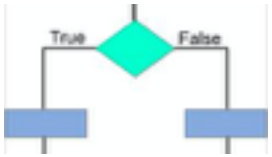
JavaScript:
Tipos, objetos y clases

Tipos, objetos y valores (ES3 y ES5)

- **Tipos** de JavaScript
  - **number**
    - Literales de números: **32**, **1000**, **3.8**

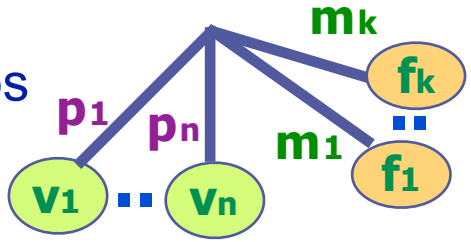
  - **boolean**
    - Los literales son los valores **true** y **false**



  - **string**
    - Los literales de string son caracteres delimitados entre comillas o apóstrofes
      - **"Hola, que tal", 'Hola, que tal',**
    - Internacionalización con Unicode: **'Γεια σου, ίσως', '嗨，你好吗'**

  - **undefined**
    - **undefined:** representa un valor no **definido** todavía

- **Objeto**: agregación estructurada de propiedades y métodos
  - Se agrupan en **clases:** Object, Array, Date, ...
    - Objeto **null**: valor especial que representa objeto nulo



https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Values,_variables,_and_literals

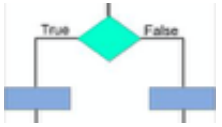
Operador typeof (ES3 y ES5)

- El operador **typeof** permite conocer el tipo de un valor
  - Devuelve un **string** con el **nombre del tipo**
    - "number", "string", "boolean", "undefined", "object" y "function"
      - **Todos los objetos** (de cualquier clase) devuelven **"object"**, salvo las **funciones**

typeof 7 => "number"

typeof "hola" => "string"

typeof true => "boolean"



typeof undefined => "undefined"

typeof null => "object"

typeof new Date() => "object"

typeof new Function() => "function"

Clases y herencia

- Todos los objetos de JavaScript pertenecen a la **clase Object**
  - Javascript posee otras **clases predefinidas** que derivan de Object
    - **Date, Number, String, Array, Function**, ….
      - https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript
  - Un objeto **hereda** los métodos y propiedades de su clase
    - Un objeto puede tener además otras propiedades y métodos propios

- Métodos y clases de un objeto utilizan la **notación punto** o **array**
  - Una propiedad se accede como: **objeto.propiedad** o **objeto.["propiedad"]**
  - Un método se invoca como: **objeto.metodo(…)** o **objeto["metodo"](…)**

- Todas las clases tienen un **constructor** con el **nombre de la clase**
  - El constructor permite **crear objetos** con el **operador new**
    - Pero los literales suelen más eficientes creando objetos, arrays, etc.
      - Por ejemplo, **new Object()** crea un objeto vacío equivalente a **{}**

Algunas clases predefinidas (ES3 y ES5)

- **Object**
  - Clase raíz, suelen crearse con el literal: **{a:3, b:"que tal"}**
- **Array**
  - Colección indexable, suelen crearse con el literal: **[1, 2, 3]**
- **Date**
  - Hora y fecha del reloj del sistema, se crea con constructor: **new Date()**
- **Function**
  - Bloque de código invocado por su nombre, se crea con: **function (x) {....}**
- **RegEx**
  - Expresiones regulares se crean con el literal: **/(hola)+$/**
- **Math**
  - Modulo con constantes (propiedades) y funciones matemáticas (métodos)
- **Number, String y Boolean**
  - Clases que encapsulan valores de los tipos number, string y boolean como objetos
    - Sus métodos se aplican a los tipos directamente, la conversión a objetos es automática
- Más información:
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript

Métodos heredados

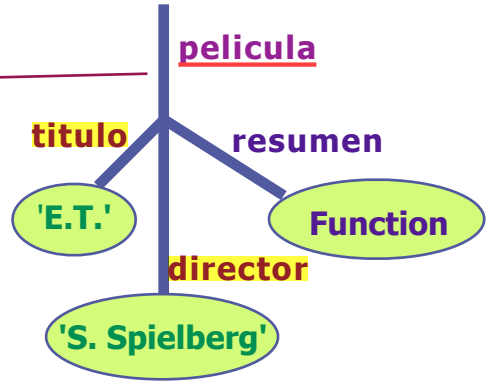
- **Método**: función que se puede invocar sobre un objeto con el operador punto: "."
- Un objeto **hereda** métodos de su **clase**, por ejemplo
  - los objetos de la **clase Date heredan** métodos como
    - **toString()**, **getDay()**, **getFullYear()**, **getHours()**, **getMinutes()**, ….. (ver ejemplo)
      - jhttps://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Date
  - También es posible definir nuevos métodos en un objeto
- En un objeto **solo** se puede invocar métodos **heredados** o **definidos** en él
  - Invocar un **método no heredado ni definido** en un objeto
    - provoca un **error_de_ejecución** y aborta el programa

```
var fecha = new Date();  // El objeto creado contiene fecha-hora de creación (ejecución)

fecha.toString()        =>  Fri Aug 08 2014 12:34:36 GMT+0200 (CEST)
fecha.toTimeString()    =>  12:34:36 GMT+0200 (CEST)
fecha.getHours()        =>  12
fecha.getMinutes()      =>  34
fecha.getSeconds()      =>  36
```

Definición de un método de un objeto

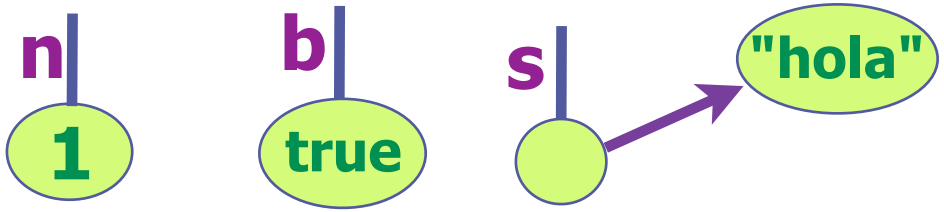
- Un **método** se puede definir asociado a un objeto
  - El nuevo método **solo pertenecerá a ese objeto** (no es de la clase)
- Al invocar un método cambia el **entorno de ejecución** de JavaScript
  - El **entorno** pasa a ser el **objeto invocado**, que se referencia con **this**
    - En el ej., **this.titulo** referencia la **propiedad titulo** del objeto sobre el que se invoca **resumen()**
      - **this** puede omitirse: en el ejemplo **this.titulo** es igual a **titulo**

```
var pelicula = {
    titulo:'E.T.',
    director:'S. Spielberg',

    resumen:function (){
        return "El director de " + this.titulo + " es " + this.director;
    }
}

pelicula.resumen()      =>      "El director de E.T. es S. Spielberg"
```

