

Continguts interactius al web. Introducció a jQuery

Marta Moreno

Javascript

Per desenvolupar un lloc web no tan sols cal preparar els elements multimèdia com són les imatges, el vídeo i el so, sinó que s'ha de ser capaç d'integrar-los. Per fer-ho, molt sovint són necessaris coneixements de JavaScript, a banda d'HTML i CSS.

JavaScript és l'únic llenguatge disponible nadiument als navegadors i per aquesta raó és el que es fa servir per afegir comportaments dinàmics i interactius al web. Atesa la seva importància, és imprescindible adquirir uns coneixements bàsics sobre aquest.

Habitualment, es fan servir biblioteques que faciliten molt la feina perquè incorporen dreceres per a les tasques més habituals o implementacions genèriques de controls que es poden fer servir en diferents projectes.

Elements interactius bàsics

La biblioteca jQuery és una de les més utilitzades i ofereix moltes possibilitats.

jQuery facilita la modificació de l'estructura del document, afegir i modificar classes CSS i afegeix moltes funcions que faciliten moltes de les tasques més comunes en treballar amb la interfície del lloc web.

Configuració dels navegadors per a la visualització d'elements interactius:

- Activar l'ús de cookies: si desactivem com que no es podrà recordar la informació de l'usuari, alguns llocs seran directament inaccessibles, per exemple els llocs web de comerç electrònic.
- Activar l'ús de JavaScript: si desactivem en alguns casos el lloc serà inservible. Per exemple, una web que inclogui mapes de Google Maps no pot funcionar sense JavaScript; en canvi, d'altres només l'utilitzen per afegir alguns efectes, i per a aquests sí que es pot presentar una solució alternativa.

Elements bàsics interactius

Quan la desactivació del JavaScript en el navegador no permet visualitzar els elements interactius de la web s'ha de mostrar un missatge informant perquè no funcionen per avisar a l'usuari de que aquestes opcions estan desactivades.

La inclusió de nous dispositius d'entrada i sortida s'han afegit controls que limiten a quins d'aquests dispositius es pot accedir.

Les Apis demanen permís a l'usuari abans de permetre utilitzar-les:

- API de globalització
- API de càmera
- API del micròfon

No tots els navegadors suporten totes les característiques previstes en HTML5, com per exemple la WebVR API que només és suportada per GoogleChrome i les versions més recents de Mozilla FireFox.

https://developer.mozilla.org/en-US/docs/Web/API/WebVR_API

Eines per a la inclusió de contingut multimèdia, JQuery

Hi han biblioteques per automatitzar les accions més repetitives simplificant la sintaxi i s'encarreguen de gestionar les diferències entre els navegadors fent servir internament diferents implementacions segons siguin o no suportades determinades característiques.

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

Utilitzarem la versió sense comprimir en l'entorn de test per poder depurar el codi i la versió comprimida en el entorn de producció.

Els fitxers de JavaScript, en ser text pla, realment no es comprimeixen, el que es fa és modificar-los fent servir altres eines de manera que s'eliminin tots els comentaris, els espais innecessaris i els salts de línia, i se substitueix el nom de les variables per d'altres més curts (normalment, una sola lletra). D'aquesta manera, la mida del fitxer es redueix i el temps de descàrrega és menor.

JQuery

- El primer pas es carregar la biblioteca de JQuery:

```
<script src="//code.jquery.com/jquery-3.1.1.js"></Script>
```

- El DOM, es la estructura del document, el DOM, Document Object Model, és una interfície que ens facilita treballar amb documents HTML.
- El DOM defineix els mètodes i les propietats que ens permeten accedir i manipular el document com si es tractés d'un arbre de nodes, on l'arrel és el node document que pot contenir qualsevol quantitat de nodes fills i les fulles que son els nodes que no tenen cap fill.
- JQuery és la capacitat de seleccionar i manipular aquests elements fàcilment, ja sigui per modificar-los, per eliminar-los o per afegir-ne de nous.
- Sempre s'ha d'afegir el codi dins de la funció que és cridada per mètode ready de jQuery, per assegurar-nos que l'objecte DOM s'ha creat.
 1. `$(document).ready(function() {`
 2. `// Aquí va el nostre codi`
 3. `});`

Simbol \$

1. `$(document).ready(function() {`
2. `// Aquí va el nostre codi`
3. `});`

El símbol \$, el document `$(document)` el que fa és crida la funció `jQuery(document)`.

