07

Desenvolupament Web en entorn Client

Cicle: Desenvolupament d'aplicacions web

Curs 2024-2025

1. Introducció

En aquest document introduirem la llibreria jquery.

2. Contingut

2.1. jQUERY.

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

jQUERY es una llibreria de javascript que pretén facilitar l'us de javascript a la teua pàgina web. «Escriu menys i fer mes».

Permet fer:

- simplificar el treball amb elements HTML/DOM.
- Manipular CSS (evitava els problemes de distints noms de les propetat css)
- AJAX
- Efectes i animacions
- nous events.

Aquesta llibreria ha sigut de les més utilitzades però està quedant-se arrere davant angular, veu, react, etc.

2.2. Incloure jQUERY a les nostres pàgines

Tenim distintes opcions:

- Incloure la pàgina des d'un CDN (**content delivery network** o **content distribution network**)

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.1/jquery.min.js"></script>
```

- Descarregar els arxius i referenciar-los des del nostre servidor (en el nostre cas xampp al vostre equip o al docker).

```
<script type="text/javascript" src="./jquery-3.6.1.min.js"></script>
```

Utilitzar l'opció de la descarrega per a no tindre problemes a l'examen perquè no està disponible internet.

2.3. Hola mon amb jQUERY

En aquest exemple afegim els events aun boto per a mostrar una finestra emergent amb un «hola mon».

La funció principal de la llibreria es $. admiteix distints valors com a paràmetre (document; identificador d'un objecte (#id), classe(.class), etiqueta (p), nom ('nom')). Esta funció retorna un objecte de tipus jquery (amb tota la funcionalitat que afegeix jquery). Si hi ha mes d'un element retornarà un vector d'objectes jquery.

El mètode ready. Este mètode s'executa quan tots els elements de la pàgina han sigut carregats. Passem el nom d'una funció sense parèntesis.

```
//referencia al document transformant-lo en un objecte jquery
//d'aquesta manera document incorpora el metode ready

var x;
x=$(document);
//quan esta tot carregat executem inicia
x.ready(inicia);


function inicia()
{
  //referencia al boto ------ #identificador_element (com a objecte jQUERY)
  var x;
  x=$("#boto");
  //amb la funcio click definim l'event click sobre el boto
  x.click(clickboto);
}

function clickboto(event)
{
  alert("Hola mon");
}
```

El nostre html simplement compte:

```
<!doctype html>
<html lang="es">
<head>
      <meta charset="utf-8">
      <title>jquery</title>
      <meta name="description" content="descripcio">
      <meta name="author" content="Jose Masip">
      <link rel="stylesheet" href="css/estilos.css">
      <script type="text/javascript" src="./jquery.min.js"></script>
      <script type="text/javascript" src="./02jquery.js"></script>
</head>

<body>
  <h1>jquery</h1>
  <p>hola mon amb jquery. al pulsar el boto</p>
  <input type="button" id="boto" value="hola mon">
</body>
</html>
```

2.4. jQUERY API.

Utilitzant el API de jQUERY podem conéixer les funcions disponibles i els seus paràmetres.

https://api.jquery.com/

També disposem d'exemples pràctics.

2.5. Alguns exemples.

Per a modificar propietats css utilitzem el metode css.

Objectejquery.css('atribut', 'valor')

```
function clickboto()
{
  //obtindre referencia a titol i modificar el tipus de lletra i color
  var x;
  x=$("#titol");
  x.css("color","#ff0000");
  x.css("background-color","#e1e1e1");
  x.css("font-family","Courier");
}
```

Si volem definir un event o realitzar una tasca sobre més d'un element podem fer-ho com al següent exemple:

```
function inicia()
{
  //referencia als botons de la pàgina
  var x;
  x=$("input");
      //Definim tots els events al mateix temps. En la variable x tenim tots els botons.
      //amb la funcio click definim l'event click sobre tots els objectes en la variable x
  x.click(clickboto);
}
```

Per a modificar atributs d'un element:

Objectejquery.attr(nom propietat)--- mostra valor de la propietat

Objectejquery.attr(nom propietat,valor) ---- canvia el valor;

Objectejquery.removeAttr(nom propietat)---- elimina la propietat

