M6-UF1-PROJECTE

Mòdul: MP06- Desenvolupament web en entorn client

UF: UF1 – Servidors web i de transferència de fitxers

Professor: Albert Guardiola

Data límit d'entrega: 24/11/2024 23:59

Mètode d'entrega: Per mitjà del Clickedu de l'assignatura. Les activitats entregades més enllà de la data límit només podran obtenir una nota de 5.

Instruccions: Veure més avall.

Resultats de l'aprenentatge:

RA1. Selecciona les arquitectures i tecnologies de programació sobre clients web, identificant i analitzant les capacitats i característiques de cadascuna.

RA2. Escriu sentències simples, aplicant la sintaxi del llenguatge i verificant la seva execució sobre navegadors web.

En aquesta pràctica, desenvoluparàs de manera guiada una aplicació JS completa. Parteix del codi que es proporciona al repositori:

https://github.com/albertetpx/m6uf1-philocards

i **segueix amb precisió les instruccions** que hi ha en aquest document**.**

L'entrega de la pràctica ha d'incloure:

-URL del **repositori de github amb el codi de l'aplicació completa** (o fins on s'hagi arribat a implementar). S'han de veure commits prou atòmics. Idealment, un al final de cada tasca.

-URL de la **GitHub Page on es publiqui la web**.

https://docs.github.com/es/pages/getting-started-with-github-pages/creating-a-github-pages-site

En aquest vídeo es pot veure la funcionalitat completa de l'aplicació:

https://www.loom.com/share/c90cd05dca88448c9ee3d14acc09b60e?sid=dc36b19d-7ab1-4ac8-bf02-697438394d5d

TASCA 1. Creació dinàmica de targetes:

Carrega el projecte. Inicialment, només es mostra una targeta (Platón), que está codificada a l'HTML. **Volem que es carregui dinàmicamente la informació continguda en la constant filosofos.**

Per fer-ho, tenim preparada la funció ***crearTarjetas*** (l'hauràs de descomentar al *window.onload*), a la qual se li passa com entrada la constant **filosofos**, que és un array d'objectes. L'objectiu d'aquesta funció és crear una targeta HTML per cada objecte present a l'array **filosofos**.

1. Observa que iterem l'array d'entrada amb el seu mètode *forEach*: a cada iteració, l'objecte amb què treballarem es diu **filosofo.**

```
function crearTarjetas(filosofos) {
      filosofos.forEach((filosofo) => {
      (...)
}
```

2. **Hem d'aconseguir que, per cada objecte filosofo, creem en el DOM una tarjeta com la que tenim d'exemple en l'HTML. L'HTML resultant ha de ser exactament igual que el que hi ha a l'HTML.**

3. Fixa't com el primer que fa la funció que li passem al *forEach* és crear la tarjeta buida.

Com que en l'HTML de mostra la tarjeta es crea com ...
 	
`<div class="card">`
	.
	.. al JS el primer que fem és ...
	`// Creamos tarjeta vacía`
	`let tarjeta = document.createElement('div');`
	`tarjeta.classList.add('card')`

4. A continuació:

....com que a dins del ***div.card*** el que trobem és ...

```
<div class="card">
      <img
src="https://upload.wikimedia.org/wikipedia/commons/thumb/d/da/Plato_Pio-
Clemetino_Inv305.jpg/1200px-Plato_Pio-Clemetino_Inv305.jpg" alt="Foto de Filósofo"
class="photo">
      <div class="card-info">
```

```
// Creamos imagen
let imagen =
document.createElement('img');
imagen.src = filosofo.imagen;
imagen.alt = `Foto de ${filosofo.nombre}`;
imagen.classList.add("photo");
tarjeta.append(imagen);

// Creamos caja de informacion
let info = document.createElement('div');
info.classList.add('card-info');
tarjeta.append(info);
```

