Objetos.

- No existe el concepto de clase propiamente dicho, ni el de herencia.
- Crear objetos:
  - Constructor genérico Object():

```
var obj = new Object();
obj.exist = 3; obj.stock = true; obj.estado = “Vend.”;
```

  - Dando una lista de propiedades:

```
var obj = {exist:3, stock:true, estado:“Vendidos” };
```

  - Empleando constructores definidos por el usuario:

```
function  Articulo( exi, sto, est) {
        this.exist = exi; this.stock = sto; this.estado = est;
}
var obj = new Articulo(3, true, “Vendidos”);
```

Objetos (II).

- Para acceder a las propiedades del objeto se utiliza el operador “.”, o el operador “[ ]”:

```
obj.exist = 45; // Ambas sentencias son equivalentes
obj[“exist”] = 45;
```

- Las propiedades y los métodos del objeto se añaden dinámicamente, sin necesidad de ser definidas con antelación:

```
function Resumen() {
        var cad = "";
        for (prop in this)
                cad += this[prop].toString() + "\n";
        return cad;
}
obj.precio = 12.4;
obj.extracto = Resumen;
alert(obj.extracto());
```

Objetos (III)

- Las propiedades y los métodos pueden ser eliminados dinámicamente con el operador delete.

```
delete obj.valor;
alert(obj.extracto());
```

- Con las propiedades y los métodos prototipo (prototype) se pueden añadir de forma dinámica propiedades y métodos al constructor.

```
function Valor() {
        return this.precio * this.exist;
}
...
var a1 = new Articulo(2,true,"Vendidos");
var a2 = new Articulo(4,true,"Vendidos");
Articulo.prototype.precio = 1.0;
Articulo.prototype.valor = Valor;
a2.precio = 15;
alert('Total:' + (a1.valor() + a2.valor()));
```

jQuery

- Jquery es un framework Javascript
- Un framework es un producto que sirve como base para la programación avanzada de aplicaciones, son unas librerías de código que contienen procesos o rutinas ya listos para usar.
- jQuery implementa una serie de clases que nos permiten programar sin preocuparnos del navegador, ya que funciona igual en todas las plataformas.
- Como usarlo:
  - Descarga la última versión del framework Jquery.com
  - Anexarlo al html <script src="jquery-x.x.x.min.js" type="text/javascript"> </script> ///x.x.x es la version
  - Ejecutar código cuando la página ha sido cargada

    window.onload = function () { … }

JQuery

- window.onload sólo se ejecutará cuando el navegador haya descargado **TODOS** los elementos de la página, (imágenes, iframes, banners,etc.).
- No hace falta esperar a toda la carga de elementos para poder ejecutar sentencias DOM. Sólo habría que hacerlo cuando el navegador ha recibido el código HTML completo y lo ha procesado al renderizar la página.
- jQuery con la siguiente sentencia incluye una manera de hacer acciones justo cuando ya está lista la página, aunque haya elementos que no hayan sido cargados del todo.

```
$(document).ready(function(){
//código a ejecutar cuando el DOM está listo para recibir instrucciones.
});
```

- Con $(document) se obtiene una referencia a la página web, con el ready() se define un evento, que se desata al quedar listo el documento para el DOM

Clases CSS

- Para alterar elementos de una página web, añadiendo y quitando clases CSS en jQuery hay dos clases llamadas addClass() y removeClass(). Se añadirán y quitarán clases del elemento con respuesta a acciones del usuario, mouseover y mouseout.