```
//referencia al document
var x;
x=$(document);
//quan esta tot carregat executem inicia
x.ready(inicia);


function inicia()
{
  //referencia al boto ------ #identificador_element
  var x;
  x=$("#boto1");
  x.click(clickboto1)
  x=$("#boto2");
  x.click(clickboto2)
  x=$("#boto3");
  x.click(clickboto3)
}

function clickboto1()
{
      var x=$("#taula");
      x.attr('border',3);
}

function clickboto2()
{
      var x=$("#taula");
      if (x.attr("border")!=undefined){
         alert(x.attr("border"));
      }
      else{
         alert("No esta definida la propietat border");
      }
}

function clickboto3()
{
      var x=$("#taula");
      x.removeAttr('border');
}
```

Per a modificar o mostrar el text d'un element utilitzem el mètode text.

Objectejquery.text() ------ obtindre el text de l'element

Objectejquery.text('nou valor') ------ modificar el text de l'element

```
function clickboto()
{
	var x=$("#paragraf");
	//accedir al text
	alert(x.text());
	//modificar el valor
	x.text("Este es el nou text");
}
```

Per a modificar o mostrar el text d'un element utilitzem el metode text.

Objectejquery.html([bloc html]) ------ Ens permet agregar un bloc de codi html (innerHTML del DOM).

Objectejquery.*html() ----- Ens retorna el bloc html*

```
function clickboto1()
{
	var x=$("#capa");
	x.html("<h1>titol</h1><p>paragraf</p>");

	//afegir al valor anterior
	//x.html(x.html()+"<h1>titol</h1><p>paragraf</p>");
}

function clickboto2()
{
	var x=$("#capa");
	alert(x.html()); //mostra etiquetes
	alert(x.text()); // mostra el text sense les etiquetes
}
```

2.6. Nous events.

mouseover i mouseout es solen utilitzar junts. jQuerydefineix l'event hover que els relaciona

```
$(element).hover([funcio entra el ratoli],[funcio ix el ratoli])
```

la primera funció s'executa amb el mouseover. la segona amb el mouseout.

```
var x;
x=$(document);
//quan esta tot carregat executem inicia
x.ready(inicia);

function inicia()
{
  //referencia al boto ------ #identificador_element
  var x;
  x=$("#capa");
  x.hover(roig,blau);
}
function roig()
{
      var x=$("#capa");
      x.css("background-color","red");
}

function blau()
{
      var x=$("#capa");
      x.css("background-color","cyan");
}
```

2.7. DOM.

Alguns mètodes per a la manipulació del DOM:

- x.append("llista"); ----- afegir al final
- x.prepend("llista"); ----- afegir al principi
- x.remove() ------ elimina el element x
- x.empty(); ------ elimina tots els elements

Exemple per a eliminar el segon element:

```
  var x;
        //obtenim tots els elements
  x=$("element");
        //mirem quants elements tenim
  var cantidad=x.length;
        //indiquem l'element que volem borrar.
        //eq retorna l'element jquery de la posicio indicada
  var y=x.eq(cantidad-2);
  y.remove();
```

2.8. Efectes.

show() ---- mostra un element que estava ocult.

show("fast")
show("normal")
show("slow")
show(milisegons)
show(milisegons,[funcio]) ---- oculta tardant els mil·lisegons indicats i després executa la funció.

Hide () ---- acepta les mateixes opcions.

```
var x;
x=$(document);
x.ready(inicia);

function inicia()
{
  var x;
  x=$("#boto");
  x.click(mostrarcapa)
  x=$("#boto1");
  x.click(ocultarcapa1);
  x=$("#boto2");
  x.click(ocultarcapa2);
  x=$("#boto3");
  x.click(ocultarcapa3);
  x=$("#boto4");
  x.click(ocultarcapa4);
}
```

```
function mostrarcapa()
{
	var x=$("#capa");
	x.show("fast");
}

function ocultarcapa1()
{
	var x=$("#capa");
	x.hide("fast");
}

function ocultarcapa2()
{
	var x=$("#capa");
	x.hide("normal");
}

function ocultarcapa3()
{
	var x=$("#capa");
	x.hide("slow");
}

function ocultarcapa4()
{
	var x=$("#capa");
	x.hide(6000,function(){alert("CAPA OCULTA");});
}
```

fadeIn() ---- mostra un element que estava ocult.

fadeOut() ----- oculta element.

Les dos accepten les mateixes opcions que el show().