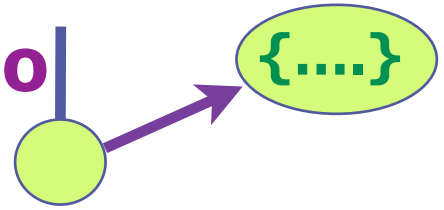


JavaScript:
Referencias a objetos

Valores y referencias



- Los tipos JavaScript se gestionan por valor o por referencia
  - **number**, **boolean** o **undefined** se gestionan por valor
    - **string** se gestiona por referencia, pero es a todos los efectos un valor,
  - **object** se gestiona por referencia

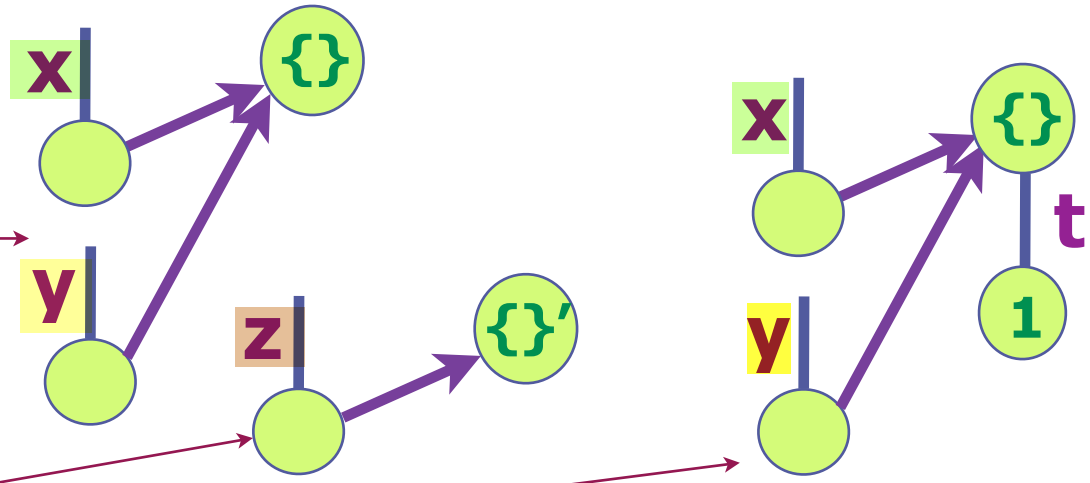


- La **asignación** copia el contenido de la variable
  - Copia el valor o el puntero según sea el contenido

- La **identidad** y la **igualdad** también se ven afectadas
  - Comparan el valor o el puntero según sea el contenido
    - Salvo con **strings** donde se comparan los valores (string apuntado) y no los punteros

Efectos laterales de las referencias a objetos

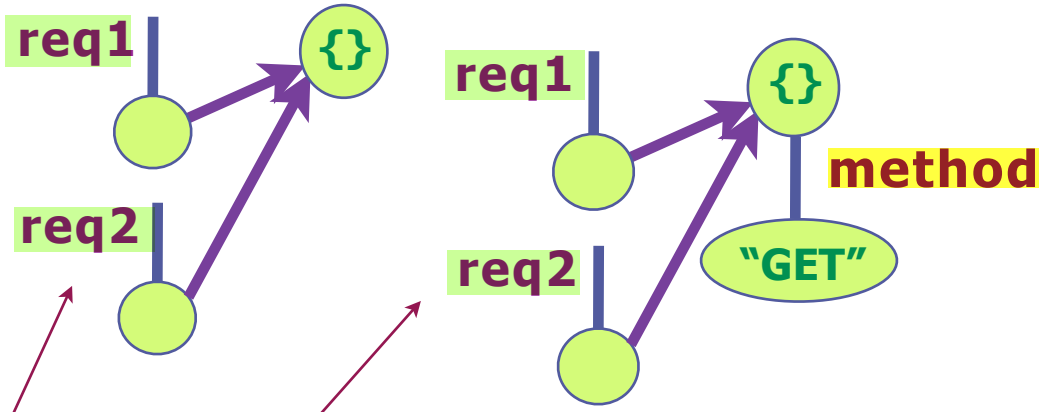
```
var x = {};  // x e y tienen la
var y = x;   // misma referencia

var z = {};  // la referencia a z
             // es diferente de
             // la de x e y

y.t = 1;

x.t     => 1   // x accede al mismo
y.t     => 1    // objeto que y
z.t     => undefined
```



- Las variables que contienen objetos
  - solo contienen la referencia al objeto
- En objeto esta en otro lugar en memoria
  - indicado por la referencia
- Al asignar una variable se copia el puntero
  - si se modifica el objeto de una de ellas
    - Se modificarán los objetos de las variables que contengan el mismo puntero
- Los **parámetros de funciones** tienen el mismo efecto lateral, cuando son objetos

Parámetros por referencia

```
71_func_reference.js                                    UNREGISTERED
1  var req = {};
2
3  function set(req1) {
4    req1.method = "GET";
5  }
6
7  function answer(req2) {
8    if (req2.method === "GET") {
9      return "Ha llegado: " + req2.method;
10   } else {
11     return "-> Error 37" ;
12   }
13 }
14
15 answer(req); // => "-> Error 37"
16
17 set(req);
18 answer(req); // => "Ha llegado: GET"
```



- Cuando pasamos objetos como parámetro
  - solo se pasa la referencia al objeto
- Si varias funciones modifican el mismo objeto
  - las modificaciones se verán en todas ellas
- Los objetos son un mecanismo muy eficaz
  - para comunicar funciones entre sí

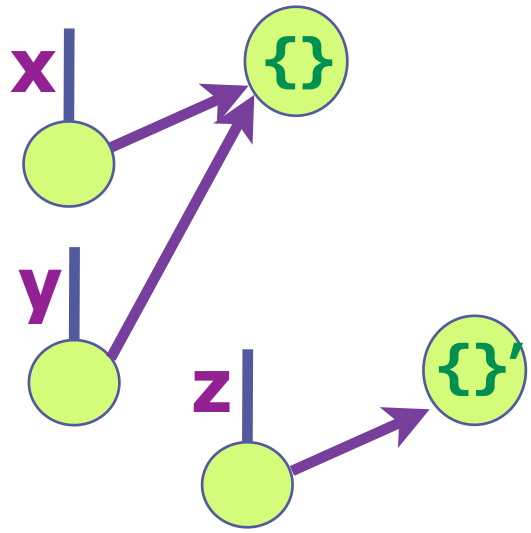
Identidad e igualdad de objetos

- Las referencias a objetos afectan a la identidad
  - porque **identidad de objetos**
    - es **identidad de referencias**
  - los objetos no se comparan
    - se comparan solo las referencias