Clases CSS

```
<html>
   <head>
      <script src="jquery-1.3.2.min.js" type="text/javascript"></script>
      <style type="text/css">  .clasecss{ background-color: #ff8800; font-weight: bold; }
      </style>
      <script>
         $(document).ready(function(){
            $("a").mouseover(function(event){
               $("#capa").addClass("clasecss");
             });
            $("a").mouseout(function(event){
               $("#capa").removeClass("clasecss");
            });
         });
       </script>
    </head>
    <body>
      <div id="capa"> Capa sobre la que voy a añadir y eliminar css </div>
      <br>
      <a href="http://www.google.com">Añadir y quitar</a>
     </body>
</html> ///3.html
```

Modificar CSS

- Tenemos un formulario con algunos elementos escondidos al marcar un checkbox, se mostrarán y al desmarcar se ocultarán.

```
<html>  <head>
        <script src="jquery-X.X.X.min.js" type="text/javascript"></script>
        <script>
             $(document).ready(function(){
                 $("#mayoria").click(function(evento){  //al clikar en el checkbox
                     if ($("#mayoria").attr("checked"))  //si esta marcado
                           $("#formmayores").css("display", "block"); //muestro el contenido oculto
                     else
                           $("#formmayores").css("display", "none"); // sino lo escondo
                 });
              });
         </script>
    </head>
    <body> <form>
             Nombre: <input type="text" name="nombre">
            <input type="checkbox" value="1" id="mayoria"> Soy mayor de edad
             <div id="formmayores" style="display: none;"> ///por defecto escondido
                  Para mayores de edad: <input type="text" name="mayores_edad">
            </div>
        </form>
     </body></html>  /////4.html
```

Efectos Rapidos

```
<html> <head>
      <script src="jquery-X.X.X.min.js" type="text/javascript"></script>
      <script>
        $(document).ready(function(){
            $("#ocultar").click(function(event){  //click sobre link ocultar
                  event.preventDefault();  //no hago el salto del link elimino su comportamiento
                  $("#capaefectos").hide("slow"); //oculto despacio
            });
            $("#mostrar").click(function(event){  ///click sobre mostrar
                 event.preventDefault();  ///elimino el comportamiento normal del link
                 $("#capaefectos").show(3000);  //muestro con una accion que dura  3 segundos
            });
        });
      </script>
   </head> <body>
       <div id="capaefectos" style="background-color: #cc7700; color:fff; padding:10px;">
              Capa para hacer efectos!
       </div>
       <p>
          <a href="#" id="ocultar">Ocultar</a> |  <a href="#" id="mostrar">Mostrar</a>
       </p>
    </body>  </html>  ///5.html
```

Callback de funciones

- Callback → función se ejecuta después de otra.
- Para ocultar una capa con un efecto de fundido (de opaco a transparente), luego moverla a otra posición y volverla a mostrar (ya en la nueva posición) con otro efecto de fundido (en este caso de transparente a opaco) podríamos pensar en hacer un código como este:

```
$("#micapa").fadeOut(2000);
$("#micapa").css({top: 300, left:200});
$("#micapa").fadeIn(2000);
```

- Será como si se ejecutasen todas a la vez, los fadeOut y fadeIn tardan 2 segundos en ejecutarse y los cambios de CSS top y left son inmediatos, asi que primero se movera a la nueva posición y luego hara los dos efectos de fundido.
- Todas las funciones jQuery pueden recibir el nombre de la función a ejecutar después de su ejecución (callback).

```
miFuncion ("parametros de la función", funcionCallback);//para funciones sin parametros
```

Callback de funciones (II)

```
miFuncion ("parametros de la funcion", function(){

        funcionCallback();

}); //funciona igual que el anterior, pero sí podemos indicar los parámetros en funcionCallback();
```

- Ejemplo

```
$("#micapa").fadeOut(1000, function(){   /////esto se ejecuta despues de fadeout
    $("#micapa").css({'top': 300, 'left':200});
    $("#micapa").fadeIn(1000); //css es inmediato, no hay que hacer callback
});
```

Core de jQuery

- Contiene una serie de clases y métodos clasificadas en diversos apartados:
  - $() (La funcion jQuery): Lla función principal, tiene diversas utilidades según los parámetros que se le envíen. Su utilidad principal es obtener elementos de la página.

