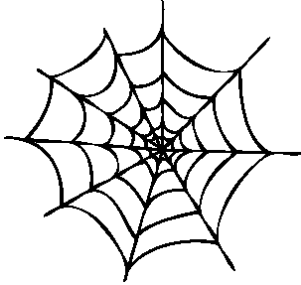
Universitat Autònoma de Barcelona



TECNOLOGIES DE DESENVOLUPAMENT PER A INTERNET I WEB

GRAU D'ENGINYERIA INFORMÀTICA - CURS 2019/2020

TEMA 1 – *LLENGUATGES DE PROGRAMACIÓ WEB*

0. INTRODUCCIÓ

1. LLENGUATGES DE PROGRAMACIÓ WEB

1.1 – Documents web

1.2 – Programació a la banda del client

1.3 – Programació a la banda del servidor

1.4 – Arquitectura model vista controlador: descripció i ús

1.5 – Aspectes de seguretat

2. EL PROTOCOL HTTP

3. PROTOCOLS DE SERVEIS

1. LLENGUATGES DE PROGRAMACIÓ WEB

1.1 – Documents web

- **Hypertext Markup Language (HTML)**
- **Cascading Style Sheets (CSS)**

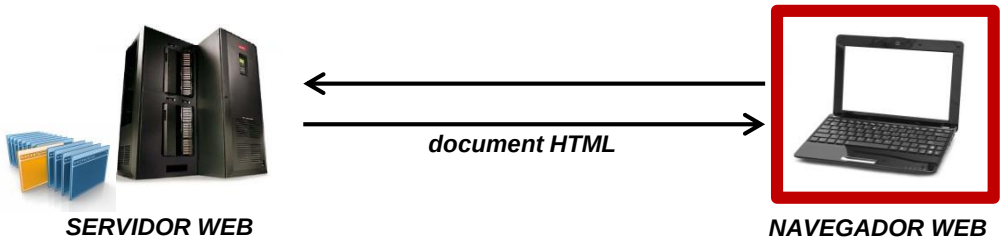
1.2 – Programació a la banda del client

1.3 – Programació a la banda del servidor

1.4 – Arquitectura model vista controlador: descripció i ús

1.5 – Aspectes de seguretat

1.1 Documents web: HTML. Introducció



La web està basada en dos blocs principals:

- El *navegador web* és una aplicació que l'usuari invoca per accedir i visualitzar una pàgina web, esdevenint un client web.
- El *servidor web* és un ordinador que emmagatzema una o vàries pàgines HTML i que respon als clients web.

Els documents transmesos són principalment HTML, que és un fitxer de text amb uns elements anomenats "tags", que transporten els continguts i també poden donar indicacions de la visualització del document. Els "tags" poden tenir atributs. Hi ha "tags" sense contingut (elements buits).

<nomtag atribut="valor"> continguts </nomtag>

<nomtag atribut="valor" />

1.1 Documents web: HTML. Evolució

1991 → **HTML 1.0**

- La primera versió de HTML debuta com un híbrid de SGML amb el "tag" ***href*** per proveir enllaços a altres documents.

"The WorldWideWeb (WWW) project aims to allow all links to be made to any information anywhere"

1995 → **HTML 2.0**

- Primer estàndard base de la web. Banc de proves per l'explosió de la web.
- Mosaic Communications allibera *Netscape*, el primer navegador web gràfic.

1996 → **HTML 2.0**

- S'introdueix el suport per taules, "image maps", i el CSS esdevé una recomanació de W3C.
- Microsoft allibera l'*Internet Explorer 3.0,* amb característiques similars al Netscape.

1.1 Documents web: HTML. Evolució

1997 → **HTML 3.2**

- S'aprova la nova versió de HTML després d'una llarga batalla entre investigadors que creuen que els atributs de text (com tipus i mides de font o colors) allunyen HTML del seu objectiu inicial d'organitzar informació i no de decorar-la.

- Netscape i Explorer protagonitzen una guerra de navegadors, durant la qual Microsoft integra Explorer a Windows (més tard perdria un judici anti-monopoli).

1999 → **HTML 4.01**

- S'aprova la nova versió de HTML, amb suport complert per CSS.

- Explorer esdevé el navegador més popular del mercat.

2000 → **XHTML**

- HTML i XML s'ajunten per crear un nou estàndard amb un codi més estructurat.

- No té èxit perquè:

1) no és compatible amb versions anteriors
2) la majoria de navegadors permeten documents XHTML mal estructurats.

1.1 Documents web: HTML. Evolució

2002

- El disseny de moltes webs comença a ser molt mal estructurat.
- Apareixen noves idees per la compartició i transmissió de continguts com weblogs i RSS: Web 2.0.
- Explorer esdevé el navegador més popular, apagant la innovació.

2005

- AJAX revitalitza el disseny de pàgines web, permeten noves funcionalitats a aplicacions com Gmail o Facebook.
- Mozilla (una comunitat de programadors de Netscape) allibera Firefox, i reapareix la guerra de navegadors.

2008

- Descontents amb XHTML, es forma el WHATWG grup amb membres de Mozilla, Apple i Opera per crear el primer "draft" de HTML5.
- Google allibera *Chrome*.

1.1 Documents web: HTML. Evolució

2010 → *2014*

- Apple deixa de donar suport a Flash i pressiona HTML5 perquè esdevingui un estàndard.
- FireFox esdevé el navegador més popular seguit de l'*Explorer*.

HTML5

- S'aprova el nou estàndard, que millora significativament els seus predecessors:

1) Afegeix elements semàntics (header, footer,...)
2) Incorpora nous atributs pels formularis (time, calendar,...)
3) Introdueix elements gràfics (svg i canvas) i multimèdia (àudio i vídeo)
4) Es creen noves APIs per geolocalització, "drag and drop", "local storage", etc.
5) Es defineix com un "living standard", que permet afegir nous elements sense deixar obsolets els vells

¡PREGUNTA D'EXAMEN!

Quins inconvenients tenien les versions anteriors a HTML5?

1.1 Documents web: HTML. Estructura

```
<!DOCTYPE html>
<html>
    <head>
        …
    </head>
    <body>
        …
    </body>
</html>
```

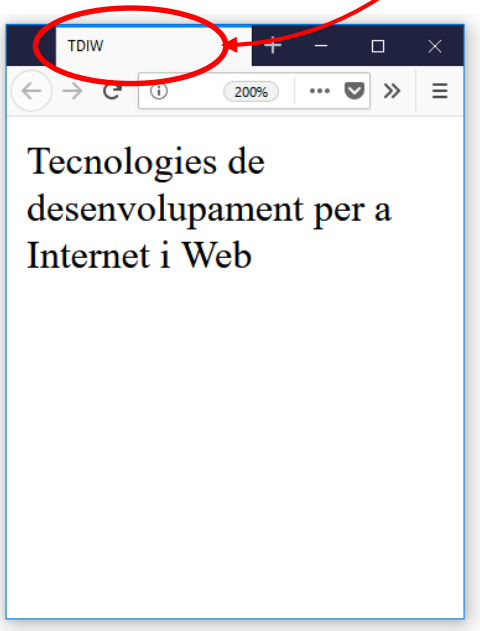
Elements estructurals:

- DOCTYPE: defineix que el document és HTML5
- HTML: element arrel
- HEAD: conté meta-informació del document
- BODY: conté el contingut del document

1.1 Documents web: HTML. Elements

Elements de meta-informació:

```
<!DOCTYPE html>
<html>
    <head>
        …
    </head>
    <body>
    </body>
</html>
```



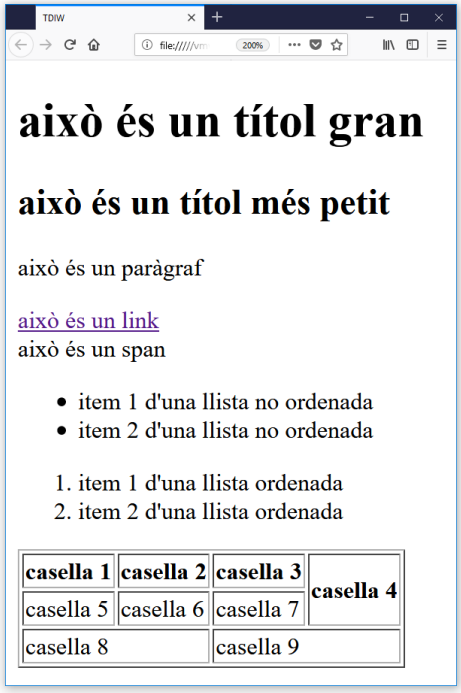
l'editor també ha d'estar configurat per UTF-8

```
<meta charset="UTF-8" />
<title>TDIW</title>
<meta name="author" content="autor de la pàgina" />
<meta name="description" content="descripció de la pàgina" />
<meta name="keywords" content="paraules clau de la pàgina" />
<meta name="robots" content="(no)index, (no)follow" />
<meta name="viewport" content="width=device-width; initial-scale=1.0" />
<link rel="stylesheet" type="text/css" href="estil.css" />
<script type="text/javascript" src="funcions.js"> </script>
```

1.1 Documents web: HTML. Elements

Elements amb informació:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

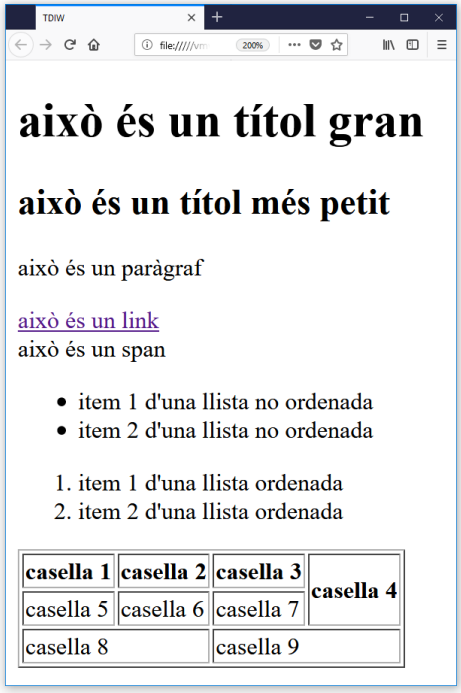



```
<h1> això és un títol gran </h1>
<h2> això és un títol més petit </h2>
<p> això és un paràgraf </p>
<a href="http://www.uab.cat" target="_self|_blank"> això és un link </a> <br />
<span> això és un span </span>
<ul>
    <li> item 1 d'una llista no ordenada </li>
    <li> item 2 d'una llista no ordenada </li>
</ul>
<ol>
    <li> item 1 d'una llista ordenada </li>
    <li> item 2 d'una llista ordenada </li>
</ol>
```

1.1 Documents web: HTML. Elements

Elements amb informació:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

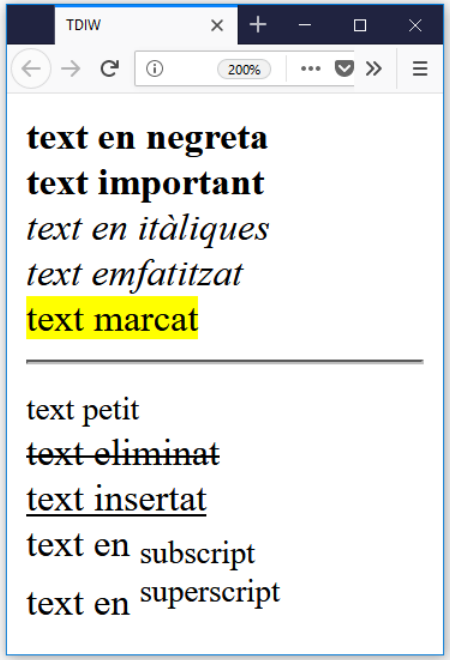



```
<table border="1">
    <tr>
        <th> casella 1 </th>
        <th> casella 2 </th>
        <th> casella 3 </th>
        <th rowspan="2"> casella 4 </th>
    </tr>
    <tr>
        <td> casella 5 </th>
        <td> casella 6 </th>
        <td> casella 7 </th>
    </tr>
    <tr>
        <td colspan="2"> casella 8 </th>
        <td colspan="2"> casella 9 </th>
    </tr>
</table>
```

1.1 Documents web: HTML. Elements

Elements de format:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```




```
<b> text en negreta </b> <br />
<strong> text important </strong> <br />
<i> text en itàliques </i> <br />
<em> text emfatitzat </em> <br />
<mark> text marcat </mark> <br />
<hr />
<small> text petit </small> <br />
<del> text eliminat </del> <br />
<ins> text insertat </ins> <br />
text en <sub> subscript </sub> <br />
text en <sup> superscript </sup> <br />
```

- Els "tags" de format es solen utilizar dins altres "tags", per exemple:

```
<p> quan volem <em>emfatitzar</em> o <del>eliminar</del>
paraules </p>
```

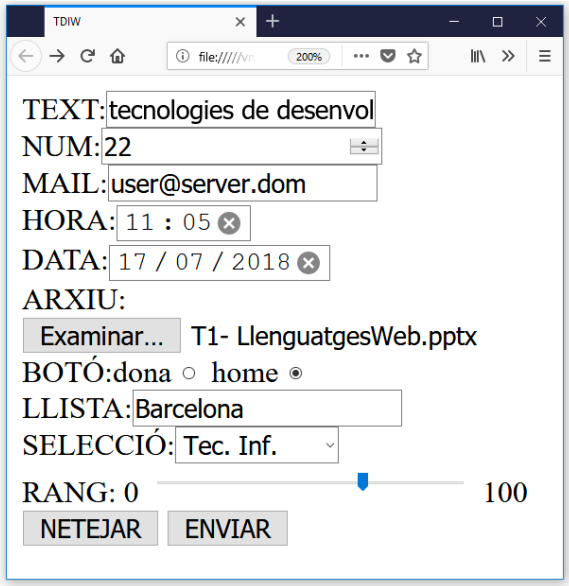
¡PREGUNTA D'EXAMEN!

Què són els elements "inline" i de "block-level" de HTML5?

1.1 Documents web: HTML. Elements

Elements de formulari:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

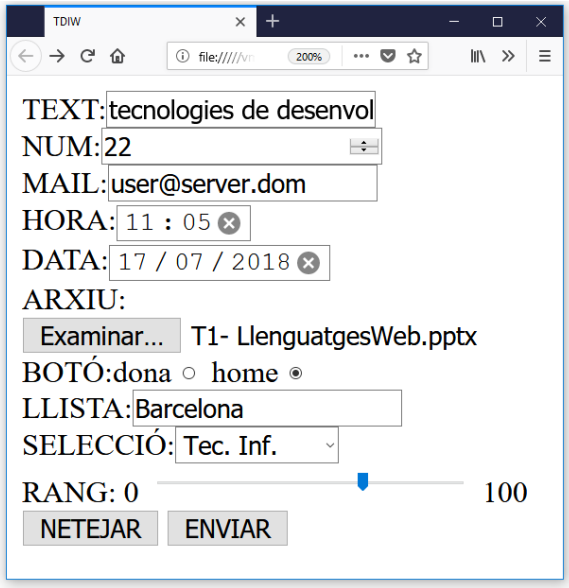



```
<form action="pag_accio.php" target="_self|_blank"
method="get|post">
    TEXT:<input type="text" name="nom"><br />
    NUM:<input type="number" name="edat"
    value="20"><br />
    MAIL:<input type="mail" name="mail"><br />
    HORA:<input type="time" name="hora"><br />
    DATA:<input type="date" name="dia"><br />
    ARXIU:<input type="file" name="arxiu"><br />
    BOTÓ:dona<input type="radio" name="genere"
    value="dona" checked> home <input type="radio"
    name="genere" value="home"><br />
```

1.1 Documents web: HTML. Elements

Elements de formulari:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```



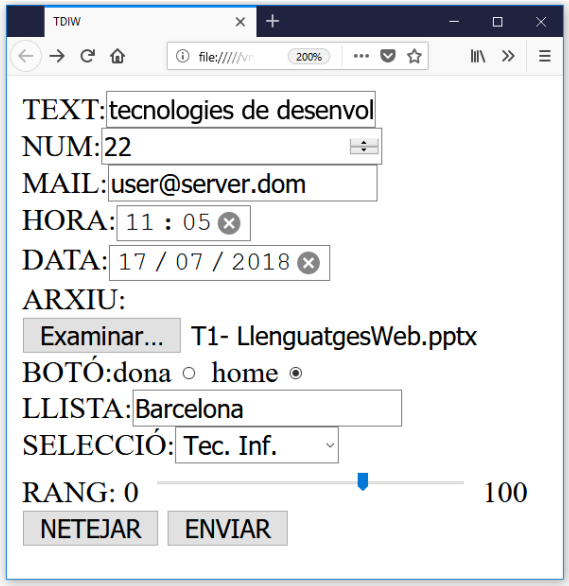

```
LLISTA:<input list="poblacio">
<datalist id="poblacio">
    <option value="Barcelona">
    <option value="Sabadell">
    <option value="Terrassa">
</datalist><br />

SELECCIÓ:<select name="mencio">
    <option value="TI">Tec. Inf.</option>
    <option value="EC">Eng. Comp.</option>
    <option value="ES">Eng. Soft.</option>
    <option value="C">Computació</option>
</select><br />
```

1.1 Documents web: HTML. Elements

Elements de formulari:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

```
        RANG: 0 <input type="range" name="satisfaccio"
        value="50" min="0" max="100"> 100<br />

        <input type="reset" value="NETEJAR">
        <input type="submit" value="ENVIAR">

    </form>
```



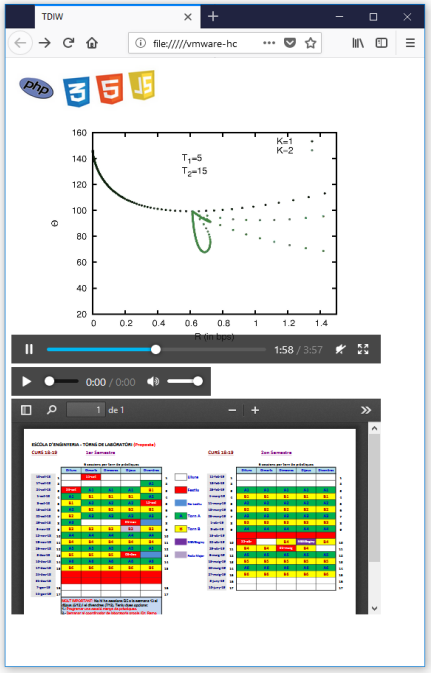
¡PREGUNTA D'EXAMEN!

Com es pot provar un formulari HTML sense enviar la informació a un servidor web?

1.1 Documents web: HTML. Elements

Elements multimèdia:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

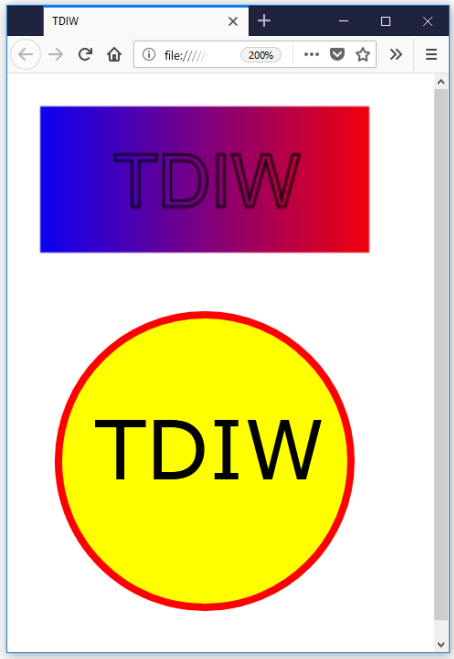



```
<img src="imatge.jpg" alt="text al." width="200" /><br />

<video width="500" controls>
    <source src="performance.mp4" type="video/mp4">
    <source src="performance.ogg" type="video/ogg">
    Navegador sense suport per vídeo.
</video><br />

<audio controls>
    <source src="entrevista.ogg" type="audio/ogg">
    <source src="entrevista.mp3" type="audio/mpeg">
   Navegador sense suport per àudio.
</audio><br />

<object width="500" height="300" data="labs.pdf"> </object>
```

1.1 Documents web: HTML. Elements

Elements gràfics:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```



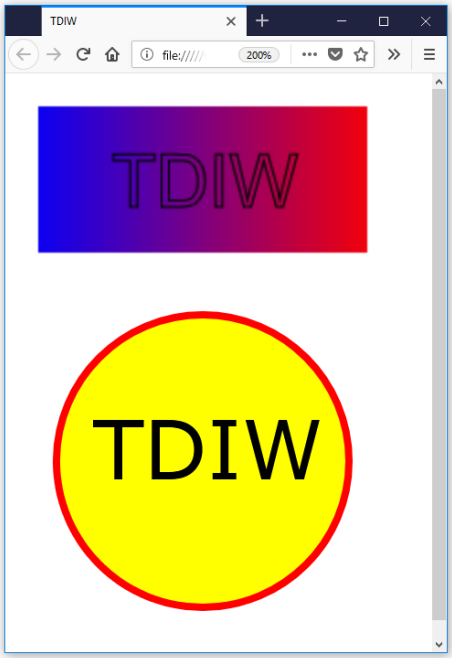

```
<canvas id="dibuix" width="200" height="100">
    Navegador sense suport per canvas.
</canvas>

<script>
    var c = document.getElementById("dibuix");
    var ctx = c.getContext("2d");
    var grd = ctx.createLinearGradient(0, 0, 200, 0);
    grd.addColorStop(0, "blue");
    grd.addColorStop(1, "red");
    ctx.fillStyle = grd;
    ctx.fillRect(10, 10, 180, 80);
    ctx.font = "40px Arial";
    ctx.strokeText("TDIW", 50, 65);
</script>
```

1.1 Documents web: HTML. Elements

Elements gràfics:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

```
<svg width="200" height="200">
    <circle cx="100" cy="100" r="80" stroke="red"
    stroke-width="4" fill="yellow" />
    <text fill="#000000" font-size="45" font-
    family="Verdana" x="40" y="110">TDIW</text>
</svg>
```



Diferències entre Canvas i SVG:

Canvas	SVG
Dibuixa gràfics amb JavaScript	Descriu dibuixos utilitzant XML
Renderitzat píxel a píxel. Canvis de posició o mida han de renderitzar de nou el gràfic	Ideal per aplicacions amb àrees de renderitzat molt grans (e.g., Google maps)
Renderitzat de text pobre	Renderitzat lent (gràfics complexes)
Ideal per jocs gràfics	No recomanat per jocs
Sense suport per events	Els elements estan en el DOM (s'hi poden enllaçar events)

¡PREGUNTA D'EXAMEN!

