Aplicaciones Móviles Multiplataforma

LABORATORIO N° 01

React JS

Alumno(s):		***Nota***	
Grupo:		***Ciclo:V***	

Criterio de Evaluación	*Excelente (4pts)*	*Bueno (3pts)*	*Requiere mejora (2pts)*	*No acept. (0pts)*	*Puntaje Logrado*
Entiende cómo funciona los componentes					
Utiliza Next-Gen Javascript					
Desarrolla aplicaciones web con ReactJS					
Realiza con éxito lo propuesto en la tarea					
Es puntual y redacta el informe adecuadamente					

Laboratorio 07:
React JS

Objetivos:

Al finalizar el laboratorio el estudiante será capaz de:

- Entender el funcionamiento de React JS
- Desarrollar aplicaciones web enfocadas a componentes
- Implementación correcta de Next-Gen Javascript

Seguridad:

- Ubicar maletines y/o mochilas en el gabinete del aula de Laboratorio.
- No ingresar con líquidos, ni comida al aula de Laboratorio.
- Al culminar la sesión de laboratorio apagar correctamente la computadora y la pantalla, y ordenar las sillas utilizadas.

Equipos y Materiales:

- Una computadora con:
  - Windows 7 o superior
  - VMware Workstation 10+ o VMware Player 7+
  - Conexión a la red del laboratorio
- Máquinas virtuales:
  - Windows 7 Pro 64bits Español - Plantilla
- Instalador de node.js

Procedimiento:

Lab Setup

1. **Instalación y configuración de ReactJS**
   1.1. Para instalar React, podemos hace uso de su herramienta de generación de proyectos. La podemos instalar con el siguiente comando:

```
~$ npm install -g create-react-app
```

   1.2. En caso tuviera algún problema en la instalación, repita el comando anterior con la instrucción **sudo**

```
~$ sudo npm install -g create-react-app
```

   1.3. Ahora, procedemos a crear nuestro proyecto en React.

```
~$ create-react-app lab07
```

   1.4. Esto dejará listo nuestro entorno de trabajo, por lo que nos situaremos en la carpeta generada y posteriormente iniciaremos el proyecto con **npm start**

```
$ cd lab07/
/lab07$ npm start
```

2. Aplicación en ReactJS

2.1. Esta parte del laboratorio será guiada. El instructor explicará paso a paso como llegar a la aplicación final, un listado de alumnos.

Aprovecha esta explicación en vivo para hacer todas las preguntas posibles, ya que se cubrirán los conceptos básicos de ReactJS. El código final del laboratorio lo puedes encontrar en el siguiente repositorio.

https://github.com/fnaquira/dawa_2018_lab07

3. Next-Gen Javascript

3.1. let y const básicamente reemplazan a var. La diferencia está en que const se utiliza cuando se crea una constante, es decir, una variable cuyo valor inicial no se va a modificar, mientras que let crea una variable a la espera de una modificación.

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const

3.2. **ES6 Arrow functions**, son en realidad una forma distinta de crear funciones en Javascript. Además de tener una sintaxis más corta, ofrecen ventajas al mantener el contexto de this.

Una arrow-function puede parecer extraña pero en realidad es simple.

```
function callMe(name) {
  console.log(name);
}
```

Puede ser escrito también así:

```
const callMe = function(name) {
  console.log(name);
}
```

Puede ser finalmente invocado así:

```
const callMe = (name) => {
  console.log(name);
}
```

Importante: cuando no se tienen argumentos, se puede dejar el paréntesis vacío.

```
const callMe = () => {
  console.log('Max!');
}
```

O cuando se tiene un solo argumento, se pueden omitir los paréntesis.

```
const callMe = name => {
  console.log(name);
}
```

O también se puede dar el caso de que se retorna directamente un valor, por lo que esto:

```
const returnMe = name => name
```

Es equivalente a esto:

```
const returnMe = name => {
  return name;
}
```

3.3. Import y Export

En proyectos de React (en realidad, en todo proyecto moderno de Javascript) puedes separar el código entre múltiples archivos Javascript, llamados módulos. Se recomienda hacer esto para mantener cada archivo o módulo administrable y enfocado a solucionar un problema.

Para acceder a la funcionalidad de otro archivo, necesitas exportarlo (hacerlo disponible) e importarlo (obtener acceso)

Existe dos tipo de exportación: default (unnamed) y named exports:

Default => export default …

Named => export const someData = …

Puede importar default exports de la siguiente manera:

```
import someNameOfYourChoice from './path/to/file.js';
```

El nombre, someNameOfYourChoice es de tu elección.

En cambio, en los named export, el nombre es inalterable.

```
import { someData } from './path/to/file.js';
```

Un archivo puede contener un default export y una ilimitada cantidad de named exports. Puedes combinar esto en un solo archivo.

Cuando importas named exports, puedes también importar todos los named exports del archivo de una sola vez con la siguiente sintaxis:

```
import * as upToYou from './path/to/file.js';
```

upToYou es simplemente un objeto Javascript que contiene todas las variables y funciones exportadas. Por ejemplo, si tu exportas const someData = ... (/path/to/file.js) puedes acceder en upToYou de la siguiente manera: upToYou.someData.

3.4. Spread Operator

El spread operator tiene una sintaxis muy simple: …

Si, el operador es solamente tres puntos. Este operador te permite separar elementos de un array o separar aparte las propiedades de un objeto, por ejemplo:

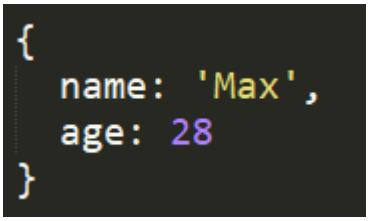
```
const oldArray = [1, 2, 3];

const newArray = [...oldArray, 4, 5]; // Ahora esto es [1, 2, 3, 4, 5];
```

Aquí hay un ejemplo del spread operator usado en un objeto.

```
const oldObject = {
    name: 'Max'
};
const newObject = {
    ...oldObject,
    age: 28
};
```

El nuevo objeto (newObject) sería:



El spread operator es extremedamente útil para clonar arrays y objetos. Dado que ambos referencian tipos (y no primitivos), copiarlos de forma segura (por ejemplo, prevenir futuras mutaciones del objeto original copiado) puede ser dificultoso.

4. **JS Array functions**

4.1. Aunque no son precisamente funciones nuevas, también son muy importantes, las funciones de arreglos en Javascript.

Aquí hay un listado de las más comunes durante el uso de Javascript moderno, ya que es importante saber referenciar correctamente arreglos y objetos de manera que no afecten posibles variables inmutables.

- map() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- find() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find
- findIndex() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/findIndex
- filter() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
- reduce() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce?v=b
- concat() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/concat?v=b
- slice() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/slice
- splice() => https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/splice

5. **Finalizar la sesión**

5.1. Apagar el equipo virtual

5.2. Apagar el equipo

Conclusiones:

Indicar las conclusiones que llegó después de los temas tratados de manera práctica en este laboratorio.