

Fundamentos de la Web

Bloque II: Tecnologías de cliente web

Tema 8 – jQuery

jQuery



<http://jquery.com>

- En **2006** los navegadores web ejecutaban de forma ligeramente diferente el **mismo código JavaScript** (sobre todo al manipular el DOM). Especialmente **Internet Explorer**
- La **API DOM era muy limitada** (no como ahora). Operaciones sencillas requerían escribir **código complejo** (que había que **repetir** en cada web y tenía que ser específico para cada navegador)

- En 2006 se crea la librería jQuery
 - Ofrece **operaciones de alto nivel** en el DOM (hide, toggle...)
 - **Unifica las diferencias** que existen en los navegadores
 - Para seleccionar elementos de la página se usa la misma sintaxis que los **selectores CSS** (antes no se podía con la API DOM)
 - Trabajaba directamente con **conjuntos de elementos**, de forma que no es necesario procesar cada elemento de forma individual (**no hay que hacer el for**)

- ¿Sigue siendo útil jQuery?

- Los navegadores ahora son muy compatibles entre sí.
(El código JavaScript se comporta de la misma forma)
- La API DOM ofrece APIs de alto nivel (similares a las de jQuery)

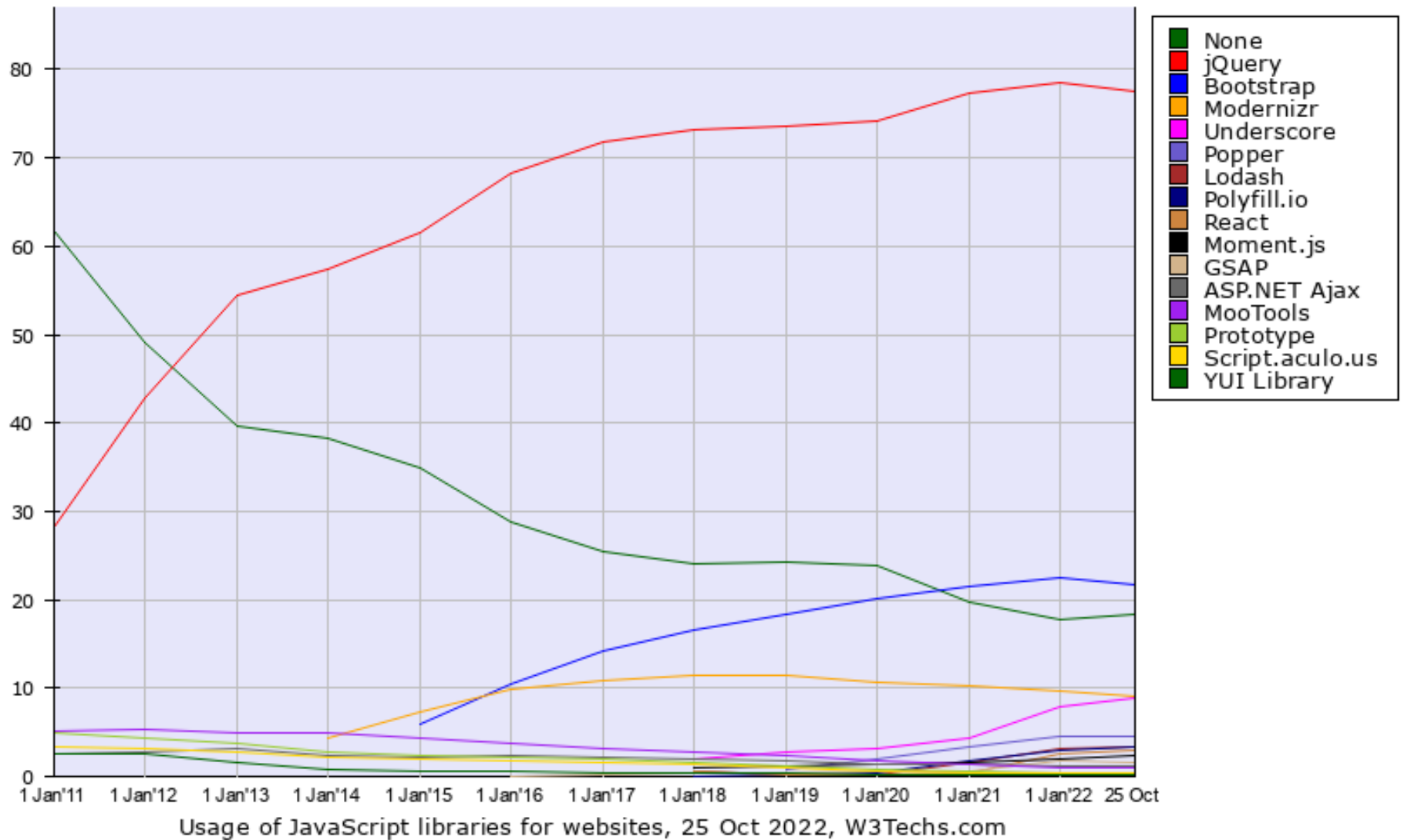
<http://youmightnotneedjquery.com/>

- Los frameworks de alto nivel cada vez se usan más (y no se necesita jquery)
- En la mayoría de las webs creadas hoy jQuery ya no se usa

- **Pero sigue siendo muy popular**
 - Bootstrap 4 necesitaba jQuery (Bootstrap 5, publicado en Mayo del 2021, ya no lo necesita)
 - En Octubre del 2022 jQuery se usa en el **74%** de los 10 millones de sitios web más populares

https://w3techs.com/technologies/history_overview/javascript_library/all/y

jQuery



- ## Uso de jQuery

- La librería se puede descargar de su página web para desarrollo local
- Cuando se publica una página web que usa la librería, en vez de servir la librería en el servidor web, es mejor usar una ruta en una **red de distribución de contenido (CDN)**