- La **identidad de objetos no es util** si no se redefine
  - En los **strings** si, porque se comparan los valores

- Igualdad (debil) de objetos **==** y **!=**
  - no tiene utilidad tampoco con objetos
    - no se debe utilizar

```
var x = {};  // x e y contienen la
var y = x;   // misma referencia

var z = {}   // la referencia a z
             // es diferente de x e y

x === y      => true
x === {}     => false
x === z      => false
```



JavaScript:
Algunos métodos de Array

Métodos de Array

Array hereda métodos de su clase

- **sort()**: devuelve el array ordenado
  - y lo guarda en la variable a
- **reverse()**: devuelve el array invertido
  - y lo guarda en la variable a
- **push(e1, .., en)**
  - añade **e1, ..,en** al final del array
    - devuelve el tamaño del array (a.length)
- **pop()**
  - devuelve el último elemento, eliminadolo del array

```
var a = [1, 5, 3];

a.sort()          =>  [1, 3, 5]
a                 =>  [1, 3, 5]

a.reverse()       =>  [5, 3, 1]
a                 =>  [5, 3, 1]

a.push(false)     =>  4
a                 =>  [5, 3, 1, false]

a.pop()           =>  false
a                 =>  [5, 3, 1]
```

Más métodos

- **join(<separador>):**
  - devuelve string uniendo elementos
    - introduce <separador> entre elementos
      - La coma de separador permite generar CSV

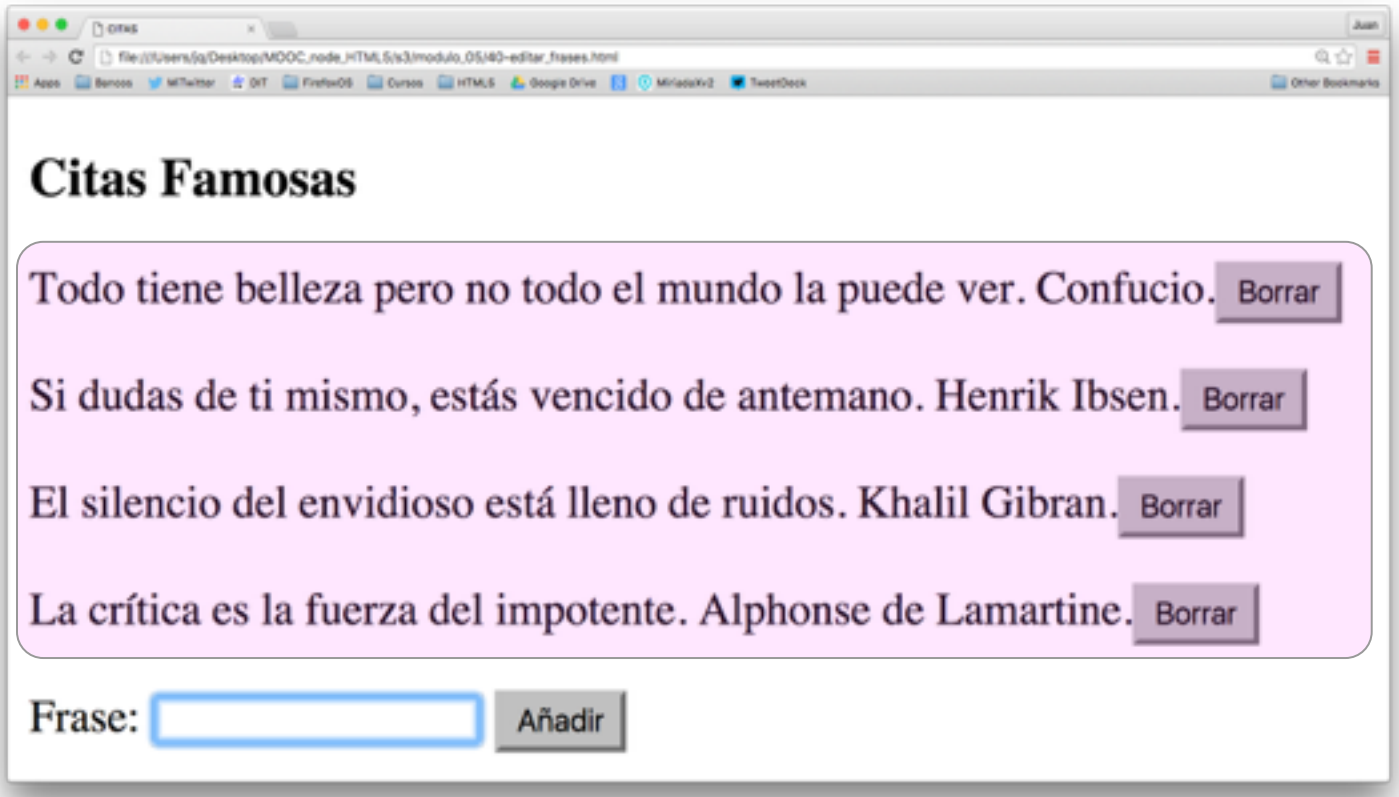
- **slice(i,j):** devuelve una rodaja
  - Indice negativo (j) es relativo al final
    - índice "-1" es igual a a.length-2

- **splice(i, j, e1, e2, .., en)**
  - sustituye j elementos desde i en array
    - por e1, e2, ..,en
  - Devuelve elementos eliminados

```
var a = [1, 5, 3, 7];

a.join(',')            => '1,5,3,7'

a.slice(1, -1)         => [5, 3]
a                      => [1, 5, 3, 7]

a.splice(1,2,true)     => [5, 3]
a                      => [1, true, 7]
```