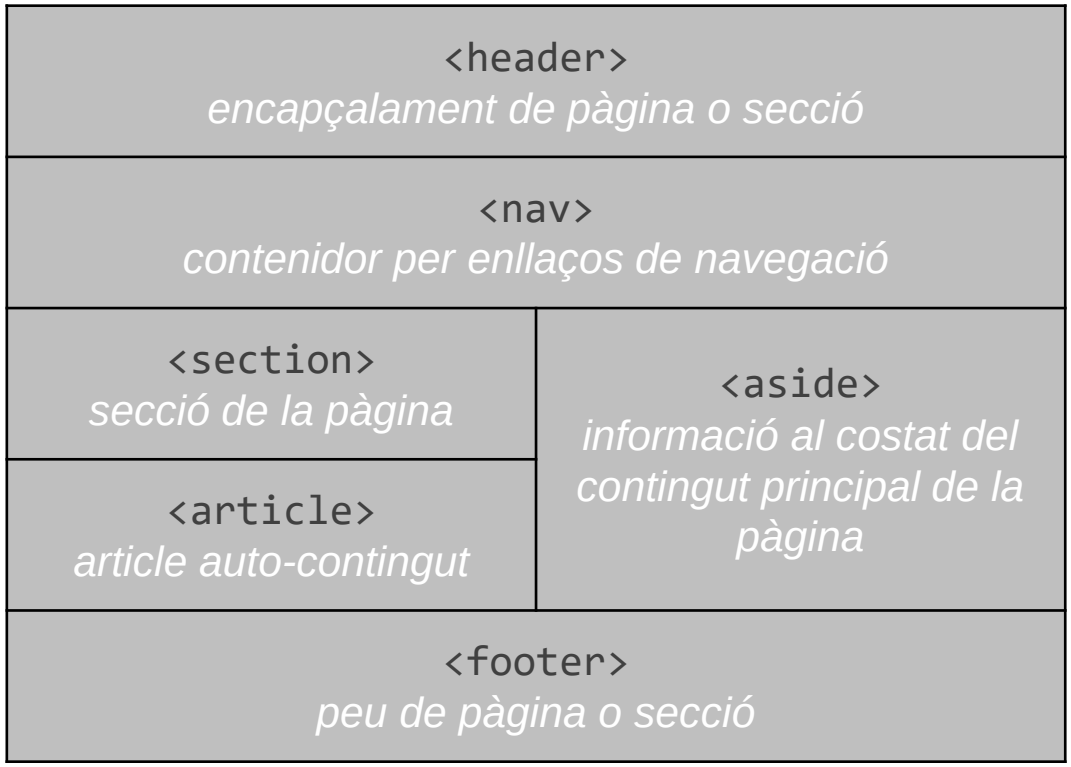
Es poden crear jocs gràfics amb SVG?

1.1 Documents web: HTML. Elements

Elements semàntics i "layout":

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

HTML5 incorpora nous elements semàntics per definir el contingut de diferents parts d'una pàgina. En general, estan pensats per organitzar la pàgina de la següent forma (però es poden canviar):



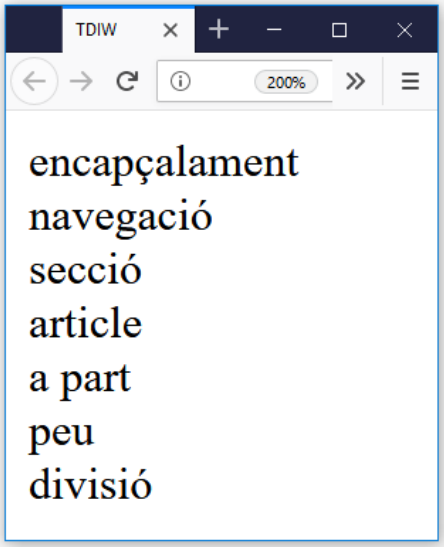
1.1 Documents web: HTML. Elements

Elements semàntics i "layout":

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

```
<header> encapçalament </header>
<nav> navegació </nav>
<section> secció </section>
<article> article </article>
<aside> a part </aside>
<footer> peu </footer>
<div> divisió </div>
```

no és un element semàntic!
pot servir per definir el "layout"



El "layout" no està definit per defecte. S'ha de programar. Existeixen vàries formes de fer-ho:

- ✗ ***Taules:*** no recomanable.No està pensat pel "layout".
- ***"CSS framework":*** fàcil però restringits a la definició.
- ***"CSS floats":*** lligats al flux del document. Poc flexibles i en alguns casos amb resultats inesperats.
- ✓ **"CSS grid":** assegura que tots els elements es mostren correctament en qualsevol mida de pantalla.
- ✓ ***"CSS flexbox":*** assegura que tots els elements es mostren correctament en qualsevol mida de pantalla.

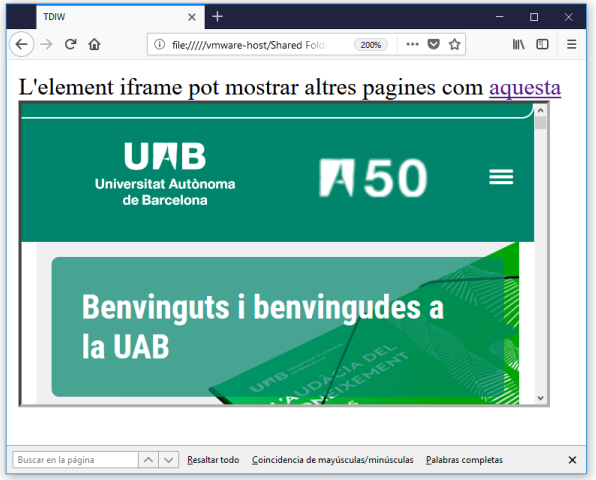
1.1 Documents web: HTML. Elements

Element <iframe>:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

L'<iframe> és un element que permet mostrar una pàgina web dins una altra pàgina. Ens pot servir pel "layout".

```
L'element iframe pot mostrar altres pàgines com
<a href="http://www.uab.cat" target="subPagina">aquesta</a>

<iframe src="./tdiw.html" name="subPagina" height="200"
width="350"> </iframe>
```



¡PREGUNTA D'EXAMEN!

Quines avantatges té l'ús d'elements semàntics en HTML5?

¡PREGUNTA D'EXAMEN!

Els elements de "layout" són els únics elements semàntics d'HTML5?

¡PREGUNTA D'EXAMEN!

L'element és un element semàntic?

1.1 Documents web: HTML. Atributs

Atributs més habituals:

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        …
    </body>
</html>
```

Molts elements d'HTML tenen atributs per proporcionar més informació sobre l'element. Els atributs sempre s'especifiquen en el "tag" d'obertura en la forma

nomAtribut="valor"

Alguns dels que ja hem vist són els següents:

```
<a href="http://www.uab.cat">
<input type="text" >
<img src="tdiw.jpg" alt="text alternatiu">
<svg width="200" height="200">
…
```

D'altres habituals són els següents:

```
<html lang="en-US">
<p title="descripció emergent">
<div id="idUnic">
<p style="color:blue; text-align:center;">
<p class="classeCSS">
```

Defineix el llenguatge de la página (útil per bots i accessibilitat)

Dóna un identificador únic a l'element (útil per CSS i JavaScript)

Ús no recomanat! Millor donar estil a través d'un fitxer CSS i l'atribut "class"

es pot utilitzar per navegar per la página:
enllaç

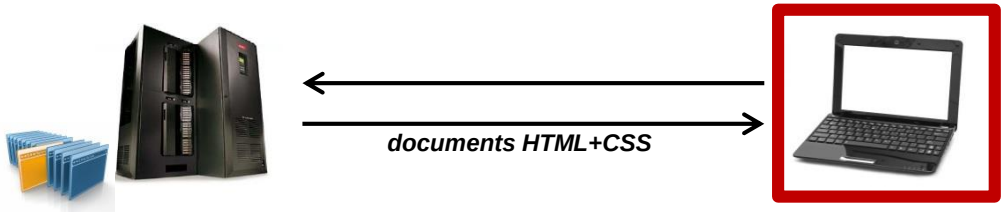
¡PREGUNTA D'EXAMEN!

Què són els atributs específics, globals i d'events?

1.1 Documents web: CSS. Introducció

HTML

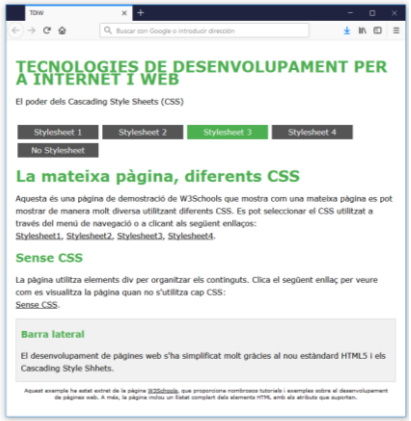
+CSS



La principal funció dels documents HTML és el transport del contingut (i.e., la informació) de la pàgina, però no haurien d'especificar res sobre com es visualitza.

CSS descriu com els continguts d'una pàgina HTML s'han de visualitzar en el navegador.

Facilita molt la feina de desenvolupament i manteniment de pàgines web: un sol fitxer CSS permet definir l'estil de totes les pàgines d'una web. Canviant aquest fitxer, canvia l'estil de *totes* les pàgines.



¡PREGUNTA D'EXAMEN!

A la versió de HTML 3.2 s'incorporaren elements i atributs per canviar fonts i colors. Això esdevingué un maldecap pels desenvolupadors. Per què?

1.1 Documents web: CSS. Introducció

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <p style="font-family=serif">paràgraf</p>
    </body>
</html>
```

CSS "inline"
Es posa dins els elements HTML.
Desavantatges: barreja contingut i visualització.

```
<!DOCTYPE html>
<html>
    <head>
        <style>
           codi CSS
        </style>
    </head>
    <body>
    </body>
</html>
```

CSS intern
Es posa dins un document HTML.
Avantatges: al moure el document HTML també l'acompanya l'estil.
Desavantatges: si es vol utilitzar l'estil a varis documents HTML, s'ha de copiar a tots, dificultant el manteniment.

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" type="text/css"
        href="estil.css" />
    </head>
    <body>
    </body>
</html>
```

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

CSS extern
Des del document HTML es referencia un fitxer .css, que conté les normes d'estil.
Avantatges: un mateix fitxer pot servir per múltiples pàgines. Fàcil de mantenir i reutilitzar.

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

El selector pot seleccionar un o més elements segons el seu:

- Tipus →

CSS
```
p { ... }
h1 { ... }
div { ... }
```

HTML
```
<p> paràgraf </p>
<h1> títol </h1>
<div> secció </div>
```

- Identificador únic →

```
#sec1 { ... }
#par1 { ... }
#imgWeb{ ... }
```

```
<section id="sec1"> secció </section>
<p id="par1"> paràgraf </p>
<img src="web.jpg" id="imgWeb" />
```

- Atribut →

```
img[alt] { ... }
a[target="_blank"] { ... }
[width*="px"] { ... }
   =  ~=  |=  ^=  $=  *=
```

```
<img src="web.jpg" alt="una web" />
<a href="" target="_blank">tdiw</a>
<img src="web.jpg" width="200px" />
```

- Classe →

```
.opcioMenu { ... }
.codi { ... }
```

```
<a href="" class="opcioMenu"> secció 1 </a>
<p class="codi"> x = 4; </p>
```

- Event →

```
a:hover { ... }
img:onClick { ... }
```

- O una combinació dels anteriors →

```
p.codi { ... }
img#imgWeb[alt] { ... }
h1, h2, p { ... }
li a:hover { ... }
```

*<a> dins un *

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

La parella de propietat/valor defineix com es visualitzarà la propietat CSS especificada. Algunes de les propietats més habituals són les següents:

Posició i mida	
position:	static, absolute, relative, fixed, sticky
right:	0px
left:	0px
top:	20px
bottom:	10px
width:	90%, 500px
max-width:	500px
height:	10%, 300px

Format i alineació de lletra	
color:	red, blue, green, ..., rgb(255, 0, 0), #FF0000
font-family:	serif, sans-serif, monospace
font-size:	2.5em, 40px
font-style:	normal, italic, oblique
font-weight:	normal, bold
text-align:	center, left, right
vertical-align:	middle, top, bottom

Fons i marges	
background-color:	lightblue
background-image:	url("tdiw.jpg")
background-repeat:	no-repeat, repeat-x
background-position:	right top, center top
background-attachment:	fixed, scroll
border-style:	dotted, dashed, double, ridge
border-top-style:	dotted
border-width:	10px, medium
border-color:	red
border-radius:	10px
padding:	5px
margin:	5px

Taules	
border-...	...
background-...	...
border-spacing:	10px
border-collapse:	separate, collapse
caption-side:	top, bottom

estil de caixa CSS

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

La parella de propietat/valor defineix com es visualitzarà la propietat CSS especificada. Algunes de les propietats més habituals són les següents:

Posició i mida	
position:	static, absolute, relative, fixed, sticky
right:	0px
left:	0px
top:	20px
bottom:	10px
width:	90%, 500px
max-width:	500px
height:	10%, 300px

Taules	
border-...	...
background-...	...
border-spacing:	10px
border-collapse:	separate, collapse
caption-side:	top, bottom

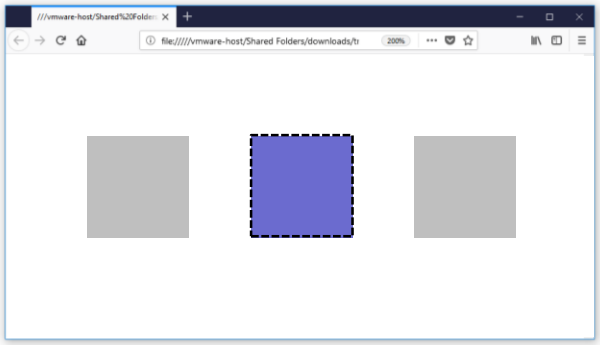
Format i alineació de lletra

font-fa
font
font-
font-w
text-
vertical-

Fons i marges

eat-x
r top

-static: *els elements es situen de forma estàtica i no es veuen influenciats per les propietats de "top", "bottom", "left", i "right". Opció per defecte.*



estil de caixa CSS

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

La parella de propietat/valor defineix com es visualitzarà la propietat CSS especificada. Algunes de les propietats més habituals són les següents:

Posició i mida	
position:	static, absolute, relative, fixed, sticky
right:	0px
left:	0px
top:	20px
bottom:	10px
width:	90%, 500px
max-width:	500px
height:	10%, 300px

Taules	
border-...	...
background-...	...
border-spacing:	10px
border-collapse:	separate, collapse
caption-side:	top, bottom

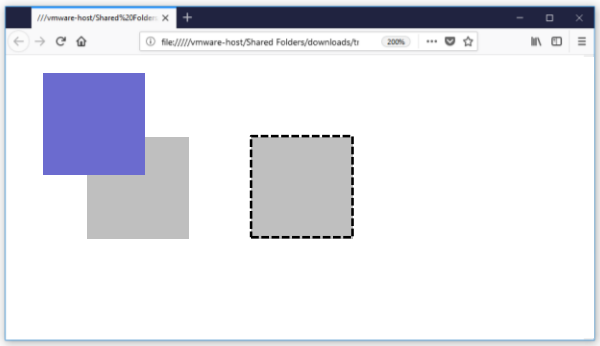
Format i alineació de lletra

font-fa
font
font-
font-w
text-
vertical-

Fons i marges

eat-x
r top

-static: *els elements es situen de forma estàtica i no es veuen influenciats per les propietats de "top", "bottom", "left", i "right". Opció per defecte.*
-absolute: *els atributs anteriors situen l'element a la pàgina considerant si està dins un altre element.*



top: 10px
left: 20px

estil de caixa CSS

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

La parella de propietat/valor defineix com es visualitzarà la propietat CSS especificada. Algunes de les propietats més habituals són les següents:

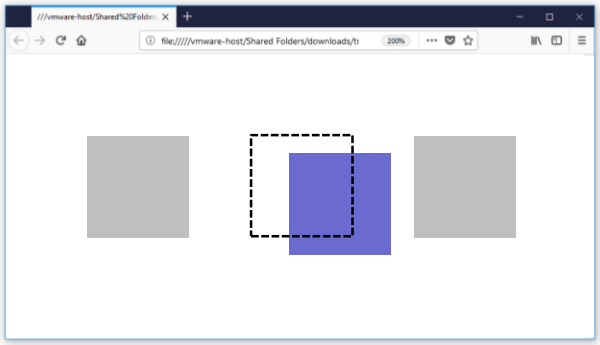
Posició i mida	
position:	static, absolute, relative, fixed, sticky
right:	0px
left:	0px
top:	20px
bottom:	10px
width:	90%, 500px
max-width:	500px
height:	10%, 300px

Format i alineació de lletra	
font-fa	
font	
font-	
font-w	
text-	
vertical-	

Fons i marges
eat-x
r top

Taules	
border-...	...
background-...	...
border-spacing:	10px
border-collapse:	separate, collapse
caption-side:	top, bottom

-static: *els elements es situen de forma estàtica i no es veuen influenciats per els atributs de "top", "bottom", "left", i "right". Opció per defecte.*
-absolute: *els atributs anteriors situen l'element a la pàgina considerant si està dins un altre element.*
-relative: *els atributs anteriors situen l'element de forma relativa a la posició "static".*



estil de caixa CSS

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

La parella de propietat/valor defineix com es visualitzarà la propietat CSS especificada. Algunes de les propietats més habituals són les següents:

Posició i mida	
position:	static, absolute, relative, fixed, sticky
right:	0px
left:	0px
top:	20px
bottom:	10px
width:	90%, 500px
max-width:	500px
height:	10%, 300px

Taules	
border-...	...
background-...	...
border-spacing:	10px
border-collapse:	separate, collapse
caption-side:	top, bottom

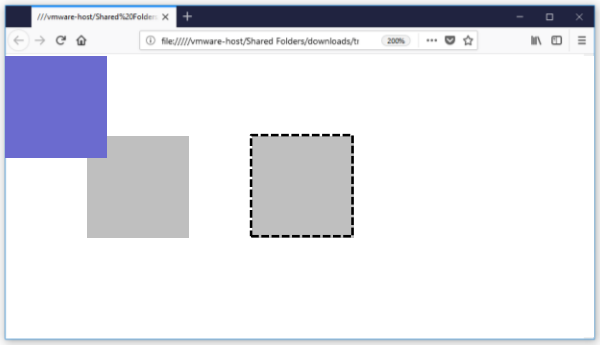
Format i alineació de lletra

font-fa
font
font-
font-w
text-
vertical-

Fons i marges

eat-x
r top

-static: *els elements es situen de forma estàtica i no es veuen influenciats per els atributs de "top", "bottom", "left", i "right". Opció per defecte.*
-absolute: *els atributs anteriors situen l'element a la pàgina considerant si està dins un altre element.*
-relative: *els atributs anteriors situen l'element de forma relativa a la posició "static".*
-fixed: *els elements es situen de forma fixa a la vista segons els atributs anteriors.*



top: 0px
left: 0px

estil de caixa CSS

1.1 Documents web: CSS. Sintaxi

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Un full d'estil està composat d'un (conjunt) de regles d'estil. Cada regla té un selector i una o vàries declaracions composades de propietat i valor.

La parella de propietat/valor defineix com es visualitzarà la propietat CSS especificada. Algunes de les propietats més habituals són les següents:

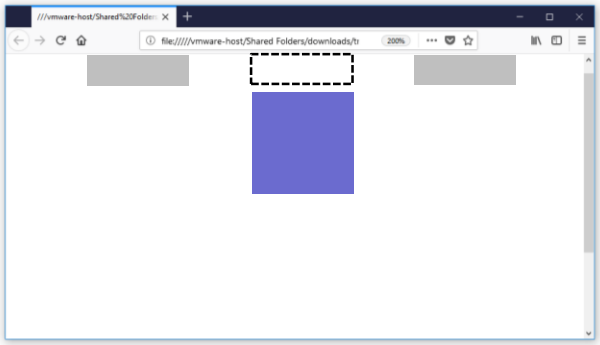
Posició i mida	
position:	static, absolute, relative, fixed, sticky
right:	0px
left:	0px
top:	20px
bottom:	10px
width:	90%, 500px
max-width:	500px
height:	10%, 300px

Taules	
border-...	...
background-...	...
border-spacing:	10px
border-collapse:	separate, collapse
caption-side:	top, bottom

Format i alineació de lletra

Fons i marges

-static: *els elements es situen de forma estàtica i no es veuen influenciats per els atributs de "top", "bottom", "left", i "right". Opció per defecte.*
-absolute: *els atributs anteriors situen l'element a la pàgina considerant si està dins un altre element.*
-relative: *els atributs anteriors situen l'element de forma relativa a la posició "static".*
-fixed: *els elements es situen de forma fixa a la vista segons els atributs anteriors.*
-sticky: *els elements es situen de forma relativa a la vista fins que es fa una scroll, moment en el qual es situen segons els atributs anteriors.*



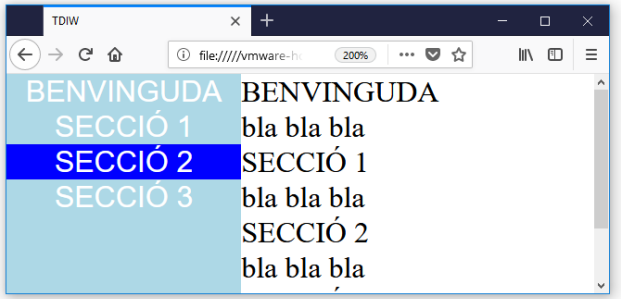
top: 20px

estil de caixa CSS

1.1 Documents web: CSS. Navegació

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

Les pàgines web solen tenir una barra de navegació a partir de la qual es pot anar a qualsevol secció de la web. CSS ens permet definir aquestes barres de diferents formes. Per exemple:



HTML