5. En un moment donat, a l'HTML hi ha més d'un ***div.skill*** a dins d'un ***div.skills***:

```
<div class="skills">
       <div class="skill">
           (…)
       </div>
       <div class="skill">
           (…)
       </div>
       (…)
</div>
```

Per això a JS farem;

```
// Añadimos caja de habilidades
        let habilidades = document.createElement('div');
        habilidades.classList.add('skills');
        info.append(habilidades);
        // Añadimos una a una las habilidades
        for (let infoHabilidad of filosofo.habilidades) {
            // Añadimos una caja de habilidad

            // Añadimos contenido caja de habilidad
```

	// 1.Icono de habilidad // 2.Etiqueta de habilidad // 2.Barra de habilidad } Fixa't com iterem **filosofo.habilidades** per saber quants ***div.skill*** hem de crear a dins del ***div.skills***.

6. **ÉS MOLT IMPORTANT ESTAR CONCENTRAT, SER METÒDIC, I ANAR ANOMENANT LES VARIABLES AMB CRITERI**. Ves comprovant poc a poc al navegador com va quedant les targetes creades des de l'script.

7. La comprovació d'aquesta activitat serà que aconsegueixis crear, al costat de la tarjeta original de Plató, la resta de tarjetes, iguals que la d'exemple. **No et preocupis pels estils. Si vas posant les classes correctament als elements creis al JS, el full d'estils s'encarregarà de tot.**

8. Quan aconsegueixis completar aquesta tasca, ja podràs comentar la tarjeta d'exemple a l'HTML perquè només es presentin les tarjetes creades al DOM per la funció ***crearTarjetas*****.**

TASCA 2. Ara afegirem a cada targeta un botó per poder esborrar-la.

1. A la funció *crearTarjetas*, just abans de fer el darrer *append* de cada tarjeta al seu contenidor...

```
--->AQUÍ VA EL NOU CODI<---
// Añadimos tarjeta creada al contenedor de tarjetas
  let contenedor = document.querySelector('.cards-container');
  contenedor.append(tarjeta);
```

... crea un nou div:

   a. Afegeix-li com a innerHTML el caràcter "&#x2716" (aspa).
   b. Afegeix-li la classe 'botonEliminar'.
   c. Afegeix-li un listener per l'event de 'click', associat a la funció *eliminarTarjeta*.

2. **Completa la funció *eliminarTarjeta*** perquè, en clicar el botó en forma d'aspa, s'elimini la tarjeta corresponent. Recorda que la tarjeta (*div.card*) és el pare del botó (*div.botonEliminar*). Només cal posar a la funció *eliminarTarjeta* una sola línia de codi.

3. Comprova que la nova funcionalitat funciona correctament.

TASCA 3. Es tractaria d'afegir ara al <aside> un formulari per crear noves targetes.

1. Descomenta el formulari que hi ha a l'HTML.
2. Anem a programar la funció ***crearNuevaTarjeta*** perquè creii una nova tarjeta a partir de la informació del formulari.
3. En primer lloc, crea un objecte *nuevoFilosofo* amb la mateixa estructura que els que hi ha a l'array filosofos.

```
function crearNuevaTarjeta(event){
    event.preventDefault();
    let nuevoFilosofo = {};
    nuevoFilosofo.nombre = document.querySelector('.create-card-form .nombre').value;
    nuevoFilosofo.imagen = document.querySelector('.create-card-form .foto').value;
    nuevoFilosofo.pais = {};
    nuevoFilosofo.pais.nombre = document.querySelector('.create-card-form .pais').value;
    // Completar la función

    // crearTarjetas(nuevoFilosofo);
}
```

Observa com anem poblant l'objecte ***nuevoFilosofo*** extraient els valors dels <input> del formulari.

4. Observa també com fem volem fer servir la funció que ja tenim codificaca (*crearTarjetas*) per crear un nova tarjeta a partir del formulari. **Però la funció *crearTarjetas* espera un array com entrada**, ja que el primer que fa és un *forEach*. Com podem solucionar el problema sense tocar la funció *crearTarjetas*?