Citas

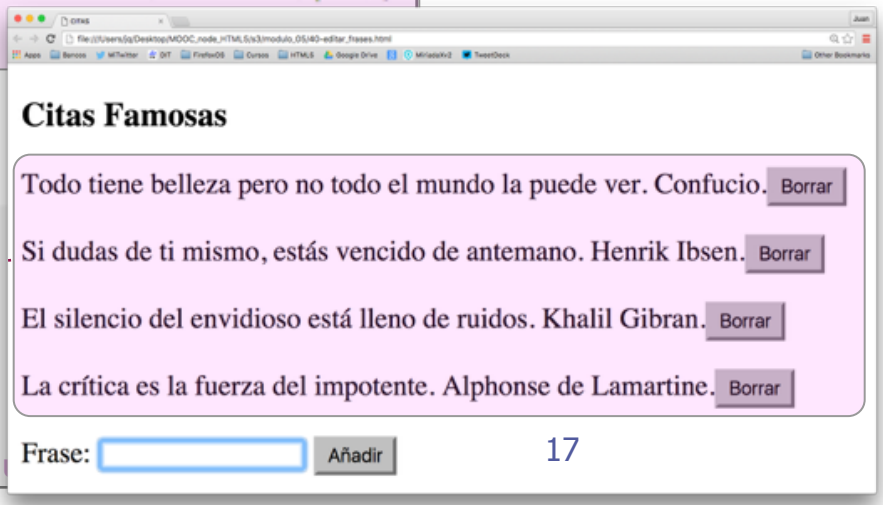
- Este ejemplo utiliza un array para gestionar citas
  - Una cita es una frase de un autor conocido
- La aplicación ilustra
  - **splice(..)** para borrar las citas existentes
  - **push(..)** para añadir nuevas citas al fina



Citas

```
<!doctype html> <html> <head><meta charset="utf-8"/>
<script type="text/javascript" src="jquery-2.1.4.min.js" ></script>
<script>
var vista, frase_nueva;

var frases = [
  "Todo tiene belleza pero no todo el mundo la puede ver. Confucio.",
  "Si dudas de ti mismo, estás vencido de antemano. Henrik Ibsen.",
  "El silencio del envidioso está lleno de ruidos. Khalil Gibran.",
  "La crítica es la fuerza del impotente. Alphonse de Lamartine."
]

$(function (){
  vista = $("#vista");
  frase_nueva = $("#frase_nueva");
  generar_vista();

  $("#b1").on("click", function() {
    frases.push(frase_nueva.val());
    generar_vista();
  });
});

function generar_vista(){
  vista.html("");
  frases.forEach(function(elem,i){
    vista.append("<p>"+frases[i]+"<button onclick='borrar("+i+")'>"+"Borrar</button></p>");
  });
};

function borrar(i){ frases.splice(i,1); generar_vista(); }

</script>
</head> <body>
  <h3>Citas Famosas</h3>

  <div id="vista"> </div>

  Frase: <input type="text" id="frase_nueva" />
  <button id="b1"> Añadir </button>
</body> </html>
```

- La aplicación permite
  - **Añadir** citas al final con **push(..)**
  - **Borrar** las citas nuevas con **splice(..)**



Citas

```
<!doctype html> <html> <head><meta charset="utf-8"/>
<script type="text/javascript" src="jquery-2.1.4.min.js" ></script>
<script>
var vista, frase_nueva;

var frases = [
  "Todo tiene belleza pero no todo el mundo la puede ver. Confucio.",
  "Si dudas de ti mismo, estás vencido de antemano. Henrik Ibsen.",
  "El silencio del envidioso está lleno de ruidos. Khalil Gibran.",
  "La crítica es la fuerza del impotente. Alphonse de Lamartine."
]

$(function (){
  vista = $("#vista");
  frase_nueva = $("#frase_nueva");
  generar_vista();

  $("#b1").on("click", function() {
    frases.push(frase_nueva.val());
    generar_vista();
  });
});

function generar_vista(){
  vista.html("");
  frases.forEach(function(elem,i){
    vista.append("<p>"+frases[i]+"<button onclick='borrar("+i+")'>"+"Borrar</button></p>");
  });
};

function borrar(i){ frases.splice(i,1); generar_vista(); }

</script>
</head> <body>
  <h3>Citas Famosas</h3>

  <div id="vista"> </div>

  Frase: <input type="text" id="frase_nueva" />
  <button id="b1"> Añadir </button>
</body> </html>
```

El **array** actúa de **contenedor de citas.** Permite **añadir** o **borrar** citas.
OJO! Los cambios se pierden al salir de la página Web.

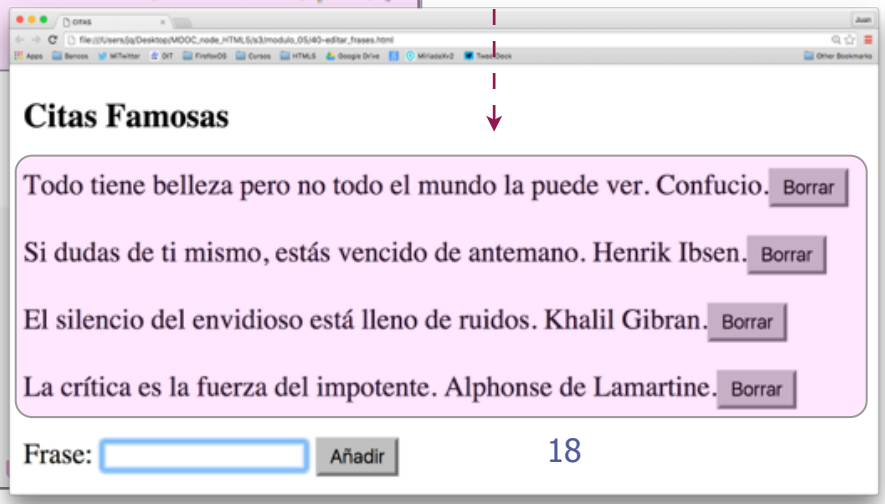
Inicia variables, genera el HTML de las citas y programa evento Añadir.

push() se usa en el manejador del evento para **añadir** un elemento al **final del array**.

