Desarrollo de Aplicaciones I – FIUBA

Ejercicio 1

1) Generar un sitio Web con la misma estructura que el ejercicio de maquetado y agregarle la carpeta src.

2) Dentro de esta carpeta generar el archivo “main.ts” e incluir el archivo “tsconfig.json” con la configuración para el compilador de TS como se detalla a continuación.

```
{
    "compilerOptions": {
        "target": "es6",
        "sourceMap": false,
        "outDir": "../js/"
    }
}
```

3) Incluir en el archivo “index.html” el archivo de JS generado “main.js” para que se ejecute al cargar la página.

4) Agregar dentro del archivo main.ts el código:

```
console.log(“Hola mundo”);
```

5) Compilar el src* con el comando:

```
tsc –watch
```

*o si utilizamos Docker, ejecutar el script que lanza el contenedor dentro de src:

```
./compile_ts.sh . ../js/
```

6) Hostear el sitio web* con el servidor estático:

```
http-server . -c-1
```

Siendo “.” el path al directorio actual.

*o si utilizamos docker, ejecutar el script que lanza el contenedor:

```
./serve_http.sh . 8000
```

7) Abrir el navegador y el “inspector” y chequear el funcionamiento del programa.

Ejercicio 2

1) Dentro del archivo main.ts del ejercicio anterior, definir la clase “Main” y dentro de la misma definir un método “main”. Dentro del método imprimir un mensaje.

2) Utilizar la clásica función “onload” de JS para crear un objeto del tipo Main y ejecutar el método “main()”.

Ejercicio 3

1) Definir la clase “User” con los atributos privados “id” (numérico) “name”, “”email” (textos) y “isLogged” (booleano). El constructor de la clase debe recibir el ID, el email y el nombre.

2) Agregar getters y setters para los atributos.

3) Agregar el método “printInfo” el cual imprime por consola los tres atributos.

4) En la clase Main, en el método “main” definir un array de 3 objetos “User”. Luego iterar el array y ejecutar el método printInfo() de cada objeto.

Ejercicio 4

1) Agregar en la maqueta HTML un botón con el id “boton”.

2) Crear la clase MyFramework la cual deberá tener el método “getElementById()” y devolverá el tipo de dato “HTMLElement”.

3) En el main, crear un objeto MyFramework y ejecutar el método para obtener la referencia del botón que se encuentra en el DOM.

4) Modificar el texto del botón desde el programa. (utilizar el atributo textContent del objeto)

Ejercicio 4-2

1) Agregar la clase User definida en el Ejercicio 3 al proyecto del ejercicio 4.

2) A la clase Main, agregarle el método “mostrarUsers” que reciba un array de objetos User y los imprima por la consola.

3) En el método main, definir un array de 3 objetos User e invocar al método “mostrarUsers” y pasarle el array.

Ejercicio 5

1) Partiendo del ejercicio 4, agregar un listener para escuchar el evento de click del botón, para ello, definir en la clase Main el siguiente método:

```
evento(ev:Event):void {
     console.log("se hizo click!");
}
```

2) Setear el listener el elemento botón mediante el método "addEventListener" de la siguiente manera:

```
objBoton.addEventListener("click",this.evento);
```

3) Verificar el funcionamiento.

4) Imprimir dentro del método "evento" el objeto "this" y verificar si es del tipo Main.

```
console.log(this);
```

Ejercicio 6

1) Partiendo del ejercicio 5, cambiar la manera de escuchar el evento utilizando interfaces. Para ello, se deberá implementar la interface "EventListenerObject", esto nos obligará a implementar el método:

```
handleEvent(evt: Event): void
{
}
```

2) Escribir dentro del método un mensaje e imprimir el valor de "this".

3) En la llamada a "addEventListener" ahora se deberá pasar el objeto que implementa la interfaz (this):

```
objBoton.addEventListener("click",this);
```

4) Probar y verificar que el objeto "this" que se imprime en el evento sea del tipo Main

Ejercicio 7

1) Agregar a la clase "MyFramework" un método que reciba un objeto "Event" y devuelva el objeto HTMLElement asociado, se debe llamar "getElementByEvent".

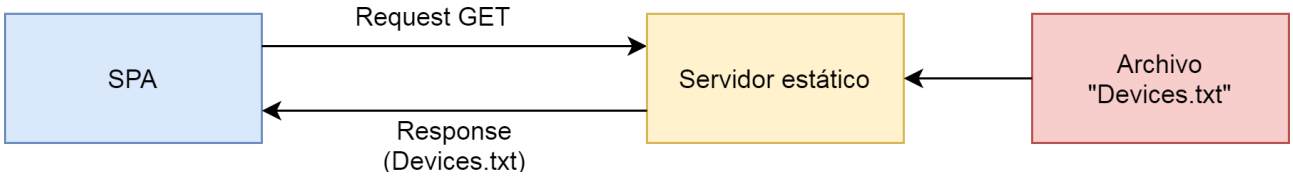
Dentro del método colocar:

```
return <HTMLElement>evt.target
```

2) Utilizar este método en el ejercicio anterior para obtener el elemento sobre el que se produjo el evento. Cambiar el texto al hacer click sobre el botón y mostrar un contador de clicks.

