker

Este componente nos sirve para que el usuario pueda introducir fechas de una forma cómoda y además sabiendo que son fechas válidas



Ejemplo: En la mayoría de webs de viajes o reservas encontraremos este componente

Jquery UI

Widgets > DatePicker

Para crear un datePicker simplemente tenemos que crear un elemento input

```
<label>Introduce una fecha: </label>
<input id="myDate" />
```

```
<script type="text/javascript">
        $(function(){
                $("#myDate").datepicker();
        });
</script>
```

Jquery UI

Widgets > Datepicker

Nuevamente disponemos de opciones para configurarlo

```
<script type="text/javascript">
 $(function(){
        var pickerOpts = {
        ...
        };
        $("#myDate").datepicker(pickerOpts);
});
</script>
```

```
var pickerOpts = {
minDate: new Date(),
maxDate: "+10"
};
```

O también: semanas, meses o años: +3w, +2m, +1y

```
var pickerOpts = {
dateFormat:"dd MM yy"
};
```

Formatos

```
var pickerOpts = {
changeMonth: true,
changeYear: true
};
```

```
var pickerOpts = {
showOn: "button",
buttonText: "Open Picker"
};
```

```
var pickerOpts = {
showButtonPanel: true
};
```

Jquery UI

Widgets > Slider

Para acceder al valor seleccionado, debemos añadir una función:

```
$(function(){
var pickerOpts = {
        onSelect: function(dateText, inst) {
                var fechaComoTexto=dateText;
                var fechaComoObjeto=$( "myDate" ).datepicker('getDate');
                alert(fechaComoTexto);
        }
};
$("#myDate").datepicker(pickerOpts);
});
```