La función generar_vista() genera dinámicamente el HTML, que muestra cada **cita con su botón de borrar,** utilizando un bucle que lo inserta como el HTML interno del elemento: ** .**

Hacer **click** en el botón **borrar** invoca **borrar(i)** que elimina la cita **i** del array.

borrar(i) elimina el elemento **i** de **frases** con **splice(i,1).**



JavaScript
Objetos y arrays: ejemplo citas

Objetos y Arrays

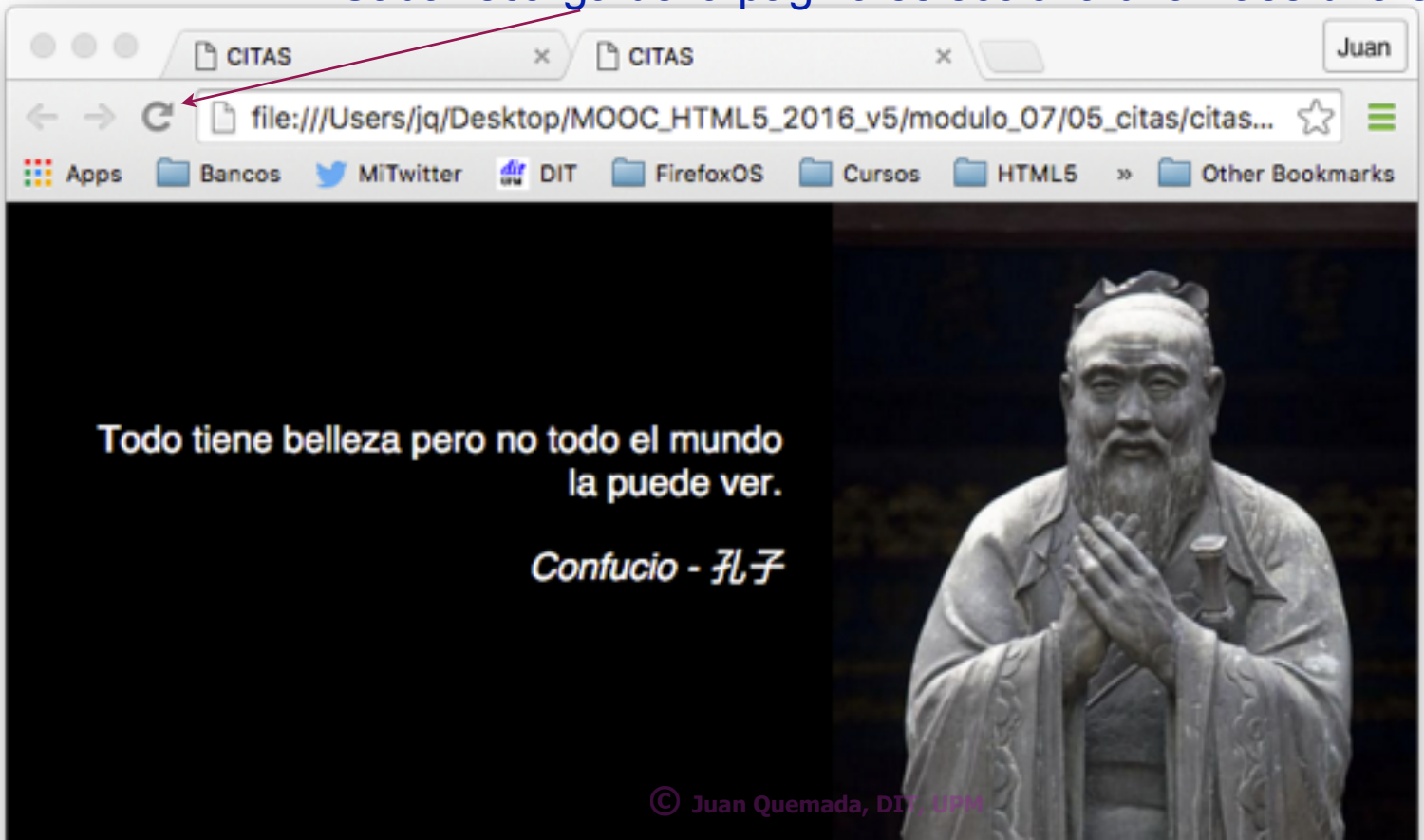
- **Objetos** y **arrays** permiten definir y gestionar recursos de datos
  - De forma que resulten muy sencillos de utilizar en programas, bases de datos, …
- Un **objeto** puede definir la **estructura** de los datos de un recurso
  - Los objetos se suelen construir con el **literal de objeto**:
    - **{ propiedad_1:valor_1, …….., propiedad_n:valor_n }**
      - Cada propiedad guarda un valor, igual que una variable
  - Por ejemplo, los objetos nos permiten dar el título y el director de películas favoritas

```
var et           = { titulo: "E.T.",          director: "Steven Spielberg" };
var star_wars_IV = { titulo: "Star Wars",     director: "George Lucas" };
var psicosis     = { titulo: "Psicosis",      director: "Alfred Hitchcock" };
var placido      = { titulo: "Placido",       director: "Luis García Berlanga" };
…….
```

- Un **array** define una **colección indexable** de recursos
  - Un array se suele construir con el **literal de array**:
    - **[recurso_1, recurso_2, …, recurso_n]**
  - Por ejemplo, podemos agrupar los objetos anteriores en un array de películas favoritas
    - **var mis_peliculas_favoritas = [psicosis, et, placido, star_wars_IV, …..];**
      - donde los elementos de este array son las referencias a las variables que contienen los objetos

Frase del día

- Este ejemplo muestra una cita seleccionada al azar
  - Una cita es una frase de un autor conocido
    - Cada recarga de la página selecciona una frase diferente



Frase del día

```
<!doctype html><html><head><meta charset="utf-8"><title>CITAS</title>
<link rel="stylesheet" type="text/css" href="citas.css" />
<script type="text/javascript" src="jquery-2.1.4.min.js" ></script>
<script>

var galeria = [
  { persona:"Confucio - 孔子",
    frase: "Todo tiene belleza pero no todo el mundo la puede ver.",
    foto:"http://3.bp.blogspot.com/-VlsuMSoivLU/VeMlD-
      LymUI/AAAAAAABKPU/Q8PYwsFbqxg/s1600/confucio.jpeg"
  },
  { persona:"Henrik Ibsen - hɛnɪɪk 'jo:hɑn 'ɪpsən",
    frase:"Si dudas de ti mismo, estás vencido de antemano.",
    foto:"https://ebooks.adelaide.edu.
      au/i/ibsen/henrik/gosse/images/bust2.jpg"
  },
  .....
];

$(function (){
  var i = Math.floor(Math.random()*galeria.length);

  $("#persona").html(galeria[i].persona);
  $("#frase").html(galeria[i].frase);
  $("#foto").attr("src", galeria[i].foto);
});
</script>
</head><body>

  <div id="caja">
    <img id="foto" alt="" />
    <div class="textos">
      <p id="frase"></p>
      <p id="persona"></p>
    </div>
  </div>
</body>
</html>
```