Ejercicio 8

Agregaremos a “MyFramework” un método que nos permita hacer requests del tipo GET. De esta forma, al iniciar la aplicación se ejecutará un request GET al archivo “devices.txt” (el cual deberemos crear) para que la página pueda leer la información que hay en él y posteriormente mostrar la lista de dispositivos de la “smart home”.



1) En MyFramework, definiremos una interfaz que nos permitirá definir el evento que se ejecute al llegar la respuesta del servidor:

```
interface GETResponseListener{
     handleGETResponse(status:number, response:string):void;
}
```

La función que realice al request, deberá recibir un objeto del tipo “GETResponseListener” como argumento, cuando se reciba la respuesta del server, a dicho objeto se le ejecutará el método “handleGETResponse”.

2) Crear el archivo “Devices.txt” en la raiz del sitio web, dentro del mismo colocar el texto “Hola mundo”.

3) Agregar el método “requestGET” a la clase “MyFramework” con la siguiente firma:

```
requestGET(url:string, listener: GETResponseListener):void
```

Dentro del método, podremos usar el clásico objeto de JS XMLHttpRequest para realizar el request del tipo GET.

Ejemplo de request GET:

```
let xhr: XMLHttpRequest;
xhr = new XMLHttpRequest();

xhr.onreadystatechange = function()
{
     if(xhr.readyState == 4)
     {
          if(xhr.status == 200)
          {
               listener.handleGETResponse(xhr.status,xhr.responseText);
          }
          else
          {
               listener.handleGETResponse(xhr.status,null);
          }
     }
};
```

```
xhr.open('GET', url, true);
xhr.send(null);
```

4) Para usar este método al iniciar la aplicación web, agregaremos una llamada al mismo en el método “main”:

```
objMyFramework.requestGET(“Devices.txt”,this);
```

Como primer argumento pasamos la url (en este caso el nombre del archivo ya que estará hosteado en la raiz del sitio), y como segundo argumento, un objeto que implemente la interfaz “GETResponseListener”, por lo que deberemos hacer que la clase “Main” implemente la interfaz, asi podremos pasarle “this” como argumento, y el método que se ejecutará al recibir la respuesta del server, estará en la clase “Main”.

```
class Main implements EventListenerObject, GETResponseListener {
```

5) Al implementar la interfaz, estaremos obligados a agregar el método definido en la interfaz, en la clase Main, por lo que se debe agregar el método “handleGetResponse()”. Imprimir dentro del mismo el texto que llegó desde el server.

Ejercicio 9

1) Cambiar el contenido del archivo “Devices.txt” por un texto con formato JSON. De esta forma se podrán convertir en variables de TS/JS y la información podrá ser representada en el sitio web. El formato del JSON sera:

```
[
  {
    "id":"1",
    "name":"Lampara 1",
    "description":"Luz living",
    "state":"0",
    "type":"0"
  } ,
  {...}
]
```

2) Definir la interfaz para parsear el texto en formato JSON, llamada “DeviceInt”

3) En el método de la clase Main, “handleGETResponse” parsear el texto JSON y transformarlo en un array de objetos “DeviceInt”:

```
let data: DeviceInt[] = JSON.parse(response);
```

4) Imprimir el objeto data para verificar que los datos sean los cargados en el archivo “Devices.txt”.

Ejercicio 10

En este ejercicio, mostraremos la lista de dispositivos en la página web.

1) Tomar de la maqueta original de “smart house” la lista “ul”, definida en el HTML, y dejarla sin elementos “li” (es decir, sin ítems). Agregarle el id “devicesList”:

```
<ul class='collection' id='devicesList'></ul>
```

2) en Main.ts, crear la clase “ViewMainPage” la cual tendrá el método “showDevices” con la siguiente firma:

```
showDevices(list: DeviceInt[]):void
```

Dentro de este método, que recibe el array de objetos creados desde el texto JSON, se deberán agregar elementos “li” al elemento “ul” que tiene el id “devicesList”.

NOTA1: Para obtener el elemento ul, usar el método “getElementById” de la clase MyFramework.
NOTA2: Para agregar html a un elemento, cargar el texto en el atributo “innerHTML”

3) Ejecutar el método “showDevices” luego de parsear el JSON para que se vean en la página.

Ejercicio 11

En este ejercicio, detectaremos el click sobre los elementos “switch” de cada device de la lista.

1) Agregar un id a cada switch de la lista (elementos “input” generados por showDevices()) con el formato “dev_”+id, donde “id” es el id del dispositivo, por lo que los ids de los elementos html quedarán : “dev_1”, “dev_2”, etc.