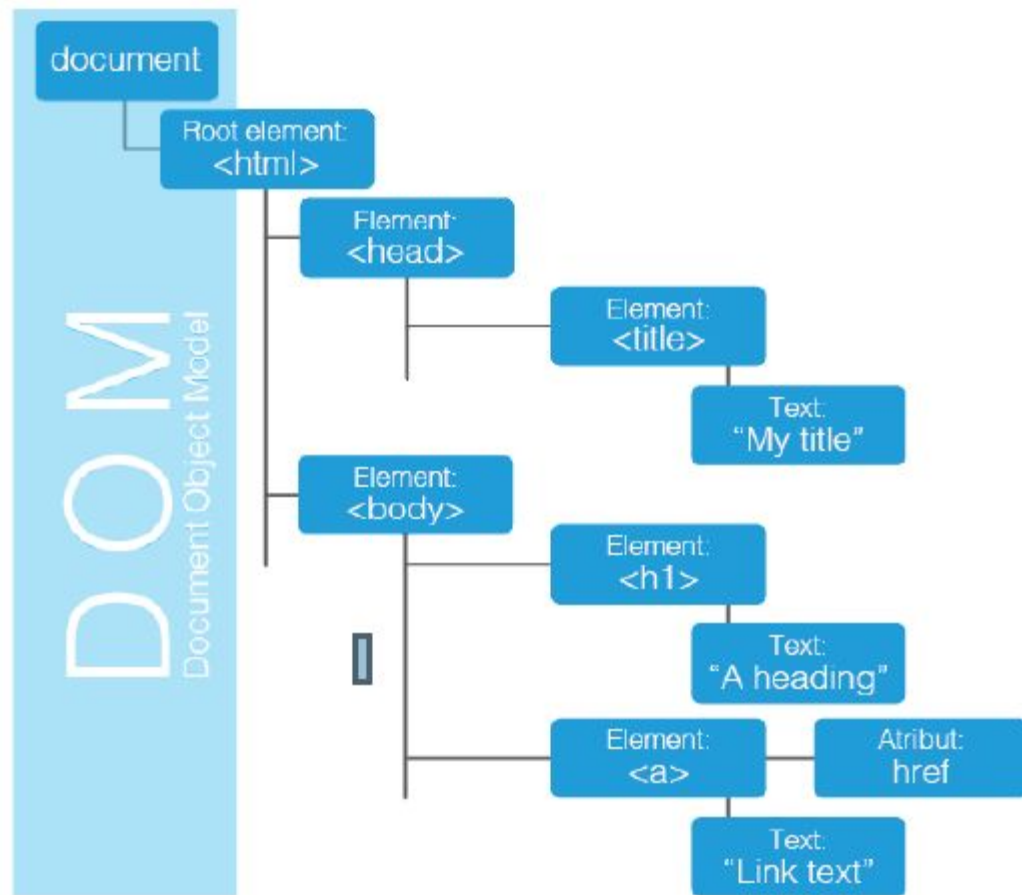
El paràmetre `document` de la funció `jQuery` és el pare de l'arbre DOM.

El DOM és l'objecte que es crea al navegador després de carregar el codi HTML de la pàgina web, es una estructura de dades.

DOM

El mètode ready sempre passarem l'objecte document com a paràmetre, i s'activa quan estigui creat tot l'arbre DOM de la pàgina web.

Tots els elements de la pàgina web són descendents del document.



Explicació mètode ready()

1. `$(document).ready(function() {`
2. `// Aquí va el nostre codi`
3. `});`

Aquest codi el que fa és dir-li a jQuery(\$) que es vol executar el codi (`function(){};`) una vegada el document (`document`) estigui creat (`ready`).

El mètode `ready()` té una funció similar a:

```
<body onLoad="funcioACrida();">
```

És incompatible amb aquesta. Si es fa servir un no s'ha de fer servir l'altre.

Aquest codi té dues parts, la crida a la funció jQuery que retorna un objecte i després la crida a un mètode d'aquest objecte.

`$(document).ready()`

No és possible interactuar de forma segura amb el contingut de la pàgina fins que el document no es trobi preparat per la seva manipulació. JQuery permet detectar que la pàgina ja s'ha carregat sencera.

```
$(document).ready(function() {  
    console.log('el documento está preparado');  
});
```

Existe una forma abreviada para `$(document).ready()` la cual podrá encontrar algunas veces; sin embargo, es recomendable no utilizarla en caso que este escribiendo código para gente que no conoce jQuery.

Forma abreviada para `$(document).ready()`

```
$(function() {  
    console.log('el documento está preparado');  
});
```

Declaracion de funciones

A més és possible passar-li a `$(document).ready()` una funció definida en lloc d'una anònima:

```
function readyFn() {  
    // còdi a executar quan el document està llest  
}
```

```
$(document).ready(readyFn);
```

o

```
$(readyFn);
```

Funció jQuery

La funció jQuery converteix el paràmetre en un objecte jQuery, i aquest paràmetre pot ser un node com document , una cadena de text amb un selector CSS, com per exemple p , o una cadena de codi HTML, com per exemple <p>Hola, món</p>.

```
$(document).ready(function() {  
  // Aquí va el nostre codi  
  console.log('document:', $(document));  
  console.log('p: ', $('p'));  
  console.log('<p>Hola mon</p>: ', $('<p>Hola mon</p>'));  
});|
```

https://drive.google.com/file/d/1tUB0BScZmo_ioYzeKuNdgWX-jl_My287/view?usp=sharing

El mètode console.log

El mètode console.log mostra els valors dels paràmetres per a la consola de les eines de desenvolupador. Per poder veure el resultat heu d'anar a les opcions del vostre navegador:

- Google Chrome: botó esquerra sobre la pàg. html i Inspect
- Mozilla Firefox: botó esquerra sobre la pàg html i inspect element

Una vegada es mostri el panell d'eines, veureu una pestanya anomenada Console o Consola.

