VÍCTOR CUSTODIO

JQuery

- JQuery es una librería de JavaScript que simplifica la programación de páginas Web.
- Creado en 2006.
- Versione actuales (Febrero 2017):
  - 1.12.4
  - 2.2.4 (la línea de versiones jQuery 2.x tiene el mismo API que la linea 1.x pero no soporta Internet Explorer 6, 7 y 8).
  - 3.1.1 más ligera y rápida. Las versiones 1.X y 2.X ya no avanzan, solo se parchean algunos problemas.
- Características:
  - Simplifica la programación y reduce el número de líneas de código.
  - Permite interactuar con los contenidos de los documentos HTML y CSS.
  - Simplifica la captura y programación de eventos.
  - Permite desarrollar animaciones.
  - Soporta AJAX.
- Es software libre y de código abierto (licencias GPL y MIT).
- No es la única alternativa (MooTools, Prototype, Yahoo! UI Library, Modernizr, Dojo…) pero sí la más utilizada

HOLA MUNDO JQUERY

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<script src="./js/jquery-3.1.1.js"></script>
<script>
$(document).ready(function(){
$("#divsaludo").html("Hola mundo");
});
</script>
<title>Hola mundo, version JQuery</title>
</head>
<body>
<div id="divsaludo"></div>
</body>
</html>
```

UTILIZANDO JQUERY

- Existen dos alternativas para incluir la librería JQuery en un proyecto:
  - Descargando la librería y almacenándola en el servidor Web como un fichero JavaScript.
    - Ventaja: No necesita conexión a Internet (habitual en apps. para móviles). <script src="./js/jquery-3.1.1.js"> </script>
  - Utilizando una CDN (Content Delivery Network).
    - Ventajas:
      - Minimiza la carga de trabajo de nuestro servidor.
      - Agiliza la carga de la página.
      - <script src="http://code.jquery.com/jquery-3.1.1.js"> </script>

La FUNCIÓN JQUERY()/$()

- La función JQuery es la más importante de la librería JQuery.
- Permite seleccionar uno o más elementos de la página.
- Expresiones equivalentes para la función JQuery:
  - JQuery
  - $
- Sintaxis:
  - $(*selector*)
  - $(*selector, contexto*)
- Ejemplo:

//bloque1 es el id de un div, dentro del cual hay otro div perteneciente a la clase *divsaludo*