```
<ul id="menu">
    <li><a href="#inici">BENVINGUDA</a></li>
    <li><a href="#sec1">SECCIÓ 1</a></li>
    <li><a href="#sec2">SECCIÓ 2</a></li>
    <li><a href="#sec3">SECCIÓ 3</a></li>
</ul>
<div id="inici">BENVINGUDA<br />bla bla bla</div>
<div id="sec1">SECCIÓ 1<br />bla bla bla</div>
```

CSS

```
#menu{
background-color: lightblue;
list-style-type: none;
font-family: sans-serif;
margin: 0px;
padding: 0px;
text-align: center;
position: fixed;
width: 40%;
height: 100%;
overflow: auto;
}
```

```
#menu li:hover{
background-color: blue;
}
#menu a{
text-decoration: none;
color: white;
}
div{
margin-left: 40%;
}
body{
margin: 0px;
}
```

indica què passa quan el contingut sobrepassa la caixa de l'element (visible, hidden, scroll, auto)

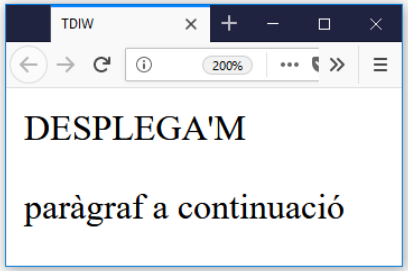
¡PREGUNTA D'EXAMEN!

Si no es fa a través de CSS, com es poden tractar els events en els elements HTML? Què és millor?

1.1 Documents web: CSS. Desplegables i animacions

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

A través de CSS podem afegir elements mòbils a la pàgina. Per exemple, podem fer desplegables o bé animacions.



HTML

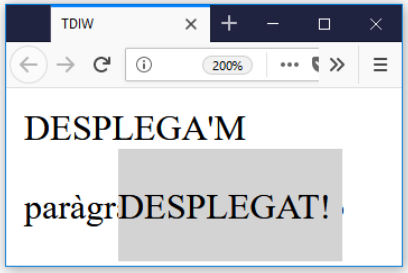
```
<div id="desplegable">
    DESPLEGA'M<br />
    <div id="desplegat">
        <p>DESPLEGAT!</p>
    </div>
</div>
<p>paràgraf a continuació</p>
```

CSS

```
#desplegat{
display: none;
position: absolute;
z-index: 1;
left: 50px;
width: 100px;
height: 50px;
background-color: lightgray;
}
#desplegable:hover #desplegat{
display: block;
}
```

indica com l'element es mostra i comporta a la pàgina (veure més endavant)

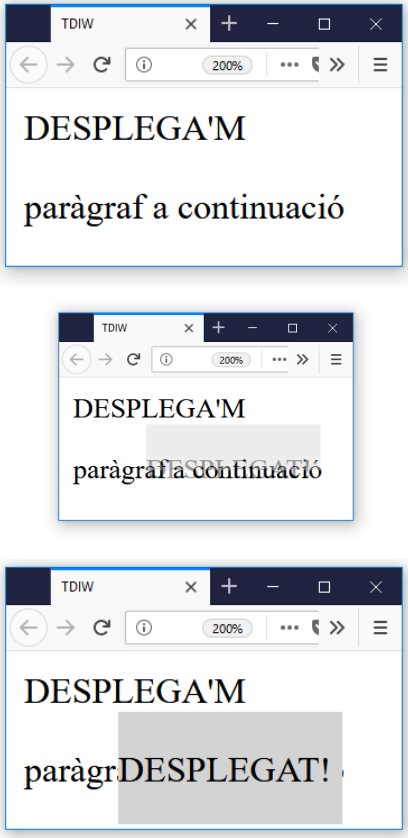
indica l'ordre de profunditat (per defecte és 0)



1.1 Documents web: CSS. Desplegables i transicions

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

A través de CSS podem afegir elements mòbils a la pàgina. Per exemple, podem fer desplegables o bé animacions.



HTML

```
<div id="desplegable">
    DESPLEGA'M<br />
    <div id="desplegat">
        <p>DESPLEGAT!</p>
    </div>
</div>
<p>paràgraf a continuació</p>
```

opacitat de l'element (0 és transparent, 1 opac)

propietats que s'utilitzaran durant la transició i temps utilitzat per canviar-les

tipus d'animació utilitzat en la transició: ease, linear, ease-in, ease-out, ...

CSS

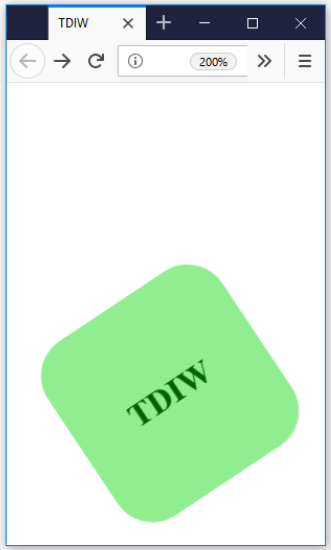
```
#desplegat{
opacity: 0;
position: absolute;
z-index: 1;
left: 50px;
width: 100px;
height: 0px;
background-color: lightgray;
transition: height 1.5s,
opacity 1.5s;
}
```

```
#desplegable:hover #desplegat{
opacity: 1;
height: 50px;
}
#desplegat{
transition-timing-function:
linear;
}
```

1.1 Documents web: CSS. Animacions

```
selector{
    propietat: valor;
    propietat: valor;
    …
}
```

CSS també ens permet canviar l'estil d'un element de forma gradual, variant una o vàries propietats en diferent instants de temps.



HTML

```
<div id="tdiw">
    <p>TDIW</p>
</div>
```

CSS

```
#tdiw{
position: relative;
display: table;
width: 100px;
height: 100px;
left: 20px;
background-color: lightgreen;
border-radius: 20px;
animation-name: rodant;
animation-duration: 2s;
animation-delay: 0.5s;
animation-timing-function:
linear;
animation-iteration-count: 2;
}
```

```
#tdiw p{
display: table-cell;
text-align: center;
vertical-align: middle;
color: darkgreen;
font-weight: bold;
}
@keyframes rodant{
0% {top: 0px;}
50% {top: 100px; transform:
rotate(360deg);}
100% {top: 0px;}
}
```

¡PREGUNTA D'EXAMEN!

Es poden crear jocs amb les animacions CSS?

1.1 Documents web: CSS. "Media query"

Les "media query" permeten comprovar diversos aspectes del dispositiu on es mostra la pàgina (com mida de la pantalla i/o navegador, resolució i orientació) i aplicar regles CSS diferents perquè la pàgina es mostri bé. La seva sintaxi és la següent:

```
@media not|only mediatype and|, (expression) …{
    selector{
        propietat: valor;
        ….
    }
    selector{
        propietat: valor;
        ….
    }
    …
}
```

"media query"

codi CSS

on

```
mediatype = all, print, screen, handheld, speech,...
expressions = max-width, max-height, min-width, min-height, resolution,...
```

El "mediatype" especifica el tipus de dispositiu utilitzat. A no ser que s'especifiqui "not|only", no fa falta posar-lo i s'agafa el valor per defecte ("all").

Les expressions permeten comprovar aspectes del dispositiu on es visualitza la pàgina, i són una parella de "propietat: valor"

1.1 Documents web: CSS. "Media query"

```
@media [not|only mediatype] and|, (expression) …{
   codi CSS
}
```

Alguns exemples de "media query" són els següents:

`@media screen and (min-width: 700px) and (orientation: landscape) { ... }`

S'aplica el codi CSS quan el dispositiu és una pantalla amb una resolució d'almenys 700 píxels d'amplada i l'orientació és apaïsada.

`@media handheld, (min-width: 650px), (orientation: landscape) { ... }`

S'aplica el codi CSS quan el tipus de dispositiu és portàtil, o bé té una resolució d'almenys 650 píxels d'amplada o bé l'orientació és apaïsada.

`@media screen and (max-width: 900px) and (min-width: 600px), (min-width: 1100px) {…}`

Quan l'amplada està entre 600 i 900 píxels, o bé per sobre 1100 píxels s'aplica el codi CSS.

com més ampla és la pantalla, més fosc és el blau de fons →

```
body{ background-color: lightblue; }
@media (min-width: 400px){
    body{ background-color: blue; }
}
@media (min-width: 800px){
    body{ background-color: darkblue; }
}
```

1.1 Documents web: CSS. "Flexboxs"

Els "flexboxs" estan pensats per poder definir el "layout" de la pàgina sense haver d'utilitzar altres mecanismes menys flexibles com els "floats" o les taules. Es defineixen de la següent forma:

s'utilitza un element contenidor que conté tots els elements que volem mostrar. El seu comportament es controla a través de propietats CSS

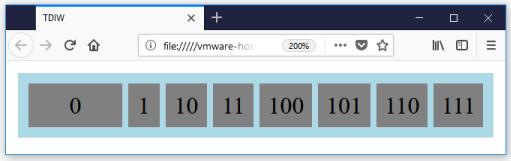
HTML

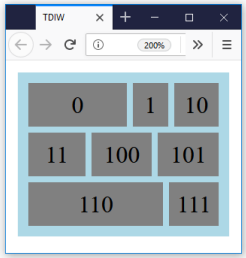
```
<div id="contenidor">
    <div style="order: 2; flex-grow: 3">1</div>
    <div style="order: 3; flex-grow: 2">10</div>
    <div style="order: 6; flex-grow: 1">101</div>
    <div style="order: 8; flex-grow: 0">111</div>
    <div style="order: 4; flex-grow: 2">11</div>
    <div style="order: 7; flex-grow: 1">110</div>
    <div style="order: 5; flex-grow: 1">100</div>
    <div style="order: 1; flex-grow: 3;
     flex-basis: 50px">0</div>
</div>
```

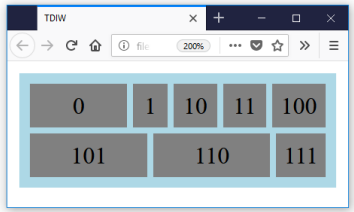
els ítems a mostrar es posen dins el contenidor i es controlen a través de propietats CSS

CSS

```
#contenidor{
background-color: lightblue;
padding: 5px;
display: flex;
flex-direction: row;
flex-wrap: wrap;
justify-content: center;
align-items: center;
align-content: center;
}
#contenidor div{
background-color: gray;
padding: 5px;
margin: 2px;
text-align: center;
}
```







1.1 Documents web: CSS. "Flexboxs"

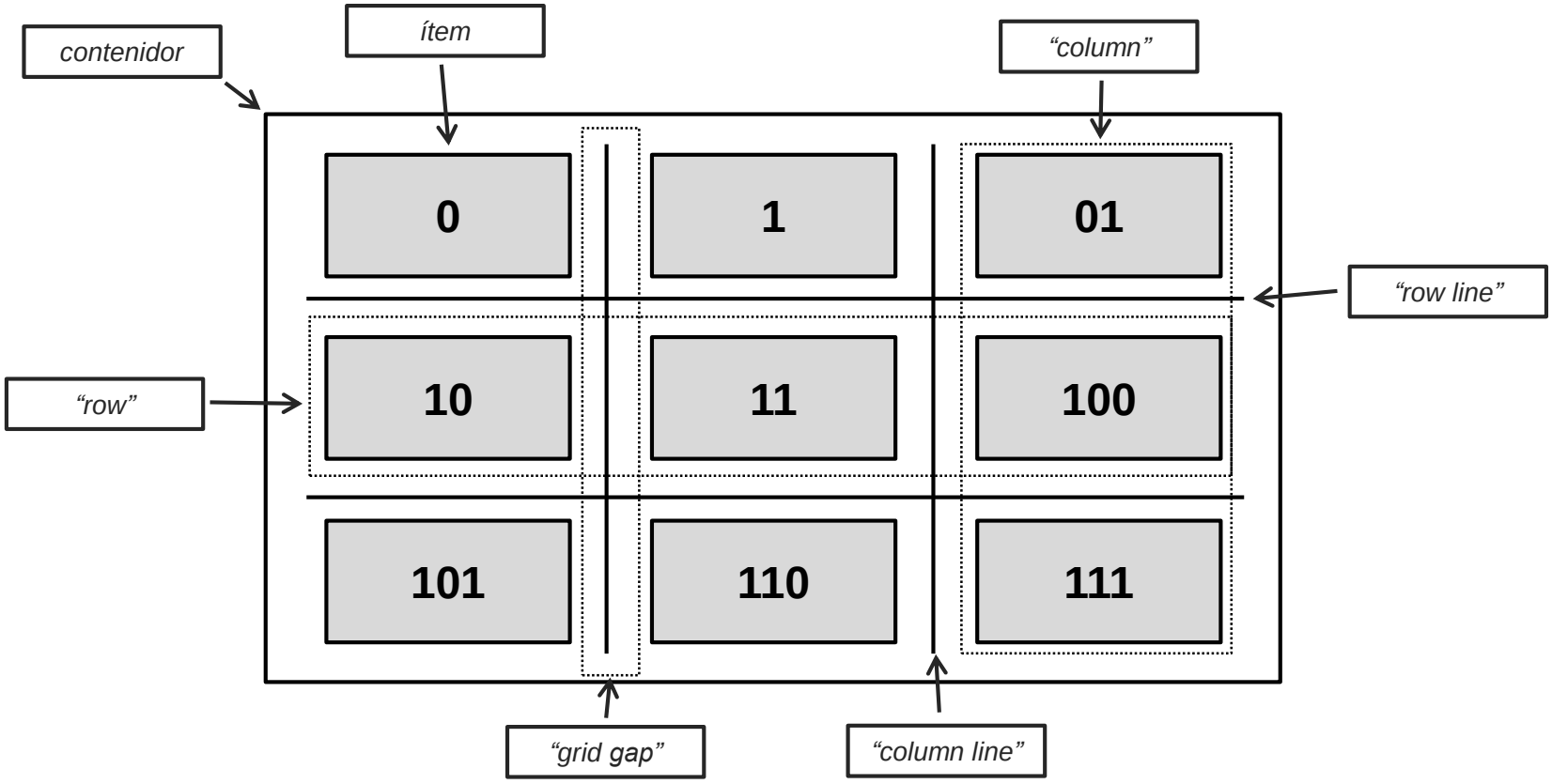
A través de les propietats del contenidor i dels ítems es pot controlar com es visualitzaran els ítems segons les mides del navegador.

Propietats del contenidor		
display:	flex	*indica que el contenidor és un "flexbox"*
flex-direction:	row, column,...	*direcció en la qual es mostren els ítems*
flex-wrap:	wrap, nowrap	*indica si es fa "wrapping" (utilització de múltiples línies)*
justify-content:	center, flex-start, flex-end, space-around, space-between	*alinea els ítems horitzontalment*
align-items:	center, flex-start, flex-end, stretch, baseline	*alinea els ítems verticalment*
align-content:	center, flex-start, flex-end, stretch, baseline	*alineació de les línies*

Propietats dels ítems		
order:	1,2,3,...	*ordre en el qual es mostren els ítems*
flex-grow:	1,2,3,...	*creixement d'un ítem respecte als altres*
flex-shrink:	1,2,3,...	*reducció d'un ítem respecte als altres*
flex-basis:	200px	*mida inicial de l'ítem*
align-self:	center, flex-start, flex-end, stretch, baseline	*alineació de l'ítem*

1.1 Documents web: CSS. "Grid"

Similar a "flexboxs", el CSS "grid" ens permet definir un "layout". És un mecanisme basat en un sistema de files i columnes, fent-lo ideal quan el "layout" de la pàgina contempla dues direccions. El "grid" es pot definir a través de files/columnes, o bé a través de "grid-areas":



1.1 Documents web: CSS. "Grid"

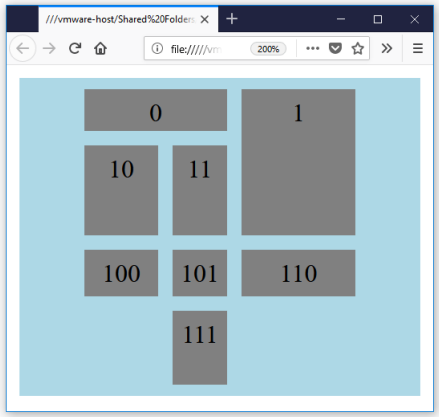
Similar a "flexboxs", el CSS "grid" ens permet definir un "layout". És un mecanisme basat en un sistema de files i columnes, fent-lo ideal quan el "layout" de la pàgina contempla dues direccions. El "grid" es pot definir a través de files/columnes, o bé a través de "grid-areas":

HTML

```
<div id="contenidor">
    <div style="grid-column: 1  / 3">0</div>
    <div style="grid-column: 3 / 4;
     grid-row: 1 / 3">1</div>
    <div>10</div>
    <div>11</div>
    <div>100</div>
    <div>101</div>
    <div>110</div>
    <div style="grid-column: 2 / 3">111</div>
</div>
```

CSS

```
#contenidor{
background-color: lightblue;
padding: 5px;
display: grid;
grid-column-gap: 5px;
grid-row-gap: 5px;
grid-template-columns: 50px auto 75px;
grid-template-rows: 30px 60px auto 50px;
justify-content: center;
align-content: center;
}
#contenidor div{
background-color: gray;
padding: 5px;
margin: 2px;
text-align: center;
}
```

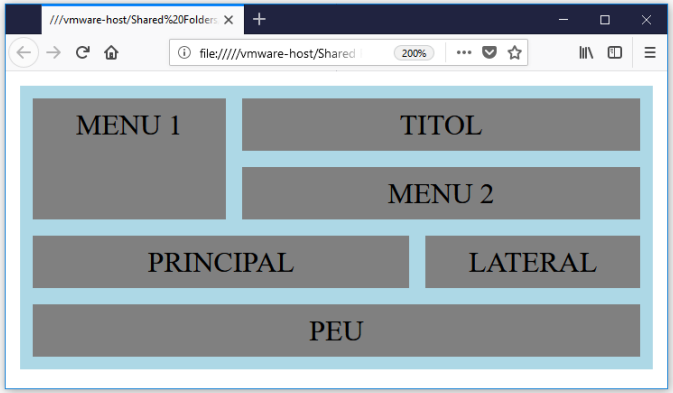


1.1 Documents web: CSS. "Grid"

Similar a "flexboxs", el CSS "grid" ens permet definir un "layout". És un mecanisme basat en un sistema de files i columnes, fent-lo ideal quan el "layout" de la pàgina contempla dues direccions. El "grid" es pot definir a través de files/columnes, o bé a través de "grid-areas":

HTML

```
<div id="contenidor">
    <div style="grid-area: titol">TITOL</div>
    <div style="grid-area: menuGran">MENU 1</div>
    <div style="grid-area: menuPetit">MENU 2</div>
    <div style="grid-area: lateral">LATERAL</div>
    <div style="grid-area: principal">PRINCIPAL</div>
    <div style="grid-area: peu">PEU</div>
</div>
```



CSS