TASCA 4. Volem ara afegir la funcionalitat d'ordenació alfabètica (en els dos sentits).

1. Afegeix al window.onload els listeners corresponents als botos d'ordenació: associar-los a les funcions ***ordenarNombreAZ*** i ***ordenarNombreZA***, respectivament.
2. Completa la funció ***ordenarNombreAZ***:

```
function ordenarNombreAZ() {
    let tarjetas = Array.from(document.querySelectorAll('.card'));
    let tarjetasOrdenadas = tarjetas.sort((tarjetaA, tarjetaB) => {
        let nombre1 = tarjetaA.querySelector('h3').innerHTML;
        let nombre2 = tarjetaB.querySelector('h3').innerHTML;
        return nombre1.localeCompare(nombre2);
    });

    // Eliminar totes les targetes de l'array 'tarjeta'
    // Completar codi

    // Afegir 'tarjetasOrdenadas' al contenidor de cards
    let contenedor = document.querySelector('.cards-container');
    // Completar codi
}
```

Observa com es fa servir el mètode ***sort*** amb una funció d'ordenació pròpia per fer l'ordenació en base als noms dels autors presents al títol de la tarjeta. Fxa't que:

a. El querySelector es pot fer sobre un element, i es buscarà només a dins d'aquest element (en aquest sub-arbre del DOM).
b. El mètode localeCompare (i de retruc la funció sencera), retornen 0 si els dos valors comparats són iguals, i un número menos o major que 0 depen de quin dels dos sigui més "gran" (tenint en compte que estem ordenant alfabèticament).
c. Cal completar la funció: eliminar del DOM els elements de l'array *tarjetas*; i a continuació afegir-hi els elements de l'array ordenat: *tarjetasOrdenadas*.

3. **Programa la funció *ordenarNombreZA***, basan-te en aquesta.

TASCA 5. Finalment, afegirem la funcionalitat de guardar les tarjetes en local, i de poder-les tornar a carregar en un altre moment.

1. Per fer-ho, farem servir el LOCAL STORAGE. El localStorage és una API del navegador que ens permet guardar dades en parells clau-valor (quelcom semblant a les cookies, pero amb menys limitacions d'espai).

   https://www.geeksforgeeks.org/local-storage-vs-cookies/

2. Implementarem aquesta nova funcionalitat a la funció **guardarTarjetas**, que llegirá el contingut de les targetes del DOM i construirà un array d'objectes amb la mateixa estructura que l'array **filosofos**. La funció **guardarTarjetas** ja està programada:

```
function guardarTarjetas(){
    let tarjetas = Array.from(document.querySelectorAll('.card'));
    localStorage.setItem('tarjetas',JSON.stringify(parsearTarjetas(tarjetas)));
}
```

   Cal observar que:

   a. El *HTMLCollection* que retorna el *querySelectorAll* es transforma a array amb el mètode **Array.from**. Això permet treballar amb l'array amb més facilitat.
   b. El passarem a la funció ***parsearTarjetas***, que retornarà un array d'objectes amb l'estructura de l'array **filosofos.**
   c. El que retorna la funció *parsearTarjetas* és serialtzat amb *JSON.stringify* i emmagatzemat en el *localStorage*.

3. Pel que fa a la funció ***parsearTarjetas***, cal anar recorrent el DOM (concretament, les tarjetes (*.cards*)) i creant l'array d'objectes:

```
let filosofosParseados = [];
    for (let tarjeta of tarjetas){
        let filosofo = {};
        filosofo.nombre = tarjeta.querySelector('.nombre').innerHTML;
        filosofo.imagen = tarjeta.querySelector('.photo').src;
        filosofo.pais = {};
        filosofo.pais.nombre = tarjeta.querySelector('.pais').innerHTML;

        (…)
        let habilidades = tarjeta.querySelectorAll('.skill');
        for (let habilidad of habilidades){
            let habilidadParaGuardar = {};
```

```
            (…)
        }
        filosofosParseados.push(filosofo);
    }
    return filosofosParseados;
```