```
function ocultarcapa3()
{
	var x=$("#capa");
	x.fadeOut("slow");
}

function ocultarcapa4()
{
	var x=$("#capa");
	x.fadeOut(6000);
}
```

fadeTo() ----modifica la opacitat d'un element.

Passa del estat actual fins a la indicada.
Encara que siga transparent segueix ocupant l'espai

fadeTo(velocitat, valor)
velocitat =slow/normal/fast/milisegons
opacitat= número entre 0 i 1. 0 es transparent

```
function opacitat1()
{
	var x=$("#capa");
	x.fadeTo("fast",1);
}

function opacitat2()
{
	var x=$("#capa");
	x.fadeTo("normal", 0.5);
}

function opacitat3()
{
	var x=$("#capa");
	x.fadeTo("slow", 0);
}

function opacitat4()
{
	var x=$("#capa");
	x.fadeTo(6000, 1);
}
```

2.9. Iteració en tots els elements.

each permet iterar amb tots els elements del vector o objecte jquery.

Objectejquery.each(funcio) ----- executarem la funció sobre cadascun dels elements.

Al següent exemple mostrem el text dels paràgrafs de la pàgina:

```
function clickboto()
{
	var x=$("p");
	x.each(function (){
		var x=$(this);
		alert(x.text());
	});
}
```

2.10. ajax.

Vorem una introducció als mètodes existents per a fer peticions ajax a la llibreria jquery.

Mes informació a *https://api.jquery.com/category/ajax/*

El primer mètode que podem utilitzar es el **load**. Aquest mètode carrega les dades des del servidor i modifica el HTML de l'objecte des de que hem invocat la petició ajax.

load(pagina, paràmetres, funció final)

```
function clickboto()
{
	var x=$("#capa");
	//automàticament substitueix el text de la capa amb el contingut de l'arxiu
	x.load("./ajax/hola.xml");
	alert(x.html());
}
```

Arxiu xml

```
<hola>hola mon</hola>
```

Si verifiquem el nostre codi executant la petició ajax vorem que pareix que funciona adequadament a la capa. Però al verificar la finestra emergent vegem que ens apareixen les etiquetes del XML que el navegador interpreta com a etiquetes creades per nosaltres com permet HTML5. Com aquestes etiquetes

no tenen definit cap format especial no representa res. El load sols serveix per a carregar les dades i mostrar-les directament sense tindre que formatar-les.

El mètode **get** realitza una petició de dades al servidor a traves d'enviar les dades al servidor utilitzant el mètode del mateix nom.

```
$.get(nom pagina,parametres a enviar, funcio que rep les dades del
servidor)
jQuery.get(nom pagina,parametres a enviar, funcio que rep les dades
del servidor)
```

A l'exemple enviem una variable 'variable' al servidor que generarà les dades oportunes i les retornara; a la funció repDades les formataríem i les mostraríem per pantalla. Codi:

```
function clickboto()
{
   //$.get("ajax/20ajax.php", "variable=1" ,repDades);
   $.get("ajax/20ajax.php?variable=3", "" ,repDades);
}

function repDades(d)
{
    alert(d);
}
```

Idem al mètode anterior però per a enviar les dades per POST tenim **post**.

```
$.post(nom pagina,parametres a enviar, funcio que rep les dades del servidor)

jQuery.post(nom pagina,parametres a enviar, funcio que rep les dades)
```

Mateix exemple que amb el get:

```
function clickboto()
{
     $.post("ajax/21ajax.php", "variable=3" ,repDades);
}
```

```
function repDades(d)
{
      alert(d);
}
```

El mètode ajax es l'opció que ens dona mes possibilitats de configuració. A continuació podem vore algunes de les opcions que ens permet:

```
ajax([objecte literal]) --------Retorna un objete XMLHttpRequest

$.ajax({
      async:true,           //asíncrona (true) / síncrona (false)
      type: "POST",          //mètode enviem les dades (get/post)
      dataType: "html",      //tipus de dades vas a recuperar ("html","xml","json","script")
      contentType: "application/x-www-form-urlencoded",    //com empaquetem les dades per a enviar-les
      url:"pagina.php",           //pagina rep les dades
      data:"variable=3",           //dades enviem al servidor
       beforeSend:iniciEnviament,   //funció que s'executa abans d'enviar les dades al servidor
       success:RepDades,           //funció s'executa quan finalitza l'enviament de dades per part del servidor
(tot OK)
       timeout:4000,               //temps màxim que esperarem les dades
       error:problemes             //funció que executem si hi ha algun error
 });
```

Exemple equivalent al del load:

```
$.ajax({url: "hola.xml", async: false, success: function(result){
      $("div").html(result);
    }});
```

També ens podem trobar amb la següent opció per a fer la petició ajax:

```
$.ajax( "example.php" )
    .done(function() {
          alert( "tot OK" );
     })
    .fail(function() {
           alert( "error" );
    })
    .always(function() {
          alert( "completat" );
    });
```

2.11. Tractar XML amb jquery.

Com tenim que treballar amb un arxiu XML i no podem utilitzar el getElementsByTagName() com feiem a ajax.

El Php genera un xml on es substituirien les variables per les dades de la BD.

```
$xml="<?xml version=\"1.0\"?>\n";
$xml.="<persona>\n";

$xml.="<nom>".$nom."</nom>\n";
$xml.="<cognom>".$cognom."</cognom>\n";
$xml.="<direccio>".$direccio."</direccio>\n";

$xml.="</persona>\n";

echo $xml;
```

Per a imitar el getElementsByTagName podem utilitzar la funció find de jquery. El problema que tenim es que les dades que rebem no son un objecte jquery és un xml. Per tant, cal transformar aquestes a jquery.

```
alert(d); // mostra el xml

//  alert(d.find("nom").html()); error -----> d no te la funcio find per no ser jquery

dades=$(d); //transformar xml en objecte jquery

alert(dades.find("nom").html());
```

El nostre arxiu jquery a la funció que rep les dades cal transformar el XML a jquery com hem dit.

```
function repDades(d)
{
      dades=$(d); //transformar xml en objecte jquery
   /* si tinguerem mes d'una persona fariem un each
   $.each(dades, function(i, field){
       });*/
      $("#detalls").html("nom:"+dades.find("nom").html()+"<br>Cognom:"
+dades.find("cognom").html()+"<br>Direccio:"+dades.find("direccio").html());
}
```

2.12. **Animacions.**

El mètode de jQUERY animate() ens permet crear animacions personalitzades. Aquesta funcionalitat ens la oferia abans de que es poguera fer mitjançant css.

Sintaxis:

ObjectejQUERY.animate({params},speed,callback);

El primer paràmetre ens serveix per a establir la propietat css que volem animar. Amb el segon paràmetre opcional especifiquem la duració de l'efecte (mil·lisegons, low, fast). El tercer paràmetre opcional és la funció a executar una volta finalitzat l'efecte o animació.

Contingut del html:

```
<!doctype html>
<html lang="es">
<head>
      <meta charset="utf-8">
      <title>jquery</title>
      <meta name="description" content="descripcio">
      <meta name="author" content="Jose Masip">
      <link rel="stylesheet" href="css/estilos.css">
      <script type="text/javascript" src="./jquery-3.6.0.min.js"></script>
      <script type="text/javascript" src="./27jquery.js"></script>
</head>

<body>
  <h1>AJAX jquery</h1>
      <input type="button" id="boto1" value="animate left 250 ">
```

```
    <input type="button" id="boto2" value="animate +50">
    <input type="button" id="boto3" value="animate 2 valors"><br>
    <input type="button" id="boto4" value="animate +50 slow ">
    <input type="button" id="boto5" value="animate +50 6000ms"><br>