Mètode console.log

Si desplegueu els continguts de cada objecte podreu veure que la seva estructura és diferent:

- L'objecte creat pel document no té la propietat prevObject perquè no hi ha cap objecte anterior, ni la propietat selector.
- L'objecte generat a partir del selector p conté tant un objecte anterior (prevObject), que es correspon amb el document (és lògic, perquè l'element està inclòs en el document), i la propietat selector.
- L'objecte generat a partir de la cadena de text no té cap element previ perquè no s'ha afegit al document; tampoc no té context, per la mateixa raó, i no es pot seleccionar.

El objecte console té els següents mètodes que ens poden ajudar a depurar l'aplicació:

- console.log() para enviar y registrar mensajes generales.
- console.dir() para registrar un objeto y visualizar sus propiedades.
- console.warn() para registrar mensajes de alerta.
- console.error() para registrar mensajes de error.

Selecció d'elements

El concepte més bàsic de jQuery és el de seleccionar alguns elements i realitzar accions amb ells.

La biblioteca suporta gran part dels selectors CSS3 i varis més no estandaritzats.

A <https://api.jquery.com/category/selectors/> es pot trobar una completa referència sobre els selectors de la biblioteca.

Tipus de selecció d'elements:

- selecció d'elements en base a la seva id
- selecció d'elements en base al nom de classe
- selecció d'elements pel seu atribut
- selecció d'elements en forma de selector CSS
- pseudo-selectors

Selecció d'elements

- Selecció d'elements en base al seu id:

`$('#myId');` // els ids han de ser únics per pàgina

- Selecció d'elements en base al nom de classe

`$('#div.myClass');` // si s'especifica el tipus d'element es guanya temps

- Selecció d'elements per el seu atribut:

`$('#input[name=first_name]);` // pot ser molt lent

- Selecció d'elements en forma de selector CSS:

`$('#contents ul.people li');`

Pseudo-selectors

`$(‘a.external:first’);` // selecciona el primer element `<a>` amb la classe `‘external’`

`$(‘tr:odd’);` // selecciona tots els elements `<tr>` imparells d’una taula

`$(‘#myForm :input’);` // selecciona tots els elements del tipus `input` dins del formulari `myForm`

`$(‘div:visible’);` // selecciona tots els `divs` visibles

`$(‘div:gt(2)’);` // selecciona tots els `divs` excepte els primers

`$(‘div:animated’);` // selecciona tots els `divs` actualment animats

Exemple selecció d'elements, propietat length

```
HTML
1 <div>
2   <p>Primer paràgraf</p>
3   <p>Segon paràgraf</p>
4   <p>Tercer paràgraf</p>
5   </div>
6   <p>Quart paràgraf</p>

CSS

JS
1 $(document).ready(function() {
2   console.log('p: ', $('p'));
3 });
```