```
$(document).ready(function(){
 ¡Query(".divsaludo", "#bloque1").html("Hola, mundo");
});
```

Selectores básicos

- Selectores básicos:
- Selector universal. Selecciona todos los elementos del documento:
  - jQuery(*)
- Selector de tipo. Selecciona todos los elementos de un tipo:
  - jQuery("h1")
- Selector de clase. Selecciona todos los elementos de una clase:
  - jQuery(".clase")
- Selector de identificador. Selecciona el elemento con el id indicado:
  - jQuery("#identificador")
- Selector de tipo y clase. Selecciona todos los elementos de una clase dentro de un tipo:
  - jQuery("div.clase")
- Selector de tipo y de identificador. Selecciona el elemento con el identificador indicado dentro de un tipo:
  - jQuery("div#clase")
- Selector de grupo. Permite combinar selectores separándolos por comas:
  - jQuery("h1, h2,title")

Selectores de formularios

- jQuery(“:text”) en lugar de “[type=text]”
- jQuery(“:password”)
- jQuery(“:input”)
- jQuery(“:checkbox”)
- jQuery(“:checked”) //Elementos seleccionados
- jQuery(“:radio”)
- jQuery(“:selected”) //Elementos seleccionados
- jQuery(“:submit”)
- jQuery(“:button”)
- jQuery(“:reset”)
- jQuery(“:disabled”)//Elementos desactivados
- jQuery(“:enabled”)//Elementos activados
- jQuery(“:focus”)//Elemento activo en el momento de la ejecución

Selectores de atributos

- Selector de atributo básico. Selecciona los elementos que tengan el atributo indicado:
  - jQuery(“[align]”)
  - jQuery("div[align]“)
- Selector de atributo con determinado valor:
  - jQuery("div[align='center']")
- Selector de atributo con determinado valor distinto:
  - jQuery("div[align!='center']")
- Selector de atributo a partir de la evaluación de cadenas:
- jQuery("a[href^='http://www']") //Empiezan por la cadena
- jQuery("a[href$='.com']") //Terminan por la cadena
- jQuery("a[href*=‘google']") //Contienen la cadena
- Selector de atributos combinados:
- jQuery("[align][lang]")

Más Selectores

- Otros selectores:
- Básicos:
  - http://api.jquery.com/category/selectors/basic-filter-selectors/
- De contenido:
  - http://api.jquery.com/category/selectors/content-filter-selector/
- De jerarquía:
  - http://api.jquery.com/category/selectors/jquery-selector-extensions/
- Child-filter:
  - http://api.jquery.com/category/selectors/child-filter-selectors/
- De visibilidad:
  - jQuery(:hidden)
  - jQuery(:visible)
- De objetos del navegador:
  - jQuery(window) //selecciona la ventana del navegador
  - jQuery(document) //selecciona el documento cargado en la ventana
- API SELECTORES: http://api.jquery.com/category/selector

Eventos

Eventos

- La asociación de eventos a componentes en JQuery se realiza a través de la función JQuery:
  - $(*selector*).*evento*(*función*(){…})
- Ejemplo:
  - $(document).ready(function(){
    $("#miboton").click(function() { alert("Buenos días"); }) });
- $(*selector*).*evento*(*función*(e){…})
- Ejemplo:
  - $(document).**ready(function(e){**
    e.preventDefault();//Desactiva la respuesta por defecto del evento
    $("#miboton").click(function() { alert("Buenos días"); }) });
- http://api.jquery.com/category/events/

Eventos

- Eventos más utilizados:
  - blur()
  - change()
  - click()
  - dblclick()
  - error()
  - focus()
  - focusin()
  - focusout()
  - hover()
  - keydown()
  - keypress()
  - keyup()

Eventos

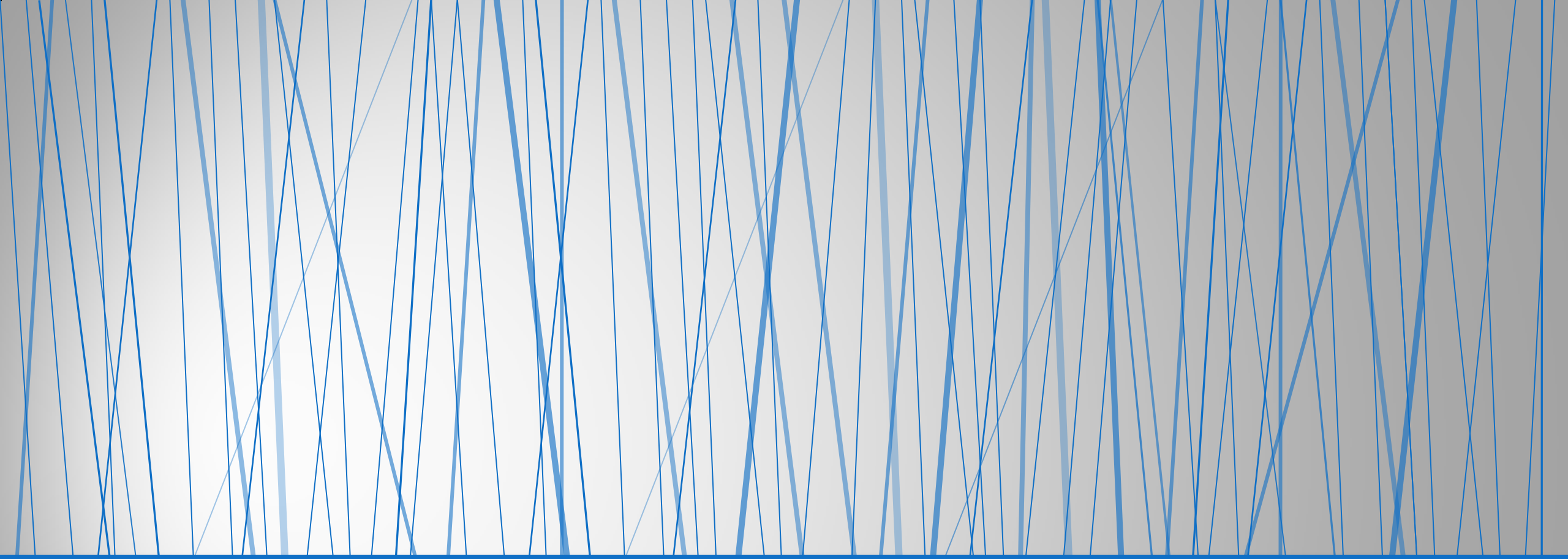
- Eventos más utilizados:
  - mousedown()
  - mouseenter()
  - mouseleave()
  - mousemove()
  - mouseout()
  - mouseover()
  - mouseup()
  - ready()

Eventos

- Otra forma de trabajar con eventos:
  - "bind" antes de la versión 1.7 y "on" ahora

    $(*selector*).on(*event,childSelector,data,function,map*)
    - **event-** Obligatorio, especifica el evento o eventos(separados por espacios.
    - childSelector – Opcional, especifica un filtro para los hijos del selector
    - Data- opcional, especficia datos adicionales a pasar a la función
    - **Function-** Obligatorio, especifica la función a ejecutar cuando ocurra el evento
    - Map – Especifica un mapa de eventos, con uno o varios eventos y funciones a ejecutar cuando ocurra el evento({event:function,event:function,…})

  - http://api.jquery.com/on/



JQUERY HTML

¡Query DOM Manipulación

- Una parte muy importante de ¡Query es la posibilidad de manipular el DOM.
- ¡Query viene con un montón de métodos DOM relacionados que hacen que sea fácil de acceder y manipular los elementos y atributos
  - Get
  - Set
  - Add
  - Remove
  - CSS Classes
  - Dimensiones

JQUERY HTML- GET

- Cuatro simples, pero útiles, métodos de ¡Query para la manipulación de DOM son:
  - text () - Establece o devuelve el contenido de texto de los elementos seleccionados
  - html () - Establece o devuelve el contenido de elementos seleccionados (incluyendo el formato HTML)
  - val () - Establece o devuelve el valor de los campos del formulario
  - attr () método se utiliza para obtener o establecer el valor de los atributos de una etiqueta seleccionada

JQUERY HTML- GET

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    alert("Text: " + $("#test").text());
  });
  $("#btn2").click(function(){
    alert("HTML: " + $("#test").html());
  });
});
</script>
</head>

<body>
<p id="test">This is some <b>bold</b> text in a paragraph.</p>
<button id="btn1">Show Text</button>
<button id="btn2">Show HTML</button>
</body>
</html>
```