    <div id="capa" style="position:absolute;"><h1>ANIMATE</h1></div>
</body>
</html>
```

Contingut de 27jquery.js

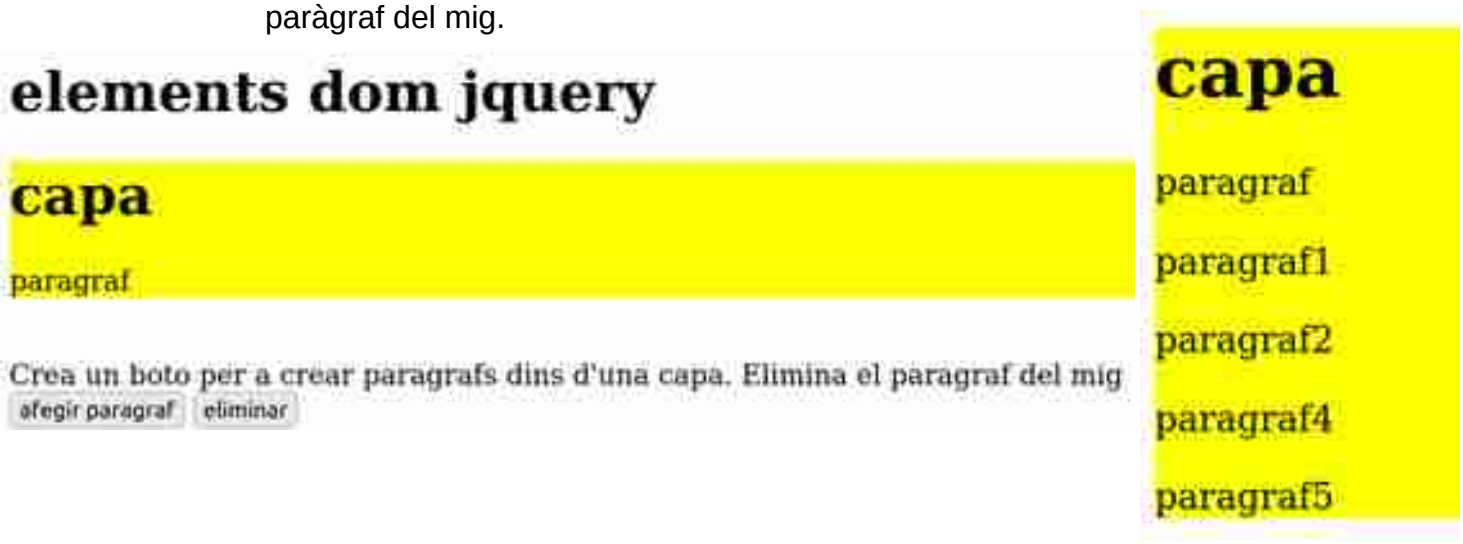
```
var x;
x=$(document);
x.ready(inicia);

function inicia()
{
  var x;
  x=$("#boto1");
  x.click(clickboto1);

  x=$("#boto2");
  x.click(clickboto2);

  x=$("#boto3");
  x.click(clickboto3);

  x=$("#boto4");
  x.click(clickboto4);

  x=$("#boto5");
  x.click(clickboto5);
}

function clickboto1()
{
  $("#capa").animate({left:'250px'});
}

function clickboto2()
{
  $("#capa").animate({left:'+=50px'});
}
```

```
function clickboto3()
{
  $("#capa").animate({top:'+=50px', left:'+=50px'});
}

