Manual jQuery-3.1.1

1.3.6- Extra: Bloquear el comportamiento normal de un enlace.....2

1.4- Básicos jQuery: añadir y quitar clases CSS sobre elementos.....2

1.5- Mostrar y ocultar elementos de la página con jQuery (prop/display…).....2

1.6- Efectos rápidos con jQuery (hide/show).....3

1.7- Callback de funciones jQuery (fadeout/fadein).....3

2.2.1- Función jQuery enviando un selector y un contexto.....4

2.4.1- Cómo funciona each.....4

2.4.2- Retornando valores en la función que enviamos a each.....5

2.5.2- Propiedad length del objeto jQuery.....5

2.6.3- Ejemplo completo de los métodos data() y removeData() del Core de jQuery.....5

4.3.3- Ejemplo completo sobre los métodos de dimensiones y posición de elementos.....6

5.3.1- Averiguar la posición del ratón al hacer clic.....7

6.2.1- Ejemplo de un plugin en jQuery.....8

1.3.6- Extra: Bloquear el comportamiento normal de un enlace

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $("a").click(function (evento) {
      alert("ok");
      evento.preventDefault();
    });
  });
</script>
```

Como podemos observar, simplemente llamamos a la etiqueta de nombre ‘a’ en nuestro código y le añadimos una función anónima cuando pinchamos sobre dicho enlace. Con el método ‘preventDefault’ impedimos que se ejecute la acción propia de un enlace.

1.4- Básicos jQuery: añadir y quitar clases CSS sobre elementos

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $("a").mouseover(function (event) {
      $("#capa").addClass("classcss")
    });
    $("a").mouseout(function (event) {
      $("#capa").removeClass("classcss")
    });
  });
</script>
```

Para añadir y quitar clases de un elemento utilizamos la función ‘.addClass’ y ‘removeClass’, respectivamente. Ambas funciones requieren un parámetro, que es el nombre de la clase.

1.5- Mostrar y ocultar elementos de la página con jQuery (prop/display…)

```
<script type="text/javascript">
  $(document).ready(function () {
    $("#mayoria_edad").click(function (event) {
      if($("#mayoria_edad").prop("checked")){
        $("#formulariomayores").css("display", "block");
      }else{
        $("#formulariomayores").css("display", "none");
      }
    });
  });
</script>
```

Para cambiar propiedades de CSS utilizamos la función ‘css’. Debemos indicar dos parámetros donde especificamos la propiedad a cambiar y el nuevo valor..

**Nota: con la nueva versión de jQuery (3.1.1), la función ‘attr’ fue sustituida por ‘prop’ para acceder al valor de una propiedad del elemento seleccionado.*

1.6- Efectos rápidos con jQuery (hide/show)

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $("#ocultar").click(function (event) {
      event.preventDefault();
      $("#capaefectos").hide("slow");
    });
    $("#mostrar").click(function (event) {
      event.preventDefault();
      $("#capaefectos").show(3000);
    });
  });
</script>
```

Gracias a las funciones ‘hide’/’show’ podemos ocultar/mostrar un elemento rápidamente. A estás funciones podemos pasarlo un parámetro en el que indicamos cuántos milisegundos debe durar el efecto de aparecer/desaparecer

1.7- Callback de funciones jQuery (fadeout/fadein)

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $("#pilallamadas").click(function (event) {
      event.preventDefault();
      $("#micapa").fadeOut(1000, function () {
        $("#micapa").css({
          'top': 300,
          'left': 200
        });
        $("#micapa").fadeIn(1000);
      });
    });
  });
</script>
```

Con las funciones ‘fadeOut’ y ‘fadeIn’ podemos controlar que un elemento desaparezca y reaparezca en una nueva posición. También podemos controlar el tiempo que tardará en realizarse la animación

2.2.1- Función jQuery enviando un selector y un contexto

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    var elem1 = $("#elem1");
    elem1.css("background-color", "#ff9999");

    var divs = $("div");
    divs.css("font-size", "32pt");

    var inputs = $("input", document.forms[0]);
    inputs.css("color", "red");

    var parrafos_div1 = $("p", "#div1");
    parrafos_div1.hide(5000);
  });
</script>
```

En un selector podemos indicar también un contexto para afinar más nuestra búsqueda

2.4.1- Cómo funciona each

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $("p").each(function (i) {
      if(i % 2 == 0){
        $(this).css("background-color", "#eee");
      }else{
        $(this).css("background-color", "#ccc");
      }
    });
  });
</script>
```