```
#contenidor{
background-color: lightblue;
padding: 5px;
display: grid;
grid-column-gap: 5px;
grid-row-gap: 5px;
grid-template-areas:
   'menuGran titol titol titol'
   'menuGran menuPetit menuPetit menuPetit'
   'principal principal principal lateral'
   'peu peu peu peu';
}
#contenidor div{
background-color: gray;
padding: 5px;
margin: 2px;
text-align: center;
}
```

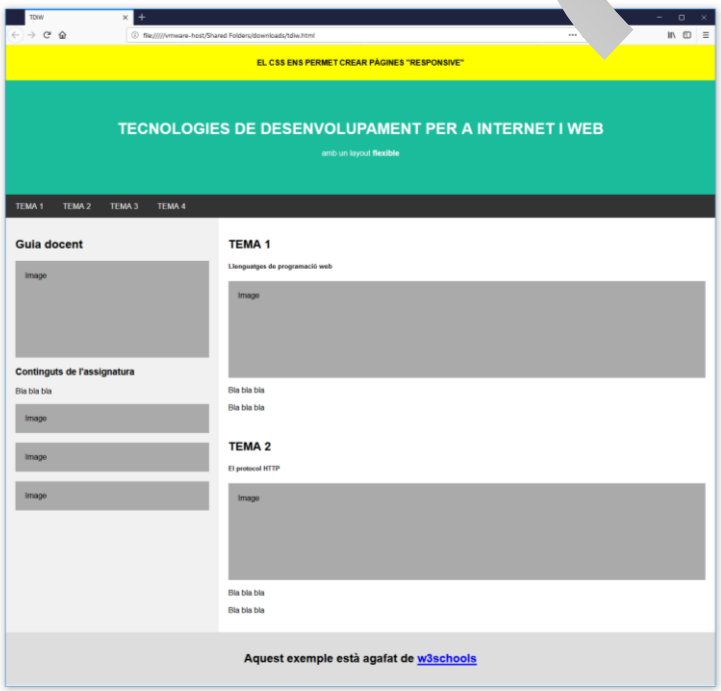
¡PREGUNTA D'EXAMEN!

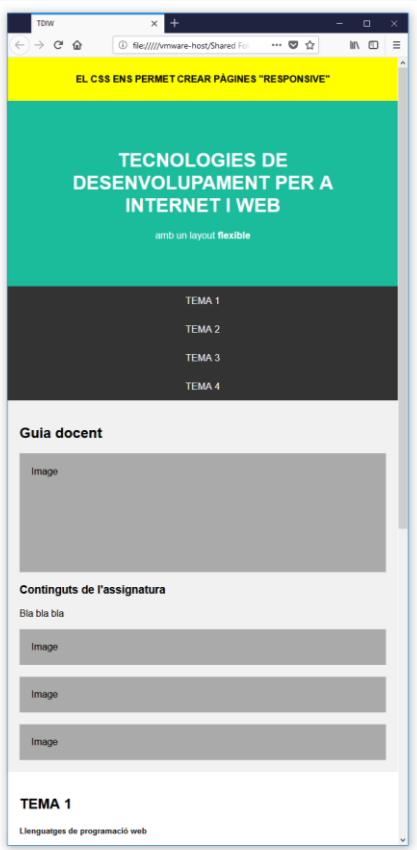
Quan és millor utilitzar CSS "flexboxs" i quan CSS "grids"?

1.1 Documents web: CSS. "Responsive layouts"

Si combinem "flexboxs" i/o "grid" amb "media query" podem aconseguir que el "layout" de la pàgina sigui "responsive", i.e., es mostri adequadament en qualsevol tipus de dispositiu, resolució, mides de vista, etc. En el següent exemple, una regla de "media query" permet mostrar els continguts en una pantalla ampla i estreta:

```
@media screen and (max-width: 700px){
    .row, .navbar{flex-direction: column;}
}
```





¡PREGUNTA D'EXAMEN!

Quina és la gran avantatge de definir el "layout" amb CSS?

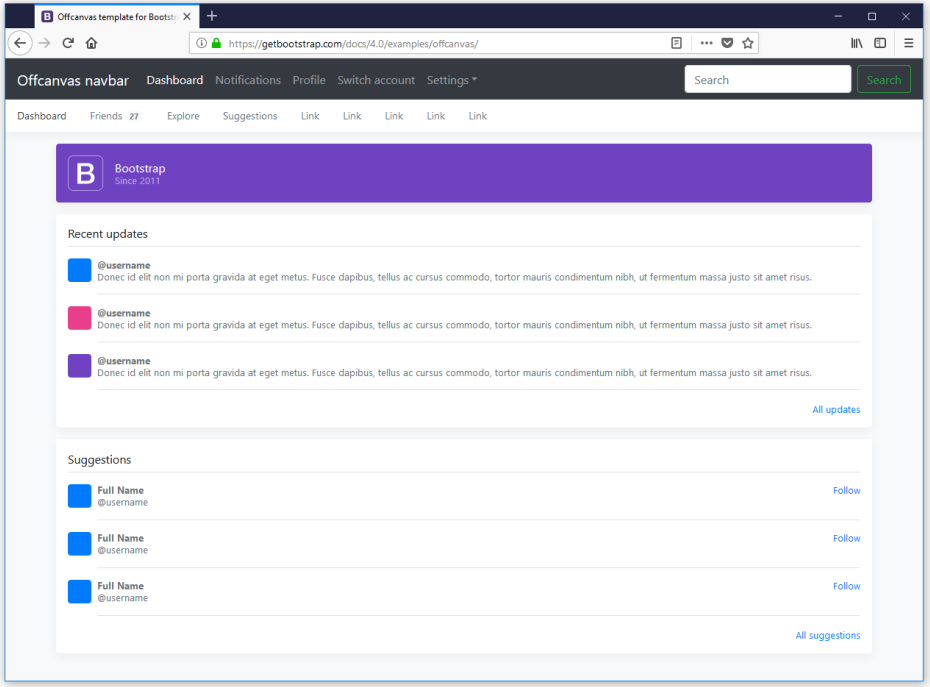
1.1 Documents web: CSS. "Frameworks"

Els CSS "frameworks" són un conjunt de classes CSS amb un estil ben definit que permeten accelerar el procés de disseny i desenvolupament d'una pàgina web. Actualment, *W3C.CSS* i b*ootstrap* són els "frameworks" més populars. Per utilitzar-los, només hem d'afegir la fulla d'estil i utilitzar les classes que ens proporcionen:

HTML

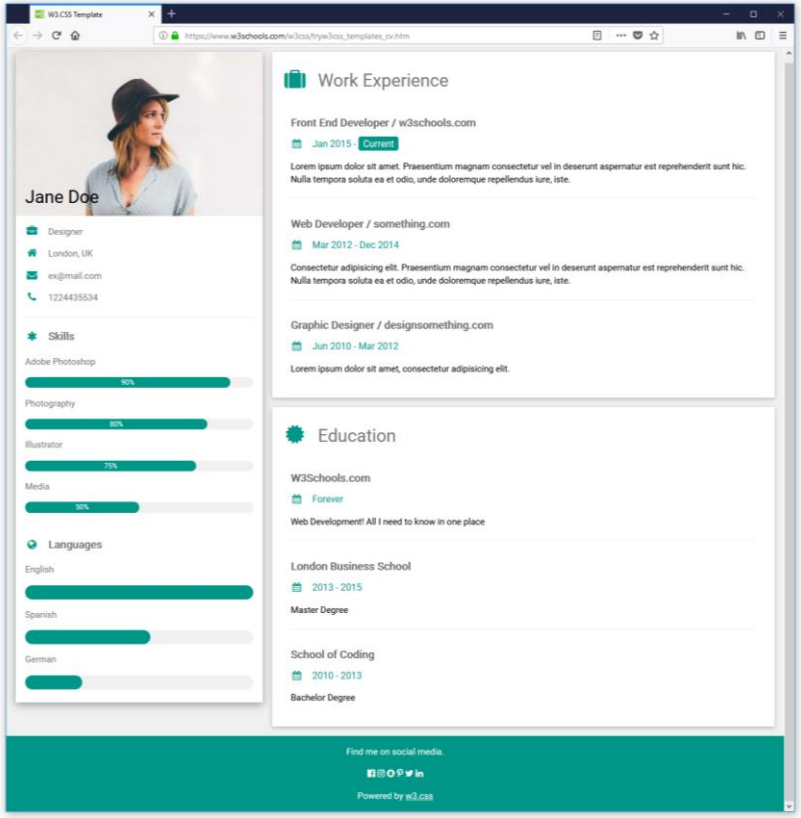
```
<!DOCTYPE html>
<html>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<body>
<header class="w3-container w3-teal">
  <h1>TECNOLOGIES DE DESENVOLUPAMENT PER A INTERNET I WEB</h1>
</header>

<article class="w3-container">
  <p>En aquesta assignatura aprendrem a programar pàgines web.</p>
</article>

<footer class="w3-container w3-teal">
  <h5>Universitat Autònoma de Barcelona</h5>
</footer>

</body>
</html>
```

1.1 Documents web: CSS. "Frameworks"



bootstrap

W3C.CSS



¡PREGUNTA D'EXAMEN!

Què és un "framework" de CSS?

¡PREGUNTA D'EXAMEN!

El "framework" de CSS és una ampliació de l'estàndard?

¡PREGUNTA D'EXAMEN!

Avantatges i desavantatges dels CSS "frameworks".

¡PREGUNTA D'EXAMEN!

Com hauríem d'organitzar els fitxers d'una web?

0. INTRODUCCIÓ

1. LLENGUATGES DE PROGRAMACIÓ WEB

- 1.1 – Documents web
- **1.2 – Programació a la banda del client**
- 1.3 – Programació a la banda del servidor
- 1.4 – Arquitectura model vista controlador: descripció i ús
- 1.5 – Aspectes de seguretat

2. EL PROTOCOL HTTP

3. PROTOCOLS DE SERVEIS

1. LLENGUATGES DE PROGRAMACIÓ WEB

1.1 – Documents web

1.2 – Llenguatges de programació a la banda del client

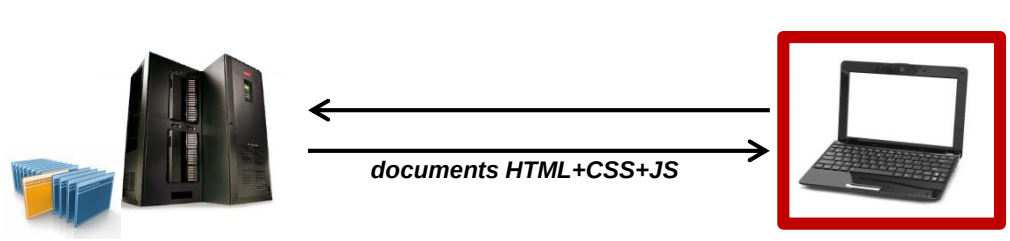
- **JavaScript**
- **AJAX**

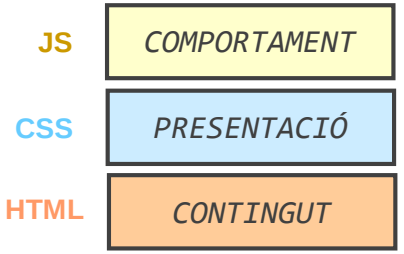
1.3 – Llenguatges de programació a la banda del servidor

1.4 – Arquitectura model vista controlador: descripció i ús

1.5 – Aspectes de seguretat

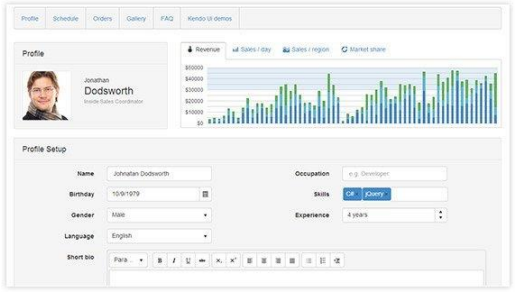
1.2 Programació al client: JavaScript. Introducció

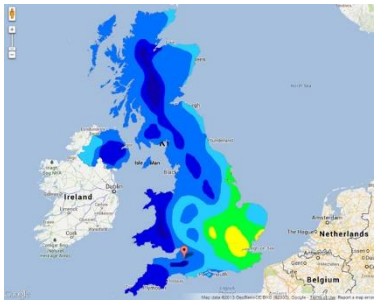




HTML transporta el contingut de la web i CSS descriu com els continguts s'han de visualitzar. Si bé aquests dos documents proporcionen bona part de tot el necessari per crear webs, en moltes pàgines és necessari poder programar alguns aspectes per dotar a la web de més interactivitat / seguretat / intel·ligència / disseny / etc. Ens determina el comportament de la web.

JavaScript és el llenguatge de programació (o de "client-side scripting") per clients HTTP. L'interpreta el navegador i permet modificar qualsevol aspecte de la pàgina HTML i de la fulla d'estil CSS. A més, a través de JavaScript podem utilitzar motors 3D per aplicacions de realitat virtual o jocs.





¡PREGUNTA D'EXAMEN!

JavaScript és un llenguatge interpretat o compilat?
Per què?

¡PREGUNTA D'EXAMEN!

Com pot ser que es puguin fer jocs amb gràfics 3D a la web?

1.2 Programació al client: JavaScript. Introducció

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>
        <button type="button"
         onClick="codi JavaScript">botó</button>
    </body>
</html>
```

JavaScript "inline"

Es posa dins elements HTML.
Avantatges: pràctic per fer crides a funcions JavaScript en elements HTML.
Desavantatges: si es posa codi JavaScript "inline", el manteniment es complica.

```
<!DOCTYPE html>
<html>
    <head>
        <script>
           codi JavaScript
        </script>
    </head>
    <body>
    </body>
</html>
```

JavaScript intern

Es posa dins un document HTML. Pot ser en el "head" o en el "body".
Avantatges: al moure el document HTML també l'acompanya el codi. Útil quan el codi és particular per una sola pàgina.
Desavantatges: si es vol utilitzar el codi JavaScript a varis documents, s'ha de copiar a tots, dificultant el manteniment.

```
<!DOCTYPE html>
<html>
    <head>
        <script src="fitxerJS.js"></script>
    </head>
    <body>
    </body>
</html>
```

```
function tdiw(){
    //codi JavaScript
}
```

JavaScript extern

Des del document HTML es referencia un fitxer .js, que conté el codi JavaScript.
Avantatges: un mateix fitxer pot servir per múltiples pàgines. Fàcil de mantenir i reutilitzar.

¡PREGUNTA D'EXAMEN!

En termes de velocitat a l'hora de carregar la pàgina, on és millor tenir el codi JavaScript?

1.2 Programació al client: JavaScript. Sintaxi

Constants, variables, "arrays", objectes, operadors i comparadors:

```
const PI = 3.1415926;
```

```
var enter = 3;
var flotant = 3.14;
var cadena1 = "TDIW" + " " + "UAB"; //"TDIW UAB"
var cadena2 = 5 + 4 + "6"; //"546"
```

operadors aritmètics:
+ - * / % ++ --
assignacions:
= += -= *= /= %=
comparacions:
== != > < >= <=
operadors lògics:
&& || !

```
var elementsHTML = ["head", "body",
                    "article"];
var numeros = [7, 3, 9];
var notes = [];
```

```
var element = elementsHTML[0];
var longitud = elementsHTML.length;
var numerosOrdenat = numeros.sort();
notes[0] = 9; //afegeix
notes.push(8); //afegeix al final
notes.shift(5); //afegeix al principi
numeros.pop(); //elimina últim
numeros.shift(); //elimina primer
delete elementsHTML[0]; //[0] "undefined"
var cadenaNums = numeros.toString();
```

```
var estudiant = {nom:"manel", cognom:"garcia", edat:25};
var cognom = estudiant.cognom;
```

¡PREGUNTA D'EXAMEN!

Per què és recomanable evitar l'ús de l'operador "new" a JavaScript?

1.2 Programació al client: JavaScript. Sintaxi

Estructures condicionals, iteratives, i funcions:

```
if(x == 7){
  y = 4;
}else{
  y = 2;
}
```

```
var i;
var r = 0;
for(i = 0; i < numeros.length; i++){
    r += numeros[i];
}
```

```
switch(x){
case 1:
  y = 3;
  break;
case 2:
  y = 7;
  break;
default:
  y = 0;
}
```

```
var i = 0;
var r = 0;
while(i < numeros.length){
    r += numeros[i];
    i++;
}
```

```
var i = 0;
var r = 0;
do {
    r += i;
    i++;
}while(i < 10);
```

```
function tdiw(p1, p2, p3){
    return(p1 + p2 + p3);
}
var x = tdiw(1, 2, 3);
```

1.2 Programació al client: JavaScript. Sintaxi

Mètodes amb strings:

```
var cadena = "Tecnologies de desenvolupament per a Internet i Web";
var long = cadena.length;
var pos1 = str.indexOf("per");
var pos2 = str.lastIndexOf("per");
var pos3 = str.search("per");
var cadena2 = cadena.slice(0, 10); //des de la posició 0 a la 10
var cadena3 = cadena.substring(0, 10); //des de la posició 0 a la 10
var cadena4 = cadena.substring(11, 2); //des de la posició 10 i 2 més
var cadena5 = cadena.replace("Web", "web");
var cadena6 = cadena.toUpperCase();
var cadena7 = cadena.toLowerCase();
var cadena8 = cadena.trim(); //elimina espais en blanc a l'inici i final
var car = cadena.charAt(0);
var array = cadena.Split(" ");
```

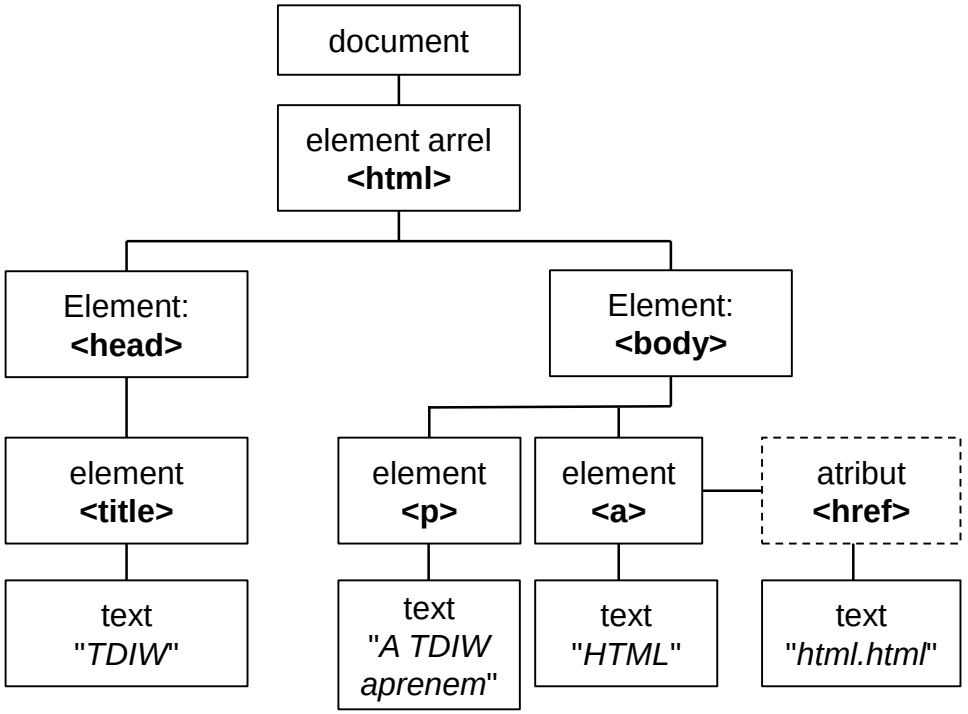
```
var num = "1234";
var cadena9 = num.toString();
var num2 = parseInt(cadena9);
var num3 = parseFloat(cadena9);
```

1.2 Programació al client: JavaScript. DOM

El "Document Object Model" (DOM):

El DOM és un model en forma d'arbre dels objectes d'una pàgina. El DOM permet que, a través de JavaScript, la pàgina pugui ser dinàmica. Es poden canviar, afegir, i eliminar elements, atributs, estils, i events.

```
<!DOCTYPE html>
<html>
    <head>
        <title>TDIW</title>
    </head>
    <body>
        <p>A TDIW aprenem</p>
        <a href="html.html">HTML</a>
    </body>
</html>
```



1.2 Programació al client: JavaScript. Objectes del DOM

Principals objectes del DOM definits a JavaScript:

Per poder manipular el DOM, JavaScript ens proporciona una sèrie d'objectes amb mètodes i propietats ben definides a través dels quals es pot canviar qualsevol aspecte de la web. Els més importants son els següents:

- *document:* element arrel del document HTML. A través d'aquest objecte podem referenciar qualsevol element. Està instanciat en qualsevol codi JavaScript. Per exemple:

HTML
```
<div id="idUnic">TDIW</div>
```
JS
```
var elementHTML = document.getElementById("idUnic");
```

- *element:* és un element HTML, com un o <p>. S'instancia a partir de l'objecte *document*. A través d'aquest objecte podem obtenir o modificar el contingut de l'element. Per exemple:

JS
```
elementHTML.innerHTML = "Tecnologies Internet i Web";
```
HTML
```
<div id="idUnic"> Tecnologies Internet i Web </div>
```

1.2 Programació al client: JavaScript. Objectes del DOM

Principals objectes del DOM definits a JavaScript:

- *style:* representa l'estil CSS corresponent a un element HTML. A través seu podem obtenir o modificar qualsevol aspecte de l'estil de l'element. Per exemple:

```
elementHTML.style.display = "none";
```

- *attribute:* representa un atribut d'un element HTML. S'accedeix a partir de l'element. Atenció! no confondre amb la propietat d'un element HTML. Per exemple:

HTML

```
<a id="aTema2" href="#tema2">Tema 2</div>
```

JS

```
var enllac = document.getElementById("aTema2");
var atribut = enllac.attributes[1].value; //#tema2
var propietat = enllac.href; //http://www.tdiw.cat#tema2
```

- *events:* permeten manipular els events que succeeixen en qualsevol element HTML. Normalment s'utilitzen en combinació amb funcions. Per exemple:

HTML

```
<button onClick="funcioJS()">Clica'm</div>
```

JS

```
funcioJS(){ alert("TDIW"); }
```

mostra una finestra emergent

1.2 Programació al client: JavaScript. Objectes del DOM

Principals objectes del DOM definits a JavaScript:

- *navigator:* aporta informació del navegador. Per exemple:

```
var navegador = navigator.appName;
var idioma = navigator.language;
```

- *screen:* aporta informació de la pantalla on es mostra la pàgina. Per exemple:

```
var alcada = screen.availHeight;
var amplada = screen.availWidth;
```

- *console:* ens serveix per accedir a la consola de "debugging" del navegador. És útil per mostrar comentaris pels desenvolupadors. Per exemple:

la consola del navegador s'obre apretant F12 →

```
console.clear();
console.log("debugant JS");
console.error("error en el codi");
```

1.2 Programació al client: JavaScript. Objectes del DOM

Principals objectes del DOM definits a JavaScript:

Els mètodes i propietats d'aquests objectes ens permeten manipular qualsevol aspecte del que fan referència. Els següents enllaços ens proporcionen la referència completa d'aquests objectes:

JavaScript *document* object reference

JavaScript *element* object reference

JavaScript *style* object reference

JavaScript *attribute* object reference

JavaScript *events* object reference

JavaScript *navigator* object reference

JavaScript *screen* object reference

JavaScript *console* object reference

1.2 Programació al client: JavaScript. Altres

Tractament d'excepcions:

Qualsevol programa pot generar errors en el moment en què s'executa. Els errors poden estar causats pel programador, les dades d'entrada, o bé altres aspectes inesperats. JavaScript incorpora el tractament d'errors a través de les excepcions. El seu funcionament és similar al d'altres llenguatges:

```
var x = 0;
try{
    if(isNaN(y)){
        throw "y is not a number";
    }
    x = y + 3;
}
catch(err){
    alert("ERROR: " + err);
}
finally{
    alert("RESULTAT: " + x);
}
```

dins el bloc del try { } *hi va el codi del qual volem gestionar els errors*

el throw *és una instrucció opcional i ens permet llançar excepcions a més a més de les pròpies que llança el JavaScript*

dins el bloc del catch { } *hi va el codi que s'executa NOMÉS quan s'ha produït alguna excepció*

la variable err *ens aporta informació de l'error que s'ha produït en el bloc* try { }

el bloc finally { } *conté codi que s'executa tant si hi ha hagut alguna excepció o no*

¡ha d'existir almenys un dels dos blocs!