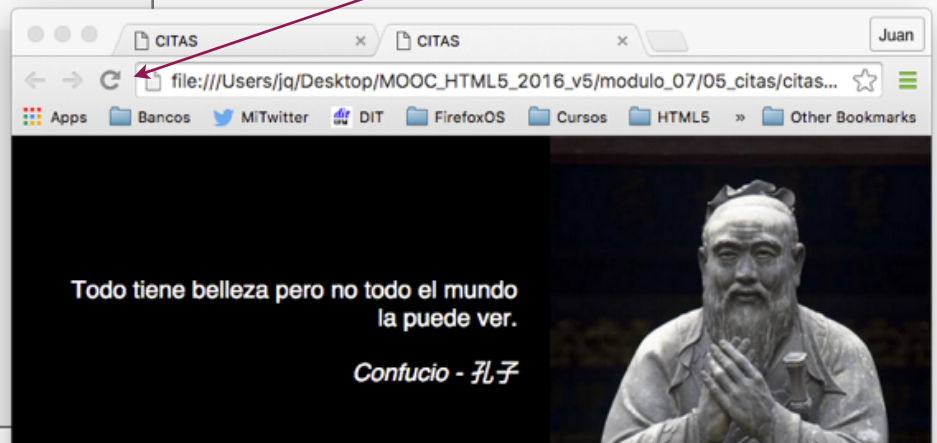
Base de datos de frases (variable galeria): array de objetos

Índice aleatorio **i** selecciona una frase de la galería

Visualiza frase, autor y foto en página Web

Elementos HTML donde visualizar frase, autor y foto

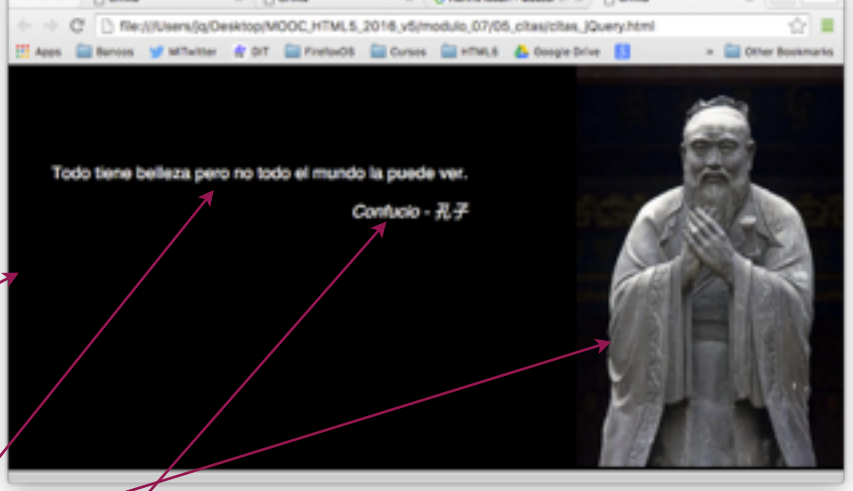
- El ejemplo muestra al cargar la página
  - Una frase de un autor conocido
    - Seleccionada al azar
      - Cada recarga de la página selecciona una frase diferente
- Las frases se guardan en la **galería**
  - La galería es un **array de objetos**
    - guardados en la **variable galería**
- Cada objeto guarda los elementos
  - El **nombre** del autor (español y original)
  - Una **frase conocida** suya
  - El **URL absoluto a una foto en Internet**
- El **evento onload** muestra la frase
  - al invocar la función **seleccion()**
    - que elige una **frase al azar** y la muestra junto con el **nombre** y su **foto**
  - El **evento onload ocurre** cuando
    - y el árbol DOM **está ya construido**
  - Para **seleccionar otra frase**
    - **Recargar la página**



Frase del día:estilo CSS

```css
body {
    margin:0;
    padding:0;
    background-color:#ccc;
}
#caja {
    position:relative;
    width:100%;
    height:auto;
    background-color:#000;
    text-align:right;
}
#foto{
    height:427px;
    width:auto;
}
.textos {
    position: absolute;
    top: 20%;
    left: 5%;
    width: 50%;
    color: #FFFFFF;
    font-family: Helvetica, Arial, sans-serif;
    font-size: large;
}
#persona {
    font-style: italic;
}
```

```html
<div id="caja">
  <img id="foto" alt="" />
  <div class="textos">
    <p id="frase"></p>
    <p id="persona"></p>
  </div>
</div>
```

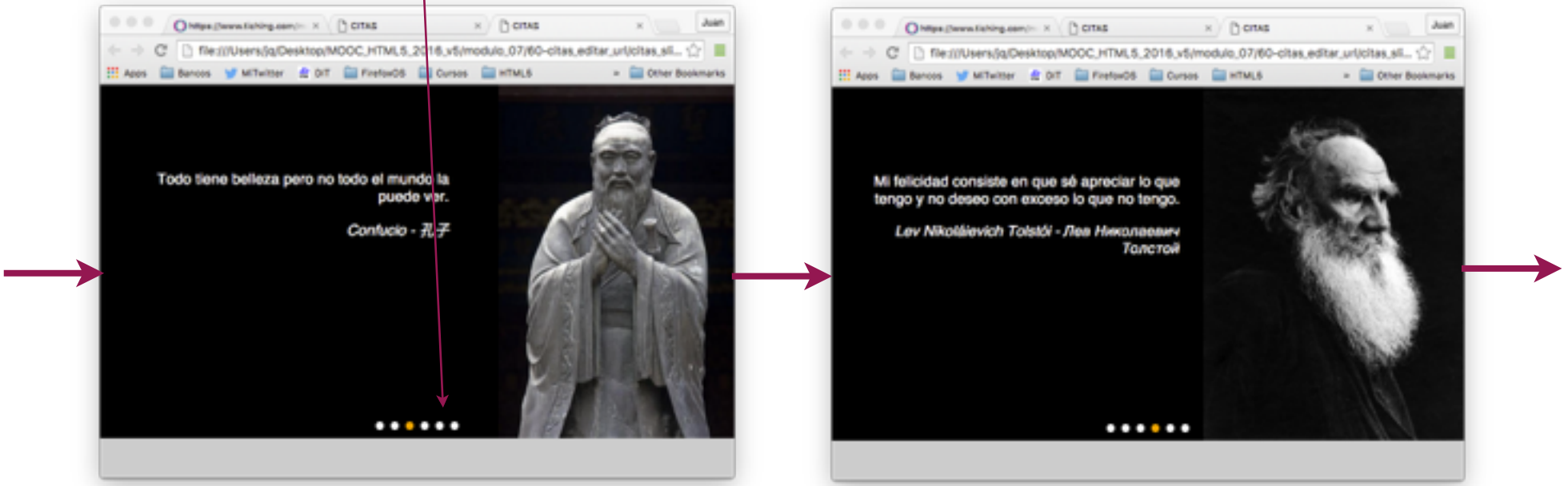


- **body**
  - se elimina margin y padding
  - fondo gris
- **#caja**
  - Posición relativa, permite posicionamiento absoluto en .textos y #persona
  - Caja ocupa todo el ancho
  - Altura automática
  - fondo negro
- **#foto**
  - Altura fija: 427 pixels
  - Anchura automática
- **.textos**
  - Posición absoluta respecto a la posición relativa de la caja
  - Font grande y en color blanco sobre fondo negro de la foto
- **#persona**
  - Párrafo en itálica, después de .textos