- Si la librería está cacheada en el navegador del cliente no se descargará de nuevo (reduciendo el tiempo de carga)

<https://releases.jquery.com/>

index.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
    <script src="https://code.jquery.com/jquery-3.6.1.min.js"
      integrity="sha256-o88AwQnZB+VDvE9tvIXrMQaPlFFSUTR+nldQm1LuPXQ="
      crossorigin="anonymous"></script>
    </script>
    <script src="app.js"></script>
  </head>
  <body>
    <h1>Title</h1>
    <p>Text...</p>
  </body>
</html>
```

app.js

```
$(document).ready(function() {  
    console.log('Document loaded');  
})
```

- `$(document).ready(...)`
 - Ejecuta la función que se pasa como parámetro cuando el **documento HTML se ha cargado**
 - Es ideal para ejecutar código de inicialización que asocia eventos a los elementos de la página **si el script se ejecuta desde el head** (antes de que se cargue el HTML)
 - Similar al evento **DOMContentLoaded** del document

app.js

```
$(document).ready(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```

- La función `$` está sobrecargada y admite muchos tipos de parámetros
 - Si se le pasa el **documento**, devuelve un objeto con métodos que permiten asociar funciones que se ejecutarán ante un determinado evento de carga (p.e. **ready**)
 - Si se le pasa un **String**, es un **selector CSS** que devuelve un conjunto con los elementos del documento **seleccionados**

app.js

```
$(document).ready(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```

- `$('div.poem-stanza').addClass('highlight');`
- En el conjunto de elementos devueltos se pueden ejecutar métodos que se ejecutarán sobre todos los elementos del conjunto:
 - **addClass(...):** Añade una clase CSS a los elementos
 - **removeClass(...):** Elimina una clase CSS a los elementos

- Es tan común ejecutar código cuando el documento se carga que se puede pasar la función directamente a \$

app.js