Primer paràgraf

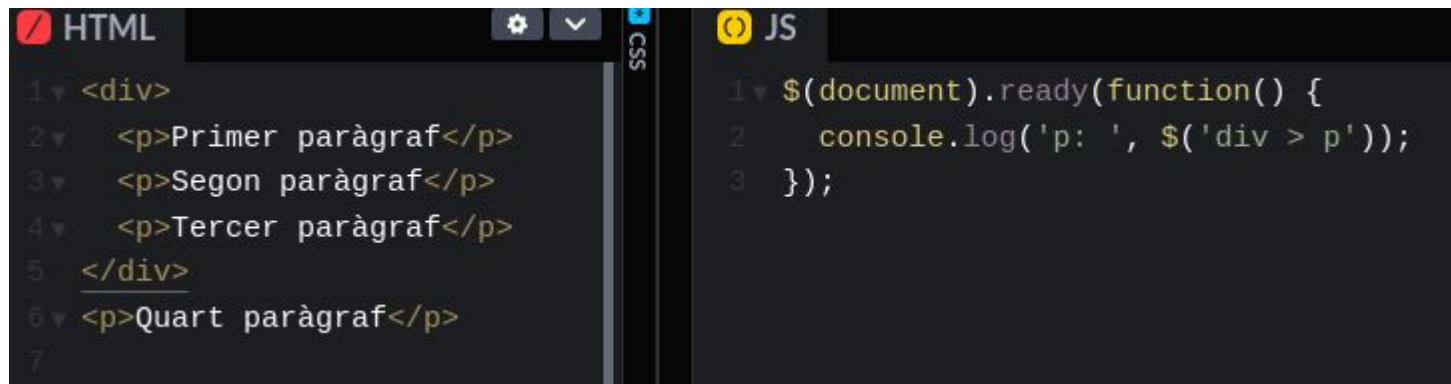
Segon paràgraf

Tercer paràgraf

Quart paràgraf



Selecciona un o mes elements



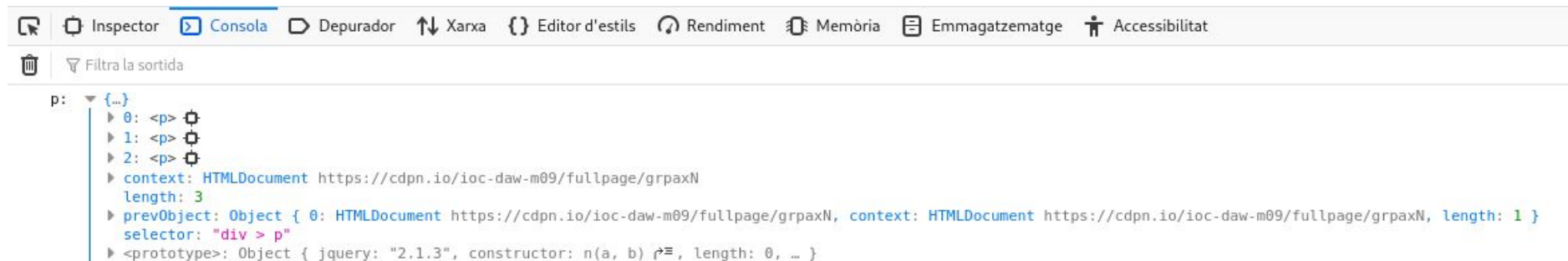
The screenshot shows a code editor with two tabs: HTML and JS. The HTML tab is active, displaying the following code:

```
1 <div>
2   <p>Primer paràgraf</p>
3   <p>Segon paràgraf</p>
4   <p>Tercer paràgraf</p>
5 </div>
6 <p>Quart paràgraf</p>
7
```

The JS tab is also visible, displaying the following code:

```
1 $(document).ready(function() {
2   console.log('p: ', $('div > p'));
3 });
```

<https://codepen.io/ioc-daw-m09/pen/grpaxN?editors=1010>



The screenshot shows a web browser's developer console. The top bar includes various tools like Inspector, Consola, Depurador, Xarxa, Editor d'estils, Rendiment, Memòria, Emmagatzematge, and Accessibilitat. The Consola tab is active, showing the output of the JavaScript code:

```
p: {
  0: <p>
  1: <p>
  2: <p>
  context: HTMLDocument https://cdpn.io/ioc-daw-m09/fullpage/grpaxN
  length: 3
  prevObject: Object { 0: HTMLDocument https://cdpn.io/ioc-daw-m09/fullpage/grpaxN, context: HTMLDocument https://cdpn.io/ioc-daw-m09/fullpage/grpaxN, length: 1 }
  selector: "div > p"
  <prototype>: Object { jquery: "2.1.3", constructor: n(a, b) ↗, length: 0, ... }
```

Modificar l'estil

```
HTML
1 <div id="primer"></div>
2 <div id="segon"></div>
3 <div id="tercer"></div>
4 <div id="quart"></div>
5

CSS
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   float: left;
6   background-color: lightgrey;
7 }

JS
1 $(function() {
2   var $quadres = $('div');
3   $quadres.css('background-color', 'red');
4   alert("El color dels quadres es: " + $quadres.css('background-color'));
5 });
6
```

<https://codepen.io/ioc-daw-m09/pen/Mywowy>



La pàgina a <https://cdpn.io> diu:

El color dels quadres es: rgb(255, 0, 0)

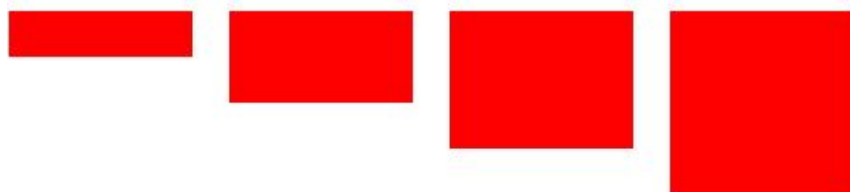
Es fa servir el mateix mètode tant per establir la propietat com per recuperar el seu valor. jQuery els distingeix segons el nombre de paràmetres, de manera que si se'n passen dos intentarà establir-la, i si només se'n passa un intentarà recuperar-la. Molts dels mètodes de jQuery funcionen d'aquesta manera.

```
HTML
1 <div id="primer"></div>
2 <div id="segon"></div>
3 <div id="tercer"></div>
4 <div id="quart"></div>
5

CSS
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   float: left;
6   background-color: lightgrey;
7 }

JS
1 $(document).ready(function() {
2   var $quadres = $('div');
3
4   $quadres.css('background-color', 'red');
5
6   $quadres.each(function(i) {
7     var alcada = 25 + i * 25;
8     $(this).css('height', alcada + 'px');
9   });
10 });
```