JavaScript:
Carrusel con citas

Carrusel con citas

- Este ejemplo mostramos un carrusel de citas
  - Cada 2 segundos se pasa a la siguiente cita
- La botonera inferior muestra la cita que se está mostrando
  - Se puede cambiar la cita clicando en la botonera



Carrusel de citas

```
<!doctype html><html>  <head><meta charset="utf-8"><title>CITAS</title>
<link rel="stylesheet" type="text/css" href="css/citas_slide.css" />

<script type="text/javascript" src="javascript/jquery-2.1.4.min.js" ></scr
<script type="text/javascript" src="javascript/galeria.js" ></script>
<script>
var t;

$(function (){
  galeria.forEach(function(elem,i) {
    $("ul#selector").append("<li><a onClick='select("+i+")'></a></li>");
  });

  select(0);
});

function select(i){
  $("nav a")
    .removeClass("on off")
    .addClass(function(j){return(j===i)?"on":"off";});

  $("#persona").html(galeria[i].persona);
  $("#frase").html(galeria[i].frase);
  $("#foto").attr("src", galeria[i].foto);

  clearTimeout(t);
  t = setTimeout( function(){select((i + 1) % galeria.length);}, 2000);
}
</script>
</head>
```

```
<body>
<div class="contenido">
  <nav>
    <ul id="selector"></ul>
  </nav>
  <section id="caja">
    <img id="foto" alt="" />
    <div class="textos">
      <p id="frase"></p>
      <p id="persona"></p>
    </div>
  </section>
</div>
</body>
</html>
```

Carrusel de citas

```
<!doctype html><html>  <head><meta charset="utf-8"><title>CITAS</title>
<link rel="stylesheet" type="text/css" href="css/citas_slide.css" />

<script type="text/javascript" src="javascript/jquery-2.1.4.min.js" ></scr
<script type="text/javascript" src="javascript/galeria.js" ></script>
<script>
var t;

$(function (){
  galeria.forEach(function(elem,i) {
    $("ul#selector").append("<li><a onClick='select("+i+")'></a></li>");
  });

  select(0);
});

function select(i){
  $("nav a")
    .removeClass("on off")
    .addClass(function(j){return(j===i)?"on":"off";});

  $("#persona").html(galeria[i].persona);
  $("#frase").html(galeria[i].frase);
  $("#foto").attr("src", galeria[i].foto);

  clearTimeout(t);
  t = setTimeout( function(){select((i + 1) % galeria.length);}, 2000);
}
</script>
</head>
```

```
<body>
<div class="contenido">
  <nav>
    <ul id="selector"></ul>
  </nav>
  <section id="caja">
    <img id="foto" alt="" />
    <div class="textos">
      <p id="frase"></p>
      <p id="persona"></p>
    </div>
  </section>
</div>
</body>
</html>
```

La galería se saca a un fichero JS exterior (mismo código)

Botonera con HTML se genera dinámicamente

El evento de selección de cita es un atributo lleva el índice **i** a la cita en el array.

Mostrar primera cita al comenzar.

Colorear en marrón el botón de la cita mostrada

Mostrar los 3 componentes de la cita: frase, autor y foto.

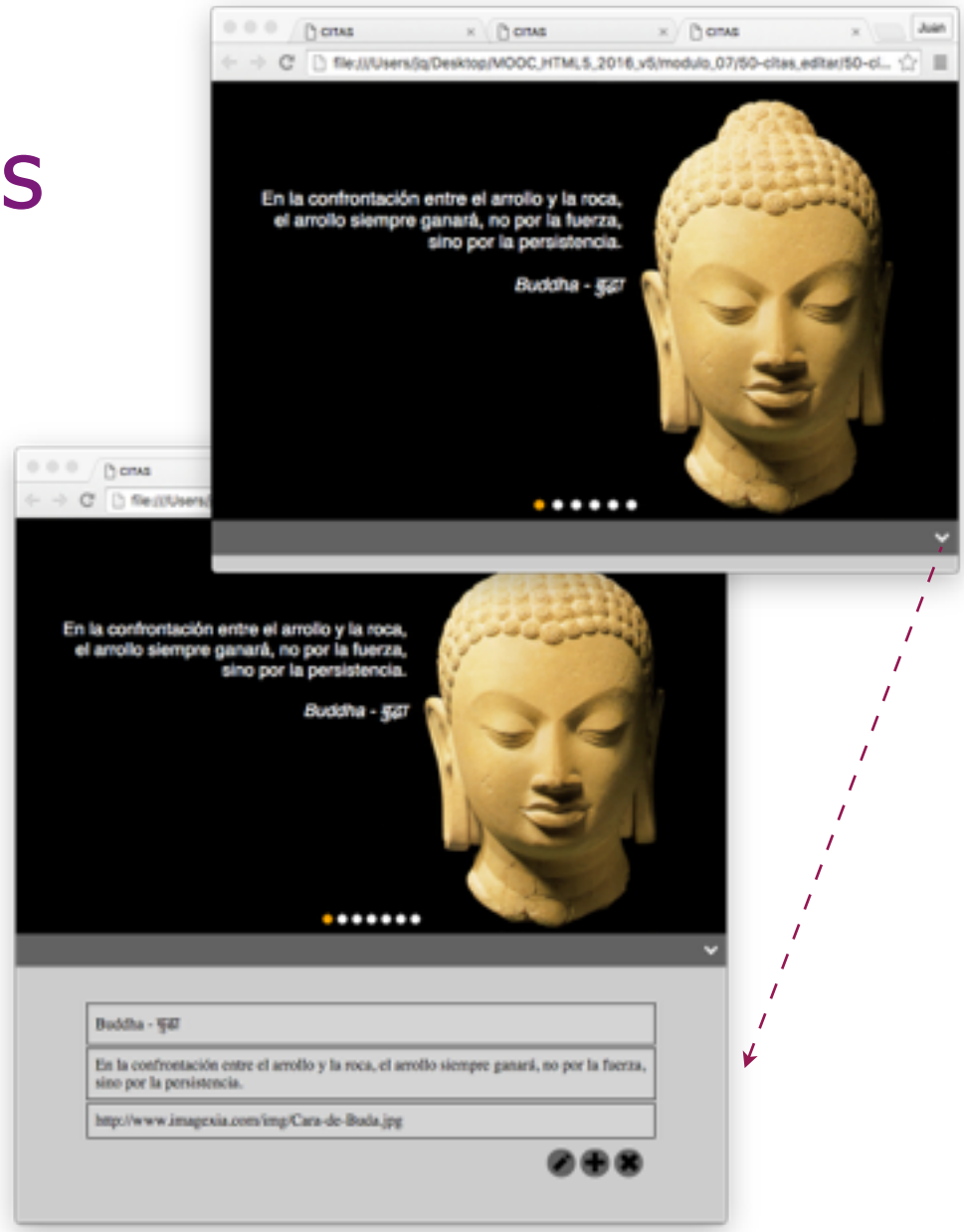
Borrar temporizador anterior, por si el anterior no ha vencido todavía (al hacer click en botonera).