```
$(document).ready(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```



app.js

```
$(function() {  
    $('div.poem-stanza').addClass('highlight');  
});
```

Seleccionar elementos

- Selección de elementos con selectores CSS
 - Basados en elementos, clases e ids

```
$('#selected-plays > li').addClass('horizontal');
```

- Basados en atributos con expresiones regulares

```
$('.img[alt]').addClass('imgstyle');
$('.a[href^="mailto:"]').addClass('mailto');
$('.a[href$=".pdf"]').addClass('pdflink');
$('.a[href^="http"]').addClass('henrylink');
```

Seleccionar elementos

- Selectores propios de jQuery (no CSS)
 - Elementos pares (empezando con 0)

```
$( 'tr:even' ).addClass( 'alt' );
```

- Hijos pares (empezando con 1)

```
$( 'tr:nth-child(odd)' ).addClass( 'alt' );
```

<http://api.jquery.com/category/selectors/>

Seleccionar elementos

- **Seleccionar elementos con operaciones**

- Filtrar los elementos usando funciones (el elemento es this)

```
$('#a').filter(function() {
    return this.hostname && this.hostname!=location.hostname;
}).addClass('external');
```

- Navegar por el DOM desde un elemento

```
$('#td:contains(Henry)').next().addClass('highlight');
$('#td:contains(Henry)').parent().find('td:eq(1)')
    .addClass('highlight').end().find('td:eq(2)')
    .addClass('highlight');
```


Seleccionar elementos


- **Acceso a los elementos nativos DOM**
 - Se puede acceder a los **elementos nativos del árbol DOM** con el método `get(...)`
 - Como los objetos devueltos por jQuery pueden tener muchos nodos del árbol DOM, el método `get()` permite indicar cuál de los nodos es (aunque solo haya uno)

```
let tagName = $('#my-element').get(0).tagName;
```

- Los eventos se asocian a los elementos con el método **on**

```
$('#button1').on('click', function() {  
    $('body').addClass('large');  
});
```

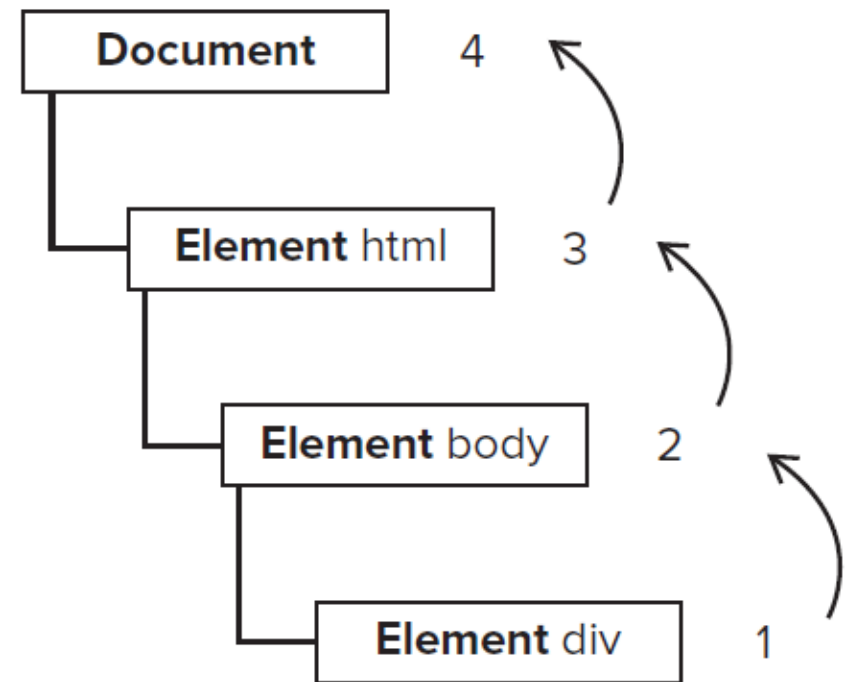
```
$('#button2').on('click', function() {  
    $(this).addClass('selected');  
});
```



this apunta al nodo nativo en el que se ha generado el evento.
\$(this) convierte el nodo nativo en objeto jQuery