<https://codepen.io/ioc-daw-m09/pen/GZJEpP>



jQuery ens proporciona un mètode per recórrer a tots els elements i aplicar-hi els canvis de manera individual, substituint el codi JavaScript pel següent

Diferents usos d'each a jQuery

jQuery té dos mètodes anomenats each que fan coses diferents. Si es crida a partir d'un objecte amb una selecció d'elements, el que fa és recórrer a aquests elements i aplicar la funció cridada a cada element.

En canvi, si es crida directament sobre jQuery (`$.each(array, funció)`), el que fa és recórrer a l'array i aplicar la funció a cada element d'aquest.

En aquest últim cas seria més adient dir que el que hem fet és cridar la funció each de la biblioteca per diferenciar-lo del primer cas, en què s'ha cridat el mètode each d'una instància d'un objecte concret.

mètode each()

S'ha afegit una crida al mètode each i s'ha passat com a paràmetre una funció. Aquesta funció és cridada per a cadascun dels elements de l'objecte jQuery, en aquest cas els quatre elements div . El valor del paràmetre i es correspondrà en cada cas amb l'índex de l'objecte (sent 0 el primer i 3 l'últim en aquest exemple).

Dins de la funció, el que es fa és calcular l'alçada de cada element amb la fórmula $25 + i * 25$; a continuació es crea un nou objecte jQuery que fa referència a aquest element amb `$(this)` (com que no es fa res més amb aquest no cal guardar-lo a cap variable), i es crida el mètode css com en el cas anterior, però aquest cop per a la propietat height .

Fixeu-vos que per poder manipular cadascun d'aquests elements s'ha hagut de cridar de nou la funció jQuery passant l'element (this fa referència a l'element processat).

```
$(document).ready(function() {  
    var $squadres = $('div');  
    $squadres.css('background-color', 'red');  
    $squadres.each(function(i) {  
        var alcada = 25 + i * 25;  
        $(this).css('height', alcada + 'px');  
        if (i === 1) {  
            return false;  
        }  
    });  
});
```



Hi ha dues maneres d'accedir a aquests elements: a través del mètode `get` o directament com si es tractés d'un array especificant l'índex.

```
<div id="primer"></div>
<div id="segon"></div>
<div id="tercer"></div>
<div id="quart"></div>
```

```
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   float: left;
6   background-color: lightgrey;
7 }
```

```
1 $(document).ready(function() {
2   var $quadres = $('div');
3
4   $($quadres[2]).css('background-color', 'red');
5
6   $($quadres.get(0)).css('background-color', 'green');
7
8 });
```



Explicació:

```
$(document).ready(function() {  
    var $quadres = $('div');  
  
    $($quadres[2]).css('background-color', 'red');  
  
    $($quadres.get(0)).css('background-color', 'green');  
});
```

S'ha de tenir en compte que encara que és possible, no és recomanable accedir a l'element com si es tractés d'un array. S'ha de fer servir el mètode `get` perquè és el que la interfície de jQuery ens garanteix que funcionarà correctament.

En el primer cas s'ha seleccionat el tercer element com si es tractés d'un array.

Fixeu-vos que per poder cridar el mètode `css` s'ha hagut de tornar a cridar la funció jQuery passant-li com a argument el node emmagatzemat a la tercera posició de l'array: `($quadres[2])`

En el segon s'ha fet correctament, cridant el mètode `get` passant com a argument la posició 0 de l'array, a continuació s'ha passat el node retornat a la funció jQuery i s'ha cridat el mètode `css` per canviar la propietat `background-color` a verd.

El mètode `get` retorna el node seleccionat de la posició; no es tracta d'un objecte jQuery. Un dels tipus de paràmetres que es pot passar a la funció jQuery és precisament un node. Així doncs, el que s'ha fet és crear un nou objecte jQuery a partir d'aquest node i seguidament cridar el mètode `css`.

Afegir classes dels nodes

HTML

```
1 <div class="vores"></div>
2
```



CSS

```
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   background-color: lightgrey;
6 }
7
8 .vores {
9   border-radius: 50px;
10 }
11
12 .vermell {
13   background-color: red;
14 }
15
16 .gran {
17   width: 200px;
18   height: 200px;
19 }
```

JS

```
1 $(document).ready(function() {
2   var $quadre = $('div');
3   $quadre.addClass('vermell');
4   $quadre.addClass('gran');
5 });
```

En aquest cas s'han afegit dues classes CSS amb el mètode `addClass`, una anomenada `vermell` i una `gran`, que són afegides en temps d'execució i fan que el quadre tingui 200x200 px de mida i color vermell en lloc de gris.