JQUERY HTML- SET

- Los mismos 4 mismos métodos, pero pasando parámetros a insertar

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
  $("#test3").val("Dolly Duck");
});
$("button").click(function(){
  $("#w3s").attr("href",“www.google.es");
});
```

JQUERY HTML- AÑADIR

- Añadir Nuevo Contenido HTML
  - append () – Inserta contenido al final del elemento seleccionado
  - prepend () – Insert contenido en el comienzo de los elementos seleccionados

JQUERY HTML - AÑADIR

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
function appendText()
{
var txt1="<p>Text.</p>";               // Create text with HTML
var txt2=$("<p></p>").text("Text.");  // Create text with jQuery
var txt3=document.createElement("p");
txt3.innerHTML="Text.";                // Create text with DOM
$("body").append(txt1,txt2,txt3);        // Append new elements
}
</script>
</head>
<body>

<p>This is a paragraph.</p>
<button onclick="appendText()">Append text</button>

</body>
</html>
```

JQUERY-HTML - ELIMINAR

- Para eliminar elementos y contenido, hay principalmente dos métodos de ¡Query:
  - remove () - Elimina el elemento seleccionado (y sus elementos secundarios)
    - Se le puede añadir filtros pasados por parámetros(como un selector)
  - empty () - Elimina los elementos secundarios del elemento seleccionado

JQUERY-HTML - ELIMINAR

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").empty(".italic");
  });
});
</script>
</head>
<body>
<p>This is a paragraph in the div.</p>
<p class="italic"><i>This is another paragraph in the div.</i></p>
<p class="italic"><i>This is another paragraph in the div.</i></p>
<button>Remove all p elements with class="italic"</button>
</body>
</html>
```

JQUERY-HTML – Manipulando CSS

- ¡Query tiene varios métodos para la manipulación de CSS. Veremos los métodos siguientes:
  - addClass () - Añade una o más clases a los elementos seleccionados
  - removeClass () - Elimina una o más clases de los elementos seleccionados
  - toggleClass () - Cambia entre añadir / eliminar las clases de los elementos seleccionados
  - css () - Establece o devuelve el atributo de estilo

JQUERY-HTML – Manipulando CSS

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("h1,h2,p").toggleClass("blue");
  });
});
</script>
<style>
.blue
{
color:blue;
}
</style>
</head>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Toggle class</button>
</body>
</html>
```

JQUERY-HTML – Manipulando CSS

- css(“propertyName”) para recuperar valor
- css(“propertyName”, “value”); para insertar valor
- Para Multiples valores (map)
  - css({"*propertyname*":"*value*","*propertyname*":"*value*",...});