Con la función ‘each’ recorremos todos los elementos que hayas seleccionado con el selector y gracias a una variable podemos especificar a cada elemento diferentes propiedades

2.4.2- Retornando valores en la función que enviamos a each

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $("div").each(function (i) {
      var elemento = $(this);
      if (elemento.html() == "white")
        return true;
      if(elemento.html() == "nada")
        return false;
      elemento.css("color", elemento.html());
    })
  })
</script>
```

Dentro de un ‘each’ también podemos retornar valores y trabajar con los mismo

2.5.2- Propiedad length del objeto jQuery

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    var ElementosMitexto = $(".mitexto");
    alert('Hay ' + ElementosMitexto.length + " elementos de la clase mitexto");
  });
</script>
```

Con la propiedad ‘length’ obtenemos el número de elementos que hemos almacenado con el selector

2.6.3- Ejemplo completo de los métodos data() y removeData() del Core de jQuery

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $('#guardar').click(function (event) {
      var valor = $("#valor").prop("value");
      $("#division").data("midato", valor);
      $("#division").html('He guardado en este elemento (id="division") un dato' +
      'llamado "midato" con el valor "' + valor + '"');
    });
    $('#leer').click(function (event) {
      valor = $('#division').data("midato");
      $('#division').html('En este elemento (id="division") leo un dato llamado' +
      '"midato" con el valor "' + valor + '"');
    });
    $('#eliminar').click(function (event) {
      $('#division').removeData("midato");
      $('#division').html('Acabo de eliminar del elemento (id="division") el ' +
      'dato llamado "midato"');
    });
  });
</script>
```

Con la función ‘data’ podemos recoger un valor para luego tratarlo más adelante. Para borrar el valor almacenado en ‘data’ utilizamos ‘removeData’. Cuando invocamos la función ‘removeData’ debemos especificar el nombre que le añadimos indicado a ese valor.

4.3.3- Ejemplo completo sobre los métodos de dimensiones y posición de elementos

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  function dimensionCapa(capa) {
    capa = $(capa);
    var dimensiones = "";
    dimensiones += "Dimensiones internas: " + capa.innerWidth() + "x" + capa.innerHeight();
    dimensiones += "\nDimensiones externas: " + capa.outerWidth() + "x" + capa.outerHeight();
    alert(dimensiones);
  }
  function posicionCapa(capa) {
    capa = $(capa);
    var posicion = "";
    posicion += "Posición relativa al documento:\nLEFT: " + capa.offset().left +
                "\nTOP:" + capa.offset().top;
    posicion += "\nPosición si no tuviera margen:\nLEFT: " + capa.position().left +
                "\nTOP: " + capa.position().top;
    alert(posicion);
  }
```

```
  $(document).ready(function () {
    $('#botondimensiones').click(function () {
      dimensionCapa('#capa1');
    });
    $('#botonposicion').click(function () {
      posicionCapa('#capa1');
    });
    $('#botontamano').click(function () {
      $('#capa1').css("width", 200);
    });
    $('#botonmargen').click(function () {
      $('#capa1').css("margin", 20);
    });
    $('#botondimensionesc2').click(function () {
      dimensionCapa('#capa2');
    });
    $('#botonposicionc2').click(function () {
      posicionCapa('#capa2');
    });
  });
</script>
```

Gracias a las funciones especificadas más arriba podemos determinar el tamaño y la posición del elemento deseado

5.3.1- Averiguar la posición del ratón al hacer clic

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  $(document).ready(function () {
    $(document).click(function (e) {
      alert('X: ' + e.pageX + ' Y: ' + e.pageY);
    });
  });
</script>
```

Gracias a las propiedades ‘pageX’ y ‘pageY’ obtenemos las coordenadas en píxeles de dónde se encuentra localizado, en el caso, el ratón

6.2.1- Ejemplo de un plugin en jQuery

```
<script src="jquery-3.1.1.min.js" charset="utf-8"></script>
<script type="text/javascript">
  jQuery.fn.parpadea = function () {
    this.each(function () {
      elem = $(this);
      elem.fadeOut(250, function () {
        $(this).fadeIn(250);
      });
    });
    return this;
  }
  $(document).ready(function () {
    $('.parpadear').parpadea();
    $('#botonparpadear').click(function () {
      $('.parpadear').parpadea();
    });
  });
</script>
```

Con el código de arriba podemos desarrollar un plugin gracias al cual haremos que los elementos indicados parpadeen.