function clickboto4()
{
  $("#capa").animate({left:'+=50px'},"slow");
}

function clickboto5()
{
  $("#capa").animate({left:'+=50px'},6000);
}
```

Jquery permet encadenar funcions de manera que reduïm les línies de codi.

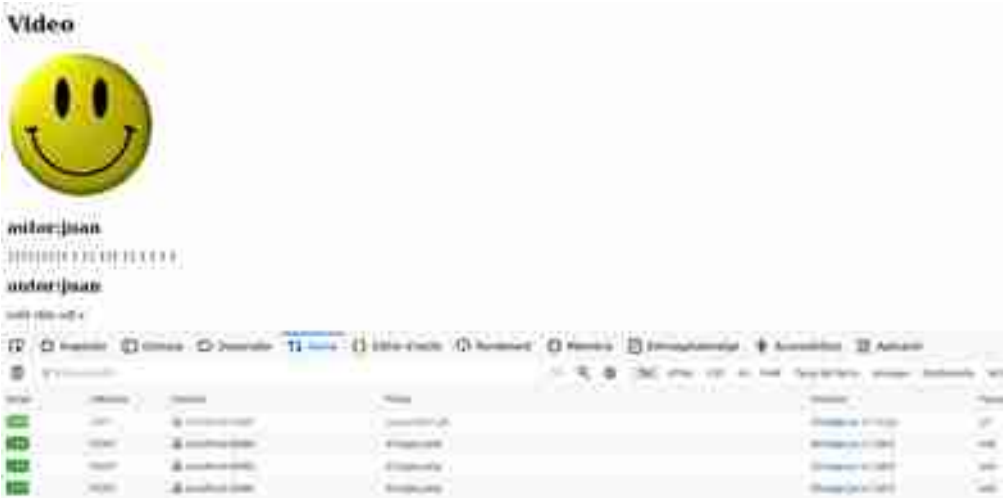
```
  $("#capa").animate({left:'+=50px'},300).animate({top:'+=50px'},1500);
```

3. **Activitats.**
   - E01. Crea un boto per a crear paràgrafs dins d'una capa. Elimina el paràgraf del mig.



   - E02. Canvia la mida del text a tots els paragrafs de la pàgina (cada volta 2 px mes).

- E03. Mostrar els comentaris d'un video. Aquest s'actualitzen cada 20 segons amb una petició ajax (importar BD comentaris_ajax)



- E04. Volem saber els horàries disponibles de tren des d'un origen a un destí. Per a això omplim el primer select des de php amb una consulta a la bd que ens trau tots els valors d'origen distints. Al seleccionar un origen averiguarem tots els destins disponibles des d'aquest a traves d'una petició ajax. Si després elegim un destí tindrem tots els horàries des d'aquest origen a aquest destí. Retornar les dades amb json (recomanació crear-lo manualment per a entendre millor com funciona)
  Recomanació: Com hi ha dos peticions ajax crear dos php distints que generaran els jsons per a separar possibles errors.

- E05. Exercici del parking realitzat a un document anterior. A aquesta Versió les dades les obtenim des de la base de dades. Retornarem les dades en format XML (enviar sempre la mínima quantitat d'informació possible).
  - Al seleccionar un client farem una petició ajax per averiguar les matricules dels seus cotxes i omplirem el select corresponent.
  - Al seleccionar una matricula omplirem els detalls del cotxe
  - **AMPLIACIÓ:** implementar el camp per buscar client amb opció d'autocompletar (si escric 2 al camp mostrar tots els clients que comencen per 2, si escric 20 mostrar tots els clients que comencen per 20)
  - **AMPLIACIÓ:** implementar el camp per buscar per matricula.

- E06. Idem a l'anterior retornant les dades en json.

4. Activitats entregables.

Entrega les següents activitats:

- E02
- E04
- E05 (Sols averiguar els cotxes d'un client)

5. Bibliografia.

- https://www.w3schools.com/jquery/default.asp
- https://jquery.com/
- https://api.jquery.com/

6. Aclariments

No intentes copiar exercicis d'altres companys o internet. ni tans sols pegar una miradeta. Si que pots consultar la sintaxis d'una funció, etc. Però no et serveix per a res intentar buscar la solució.

És el major error que pot realitzar qui comença a programar. Sols et deixarà una falsa sensació d'aprenentatge i després no sabràs solucionar els problemes per tu mateix.

Cal intentar que el resultat que genere el vostre codi siga el mes paregut possible a l'exercici (si hi ha imatge d'exemple).

Com els Navegadors no son compatibles entre sí al 100% jo corregire els examens amb FIREFOX. Per tant, es recomanable utilitzar aquest per als nostres exercicis.

Crea uns arxius per a cada exercici o apartat. Així un error a un no afectarà a la resta.

Per a entregar els exercicis crear una carpeta amb el vostre nom i comprimiu aquesta (el nom de l'arxiu comprimit serà elteunom.zip o elteunom.tar.gz). Si entregueu arxius solts o arxius genèrics sense identificar el vostre nom puc confondre'm i no puntuar-vos algun exercici al no saber de qui és.

Cal entregar tots els arxius necessaris per a l’exercici (fotos, llibreries, etc).