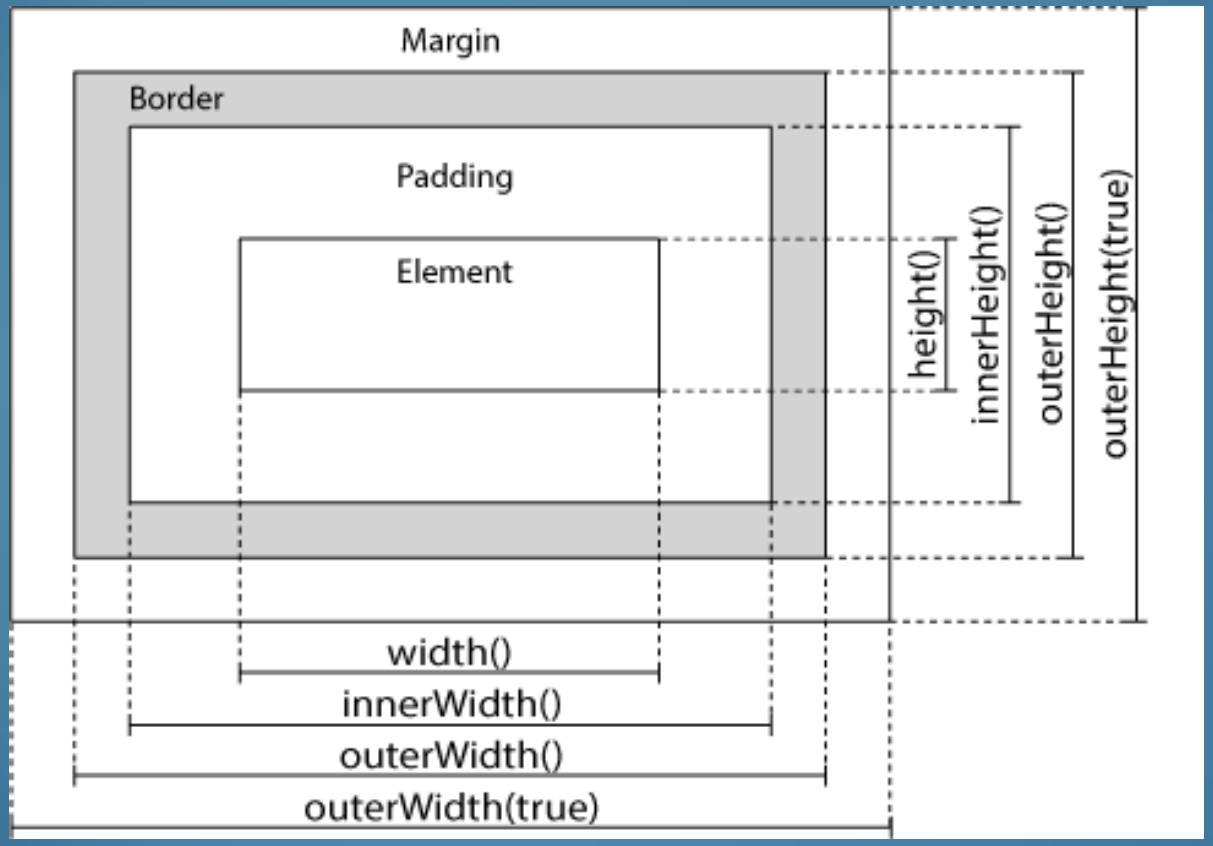
JQUERY-HTML – Manipulando CSS

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").css({"background-color":"yellow","font-size":"200%"});
  });
});
</script>
</head>

<body>
<h2>This is a heading</h2>
<p style="background-color:#ff0000">This is a paragraph.</p>
<p style="background-color:#00ff00">This is a paragraph.</p>
<p style="background-color:#0000ff">This is a paragraph.</p>
<p>This is a paragraph.</p>
<button>Set multiple styles for p</button>
</body>
</html>
```

JQUERY-HTML – Manipulando CSS

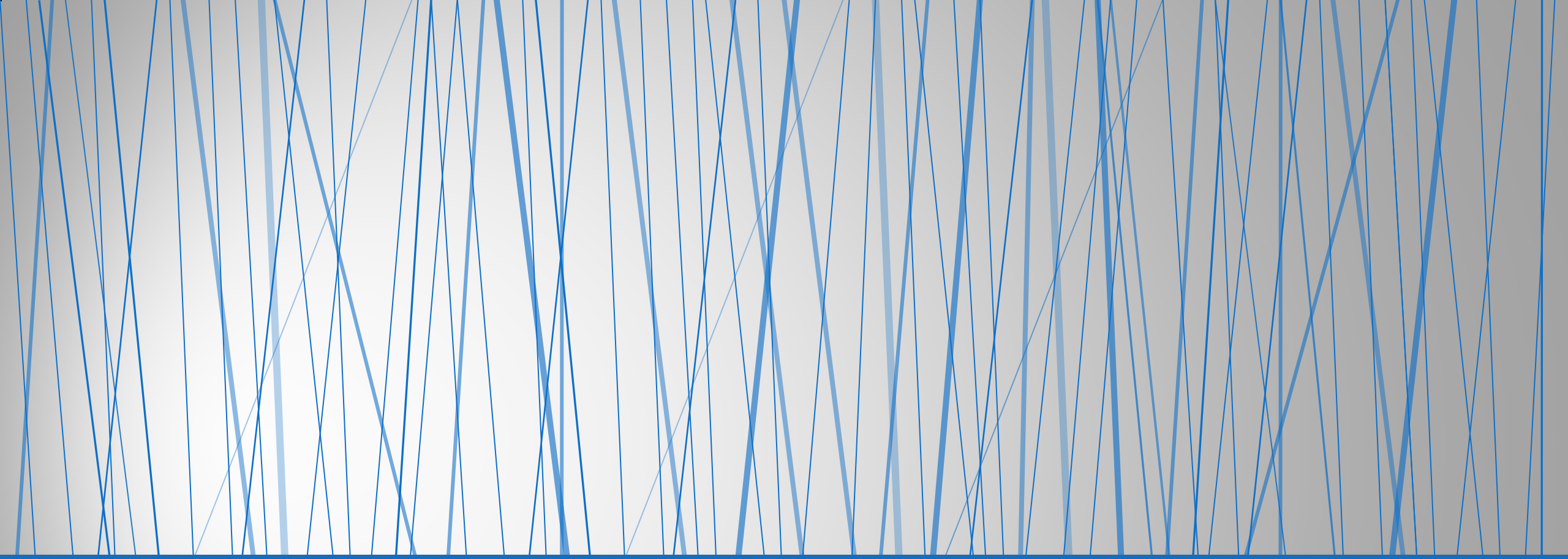
```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>css demo</title>
  <style>
  div {
    width: 60px;
    height: 60px;
    margin: 5px;
    float: left;
  }
  </style>
  <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<span id="result"> </span>
<div style="background-color:blue;"></div>
<div style="background-color:rgb(15,99,30);"></div>
<div style="background-color:#123456;"></div>
<div style="background-color:#f11;"></div>

<script>
$( "div" ).click(function() {
  var color = $( this ).css( "background-color" );
  $( "#result" ).html( "That div is <span style='color:" +
    color + ";'>" + color + "</span>." );
});
</script>