Eventos

- Los eventos **generados** en un elemento se pueden procesar en el propio elemento o en cualquiera de sus **elementos padre**
- Cuando se **configura** un manejador de eventos, el evento se ha podido producir en el propio elemento o en **cualquiera de sus hijos**
- Se dice que el evento burbujea (***bubble***)



Eventos

- La función manejadora recibe el objeto **event**
 - **this**: Objeto en el que se ha configurado el manejador
 - **event.target**: Objeto que origina realmente el evento
 - **event.stopPropagation()**: Evita que se ejecuten manejadores de los elementos padre. Se detiene el burbujeo.
 - **event.preventDefault()**: Evita que el browser realice el comportamiento por defecto para ese evento

```
$('#panel').on('click', function(event) {
    console.log('Generado en:' + event.target);
});
```

Generar HTML

- El método **append(...)** permite añadir contenido HTML a un elemento

```
<p>I would like to say: </p>
```

```
$( "p" ).append( "<strong>Hello</strong>" );
```

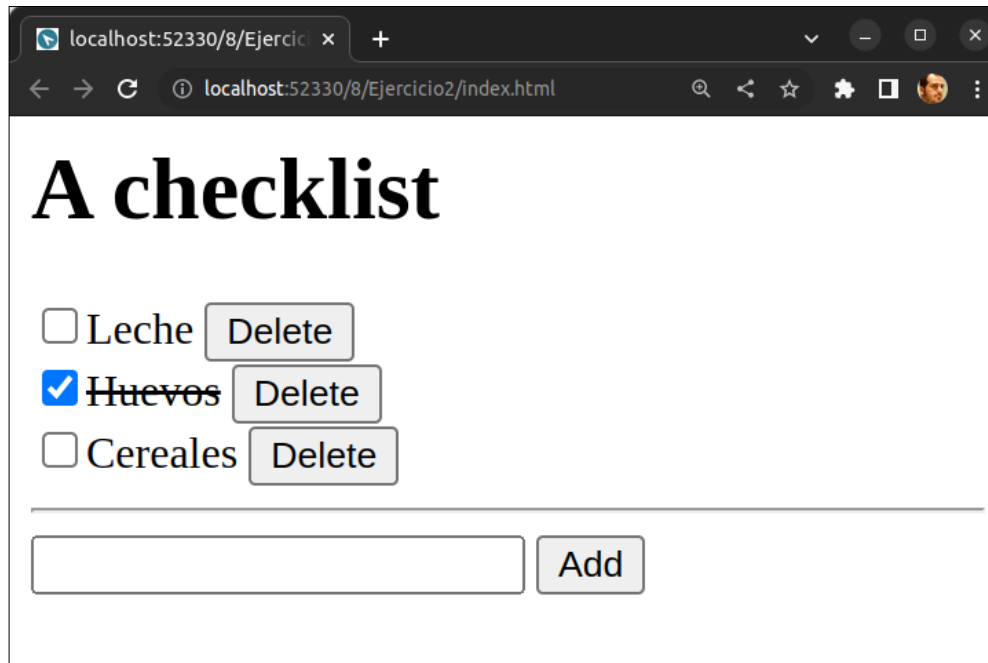
<https://api.jquery.com/append/>

Ejercicio 1

- Crea una página con campo de texto y un botón
- Cada vez que se pulse el botón, se añadirá el contenido del cuadro de texto a la página (sin borrar el contenido previo)

Ejercicio 2

- Crea una página para gestionar una lista de la compra
- Se podrán añadir, quitar y tachar elementos



localhost:52330/8/Ejercicio2/index.html

A checklist

- ☐ Leche
- ☒ Huevos
- ☐ Cereales