Eliminar classes dels nodes

Per eliminar-les és igual de fàcil, només s'ha de fer servir el mètode `removeClass`.



Events en JQuery

Detecció d'events en jQuery: <https://codepen.io/ioc-daw-m09/pen/KzpyGa>

```
HTML
1 <div class="vores"></div>
2

CSS
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   background-color: lightgrey;
6 }
7
8 .vores {
9   border-radius: 50px;
10 }
11
12 .vermell {
13   background-color: red;
14 }

JS
1 $(document).ready(function() {
2   var $squadre = $('div');
3
4   $squadre.on('mouseover', function(e) {
5     $(this).addClass('vermell');
6   })
7
8   $squadre.on('mouseout', function(e) {
9     $(this).removeClass('vermell');
10  })
11
12 });
```

click event

```
HTML
1 <div class="vores"></div>
2

CSS
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   background-color: lightgrey;
6 }
7
8 .vores {
9   border-radius: 50px;
10 }
11
12 .vermell {
13   background-color: red;
14 }

JS
1 $(document).ready(function() {
2   var $squadre = $('div');
3
4   $squadre.on('click', function(e) {
5     $(this).toggleClass('vermell');
6   })
7
8 });
```

<https://codepen.io/ioc-daw-m09/pen/RaPjvz>

efecte del mètode toggleClass

<https://codepen.io/ioc-daw-m09/pen/aNOxNR>

```
HTML
1 <div></div>
2 <div></div>
3 <div></div>
4 <div></div>
5
6

CSS
1 div {
2   width: 100px;
3   height: 100px;
4   margin: 10px;
5   float: left;
6   background-color: lightgrey;
7   transition: 1s ease;
8 }
9
10 .vores {
11   border-radius: 50px;
12 }
13
14 .vermell {

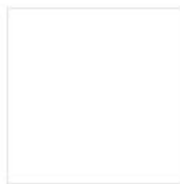
JS
1 $(document).ready(function() {
2   var $squadre = $('div');
3
4   $squadre.on('click', function(e) {
5     $(this).toggleClass('vermell');
6   })
7
8 });
```

Combinacions de CSS i jQuery per afegir animacions

```
<div></div>
<div></div>
<div></div>
<div></div>
```

```
1 div {
2   width: 100px;
3   height: 100px;
4   float: left;
5   border: 1px solid lightgrey;
6   margin: 10px;
7   opacity: 0.5;
8   transition: 0.5s ease-out;
9 }
10
11 .pols {
12   background-color: blue;
13   animation-name: pols;
14   animation-duration: 0.5s;
15   animation-iteration-count: infinite;
16   animation-direction: alternate;
17 }
18
19 @keyframes pols {
20   from {
21     transform: scale(1.0);
22     opacity: 0.5;
23   }
24   to {
25     transform: scale(1.2);
26     opacity: 1;
27   }
28 }
```

```
1 $(document).ready(function() {
2   var $quadres = $('div');
3
4   $quadres.on('mouseover', function(e) {
5     $(this).addClass('pols');
6   })
7
8   $quadres.on('mouseout', function(e) {
9     $(this).removeClass('pols');
10  })
11
12 });
```



<https://codepen.io/ioc-daw-m09/pen/reVbXb>

Explicació:

En aquest cas s'ha tornat a fer servir el mètode `on` amb els events `mouseover` i `mouseout` per detectar quan entra i quan surt el cursor dels quadres, i els mètodes `addClass` i `removeClass` per afegir o eliminar la classe `pols`, que afegeix:

- Un canvi de color del fons a blau.
- Una animació que s'ha definit amb el nom de `pols`, amb una durada de 0.5s, que es repetirà indefinidament i amb la direcció de l'animació alterna, és a dir, que quan acaba torna enrere.

```
$(document).ready(function() {  
    var $quadres = $('div');  
  
    $quadres.on('mouseover', function(e) {  
        $(this).addClass('pols');  
    })  
  
    $quadres.on('mouseout', function(e) {  
        $(this).removeClass('pols');  
    })  
});
```