</body>
```

JQUERY-HTML – Dimensiones



JQUERY-HTML – Dimensiones

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    var txt="";
    txt+="Width of div: " + $("#div1").width() + "</br>";
    txt+="Height of div: " + $("#div1").height() + "</br>";
    txt+="Outer width of div: " + $("#div1").outerWidth() + "</br>";
    txt+="Outer height of div: " + $("#div1").outerHeight();
    $("#div1").html(txt);
  });
});
</script>
</head>
<body>
<div id="div1" style="height:100px;width:300px;padding:10px;margin:3px;border:1px solid blue;background-
color:lightblue;"></div><br>
<button>Display dimensions of div</button>
<p>outerWidth() - returns the width of an element (includes padding and border).</p>
<p>outerHeight() - returns the height of an element (includes padding and border).</p>
</body>
</html>
```



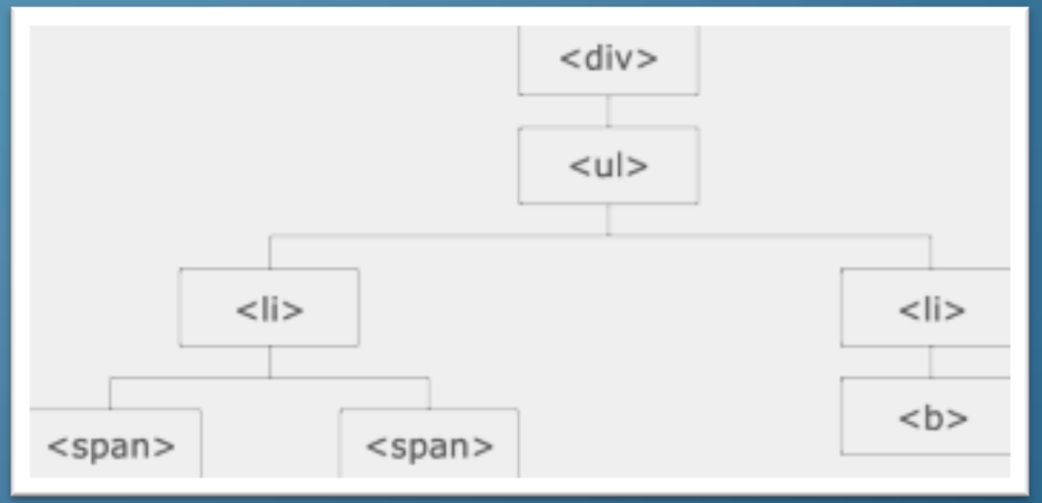
JQUERY Traversing

¿Que es traversing?

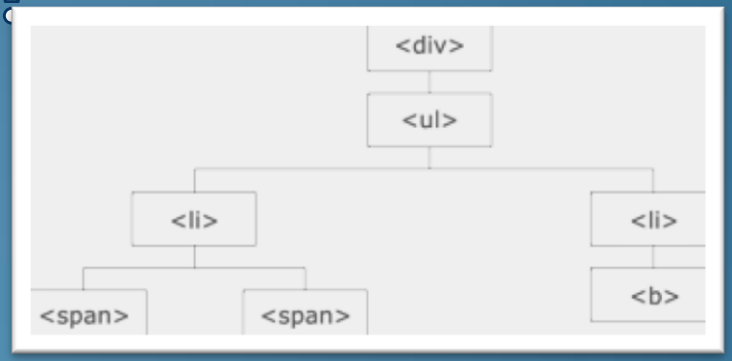
- jQuery Traversing, que significa "moverse a través de", se utilizan para "encontrar" (o seleccionar) elementos HTML en función de su relación con otros elementos. Se Comienza con una selección y pasar a través de esa selección hasta llegar a los elementos que deseamos.

¿Que es traversing?

- La imagen siguiente muestra un árbol genealógico. Con traversing jQuery , podemos movernos fácilmente hacia arriba (ancestors), abajo (descendants) y de lado (siblings) en el árbol de familia, a partir del elemento seleccionado (actual). Este movimiento se llama Traversing - o en movimiento a través del DOM.



¿Que es traversing?



- es **parent** de <ul>, y **ancestor** de todo lo que hay dentro de el.
- es **parent** de ambos <li> y **child** de
- de la izquierda es **parent** de , **child** de <ul> y **descendant** de
- es **child** de <li> de la izquierda y **descendant** de <ul> y
- Los dos <li> son **siblings** (comparten el mismo padre)
- el <li> de la derecho es **parent** de <b>, **child** de <ul> y **descendant** de
- <b> es **child** de <li> derecha y **descendant** de <ul> y

Traversing -Ancestors

- parent() -> nos da el padre (parent)
- $(document).ready(function(){
  $("span").parent();
  });
- parents() -> nos da todos los ancestros
- parentsUntil() -> nos da todos los ancestros hasta el especificado

```
$(document).ready(function(){
  $("span").parents("ul");
});
```

Traversing -Descendants

- children() -> nos da los hijos directos
- find(*) -> nos da todos los descendientes

- Se les puede pasar por parámetro un filtro para que solo nos devuelvan los descendientes que lo cumplan.
- Ejemplo

```
$(document).ready(function(){
  $("div").find("span");
});
```