2) En el método “handleGETResponse”, luego de ejecutar “showDevices”, iterar los elementos del array de devices y según el id de cada uno, obtener la referencia del objeto HTMLElement de cada “input”. Luego asignar un listener para el evento “click” para cada uno. (Volver a ver Ejercicio 6)

3) Al producirse un click sobre uno de los switch, imprimir por consola el id del dispositivo sobre el que se hizo click.

Ejercicio 12

En este ejercicio se agregará un método para hacer requests “POST” hacia el servidor. Enviando datos asociados.

1) Definiremos una interface que nos permitirá definir el evento que se ejecute al llegar la respuesta del servidor:

```
interface POSTResponseListener{
      handlePOSTResponse(status:number, response:string):void;
}
```

2) Agregar el método “requestPOST” a la clase MyFramework, el mismo deberá realizar un request del tipo POST hacia el server, enviando un diccionario de datos. El método debe tener la siguiente firma:

```
requestPOST(url:string, data:object, listener:POSTResponseListener):void
```

Ejemplo de request POST:

```
let formData:FormData = new FormData();
for(let key in data) {
      formData.append(key, data[key]);
}

let xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {
      if(xhr.readyState == 4) {
            if(xhr.status == 200)
                  listener.handlePOSTResponse(xhr.status,xhr.responseText);
            else
                  listener.handlePOSTResponse(xhr.status,null);
      }
};

xhr.open("POST", url);
xhr.send(formData);
```

3) Implementar la interfaz “POSTResponseListener” en la clase Main y escribir el método que define dicha interfaz. (handlePOSTResponse) dentro del mismo imprimir la respuesta del server.

4) Antes de poder enviar un request POST al presionar un switch, deberemos obtener en qué estado está el switch, ya que los datos que deberán enviarse al server son:

- id del dispositivo.
- Nuevo estado del switch.

Para ello, agregar en la clase ViewMainPage el método “getSwitchStateById()” el cual deberá tener la siguiente firma:

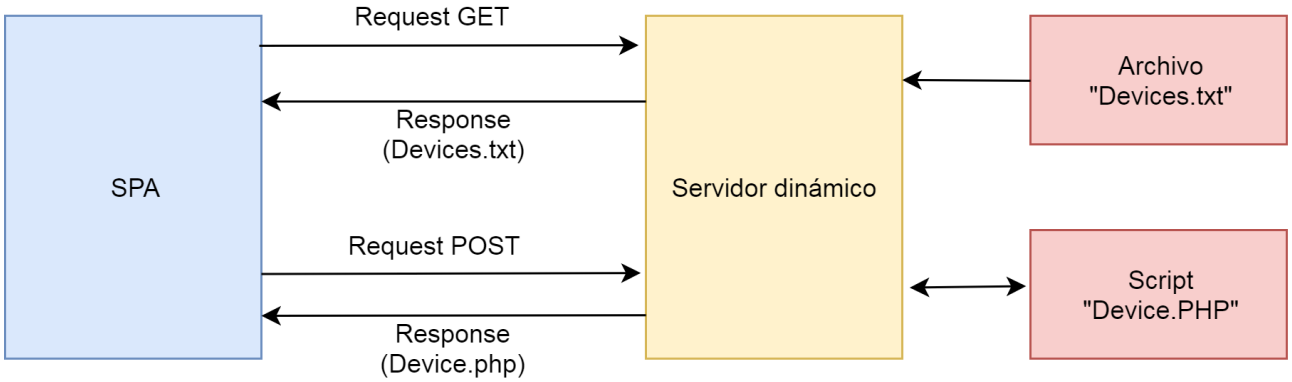
```
getSwitchStateById(id:string):boolean
```

NOTA: Castear el elemento input al tipo “HTMLInputElement” y leer el atributo “checked” para saber si está activo o no.

5) Para poder probar esta funcionalidad, deberemos crear un script PHP que reciba el request y responda algo. Y cambiar el servidor web de prueba estático por un server PHP. Para ello creamos el archivo device.php en la raiz del sitio, con el siguiente contenido:

```
<?php
     print_r($_POST);
?>
```

El diagrama de comunicación entre el cliente y el server quedaría:



6) Luego cerramos el servidor estático http-server y ejecutamos en su lugar el servidor de prueba que viene con PHP7, mediante el comando:

```
php -S localhost:8080
```

De esta manera se servirán los archivos estáticos como antes, y además los archivos PHP se interpretarán y ejecutaran.

7) En el método handleEvent (evento de click sobre un switch) en la clase Main, ejecutar el método “getSwitchStateById” para obtener el estado del switch sobre el que se hizo click, crear el diccionario para enviar por POST con los datos de id y estado, y realizar el request POST a la URL “device.php”, la cual contestará un “eco” de lo que recibió por post.