Afegir i eliminar nodes del document i manipular el DOM

```
HTML
1 <div id="avisos"></div>
2 <label>Introdueix un nombre:
3 <input type="text" id="entrada"/>
4 </label>
5 <button
6 id="validar">Validar</button>
7 <button
8 id="netejar">Netejar</button>

CSS
1 .missatge {
2   border-radius: 5px;
3   padding: 10px;
4   font-size: 1.2em;
5 }
6
7 .error {
8   background-color: #f2dede;
9   color: #d9534f;
10  border: 1px solid #d9534f;
11 }
12
13 .exit {
14   background-color: #dff0d8;
15   color: #5cb85c;
16   border: 1px solid #5cb85c;
17 }
18
19 .info {
20   background-color: #d9edf7;
21   color: #5bc0de;
22   border: 1px solid #5bc0de;
23 }

JS
1 $(document).ready(function() {
2   var $avisos = $('#avisos'),
3       $entrada = $('#entrada'),
4       $validar = $('#validar'),
5       $netejar = $('#netejar');
6   $validar.on('click', function(e) {
7     esborrarAvisos();
8     if ($.isNumeric($entrada.val())) {
9       afegirAvis('exit', 'Has introduït un nombre!');
10    } else {
11      afegirAvis('error', "S'ha d'introduir un nombre");
12    }
13    afegirAvis('info', 'El text introduït ha estat: ' + $entrada.val());
14  });
15   $netejar.on('click', function(e) {
16     esborrarAvisos();
17   });
18   function afegirAvis(tipus, missatge) {
19     var definicioNode = '<p class="missatge ' + tipus + '"><b> ' + tipus.toUpperCase() + ';</b> ' + missatge + '</p>';
20     $avisos.append(definicioNode);
21   }
22   function esborrarAvisos() {
23     $avisos.find('p').remove();
24   }
25 });
```

ERROR: S'ha d'introduir un nombre

INFO: El text introduït ha estat: Feder

Introdueix un nombre:

Validar

Netejar

Introdueix un nombre:

Validar

Netejar

Afegir i eliminar nodes del document i manipular el DOM

Primerament, s'ha definit l'estructura del document, establint els id necessaris per poder identificar clarament on s'han de mostrar els avisos , quin és el text amb l'entrada de l'usuari, i quin és el botó per validar .

A continuació s'han afegit les classes CSS, una genèrica per a tots els avisos, anomenada missatge , que estableix les vores arrodonides, el padding i la mida de la font, i altres específics per canviar el color de l'avís segons si es tracta d'un tipus o un altre: error , exit i info .

El codi JavaScript no és gaire diferent del que s'ha vist fins ara, però inclou un parell de funcions pròpies:

- `afegeixMissatge(tipus, missatge)` : afegeix el missatge del tipus passat com a argument dins del div amb id="avisos" .
- `esborrarAvisos()` : esborra tots els avisos.

Afegir i eliminar nodes del document i manipular el DOM

Al principi es troba la declaració de les variables i l'assignació als tres elements que es volen controlar: el contenidor dels avisos, l'entrada de text i el botó.

A continuació es fa servir el mètode `on` per escoltar quan es dispara l'event `click` sobre el botó, i una vegada clicat es produeixen les següents accions:

- S'esborren tots els avisos `esborrarAvisos()` .
- Es comprova si el valor introduït és o no un nombre amb `$.isNumeric($entrada.val())` .
- Si és numèric, es crida `afegeixMissatge` indicant que el tipus serà `exit` i el missatge.
- Si no ho és, es crida també `afegeixMissatge` , però aquest cop el tipus serà `error` .
- En tot cas, es crida un cop més a `afegeixMissatge` per afegir un avís de tipus `info` indicant el valor introduït.

Afegir i eliminar nodes del document i manipular el DOM

Com es pot apreciar, `$entrada` és l'objecte jQuery que conté el node corresponent al quadre de text. Per accedir al valor d'aquest (la propietat `value`) accedim cridant el mètode `val` , en aquest cas amb `$entrada.val()` .

Tota la feina referida a afegir els nodes es troba dins de la funció `afegeixMissatge` . En primer lloc, es crea la cadena de text que conté el codi HTML que es vol afegir, en aquest cas és:

- Un paràgraf (`<p>`).
- Amb les classes `missatge` i la que correspongui al tipus (`error` , `exit` o `info`).
- I el missatge passat com a argument, amb el tipus en majúscules (`tipus.toUpperCase()`) i negreta (``).

Afegir i eliminar nodes del document i manipular el DOM

En concret, la cadena composta quedaria així per a un missatge d'error:

```
<p class="missatge error"><b>ERROR: </b>S'ha d'introduir un nombre</p> .
```

Una vegada creada la cadena, s'afegeix dins del node contingut en l'objecte jQuery `$avisos` , que correspon a l'element div amb id="avisos" . Per fer això només s'ha de cridar el mètode `append` , com es veu en l'exemple:

```
$avisos.append(definicioNode) .
```

En aquests exemples s'afegeixen sempre dos missatges; fixeu-vos que no se sobreescriven, sinó que s'afegeix un darrere de l'altre.

La primera acció que es fa en clicar el botó Validar i el botó Netejar és esborrar els avisos; vegem que passa quan es crida la funció `esborrarAvisos` :

- Es crida el mètode `find` al qual es passa el paràmetre `'p'` per seleccionar tots els paràgrafs que es trobin dins de l'element contingut a l'objecte guardat a `$avisos` , que en aquest cas és l'element `div` amb `id="avisos"` .
- Com que el retorn de `find` és un objecte jQuery, es pot continuar cridant mètodes de jQuery directament, així que es crida el mètode `remove` , que fa que s'eliminin tots els nodes seleccionats, en aquest cas tots els paràgrafs.