Traversing -Descendants

- Hay muchos métodos jQuery útiles para atravesar de lado en el árbol DOM:
  - siblings() -> los hermanos (con o sin filtro)
  - next() -> el siguiente
  - nextAll() -> todos los hermanos siguientes
  - nextUntil() -> todos los siguientes hasta…
  - prev()
  - prevAll()
  - prevUntil()

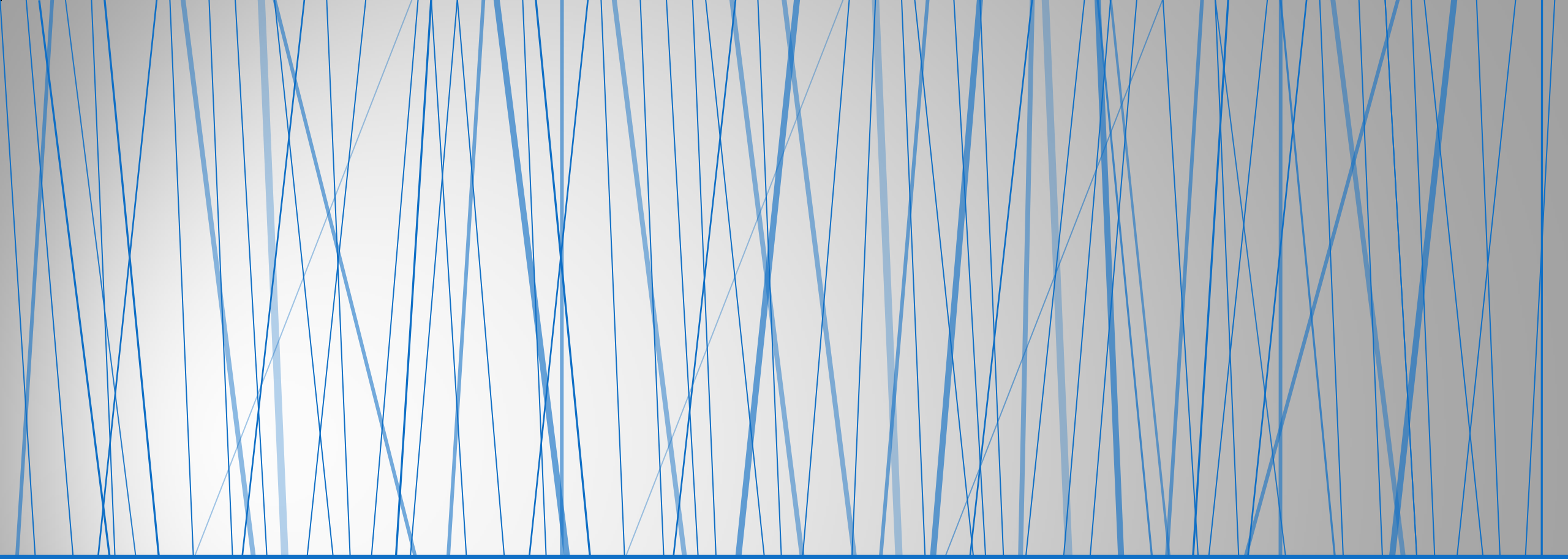
Traversing -Siblings

- Los tres métodos de filtrado más básicas son first(), last () y eq (index), que permiten seleccionar un elemento específico en función de su posición en un grupo de elementos.
  - Ejemplo

```
$(document).ready(function(){
  $("p").eq(0);                                    //  igual que $(“p”.first()
});
```

- Otros métodos de filtrado, como el filter () y not () permiten seleccionar los elementos que coinciden o no coinciden con un cierto criterio.
- Ejemplo

```
$(document).ready(function(){
  $("p").filter(".intro");                    //lo contrario sería $("p").not(".intro");
});
```



JQUERY EFFECTS

JQUERY - Ocultar y mostrar

- hide()
- show()
- toggle()

- Sintaxis : $(*selector*).hide o show o toggle (*speed*,*callback*);
  - Speed: el tiempo que tardará la acción en ms, o valor predefinido(“slow”)
  - Callback: función que se ejecutará al terminar la acción.

JQUERY - Ocultar y mostrar

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").toggle(1000);
  });
});
</script>
</head>
<body>

<button>Toggle</button>
<p>This is a paragraph with little content.</p>
<p>This is another small paragraph.</p>
</body>
</html>
```

JQUERY – Efecto desvanecimiento

- fadeIn() -> Aparición
- fadeOut() -> desaparición
- fadeToggle() -> ambos
- fadeTo() -> no se desvanece del todo, solo hasta el valor del parámetro opacity -> Sintaxys $(*selector*).fadeTo(*speed,opacity,callback*);
- Para el resto de fades la sintaxis es la común de los efectos en JQUERY:
  - Syntaxis $(*selector*).fadeXX(*speed,callback*);

JQUERY – Efecto desvanecimiento

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
    $("#div4").fadeTo("slow", 0.5);
  });
});
</script>
</head>
<body>
<p>Demonstrate fadeOut() with different parameters.</p>
<button>Click to fade out boxes</button>
<br><br>
<div id="div1" style="width:80px;height:80px;background-color:red;"></div><br>
<div id="div2" style="width:80px;height:80px;background-color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
<div id="div4" style="width:80px;height:80px;background-color:blue;"></div>
</body>
</html>
```