1.2 Programació al client: JavaScript. Altres

JavaScript Object Notation (JSON):

JSON és un format per emmagatzemar i transportar dades. No és exclusiu de JavaScript, sinó que es pot utilitzar amb qualsevol altre llenguatge. El seu ús és com segueix:

declaració utilitzant el format JSON

conversió a un objecte JavaScript

```
var dades = '{ "estudiants" : [' +
    '{"nom":"elisa", "niu":"548796" },' +
    '{"nom":"david", "niu":"478569" },' +
    '{"nom":"joan", "niu":"478599" } ]}';
var dadesObj = JSON.parse(dades);
for(var i = 0; i < dadesObj.estudiants.length; i++){
    document.body.innerHTML += dadesObj.estudiants[i].nom + "<br />";
}
```

accés

1.2 Programació al client: JavaScript. Altres

JavaScript Object Notation (JSON):

JSON és un format per emmagatzemar i transportar dades. No és exclusiu de JavaScript, sinó que es pot utilitzar amb qualsevol altre llenguatge. El seu ús és com segueix:

declaració d'un objecte JavaScript

conversió a un string JSON

```
var estudiants = [
    {nom: "elisa", niu: "548796" },
    {nom: "david", niu: "478569" },
    {nom: "joan", niu: "478599" } ];
var dades = JSON.stringify(estudiants);
document.body.innerHTML = dades;
```

¡PREGUNTA D'EXAMEN!

És una definció JSON un objecte de JavaScript? I un objecte de JavaScript és una definció JSON?

1.2 Programació al client: JavaScript. Altres

Mode estricte:

Al ser interpretat pels navegadors, alguns errors de sintaxi en JavaScript es poden interpretar com a mala sintaxi i no fer saltar cap excepció. Quan s'indica que un script o una funció s'ha d'interpretar amb mode estricte, l'intèrpret JavaScript del navegador és més estricte i fa saltar excepcions quan es troba amb qualsevol error.

```
"use strict"; //activa el mode estricte (s'ha de posar
al principi de l'script o funció)
```

Errors habituals que fan saltar excepcions només amb el mode estricte:

```
x = 5;

var y = 5;
delete y;

function hello(p1, p1);
```

1.2 Programació al client: JavaScript. Llibreries

Existeixen moltes llibreries de JavaScript que ens aporten moltes funcionalitats. Algunes d'aquestes llibreries estan pensades per facilitar la feina dels programadors, reduint la mida dels programes i simplificant moltes funcions i efectes. Es solen incloure com un fitxer en el document HTML i qualsevol codi JavaScript posterior les pot utilitzar:

```
<!DOCTYPE html>
<html>
    <head>
        <!-- inclusió de la llibreria del servidor (o també podría ser local) -->
        <script src="https://respositori.rep/libs/llibreria.min.js"></script>
        <!-- el següent fitxer ja pot utilitzar la llibreria -->
        <script src="fitxerJS.js"></script>
    </head>
    <body>
        ...
    </body>
</html>
```

1.2 Programació al client: JavaScript. Llibreries

Algunes de les més populars són:

- **jQuery:** simplifica el codi JavaScript a través d'un accés més simple als objectes i events del DOM, de la manipulació de CSS, ús d'efectes i AJAX. Té el mateix comportament en tots els navegadors.

  https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js

JS

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
```

- **angularJS:** estén el codi HTML amb directives anomenades *ng-directives*. Permet unir dades de l'aplicació amb el codi HTML i al revés.

  https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js

HTML

```
<div ng-app="">
    <p>Name: <input type="text" ng-model="name"></p>
    <p ng-bind="name"></p>
</div>
```

1.2 Programació al client: JavaScript. Llibreries

- **W3.JS:** simplifica el codi JavaScript aportant funcionalitats per manipular els objectes del DOM i CSS. Permet filtrar, ordenar i mostrar o amagar elements de forma simple. A més, també permet unir dades de l'aplicació amb elements HTML i al revés. Permet fer presentacions fàcilment.

  https://www.w3schools.com/lib/w3.js

JS

```
w3.hide("#estudiants");
w3.show("#estudiants");
w3.sortHTML("#id01", "li");
```

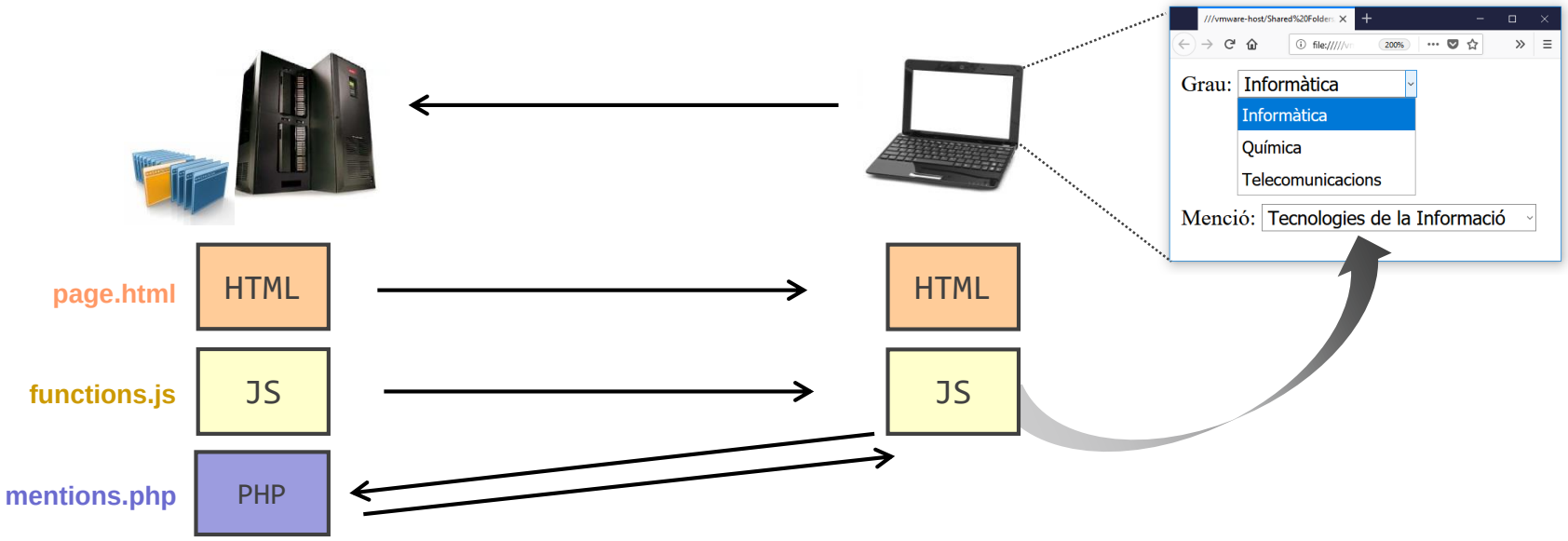
HTML

```
<ul id="estudiants">
    <li>elisa</li>
    <li>david</li>
    <li>joan</li>
</div>
```

¡PREGUNTA D'EXAMEN!

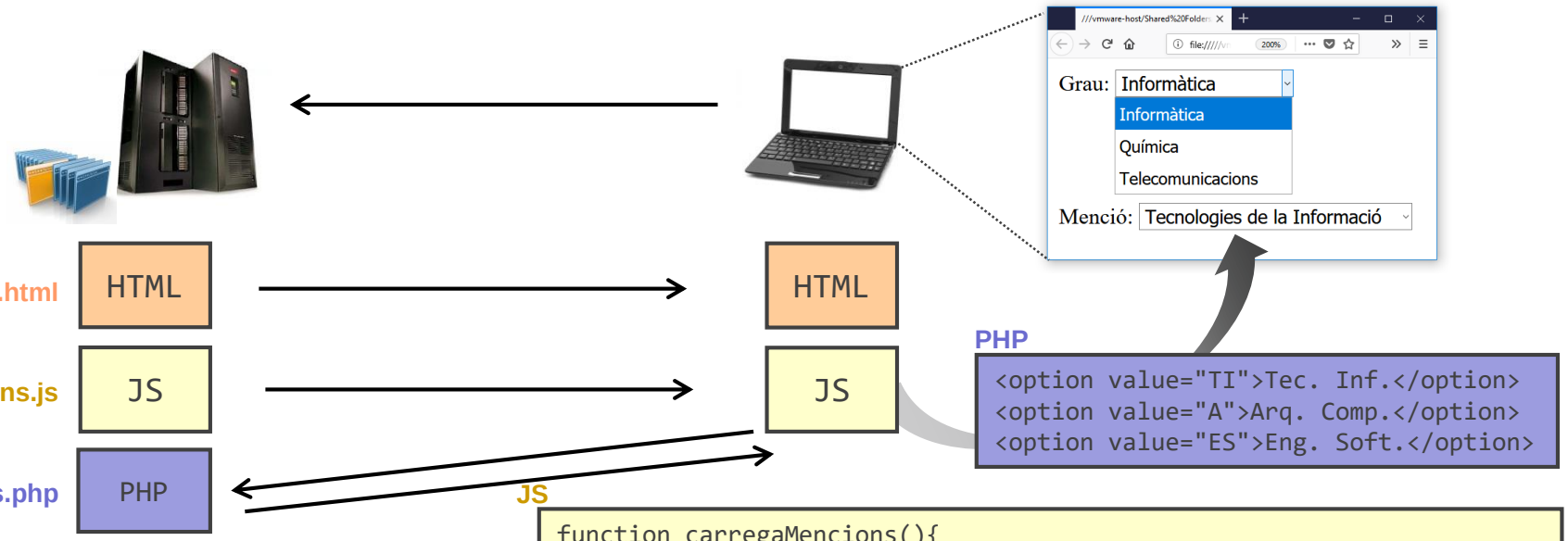
Quina és la millor llibreria de JavaScript?

1.2 Programació al client: AJAX



Asynchronous JavaScript And XML (AJAX) ***no*** és un llenguatge de programació. És una crida amb un objecte XMLHttpRequest que es realitza amb JavaScript per enviar i/o rebre dades d'un servidor remot ***des del client*** (i.e., el navegador) on es mostra una pàgina HTML. Les dades rebudes es poden processar amb JavaScript per actualitzar part del contingut de la pàgina sense haver-la de carregar tota de nou.

1.2 Programació al client: AJAX

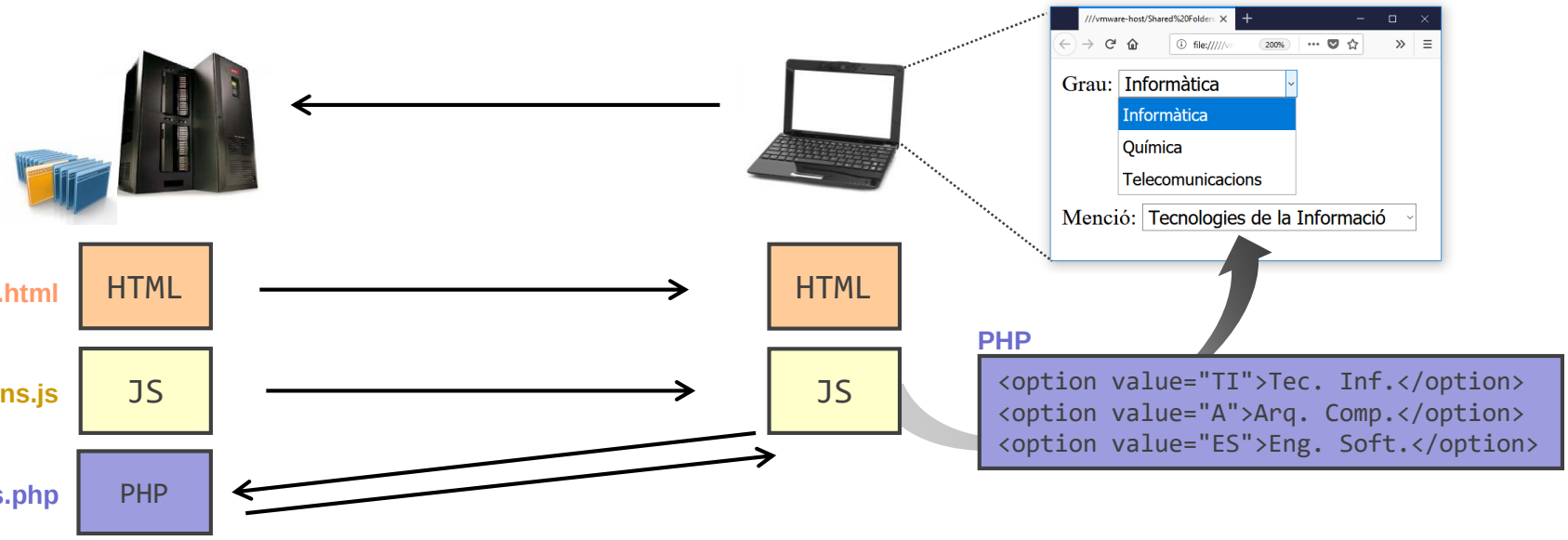


HTML

```
<form method="post" action="registre.php">
  Grau:
  <select name="grau" id="graus"
              onchange="carregaMencions();">
    <option value="I">Informàtica</option>
    <option value="Q">Química</option>
    <option value="T">Telecos</option>
  </select>
  Menció:
  <select name="mencio" id="mencions">
  </select>
</form>
```

```
function carregaMencions(){
    var xmlhttp;
    if(window.XMLHttpRequest){ //tots els navegadors
        xmlhttp = new XMLHttpRequest();
    }else{ //IE6, IE5
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange = function(){
        if(xmlhttp.readyState == 4 && xmlhttp.status == 200){
            document.getElementById("mencions").innerHTML =
                                            xmlhttp.responseText;
        }
    }
    var tagGraus = document.getElementById("graus");
    xmlhttp.open("GET","mencions.php?grau="+ tagGraus.value, true);
    xmlhttp.send();
}
```

1.2 Programació al client: AJAX



HTML

```
<form method="post" action="registre.php">
  Grau:
  <select name="grau" id="graus">
    <option value="I">Informàtica</option>
    <option value="Q">Química</option>
    <option value="T">Telecos</option>
  </select>
  Menció:
  <select name="mencio" id="mencions">
  </select>
</form>
```

JS *(amb ús de jQuery)*

```
$(document).ready(function(){
    $("#graus").change(function(){
        $.ajax({url: "mencions.php?grau=" + $("#graus").val(), success:
function(result){
            $("#mencions").html(result);
        }});
    });
});
```

¡PREGUNTA D'EXAMEN!

Es poden fer crides asíncrones a recursos que no són del mateix servidor?

0. INTRODUCCIÓ

1. LLENGUATGES DE PROGRAMACIÓ WEB

- 1.1 – Documents web
- 1.2 – Programació a la banda del client
- **1.3 – Programació a la banda del servidor**
- 1.4 – Arquitectura model vista controlador: descripció i ús
- 1.5 – Aspectes de seguretat

2. EL PROTOCOL HTTP

3. PROTOCOLS DE SERVEIS

1. LLENGUATGES DE PROGRAMACIÓ WEB

1.1 – Documents web

1.2 – Programació a la banda del client

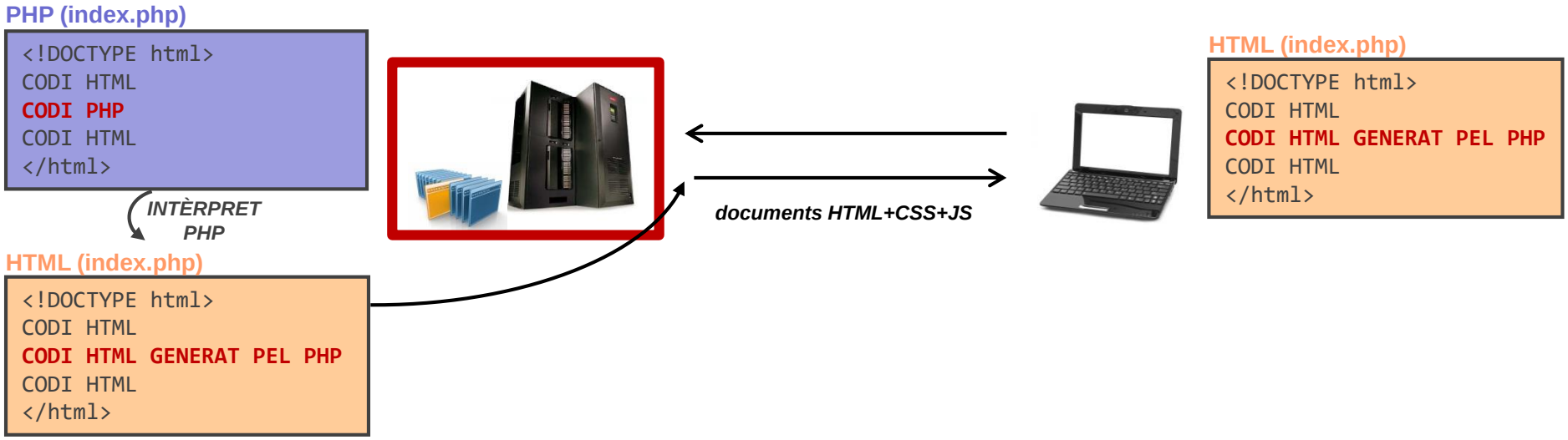
1.3 – Programació a la banda del servidor

- **Personal Hypertext Preprocessor (PHP)**

1.4 – Arquitectura model vista controlador: descripció i ús

1.5 – Aspectes de seguretat

1.3 Programació al servidor: PHP. Introducció



Els formats de document i llenguatges de programació que el client ha d'entendre (i.e., HTML, CSS i JS) formen part dels estàndards i recomanacions de la W3C i tots els navegadors els haurien de suportar.

Els servidors, en canvi, poden escollir l'ús d'un o varis llenguatges de programació ja que és només el mateix servidor el que l'interpretarà per construir el recurs (e.g., la pàgina HTML) que s'envia al client.

El "Personal Hypertext Preprocessor" (PHP) és un llenguatge de programació (o "server-side scripting") suportat en servidors web com Apache, Internet Information Services, WampServer, i molts d'altres.

¡PREGUNTA D'EXAMEN!

PHP és un llenguatge interpretat o compilat?

¡PREGUNTA D'EXAMEN!

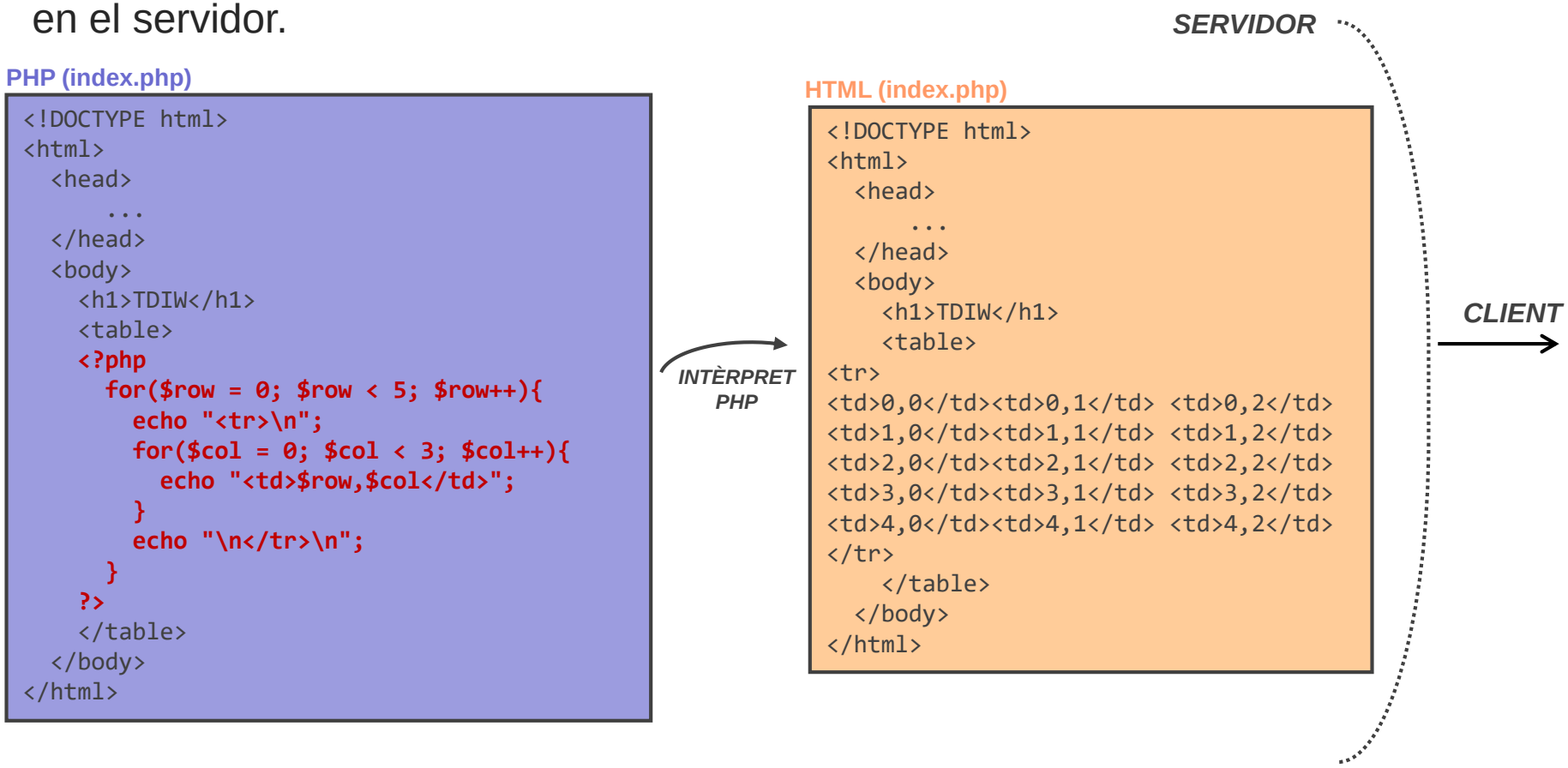
PHP podria ser un llenguatge compilat?

¡PREGUNTA D'EXAMEN!

JavaScript podria ser un llenguatge compilat?

1.3 Programació al servidor: PHP. Introducció

El codi PHP s'inserta en un document HTML en el qual se li canvia l'extensió a ".php". Abans d'enviar el document al client, el servidor interpreta tot el codi PHP del document i l'elimina del document que s'enviarà. Possiblement, el codi PHP genera elements HTML de forma dinàmica utilitzant recursos disponibles en el servidor.



¡PREGUNTA D'EXAMEN!

El client pot saber quin codi ha estat generat pel PHP i quin no?

1.3 Programació al servidor: PHP. Sintaxi

Constants, variables, "arrays", objectes, operadors i comparadors:

```
define("ASSIGNATURA", "Tecnologies de desenvolupament web");
echo ASSIGNATURA;
```

```
$enter = 3 + 0.14;
$flotant = 3.14;
$cadena = "TDIW" . " " . "UAB"; //"TDIW UAB"
echo $enter . " " . $flotant . " " . $cadena;
$boolea = true;
```

operadors aritmètics:
*+ - * / % ++ --*
assignacions:
*= += -= *= /= %=*
comparacions:
== != > < >= <=
operadors lògics:
&& || !

```
$elementsHTML = array("head", "body", "article");
$elementsHTML[3] = "section";
$numElementsHTML = count($elementsHTML); //4

$edats = array(12, 10, 24, 8);
sort($edats);
rsort($edats);

$estudiant = array("nom"=>"Jordi", "niu"=>"207319");
echo $estudiant["nom"];
$claus = array_keys($estudiant);
```

```
class Estudiant{
  function Estudiant(){
    $this->nom = "Marta";
  }
}
$e1 = new Estudiant();
echo $1->nom;
```

1.3 Programació al servidor: PHP. Sintaxi

Estructures condicionals, iteratives, i funcions:

```
if($x == 7){
  $y = 4;
}else{
  $y = 2;
}
```

```
$r = 0;
for($i = 0; $i < count($edats); $i++){
    $r += $edats[i];
}
```

```
switch($x){
case 1:
  $y = 3;
  break;
case 2:
  $y = 7;
  break;
default:
    $y = 0;
}
```

```
$i = 0;
$r = 0;
while($i < count($edats)){
    $r += $edats[i];
    $i++;
}
```

```
$i = 0;
$r = 0;
do{
    $r += $i;
    $i++;
}while($i < 10);
```

```
function tdiw($p1, $p2, $p3){
    return($p1 + $p2 + $p3);
}
$x = tdiw(1, 2, 3);
```

1.3 Programació al servidor: PHP. Sintaxi

Mètodes amb strings:

```
$cadena = "Tecnologies de desenvolupament per a Internet i Web";

$long = strlen(cadena);
$paraules = str_word_count($cadena);
$cadenaReves = strrev($cadena);
$posicio = strpos($cadena, "Internet");
$canvi = str_replace("desenvolupament", "programació", $cadena);
$arrayParaules = str_split($cadena);
$boolea = strcmp($cadena, "Tecnologies"); //false
$cadenaLow = strtolower($cadena);
$cadenaUp = strtoupper($cadena);
$cadenaTrim = trim($cadena); //elimina espais en blanc a l'inici i final
$subcadena = substr($cadena, 10, 5); //des de la posició 10 a la 15
...
```

1.3 Programació al servidor: PHP. Altres

Variables superglobals:

Les variables superglobals estan disponibles en tots els àmbits. Quan les declara el programador ens serveixen per passar informació entre diferents àmbits.

```
<?php
    function tdiw(){
        echo $GLOBALS["nom"];
    }
    ...
    $GLOBALS["nom"] = "Marta";
?>
```

Les que estan definides pel llenguatge ens aporten informació general de la petició HTTP, del client, i del servidor. PHP defineix les següents:

- `$_SERVER`: informació del servidor i la petició HTTP, per exemple,
  - `$_SERVER['SERVER_NAME']`: nom del servidor
  - `$_SERVER['SERVER_SOFTWARE']`: servidor web utilitzat
  - `$_SERVER['PHP_SELF']`: fitxer PHP on està l'script
  - `$_SERVER['SERVER_ADDR']`: adreça IP del servidor
  - `$_SERVER['REQUEST_METHOD']`: tipus de petició (GET o POST)
  - ...

1.3 Programació al servidor: PHP. Altres

Variables superglobals:

- `$_REQUEST`: conté les dades enviades des d'un formulari (sigui amb GET o amb POST), per exemple

HTML

```
<form method="post" action="register.php">
    TEXT:<input type="text" name="nom">
    <input type="submit" value="REGISTRA'T">
</form>
```

PHP (register.php)

```
<?php
    echo $REQUEST["nom"];
?>
```

- `$_POST`: conté les dades enviades des d'un formulari enviat amb una petició de tipus POST
- `$_GET`: conté les dades enviades des d'un formulari enviat amb una petició de tipus GET, o bé informació continguda en la URL, per exemple

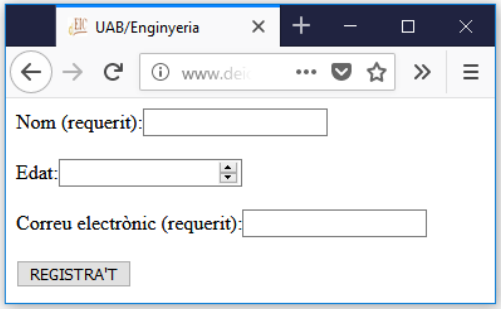
```
PETICIÓ: http://www.tdiw.org?pagina=1
```

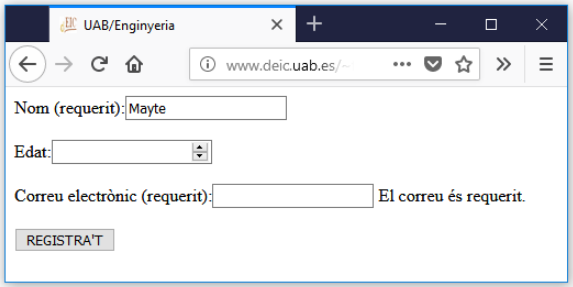
```
<?php
    echo $GET["pagina"];
?>
```

- `$_FILES`: informació dels fitxers enviats pel client
- `$_ENV`: informació del "shell" utilitzat (e.g., `$ENV['USER']`)
- `$_COOKIE`: informació de la cookie
- `$_SESSION`: informació de la sessió

1.3 Programació al servidor: PHP. Altres

Formularis (exemple sobre com recollir les dades i validar-les):






```
<?php
  $nom = $edat = $correu = "";
  $nom_err = $correu_err = "";
  if($_SERVER["REQUEST_METHOD"] == "POST"){
      if(empty($_POST["nom"])){ $nom_err = "El nom és requerit.";
      }else{ $nom = $_POST["nom"]; }
      if(!empty($_POST["edat"])){ $edat = $_POST["edat"]; }
      if(empty($_POST["correu"])){ $correu_err = "El correu és requerit.";
      }else{ $correu = $_POST["correu"]; }
  }
?>
<form action="<?php echo $_SERVER["PHP_SELF"]; ?>"
 target="_self" method="post">
    Nom (requerit):<input type="text" name="nom" value="<?php echo $nom; ?>">
    <?php echo $nom_err; ?><br />
    Edat:<input type="number" name="edat" value="<?php echo $edat; ?>"><br />
    Correu electrònic (requerit):<input type="mail" name="correu"
    value="<?php echo $correu; ?>">
    <?php echo $correu_err; ?><br />
    <input type="submit" value="REGISTRA'T">
</form>
```

¡PREGUNTA D'EXAMEN!

De quina altra forma es pot fer que un camp de formulari sigui "requerit"?

¡PREGUNTA D'EXAMEN!

Validar els camps del formulari amb PHP és l'única forma de fer-ho? Si és que no, és la més recomanable?

1.3 Programació al servidor: PHP. Altres

Inclusió de fitxers PHP:

indica el directori on està l'script (útil quan hi ha inclusions múltiples)

```php
<?php require __DIR__ . "/menu.php"; //si no troba el fitxer, atura l'execució ?>
<?php require_once __DIR__ . "/menu.php"; //com abans, però el fitxer només s'inclou un cop ?>
<?php include __DIR__ . "/menu.php"; //si no troba el fitxer, continua l'execució ?>
```

Lectura de fitxers:

acaba l'execució mostrant el missatge indicat

lectura línia a línia

lectura sencera

lectura caràcter a caràcter

```php
<?php
    $fitxer = fopen("tdiw.txt", "r") or die("No es pot obrir el fitxer");
    while(!feof($fitxer)){
        echo fgets($fitxer) . "<br />";
    }
    $a = fread($fitxer, filesize("tdiw.txt"));
    $b = str_replace("\n", "<br />", $a);
    echo $b;
    while(!feof($fitxer)){
        $c = fgetc($fitxer);
        if($c == "\n"){
            echo "<br />";
        }else{
            echo $c;
        }
    }
    fclose($fitxer);
?>
```

1.3 Programació al servidor: PHP. Altres

Escriptura de fitxers:

```
<?php
    $fitxer = fopen("tdiw.txt", "w") or die("No es pot obrir el fitxer");
    $txt = "Tecnologies de desenvolupament per a Internet i Web\n";
    fwrite($fitxer, $txt);
    fclose($fitxer);
?>
```

Tractament d'errors:

```
<?php
    function tractamentError($errno, $errstr){
        echo "<b>Error:</b> [$errno] $errstr";
        die();
    }
    set_error_handler("tractamentError", E_ALL);
    if(!file_exists("tdiw.txt")){
        trigger_error("el fitxer no existeix");
    }
    fopen("tdiw.txt", "w");
?>
```

funció pròpia pel tractament d'errors

indicació al PHP que utilitzi la funció "tractamentError" per tots els errors

llançament explícit de l'error

si no es té permisos per escriure al fitxer, PHP també llança un error

1.3 Programació al servidor: PHP. Altres

Tractament d'excepcions:

```
<?php
    try{
        if(!file_exists("tdiw.txt")){
            throw new Exception("El fitxer no existeix.");
        }else{
            ...
        }
    }catch(Exception $e){
        echo "EXCEPCIÓ: " . $e;
    }
?>
```

1.3 Programació al servidor: PHP. Establiment d'estat

PHP proporciona dos mecanismes per establir un estat amb el client: "cookies" i sessions. A través de l'estat, el servidor pot recordar informació (i.e., variables) del client que li ha estat enviada en peticions anteriors. D'aquesta forma, es pot personalitzar la pàgina per a cada client, permetent aplicacions com el comerç electrònic.

"Cookies":

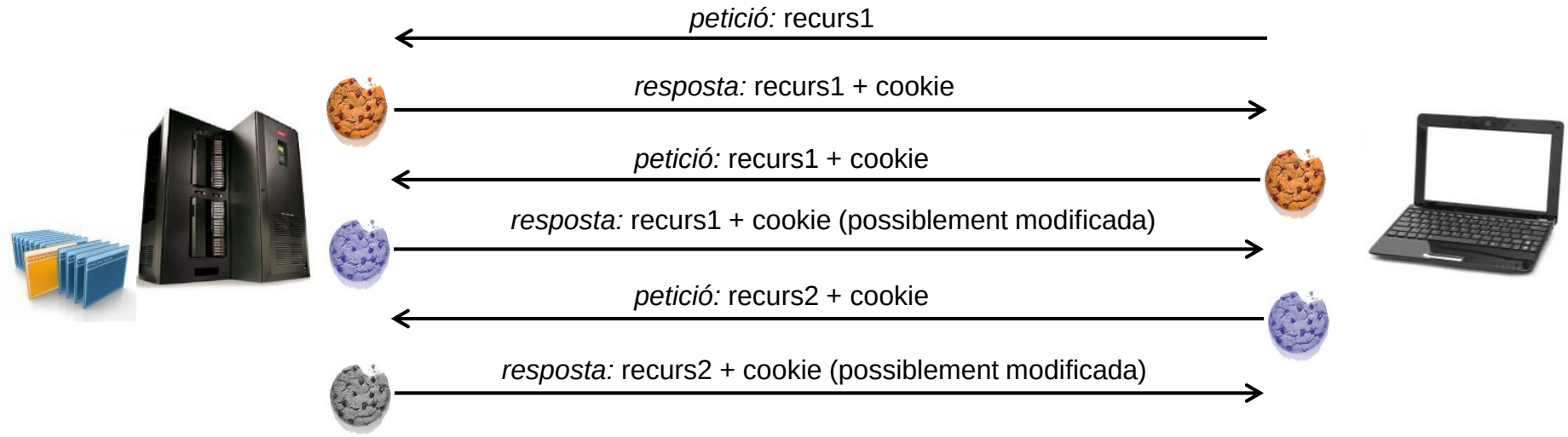
Una "cookie" és una peça d'informació que es guarda en el client i que s'envia al servidor amb cada petició que fa el client, encara que sigui de pàgines diferents. Generalment, les crea i modifica el servidor, però també ho pot fer el client. Tota la informació de la "cookie" (i.e., les variables que utilitzem) viatgen a cada petició i resposta entre client i servidor.

ATENCIÓ: les cookies s'han de crear, modificar, o eliminar abans que el fitxer generi cap sortida. S'han de posar a l'inici de tot del fitxer.
Segons l'editor que utilitzem, serà necessari eliminar un caràcter d'inici de fitxer anomenat UTF-8 BOM, per exemple amb la comanda:

```
sed '1s/^\xEF\xBB\xBF//' < original.php > modificat.php
```

1.3 Programació al servidor: PHP. Establiment d'estat

"Cookies":



¡PREGUNTA D'EXAMEN!

És recomanable guardar informació sobre l'usuari en una "cookie"?

1.3 Programació al servidor: PHP. Establiment d'estat

```php
<?php
    $cookie_nom = "estudiant";
    $cookie_valor = "miquel";
    if(!isset($_COOKIE[$cookie_nom])) {
        setcookie($cookie_nom, $cookie_valor, time() + (60 * 60 * 24 * 30), "/"); //+30 dies
        echo "Cookie " . $cookie_nom . " creada.<br />";
    }else{
        if($_COOKIE[$cookie_nom] == $cookie_valor){
            setcookie($cookie_nom, $cookie_valor . "2", time() + (60 * 60 * 24 * 30), "/"); //+30 dies
            echo "Cookie " . $cookie_nom . " modificada.<br />";
        }else{
            setcookie($cookie_nom, "" , time() - (60 * 60 * 24 * 30), "/"); //-30 dies
            echo "Cookie " . $cookie_nom . " eliminada.<br />";
        }
    }
?>
<html><body>
<?php
    echo "Cookies existents: <br />";
    for($i = 0; $i < count($_COOKIE); $i++){
      echo $i . " : ". array_keys($_COOKIE)[$i] . " = " . $_COOKIE[array_keys($_COOKIE)[$i]] . "<br />";
    }
?>
</body></html>
```

creació

modificació

eliminació

¡PREGUNTA D'EXAMEN!

Què fa el codi anterior al demanar la pàgina i al actualitzar-la?

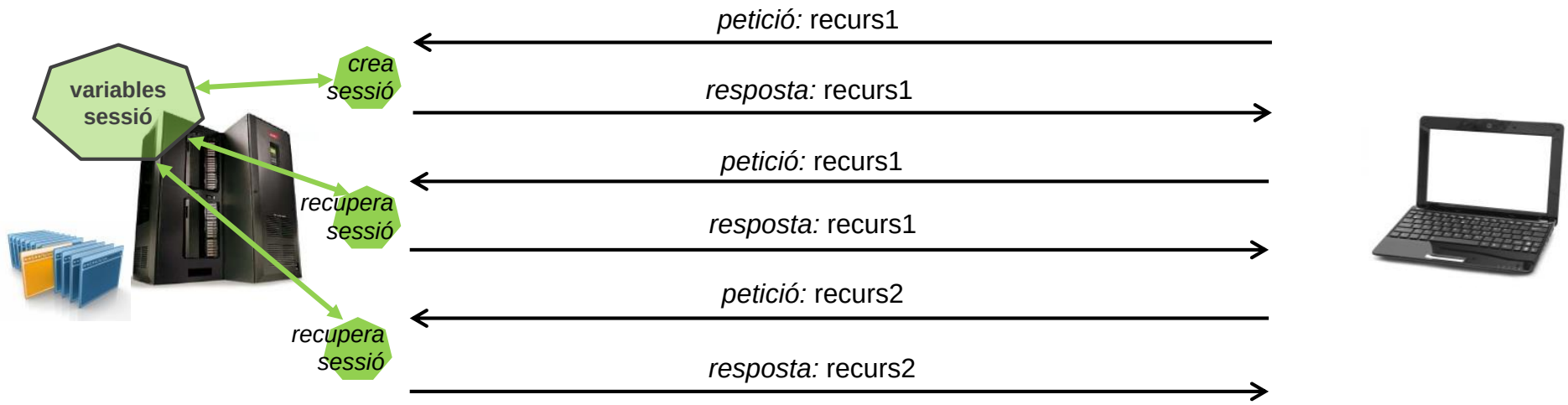
¡PREGUNTA D'EXAMEN!

Què passaria si el codi anterior estigués a dos pàgines i el client anés fent peticions alternades?

1.3 Programació al servidor: PHP. Establiment d'estat

"Sessions":

Una sessió és una peça d'informació que es guarda en el servidor i que es pot recuperar a cada petició que fa el client, possiblement demanant pàgines diferents. Les sessions només les pot crear i modificar el servidor. Tota la informació de la sessió (i.e., les variables que utilitzem) es guarda en el servidor. Si no es fa acabar implícitament, la sessió dura fins que l'usuari tanca el navegador.



¡PREGUNTA D'EXAMEN!

És recomanable guardar informació sobre l'usuari en una sessió?

1.3 Programació al servidor: PHP. Establiment d'estat