Cal observar que:

a. Primer es crea una llista buida: *filosofosParseados*.
b. Recorrem la lista de tarjetas (són encara objectes del DOM, tal com els hem recuperat amb el querySelectorAll a la funció *guardarTarjetas*)
c. Per cada tarjeta, creem un objecte buit: *filosofo.*
d. I anem creant les propietats d'aquest objecte, i assignant-les valor a partir del contingut de la tarjeta al DOM. Per exemple:

```
filosofo.nombre = tarjeta.querySelector('.nombre').innerHTML;
```

Fixeu-vos que tornem a fer servir el querySelector sobre l'element HTML: així es buscarà no sobre tot el document, sinò només en aquell sub-arbre.

e. Un cop poblat cada objecte representant d'una tarjeta, l'anem encuant a la llista:

```
filosofosParseados.push(filosofo);
```

4. Per anar depurant la funció cal saber que en les eines del programador del navegador (i.e. Chrome), podem anar inspeccionant el valor de les variables emmagatzemades al localStorage: pestanya *Application*, secció *Local Storage* (veure pàgina següent).

5. Caldrà també afegir el listener corresponent al *window.onload* per associar el click al botó Guardar Tarjetas a la funció corresponent.

6. La funció ***cargarTarjetas*** és molt ràpida de programar. Cal tenir en compte, però, que:

a. S'ha de fer servir el ***localStorage.getItem()*** per llegir el contingut del localStorage.
b. El string que llegim del localStorage s'ha de parsejar per convertir-lo en un objecte de JS.
c. Ja tenim una funció al nostre codi, implementada i provada, que converteix un array d'objectes en tarjetes HTML…
d. Finalment, caldrà afegir també el listener corresponent al botó de CargarTarjetas.


Archivo
C:/Users/alber/OneDrive/Escritorio/philocards/index.html?
Liga de Filósofos
Platón
Grecia
Corriente: Idealismo
Arma: Dialéctica
Sabiduría
Oratoria
Lógica
Innovación
Aristóteles
Grecia
Corriente: Naturalismo
Arma: Lógica
Sabiduría
Oratoria
Lógica
Innovación
Opciones
Guardar Tarjetas
Cargar Tarjetas
Ordenar por:
A->Z
Z->A
Crear Nueva Tarjeta
Nombre
Imagen (URL)
País
Bandera (URL)
Corriente Filosófica
Arma Principal
Elements
Console
Sources
Application
Application
Manifest
Service workers
Storage
Storage
Local storage
file://
Session storage
IndexedDB
Cookies
Private state tokens
Interest groups
Shared storage
Cache storage
Storage buckets
Background services
Back/forward cache
Background fetch
Background sync
Bounce tracking mitigati...
Notifications
Payment handler
Periodic background sync
Speculative loads
Push messaging
Reporting API
Frames
top
Filter
Key
Value
tarjetas
[{"nombre":"Platón","imag...
[{nombre: "Platón",…}, {nombre: "Aristóteles",…
0: {nombre: "Platón",…}
arma: "Dialéctica"
corriente: "Idealismo"
habilidades: [{habilidad: "Sabiduría", nivel
0: {habilidad: "Sabiduría", nivel: 4}
1: {habilidad: "Oratoria", nivel: 4}
2: {habilidad: "Lógica", nivel: 3}
3: {habilidad: "Innovación", nivel: 4}
imagen: "https://upload.wikimedia.org/wikipe
nombre: "Platón"
pais: {nombre: "Grecia",…}
bandera: "https://upload.wikimedia.org/wik
nombre: "Grecia"
1: {nombre: "Aristóteles",…}
2: {nombre: "Kant", imagen: "https://i.pinimg
3: {nombre: "Hume",…}
4: {nombre: "Arendt",…}