JQUERY – Efecto deslizado

- slideDown() -> deslizar hacia abajo
- slideUp() -> deslizar hacia arriba
- slideToggle() -> ambos

- Misma sintaxis que el resto

  $(*selector*).slideXXX(*speed*,*callback*);

JQUERY – Efecto deslizado

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideToggle("slow");
  });
});
</script>
 <style>
#panel,#flip{padding:5px; text-align:center; background-color:#e5eecc; border:solid 1px #c3c3c3; }
#panel{padding:50px;display:none;}
</style>
</head>
<body>
 <div id="flip">Click to slide the panel down or up</div>
<div id="panel">Hello world!</div>
</body>
</html>
```

JQUERY- ANIMACIONES

- Metodo animate()-> Similar al transition en CSS3
- Syntaxis : $(*selector*).animate({*params*},*speed*,*callback*);
  - El parámetro params requerido define las propiedades CSS para ser animadas.
  - El parámetro de velocidad opcional especifica la duración del efecto. Puede tomar los siguientes valores: "lento", "rápido", o milisegundos.
  - El parámetro callback opcional es una función a ejecutar una vez finalizada la animación.

JQUERY- ANIMACIONES

- Ejemplo simple

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({left:'250px'});
  });
});
</script>
</head>

<body>
<button>Start Animation</button>
</div>

</body>
</html>
```

JQUERY- ANIMACIONES

- Ejemplo con multiples valores

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      left:'250px',
      opacity:'0.5',
      height:'150px',
      width:'150px'
    });
  });
});
</script>
</head>

<body>
<button>Start Animation</button>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;">
</div>

</body>
</html></html>
```

JQUERY- ANIMACIONES

- Ejemplo con valores relativos

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      left:'250px',
      height:'+=150px',
      width:'+=150px'
    });
  });
});
</script>
</head>

<body>
<button>Start Animation</button>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;">
</div>

</body>
</html>
```

JQUERY- ANIMACIONES

- Ejemplo con varias animaciones

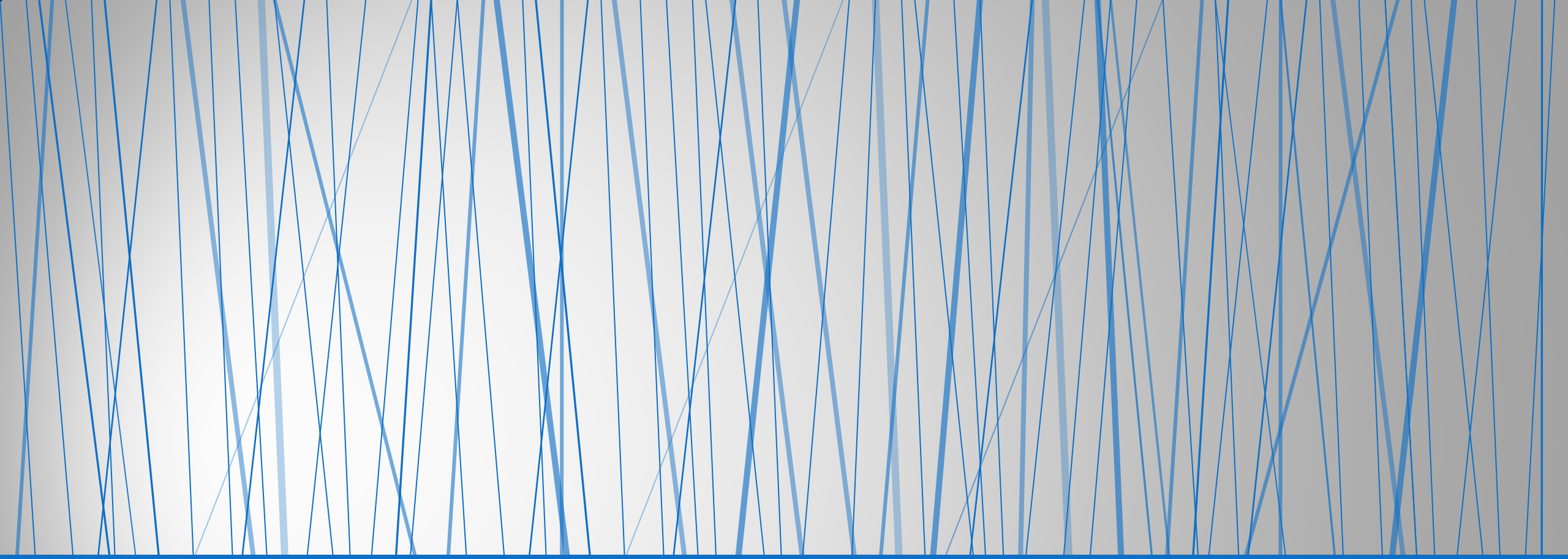
```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    var div=$("div");
    div.animate({height:'300px',opacity:'0.4'},"slow");
    div.animate({width:'300px',opacity:'0.8'},"slow");
    div.animate({height:'100px',opacity:'0.4'},"slow");
    div.animate({width:'100px',opacity:'0.8'},"slow");
  });
});
</script>
</head>

<body>
<button>Start Animation</button>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;">
</div>

</body>
</html>
```