```php
<?php
  session_start();
?>
<!DOCTYPE html>
<html><body>
<?php
  if(!isset($_SESSION["nom"])){
    $_SESSION["nom"] = "pere";
    $_SESSION["niu"] = "205489";
    echo "Variables creades: nom(" . $_SESSION["nom"] . "), niu(" . $_SESSION["niu"] .") <br />";
    session_unset();
    echo "Variables esborrades: nom(" . $_SESSION["nom"] . "), niu(" . $_SESSION["niu"] .") <br />";
    $_SESSION["nom"] = "pere";
    $_SESSION["niu"] = "205489";
    echo "Variables creades: nom(" . $_SESSION["nom"] . "), niu(" . $_SESSION["niu"] .") <br />";
  }else{
    echo "Variables ja existents: nom(" . $_SESSION["nom"] . "), niu(" . $_SESSION["niu"] .") <br />";
    session_destroy();
  }
?>
</body></html>
```

creació de la sessió

establiment de variables

neteja de variables

acabament implícit de la sessió

¡PREGUNTA D'EXAMEN!

Què fa el codi anterior al demanar la pàgina i al actualitzar-la?

¡PREGUNTA D'EXAMEN!

Què passaria si el codi anterior estigués a dos pàgines i el client anés fent peticions alternades?

¡PREGUNTA D'EXAMEN!

Com creus que estan implementades les sessions? Dit d'altra forma, com identifica el servidor les variables que corresponen a una determinada sessió?

¡PREGUNTA D'EXAMEN!

Quina diferència hi ha entre una "cookie" i una sessió? Què és millor utilitzar?

1.3 Programació al servidor: PHP. Manipulació de dades

Accés a bases de dades amb PDO:

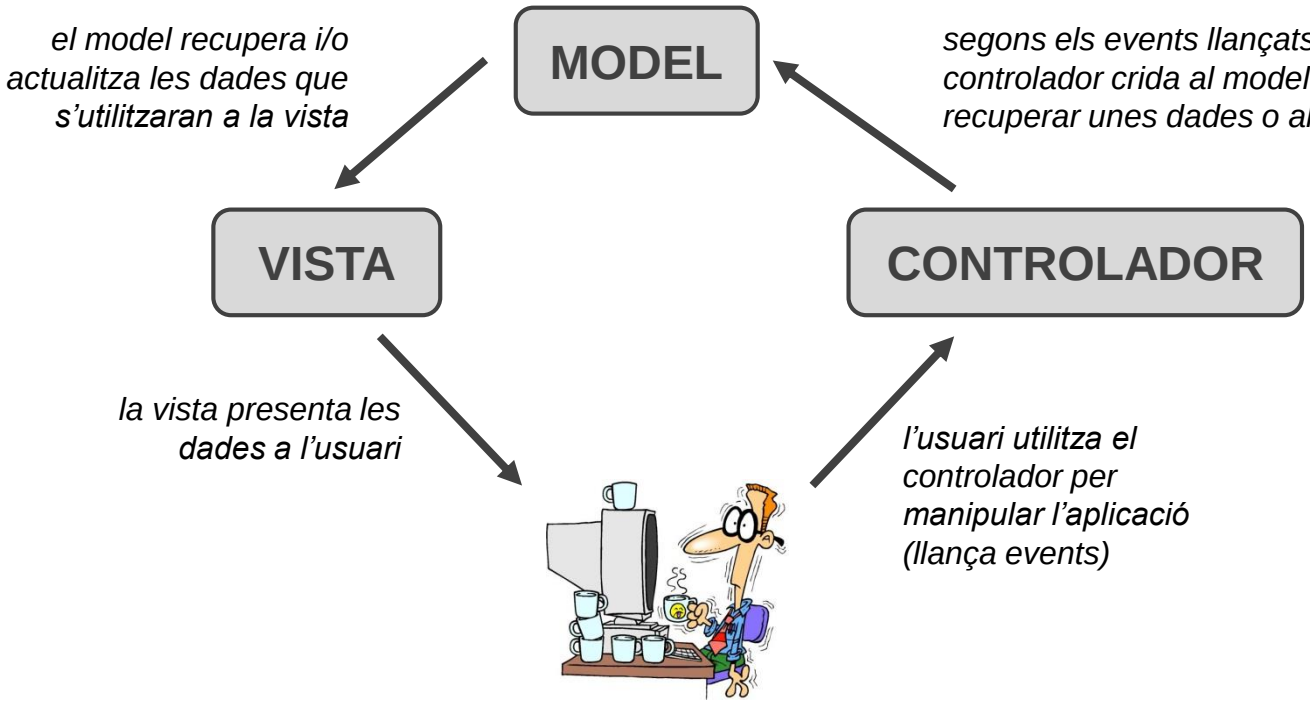
```
<?php
  $servidor = "localhost";
  $usuari = "david";
  $clau = "password";

  try{
    $connexio = new PDO("mysql:host=$servidor;dbname=myDB;charset=UTF8", $usuari, $clau);
    $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); //obliga a llançar excepcions
    $consulta = $connexio->prepare("SELECT camp1, camp2, camp3 FROM taula");
    $consulta->execute();
    $resultat = $consulta->fetchAll(PDO::FETCH_ASSOC); //retorna un array associatiu

    foreach($resultat as $fila){
      echo $fila['camp1'] . " " . $fila['camp2'] . " " . $fila['camp3'] . "<br />" ;
    }

  }catch(PDOException $e){
    echo "Error: " . $e->getMessage();
  }

  $connexio = null; //desconnecta la base de dades
?>
```

Validació de dades amb *filter_var*:

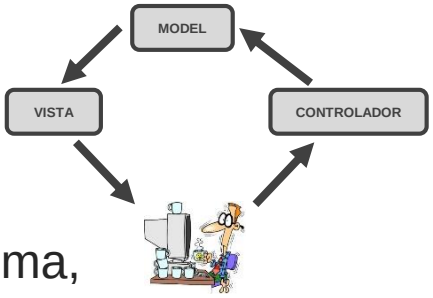
```
<?php
  if(!filter_var("maria@servidor.org", FILTER_VALIDATE_EMAIL)){ echo "Correu invalid."; }
?>
```

1.4 Arquitectura model vista controlador. Descripció

L'arquitectura Model Vista Controlador (MVC) és un patró de disseny de programari per desenvolupar interfícies d'usuari. El seu objectiu és facilitar el desenvolupament, manteniment, i (re)ús del codi font de l'aplicació. La seva utilització en la web l'ha fet molt popular.
La idea general de l'arquitectura MVC és dividir el codi de l'aplicació en tres parts separades que interactuen entre elles: el model, la vista, i el controlador:



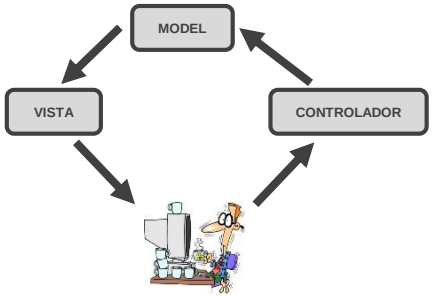
1.4 Arquitectura model vista controlador. Descripció



Les principals funcions de cadascuna de les parts de l'arquitectura MVC són les següents:

- ***MODEL***: gestiona la informació amb la qual treballa el sistema, assegurant l'accés i la integritat de les dades. Accedeix a les (base de) dades. Els paràmetres que no depenen del model els obté a través del controlador. En general, s'encarrega de la gestió de la base de dades utilitzada.
- ***VISTA***: és la part que l'usuari veu, i.e., la interfície. Representa la informació que ha recuperat el model. La representació de la informació es pot donar de qualsevol forma, e.g., amb un llistat, una gràfica, o un diagrama. Una mateixa informació es pot representar amb múltiples vistes diferents.
- ***CONTROLADOR***: s'encarrega de gestionar els events que l'usuari genera. Quan aquests events impliquen un canvi en el model i/o a la vista, crida les funcions del model perquè recuperin les dades i les passa a la vista perquè les actualitzi.

1.4 Arquitectura model vista controlador. Ús

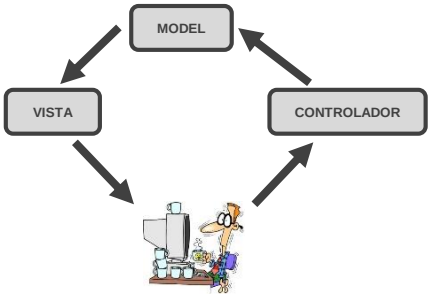


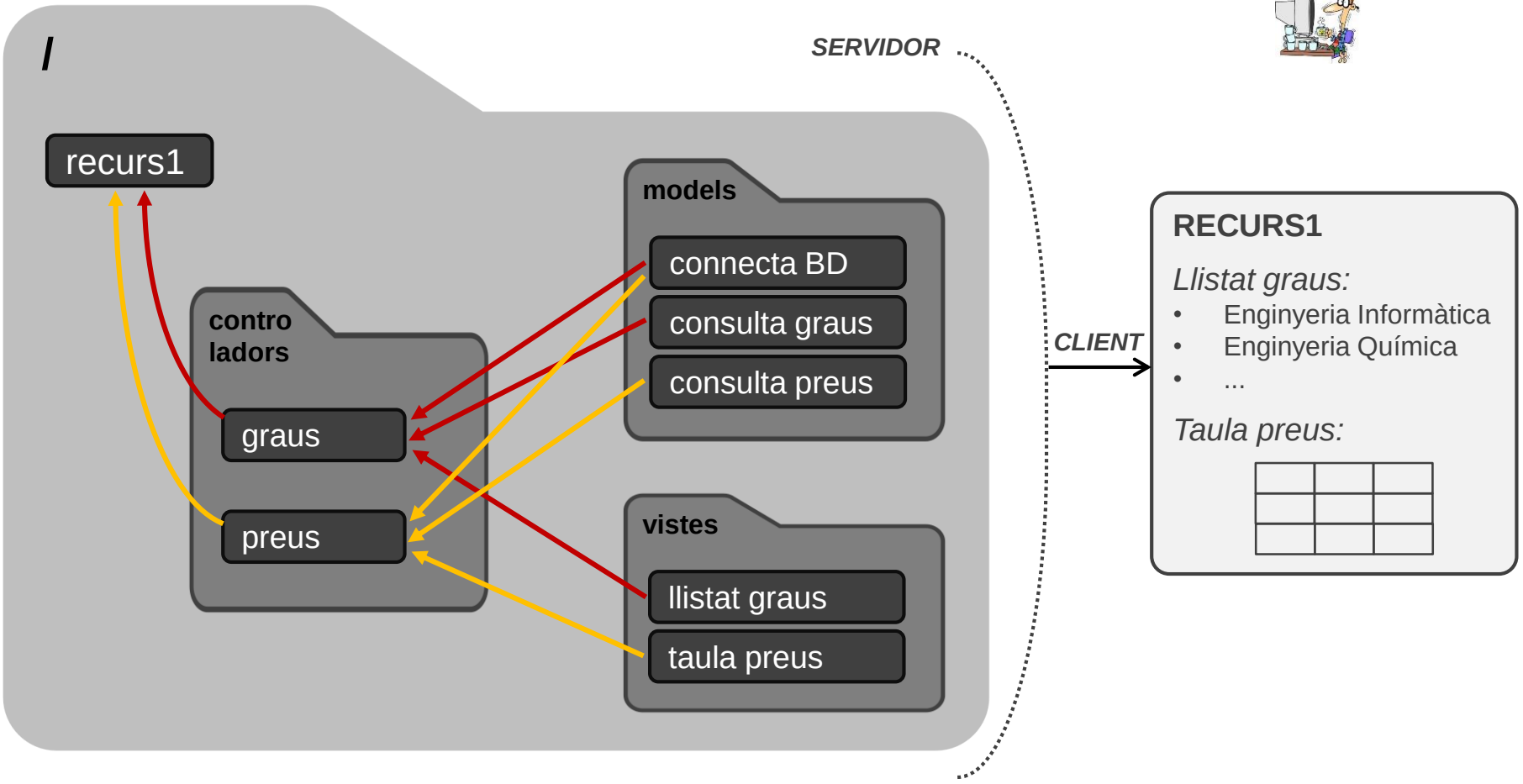
Existeixen vàries maneres d'aplicar l'arquitectura MVC en aplicacions web. Aquí en mostrem una que situa la major part de l'arquitectura en el servidor, deixant un client molt simple. El funcionament és el següent:

1) El client fa una petició al servidor demanant un recurs.
2) Per construir el recurs, el servidor crida a un (o varis) **controlador(s)**.
3) El(s) **controlador(s)** utilitzen el **model** per recuperar les dades el qual, possiblement, es connecta a una base de dades del servidor. Un cop les dades estan disponibles, el controlador les passa a la **vista**, que les representa d'una forma determinada.
4) Un cop la **vista** està construïda, el servidor envia el recurs al client.
5) El client processa els events que genera l'usuari, com l'accés a enllaços o la submissió de formularis, i els envia al servidor. El servidor passa aquests events al **controlador**, tornant al PAS 3).

1.4 Arquitectura model vista controlador. Ús

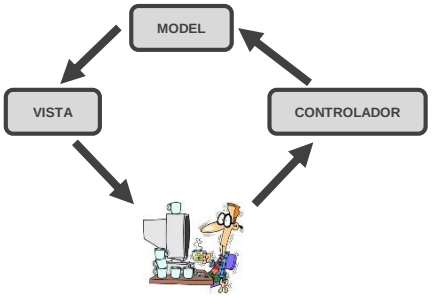
Un esquema d'implementació d'aquest tipus d'arquitectura MVC podria ser el següent:

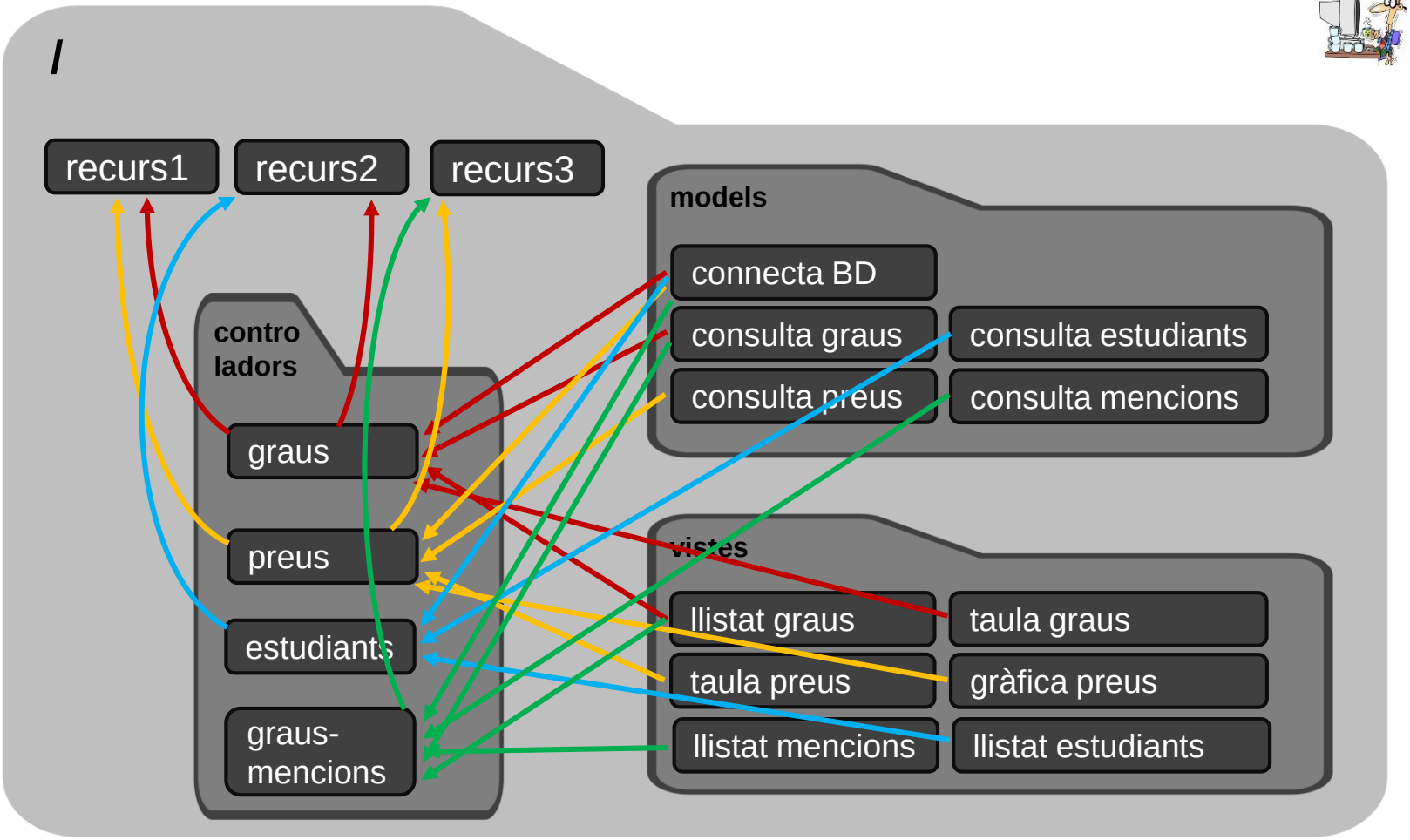




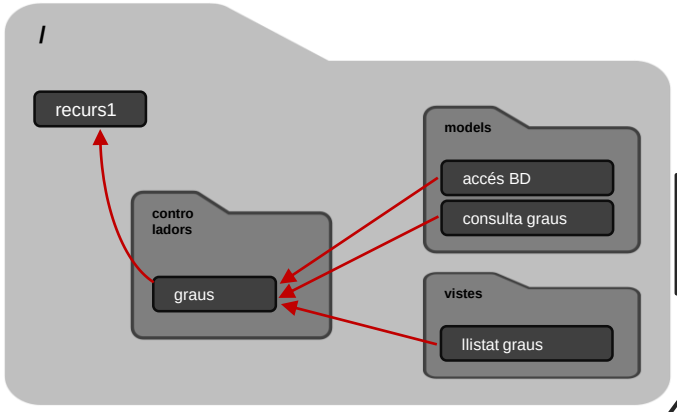
1.4 Arquitectura model vista controlador. Ús

Aquesta arquitectura facilita la re-utilització de codi i el seu manteniment. La incorporació de nous recursos aprofita codi ja existent, i canvis en el model o vista són simples perquè el codi està en un sol lloc.





1.4 Arquitectura model vista controlador. Ús



podria necessitar preparar les dades per la vista

recurs1: ./index.php

```
<html>
 [...]
 <body>
    [...]
    <?php include __DIR__ . "/controladors/graus.php" ?>
     [...]
  </body>
</html>
```

graus: ./controladors/graus.php

```
<?php
include_once __DIR__ . "/../models/connectaBD.php";
include_once __DIR__ . "/../models/consultaGraus.php";
$connexio = connectaBD();
$graus = consultaGraus($connexio);
include_once __DIR__ . "/../vistes/llistatGraus.php";
?>
```

consulta graus: ./models/consultaGraus.php

```
<?php
function consultaGraus($connexio){
  try{
    $consulta = $connexio->prepare("SELECT id,nom FROM 'graus'");
    $consulta->execute();
    $graus = $consulta->fetchAll(PDO::FETCH_ASSOC);
  }catch(PDOException $e){echo "Error: " . $e->getMessage();}
  return($graus)
}
?>
```

llistat graus: ./vistes/llistatGraus.php

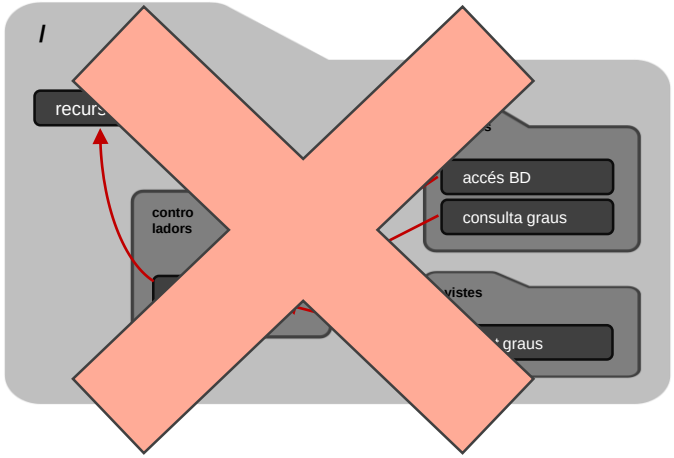
```
<ul>
<?php foreach($graus as $grau){ ?>
  <li> <?php echo $grau["id"]; ?>
       <?php echo $grau["nom"]; ?>
  </li>
<?php } ?>
</ul>
```

el codi PHP no genera "tags"

accés BD: ./models/connectaBD.php

```
<?php
function connectaBD(){
  $servidor = "localhost"; $usuari = "david"; $clau = "psw";
  try{
    $connexio = new PDO("mysql:host=$servidor;dbname=myDB;charset=UTF8", $usuari, $clau);
    $connexio->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  }catch(PDOException $e){ echo "Error: " . $e->getMessage(); }
  return($connexio);
}
?>
```

1.4 Arquitectura model vista controlador. Ús



Sense la utilització de l'arquitectura MVC el codi del model i la vista es repeteix a molts pàgines de la nostra web.

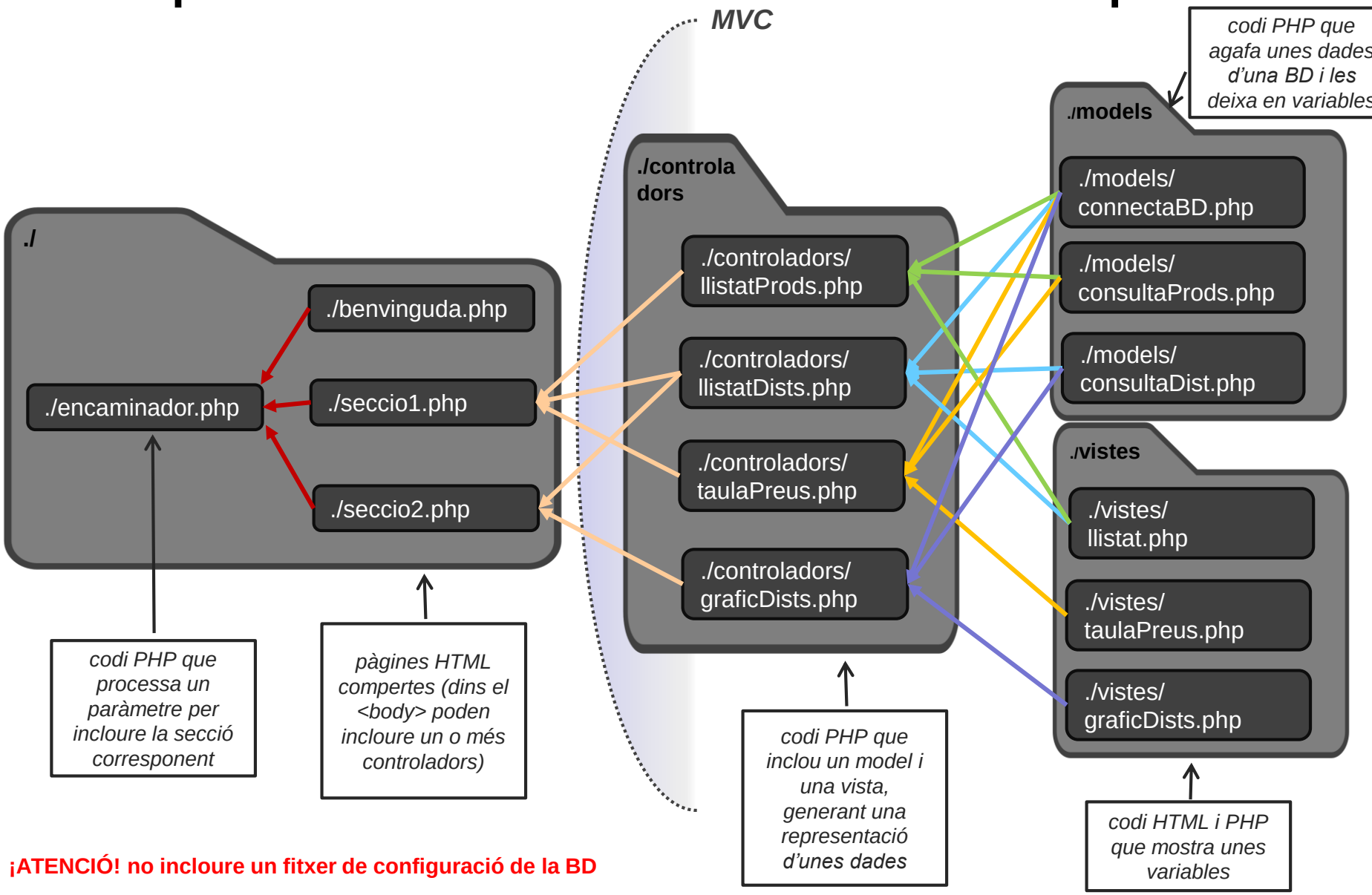
recurs1: ./index.php