    $() <=> jQuery() son equivalentes pero jQuery es más compatible con otras librerias que pueden usar tambien $()
  - Accesorios de objetos: Funciones para hacer cosas con objetos: iterar con cada uno de sus elementos, saber su tamaño, longitud, el selector o contexto, obtener sus elementos DOM, etc.
  - Trabajo con datos: Funciones para trabajar con datos y asociarlos a elementos, una forma de guardar información adicional en elementos, trabajar con colas y administrar la salida y entrada.
  - Plugins: Permiten extender los elementos jQuery para incorporar nuevos método.
  - Interoperabilidad: Funciones para evitar problemas de compatibilidad con otras librerías.

jQuery o $()

- Sirve para hacer varias cosas, según los parámetros que le pasemos. El más simple es seleccionar elementos o grupos de elementos, también para crear elementos o para hacer callback de funciones.
- Para seleccionar elementos. Recibe una expresión devolviendo un objeto jQuery donde se encuentran todos los elementos de la página seleccionados y extendidos con las funcionalidades del framework. Podemos pasarle un contexto.
    - Si no se le envía un contexto, selecciona elementos del documento completo.
    - Si indicamos un contexto conseguiremos seleccionar elementos sólo dentro de ese contexto.

```
var elem1 = $("#elem1"); //Selecciona un elemento que tiene id "elem1" y se guarda en elem1
elem1.css("background-color", "#ff9999"); //en elem1 se puede hacer cualquier cosa
var divs = $("div"); //selecciona todos los div
divs.css("font-size", "32pt"); //modifica todos los div
```

jQuery o $()

- Segundo parametro opcional:

  var inputs = $("input",document.forms[0]); //A los INPUT del primer formulario
  inputs.css("color", "red"); //se les cambia el color del texto.

  var parrafos_div1 = $("p","#div1"); // los párrafos que están en un DIV con id="div1".
  parrafos_div1.hide(); //Con hide() conseguimos que se oculten

- Pasando un html

  var nuevosElementos = $(" Elementos que creo en tiempo de ejecución.<h1>En ejecución...</h1> ");

- Crea en nuevosElementos con los elementos HTML del string. Luego podemos colocarlos en la página con appendTo().

  nuevosElementos.appendTo("body");

jQuery o $()

- **Pasando elementos:** se le puede enviar un elemento o jerarquía de elementos DOM, para extenderlos con las funcionalidades que aporta el framework para los elementos.

  var documento = $(document.body); //Obtenemos el cuerpo de la página
  documento.css("background-color", "#ff8833"); //al que le hemos agregado todas las funcionalidades de jQuery.

- Esto no funcionaría al no estar extendido

  document.body.css("background-color", "#ff8833"); // X

  $(document.body).css("background-color", "#ff8833"); // Esto si

jQuery o $()

- Pasando una función: tenemos un callback que se invoca automáticamente cuando el DOM está listo para recibir comandos.

```
$(function (){  //// o jQuery(function (){
      //Aquí puedo hacer cualquier cosa con el DOM
});
```

- Sería una abreviatura de

```
$(document).ready(function(){ // la pagina esta lista para recibir el DOM
      //Aquí puedo hacer cualquier cosa con el DOM
});
```

```
window.onload = function(){...} //no es exactamente como esta como ya vimos en la teoria de DOM ya que esta se espera a que carge toda la pagina con todos los elementos
```

- Incluso podemos hacer varios callback, para agregar distintas acciones a realizar cuando el DOM está listo, las veces que queramos, igual que cuando definíamos el evento ready() sobre el objeto document.

each del core de jQuery

- Método para realizar acciones con todos los elementos que concuerdan con una selección realizada con la función jQuery, es una manera cómoda de iterar con elementos de la página sin utilizar mucho código para definir el bucle.
- Recibe una función que es la que se tiene que ejecutar para cada elemento y dentro de esa función con la variable "this" tenemos una referencia a ese elemento del DOM. La función que se envía a each, puede recibir un parámetro que es el índice actual sobre el que se está iterando.

each del core de jQuery

```
$("p").css("background-color", "#eee"); //párrafos con colores alternos
$("p").each(function(i){
	if(i%2==0)
		$(this).css("background-color", "#eee");
	else
		$(this).css("background-color", "#ccc");
});
```

Con $("p") tengo todos los párrafos. Con each se recorren uno a uno. Para cada uno se ejecuta la función.