Declaraciones en cadena

- Hasta ahora hemos estado escribiendo jQuery declaraciones una a la vez (uno tras otro).
- Sin embargo, existe una técnica llamada encadenamiento, que nos permite ejecutar varios comandos jQuery, uno tras otro, en el mismo elemento(s).
- Sugerencia: De esta manera, los navegadores no tienen que encontrar el mismo elemento(s) más de una vez.
- Para encadenar una acción, sólo se debe anexar la acción a la acción anterior.
- Ejemplo: $("#p1").css("color","red").slideUp(2000).slideDown(2000);

Declaraciones en cadena

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function()
  {
  $("button").click(function(){
    $("#p1").css("color","red").slideUp(2000).slideDown(2000);
  });
});
</script>
</head>
<body>

<p id="p1">¡Query es divertido!!</p>
<button>Click me</button>

</body>
</html>
```



JQuery (AJAX)

AJAX

- AJAX significa Asynchronous JavaScript and XML. Esta tecnología nos permite comunicarnos con un servicio web sin tener que recargar la página. A pesar de que su nombre lo dice, XML no es requerido para usar AJAX, de hecho, últimamente se utiliza JSON.
- Con jQuery, hacer uso de AJAX es muy sencillo. Para demostrarlo, estaré utilizando la web API de Open Weather Map (http://openweathermap.org/), a la cual haré algunas solicitudes utilizando AJAX y jQuery
- En esta página se describe el API: http://openweathermap.org/current

$.ajax()

- Con ésta función se pueden hacer todas las llamadas posibles en AJAX con Jquery.
- Esta funcion recibe una url y parametros de configuración de la llamada Ajax, en los cuales podremos configurar los siguientes parámetros referidos a como se realiza la llamada, a como son y como tratar(funciones callback) los datos de respuesta, al igual que los datos de la solicitud : async,beforeSend,cache,complete,contents,contentype,context,converters,crossDomain,data,dataFilter,dataType,error,global,headers,ifModified,isLocal,jsonp,jsonpCallback,method,mimeType, username,type etc….

Atajos

- Por simplicidad Jquery creó otros métodos más sencillos de configurar para las llamadas Ajax más comunes, serán estos los que veamos a continuación.
- $.get(), $getJson(), $.post() y $().load()

jQuery.get(url [, data] [, success] [, dataType])

- Recibe 4 parametros, sólo es obligatorio el primero:
  - url: url del servicio.
  - data: datos que se envían al servidor, con formato objeto de JS
  - Success, funcion que recibe los siguientes parametros:
    - Data : objeto js
    - Status: String que indica si ha habido error o no.
  - dataType: tipo de datos esperados desde el servicio(xml,json,script,text,html)

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function() {
    $("button").click(function(){
$.get("http://rest-service.guides.spring.io/greeting", function(data){
        var text = "id:" + data.id;
        text +="content" +data.content
         $("#div1").html(text);
    });
});
});
   </script>
</head>
<body>
<div id="div1"></div><br>
<button>Ajax!</button>
</body>
```

$.getJSON()

- Igual que get, pero no es necesario indicar el dataType.
- Para trabajad con JSONP incluye callaback=? En la url.

- $.post() : igual que $.get() pero enviando datos en el cuerpo del request y no en la url.

```
<!DOCTYPE html>
<html>
<head>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
  <meta charset="utf-8">
  <title>jQuery Ajax</title>
 <script>
  function test() {
      $.getJSON( "http://api.openweathermap.org/data/2.5/weather?q=Arroyomolinos,Madrid,Spain&appid=679843b7f71414aafb0b48e9942f0e39", function(data,status) {
       alert(status);
  $('body').html('<pre>' + JSON.stringify(data,null,2)+ '</pre>');
      });
   }
 </script>
</head>
<body>
  <button onclick="test2);">click para probar </button>
</body>
</html>
```

$(selector).load()

- Atajo para cargar los datos en alguna etiqueta del dom.
- $(*selector*).load(*URL,data,callback*);
- Siguiendo el primer ejemplo , podríamos haber hecho:

```
$(document).ready(function() {
    $("button").click(function(){
                $("#div1").load("http://rest-service.guides.spring.io/greeting")
    });
});
```

JQUERY Otras funciones

Delay

- Realiza una pausa de n milisegundos.
- delay(n)
- Ejemplo de uso:

```
$("#capaopaca").fadeOut(1000).delay(2000).fadeIn(1000);
```

each

- La functión each aplicada sobre un selector itera por los elementos devueltos por este
- Syntaxis: $(*selector*).each(function(*index*,*element*)) (opcionales ambos)
- Dentro del each se puede hacer referencia a cada elemento de la iteración mediante el selector $(this) o el parámetro element.
- Se puede invocar a una función por cada iteración recogiendo el índice de la iteración por el parámetro index
- Ejemplo:

```
$("tr").each(function(i) {
if(i%2==0){
$(this).css("background-color", "#eee");
} else {
$(this).css("background-color", "#333");
$(this).css("color", "white"); }
} )
```

each

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("li").each(function(){
      alert($(this).text())
    });
  });
});
</script>
</head>
<body>
<button>Alert the value of each list item</button>
<ul>
<li>Coffee</li>
<li>Milk</li>
<li>Soda</li>
</ul>
</body>
</html>
```