Crear nuevo temporizador que muestre la siguiente cita. El manejado del evento invoca la función select con siguiente **i**.

JavaScript:
Editar carrusel con citas

Editar carrusel con citas

- Se añade un desplegable para
  - Añadir, modificar o borrar citas
- Los cajetines del desplegable
  - se rellenan
    - con el contenido de los campos del objeto mostrado
- Al abrir el desplegable se para el carrusel
  - En la imagen mostrada



Editar carrusel de citas

```
<body>
<div class="contenido">
  <nav>
    <ul id="selector"></ul>
  </nav>
  <section id="caja">
    <img id="foto" alt="" />
    <div class="textos">
      <p id="frase"></p>
      <p id="persona"></p>
    </div>
  </section>
  <div class="editar" id="editar">
    <img src="images/carat-d-white.svg" alt=""/>
  </div>
  <section id="datos">
    <div contentEditable="true" id="persona_d"></div>
    <div contentEditable="true" id="frase_d"></div>
    <div contentEditable="true" id="foto_d"></div>
    <div id="botones">
      <ul>
        <li><img src="images/edit-black.svg" alt="" id="guardar"/></li>
        <li><img src="images/plus-black.svg" alt="" id="nuevo"/></li>
        <li><img src="images/delete-black.svg" alt="" id="borrar"/></li>
      </ul>
    </div>
  </section>
  </div>
</body>
</html>
```

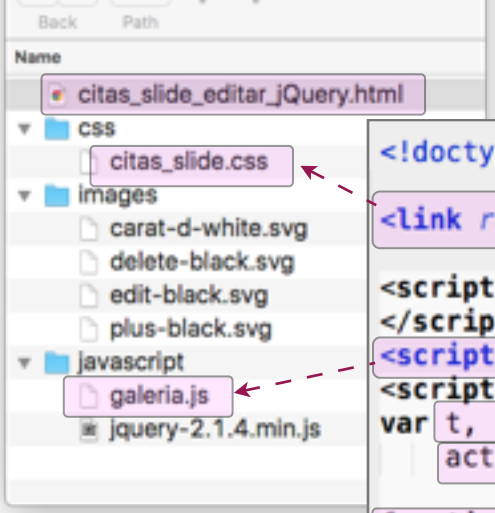
Presentación de la cita y la botonera del ejemplo anterior.

Botón para desplegar edición (en SVG)

Cajetines definidos con bloques con atributo "contentEditable"

Botones de editar, añadir y borrar (en SVG)

Editar el carrusel de citas




```
<!doctype html><html><head><meta charset="utf-8"><title>CITAS</title>

<link rel="stylesheet" type="text/css" href="css/citas_slide.css" />

<script type="text/javascript" src="javascript/jquery-2.1.4.min.js" >
</script>
<script type="text/javascript" src="javascript/galeria.js" ></script>
<script>
var t,          // temporizador en marcha
    actual;     // índice a cita mostrada

function select(i){
  ..............      // Muestra la cita i en el navegador
}

function generar_selector(){
  ..............      // genera HTML de botonera para galería actual
}

$(function (){
  generar_selector();

  $("#editar").on("click", function(){
  ..............      // manejador evento que despliega edición
  })


  $("#nuevo").on("click", function(){
  ..............      // manejador evento que añade nueva cita
  })

  select(0);          // Muestar cita 0 en navegador al comenzar
});
</script>
</head>
<body> ........ </body>
</html>
```

Editar el carrusel de citas

```
<script>
var t, actual;

function select(i){
  actual = i;

  $("nav a")
    .removeClass("on off")
    .addClass(function(j){return(j===i)?"on":"off";});

  $("#persona").html(galeria[i].persona);
  $("#frase").html(galeria[i].frase);
  $("#foto").attr("src", galeria[i].foto);

  clearTimeout(t);
  t = setTimeout( function(){select((i + 1) % galeria.length);}, 2000);
}

function generar_selector(){ // regenera la botonera
  var selector = $("#selector");

  selector.html("");

  galeria.forEach(function(elem,i) {
    selector.append("<li><a onClick='select("+i+")'></a></li>");
  });
}
```

La variable actual contiene el índice a la cita mostrada. Esta en el ámbito exterior y puede ser utilizada en cualquier función o manejador.

Guarda objeto jQuery en una variable para mayor eficiencia. Se evitan muchas búsquedas en árbol DOM.

Borra el selector anterior, porque ahora hay que eliminar el actual y generar un selector nuevo.

Función que genera el selector de citas con JavaScript.

Inicialización de la app Web

```
$(function (){
  generar_selector();

  $("#editar").on("click", function(){
    clearTimeout(t);

    $("#persona_d").html(galeria[actual].persona);
    $("#frase_d").html(galeria[actual].frase);
    $("#foto_d").html(galeria[actual].foto);

    $("#datos").css("display", "block");
  })

  $("#nuevo").on("click", function(){
    $("#datos").css("display", "none");

    actual = galeria.push({
       persona:$("#persona_d").html(),
       frase:$("#frase_d").html(),
       foto:$("#foto_d").html()
    }) - 1;

    generar_selector();

    select(actual);
  })

  select(0);
});
```

Generar html del selector de citas con JavaScript.

Para temporizador

Muestra elementos de la cita en cajetines.

Muestra bloques.

Evento para despliegue de los cajetines de edición de citas.

Oculta cajetines

Incluye nueva cita como último elem. de galeria. Guarda índice en actual.

Evento para crear una nueva cita.

Regenera la botonera de selectores de citas para incluir el elemento nuevo.

Muestra nuevo elemento (índice: actual).

Arrancar con la primera cita (índice: 0).

Final del tema

Muchas gracias!