Convé subratllar que quan un objecte jQuery conté més d'un node i es crida qualsevol dels seus mètodes, els canvis produïts (o cerques) s'aplicaran a tots ells, per exemple en cridar els mètodes `css` o `remove` .

Separar el comportament en funcions com `esborrarAvisos` i `afegirAvis` proporciona molts avantatges, entre d'altres:

- S'evita la duplicació de codi, la qual cosa fa més fàcil el manteniment, ja que si voleu fer canvis a com s'esborren els elements només heu de fer els canvis en un lloc de l'aplicació.
- Encapsula la implementació de com es porten a terme els canvis, de manera que es pot modificar sense haver de tocar el codi de l'aplicació (penseu que una aplicació complexa podeu tenir aquest codi repartit entre diferents fitxers).

Crear nodes i modificarlos abans de afegirlos al document

El codi ara és molt més llarg. Fixeu-vos en el que s'ha fet, pas per pas:

```
function afegirAvis(tipus, missatge) {  
    var $ressaltat = $('<b></b>'),  
        $avis = $('<p></p>');  
    $avis.addClass('missatge');  
    $avis.addClass(tipus);  
    $ressaltat.text(tipus.toUpperCase() + ': ');  
    $avis.append($ressaltat)  
    $avis.append(missatge);  
    $avisos.append($avis);  
}
```

1. S'han creat dos objectes jQuery, un que conté un node de tipus b (ressaltat) i un altre de tipus p , paràgraf. El codi que es passa com a paràmetres és HTML.
2. Per crear l'etiqueta ressaltada amb el tipus del missatge s'ha fet servir el mètode text , que afegeix com a text dins de l'element seleccionat el que s'ha passat per paràmetre; en aquest cas, el tipus concatenant-li el símbol ':' i un espai en blanc.
3. A l'objecte que conté l'avís s'hi ha afegit primerament la classe missatge i a continuació la classe corresponent al tipus d'avís amb el mètode addClass .
4. A continuació s'ha afegit l'objecte jQuery amb el ressaltat (\$ressaltat) al node que contindrà tot l'avís (\$avis) fent servir el mètode append .
5. Es crida un altre cop el mètode append per afegir a continuació el missatge, que és una cadena de text, i s'afegeix com a text pla.
6. Finalment, s'afegeix al contenidor d'avisos emmagatzemat a la variable \$avisos l'objecte jQuery amb les classes, el ressaltat i el missatge afegits.

<https://codepen.io/ioc-daw-m09/pen/bpVeZV>

HTML

```
1 <div id="avisos"></div>
2 <label>Introdueix un nombre:
3 <input type="text" id="entrada"/>
4 </label>
5 <button id="validar">Validar</button>
6 <button id="netejar">Netejar</button>
```

empty()

CSS

```
1 .missatge {
2   border-radius: 5px;
3   padding: 10px;
4   font-size: 1.2em;
5 }
6 .error {
7   background-color: #f2dede;
8   color: #d9534f;
9   border: 1px solid #d9534f;
10 }
11 .exit {
12   background-color: #d9f2d9;
13   color: #5cb85c;
14   border: 1px solid #5cb85c;
15 }
16 .info {
17   background-color: #d9edf7;
18   color: #5bc0de;
19   border: 1px solid #5bc0de;
20 }
```

El següent exemple és molt més simple: es modificarà la funció `esborrarAvisos` per buidar el contingut del node contingut a l'objecte `$avisos` en lloc d'eliminar tots els seus fills un per un.

JS

```
1 $(document).ready(function() {
2   var $avisos = $('#avisos'),
3       $entrada = $('#entrada'),
4       $validar = $('#validar'),
5       $netejar = $('#netejar');
6   $validar.on('click', function(e) {
7     esborrarAvisos();
8     if ($.isNumeric($entrada.val())) {
9       afegirAvis('exit', 'Has introduït un nombre!');
10    } else {
11      afegirAvis('error', "S'ha d'introduir un nombre");
12    }
13    afegirAvis('info', 'El text introduït ha estat: ' + $entrada.val());
14  });
15  $netejar.on('click', function(e) {
16    esborrarAvisos();
17  });
18
19  function afegirAvis(tipus, missatge) {
20    var $ressaltat = $('<b></b>'),
21        $avis = $('<p></p>');
22    $avis.addClass('missatge');
23    $avis.addClass(tipus);
24    $ressaltat.text(tipus.toUpperCase() + ': ');
25    $avis.append($ressaltat)
26    $avis.append(missatge);
27    $avisos.append($avis);
28  }
29  function esborrarAvisos() {
30    $avisos.find('p').remove();
31  }
32  });
```