En esa función se recibe como parámetro la variable "i" que contiene el índice actual sobre el que se itera.

Tenemos que hacer $(this) para poder invocar al método css(). extendiendo la variable this con las funcionalidades de jQuery

```
this.css("background-color", "#ccc"); //X No funciona

$(this).css("background-color", "#ccc"); // V Si funciona
```

Retornando valores en la función parametro de each

- la función que enviamos como parámetro a each() puede devolver valores y dependiendo de éstos, conseguir comportamientos como break o continue
  - Si devuelve "false" será como "break".
  - Si devuelve "true", será como "continue".
- Ejemplo de Recorrido de DIV:
  - mirar el texto que aparece.
  - Colocar el color del texto.
    - Pero si es "white", no hacer nada.
    - Si es "nada" detener el bucle.

Retornando valores en la función parametro de each

- El html sera:

```
<div>red</div>
<div>blue</div>
<div>red</div>
<div>white</div>
<div>red</div>
<div>green</div>
<div>orange</div>
<div>red</div>
<div>nada</div>
<div>red</div>
<div>blue</div>
```

- Se puede hacer con el codigo:

```
$("div").each(function(i){
        elemento = $(this);
        if(elemento.html() == "white")
            return true;
        if(elemento.html() == "nada")
            return false;
        elemento.css("color", elemento.html());
});
```

Método size() y propiedad length

- **Método size():** para obtener el número de elementos seleccionados con jQuery. El número de elementos en el objeto, como un entero.

  Al ser un metodo se usa con parentesis.

```
var parrafos = $("p");
alert ("Hay " + parrafos.size() + " párrafos en la página");
```

- **Propiedad length:** almacena directamente este valor, es más rápido y aconsejable que size().

  Al ser una propiedad se usa sin paréntesis.

```
var ElementosMitexto = $(".mitexto");
alert ("Hay " + ElementosMitexto.length + " elementos de la clase mitexto");
```

Método data()

- data(): Sirve para guardar un dato en un elemento y para consultarlo. Según los parámetros que reciba.
  - Un parámetro data(nombre): devuelve el valor con ese nombre.
  - Dos parámetros data(nombre, valor): almacena un dato, con ese nombre y ese valor.
- Podemos almacenar estos pares (dato, valor) en cualquier elemento.

```
<div id="capa">
    En esta división (elemento id="capa") voy a guardar y leer datos sobre este elemento.
</div>
$("#capa").data("midato","mivalor");
```

- Guardado un dato llamado "midato" con el valor "mivalor", se puede leer ese dato en cualquier momento.

```
alert($("#capa").data("midato"));
```

Método removeData()

- Sirve para eliminar un dato de un elemento: el funcionamiento es simple enviar por parámetro el dato que se quiere eliminar del elemento.

```
$("#capa").removeData("midato")
```

- Sobre data() y removeData():
  - **Admiten cualquier tipo de dato:** cadenas de texto, variable numérica, array, objeto Javascript o jQuery.
  - **Se guarda un dato por cada elemento del objeto jQuery selecionado:** En caso que en el objeto jQuery sobre el que estemos almacenando cosas con data() haya referencias a varios elementos de la página, el dato se almacena en todos los elementos.
  - **Los objetos se almacenan por referencia:** no se copia el objeto, sino que se asigna por referencia.

Exercicis de la primera part!!!!!!!