```
<html>
 [...]
 <body>
    [...]
    <?php
      $servidor = "localhost";
      $usuari = "david";
      $clau = "psw";
      try{
        $connexio = new PDO("mysql:host=$servidor;
                dbname=myDB;charset=UTF8", $usuari, $clau);
        $connexio->setAttribute(PDO::ATTR_ERRMODE,
                                    PDO::ERRMODE_EXCEPTION);
        $consulta = $connexio->prepare("SELECT id,nom
                                              FROM 'graus'");
        $consulta->execute();
        $graus = $consulta->fetchAll(PDO::FETCH_ASSOC);
      }catch(PDOException $e){
        echo "Error: " . $e->getMessage();
      }
    ?>
    <ul>
    <?php foreach($graus as $grau){ ?>
      <li> <?php echo $grau["id"]; ?>
           <?php echo $grau["nom"]; ?>
      </li>
    <?php } ?>
    </ul>
    [...]
  </body>
</html>
```

model

vista

1.4 Arquitectura model vista controlador. Pràctiques



¡ATENCIÓ! no incloure un fitxer de configuració de la BD

¡PREGUNTA D'EXAMEN!

L'arquitectura MVC accelera la generació dinàmica dels recursos de la web?

¡PREGUNTA D'EXAMEN!

En l'arquitectura MVC el controlador pot estar en el client?

¡PREGUNTA D'EXAMEN!

És l'arquitectura MVC l'única forma d'organitzar el codi d'una web per facilitar-ne el re-ús i manteniment?

1. LLENGUATGES DE PROGRAMACIÓ WEB

1.1 – Documents web

1.2 – Programació a la banda del client

1.3 – Programació a la banda del servidor

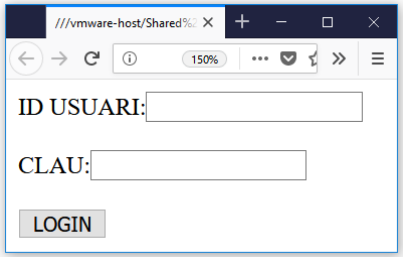
1.4 – Arquitectura model vista controlador: descripció i ús

1.5 – Aspectes de seguretat

- **SQL injection**
- **Cross-site scripting**
- **Cross-site request forgery**

1.5 Aspectes de seguretat. SQL injection

Un atac de "SQL injection" utilitza camps on habitualment l'usuari hi escriu informació que després s'utilitza en consultes SQL per injectar codi SQL maliciós. Per exemple, en un formulari d'identificació:

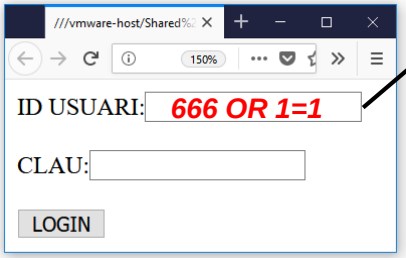


login.html

```
<form action="validacio.php" target="_self" method="post">
  ID USUARI:<input type="text" name="idUsuari"><br />
  CLAU:<input type="password" name="clau"><br />
  <input type="submit" value="LOGIN">
</form>
```

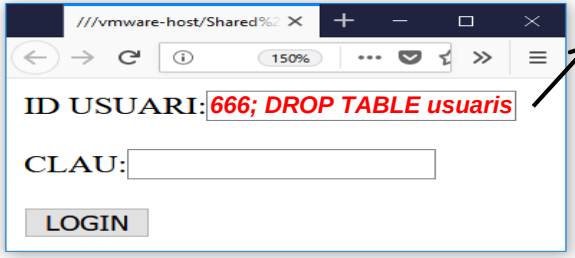
validacio.php

```
<?php
  [...]
  $SQL = "SELECT * FROM usuaris WHERE clau='" . $_POST["clau"] . "' AND idUsuari=" . $_POST["idUsuari"];
  $consulta = $connexio->prepare($SQL);
  $consulta->execute();
  [...]
?>
```




```
SELECT * FROM usuaris WHERE clau='' AND idUsuari=666 OR 1=1
```

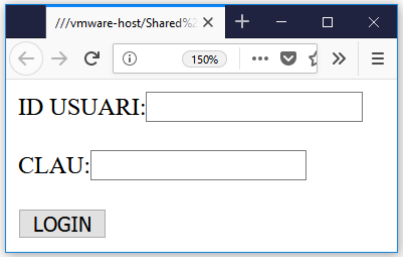
AND té precedència sobre OR




```
SELECT * FROM usuaris WHERE clau='' AND
idUsuari=666; DROP TABLE usuaris
```

1.5 Aspectes de seguretat. SQL injection

Aquest tipus d'atac es pot prevenir fent que la consulta SQL rebi com a paràmetres les entrades de l'usuari:



login.html

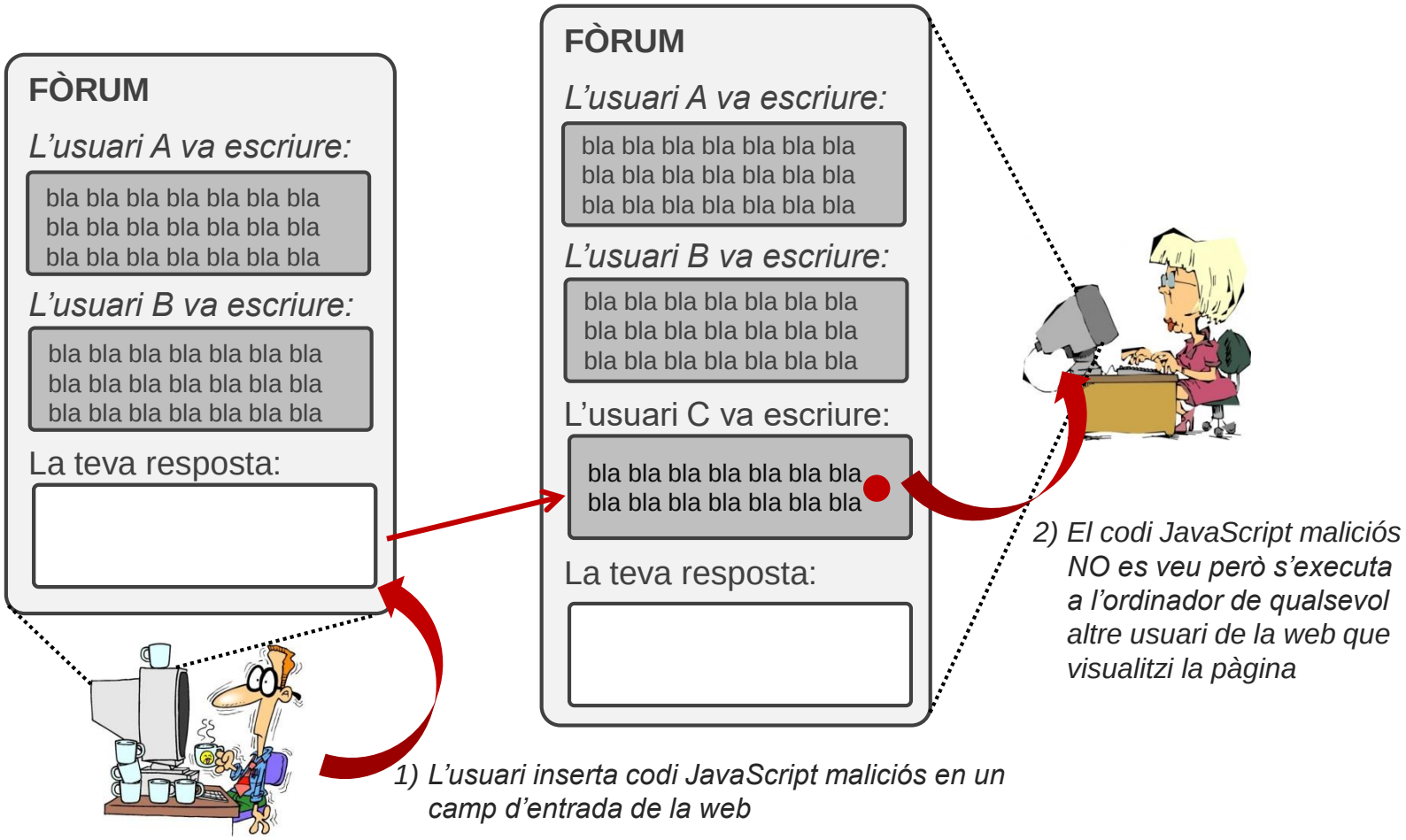
```
<form action="validacio.php" target="_self" method="post">
  ID USUARI:<input type="text" name="idUsuari"><br />
  CLAU:<input type="password" name="clau"><br />
  <input type="submit" value="LOGIN">
</form>
```

validacio.php

```
<?php
  [...]
  $SQL = "SELECT * FROM usuaris WHERE clau = ':clau' AND idUsuari = :idUsuari";
  $consulta = $connexio->prepare($SQL);
  $consulta->bindParam(":idUsuari", $_POST["idUsuari"], PDO::PARAM_INT);
  $consulta->bindParam(":clau", $_POST["clau"], PDO::PARAM_STR);
  $consulta->execute();
  [...]
?>
```

1.5 Aspectes de seguretat. Cross-site scripting

Un atac de "cross-site scripting" (XSS) executa codi JavaScript maliciós a clients d'una web aprofitant que l'entrada d'un usuari s'envia als altres usuaris de la web. Per exemple, en una web de fòrums:



1.5 Aspectes de seguretat. Cross-site scripting

Tot i ser un llenguatge segur, des de JavaScript es pot: 1) manipular el DOM, 2) fet peticions asíncrones amb AJAX, i 3) accedir a "cookies" del navegador. El tipus d'atacs que es poden portar a terme a través de l'execució de codi JavaScript maliciós són els següents:

- ***Robatori de "cookies"***: *a través de JavaScript es pot accedir a algunes de les "cookies" del navegador, de forma que es podrien enviar a usuaris maliciosos. En les "cookies" hi pot haver informació sensible d'usuaris o de les seves sessions.*

- ***Instal·lació d'un "keylogger"****: amb la instal·lació d'un "keylogger" un usuari maliciós pot enviar tots el que l'usuari escriu pel teclat a un altre servidor.*

- ***Atacs de "phishing"****: a través de la manipulació del DOM, codi JavaScript maliciós pot introduir formularis d'accés o d'informació personal de l'usuari falsos que envien la informació a servidors maliciosos.*

1.5 Aspectes de seguretat. Cross-site scripting

La prevenció dels atacs XSS passa per validar i/o reconvertir l'entrada dels usuaris a través de funcions que ens converteixin els caràcters HTML especials (i.e., < > " ' i &) en entitats HTML. Un forma habitual de fer-ho és la següent:

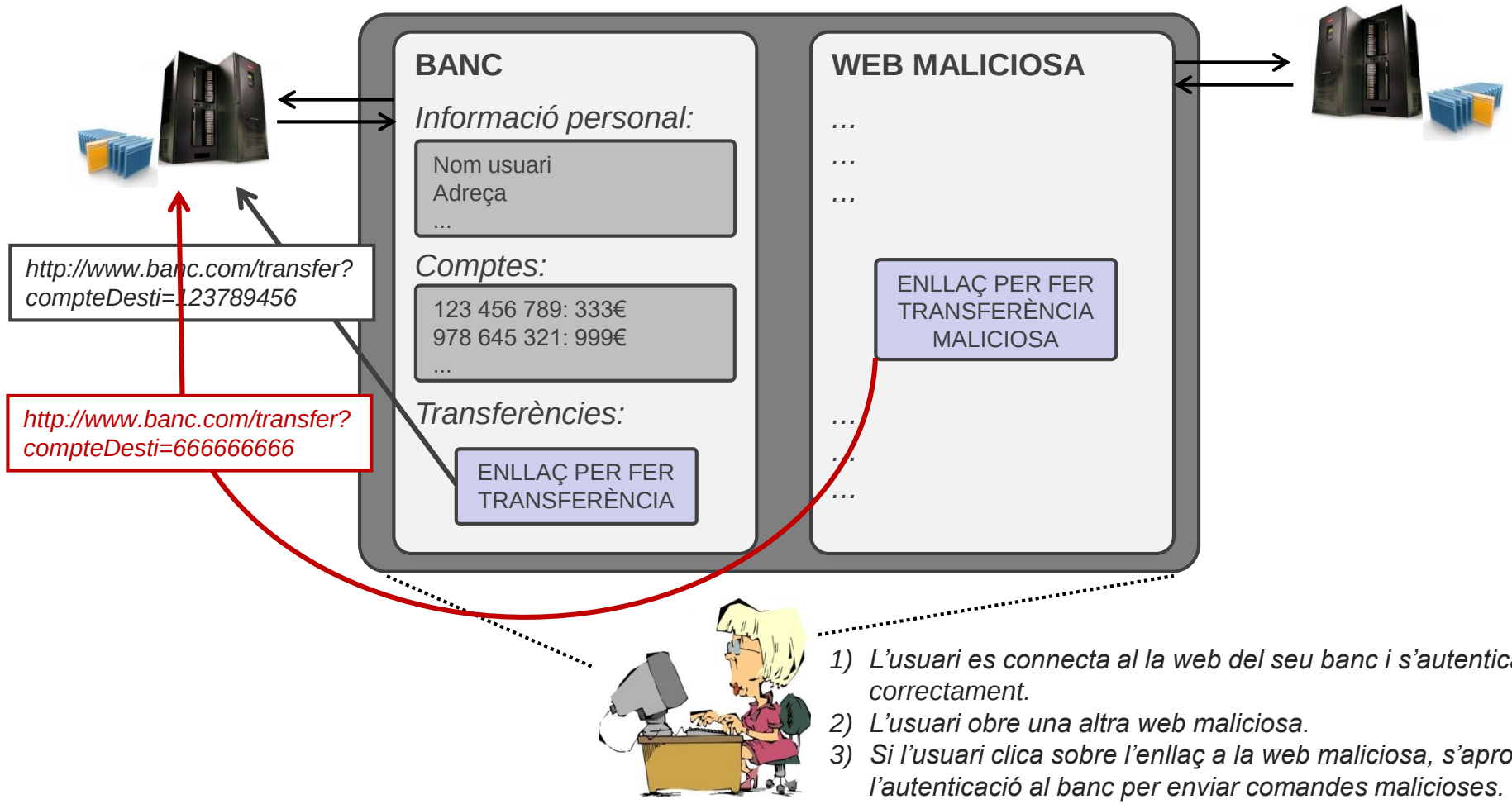
```
<?php
  [...]
  $entradaUsuariConvertida = htmlspecialchars(trim($entradaUsuari));
  [...]
?>
```

bla bla bla bla bla bla bla
<script>codi maliciós</script>
bla bla bla bla bla bla bla

bla bla bla bla bla bla bla
<script>codi maliciós</script>
bla bla bla bla bla bla bla

1.5 Aspectes de seguretat. Cross-site request forgery

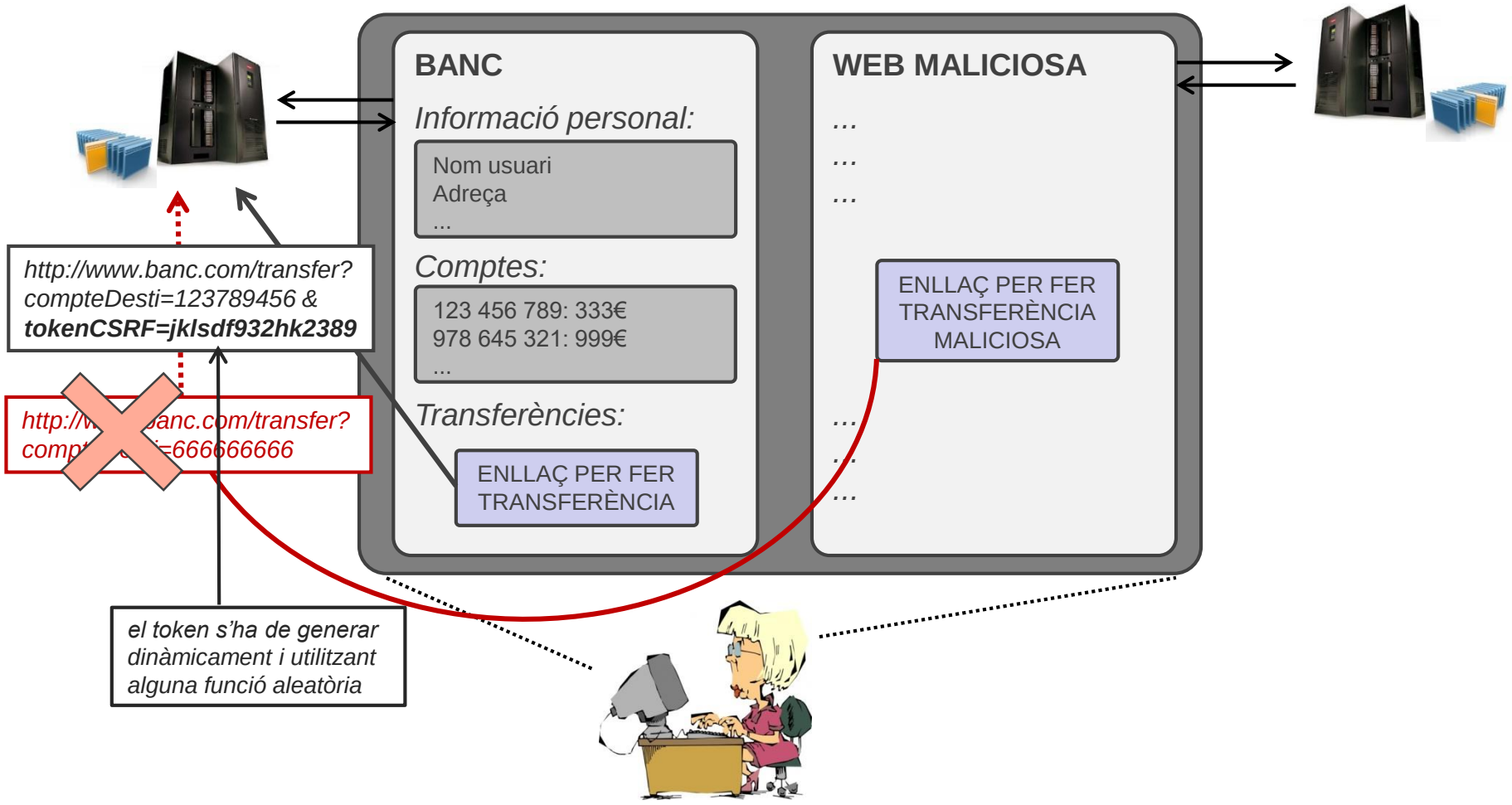
Un atac de "cross-site request forgery" (CSRF) utilitza el fet que un usuari pot estar correctament autenticat en una web per enviar peticions malicioses des d'una altra web. Per exemple:



1) *L'usuari es connecta al la web del seu banc i s'autentica correctament.*
2) *L'usuari obre una altra web maliciosa.*
3) *Si l'usuari clica sobre l'enllaç a la web maliciosa, s'aprofita l'autenticació al banc per enviar comandes malicioses.*

1.5 Aspectes de seguretat. Cross-site request forgery

La prevenció d'atacs CSRF normalment passa per afegir un CSRF token a totes les peticions de la web, de forma que un atacant no pot construir una petició vàlida.



¡PREGUNTA D'EXAMEN!

Un atac de "SQL injection" executa codi JavaScript per accedir a la base de dades?

¡PREGUNTA D'EXAMEN!

Un atac de "cross-site scripting" aprofita l'autenticació de l'usuari en una altra web per enviar comandes malicioses?

¡PREGUNTA D'EXAMEN!

Un atac de "cross-site request forgery" utilitza JavaScript per robar les "cookies" d'un usuari pertanyents a una sessió d'una altra web?