**Curso de programación reactiva RxJs.**

1. **Sección 2: Introducción a las extensiones reactivas y a la programación reactiva.**
2. **Sección 3: Observables.**
3. **Sección 4: Funciones para crear Observables.**
4. **Sección 5: Operadores básicos.**
5. **Sección 6: Operadores no tan comunes.**
6. **Sección 7: Operadores que trabajan con tiempo.**
7. **Sección 8: Ajax - Peticiones ajax usando RxJs/ajax.**
8. **Sección 9: Operadores de transformación.**
9. **Sección 10: Operadores y métodos de combinación de  
   observables.**
10. **Sección 11: Ejercicios de reforzamiento.**
11. **Sección 12: Fin del curso.**

**Sección 2: Introducción a las extensiones reactivas y a la programación reactiva.**

En esta sección aprenderemos sobre:

* Conceptos generales de RxJs y las extensiones reactivas
* ¿Qué es ReactiveX?
* ¿Cómo funciona?
* ¿Qué es el patrón observable?
* ¿Cómo leer los diagramas de canicas?
* Y otros temas

Es una sección corta llena de teoría, pero es indispensable que lo aprendamos para poder tener un mejor panorama de la programación reactiva.

**Video #07: conceptos generales.**

En esta clase empezamos con la pregunta “¿por qué usar extensiones reactivas?”, la respuesta que nos dan es porque desaseamos información en tiempo real, hoy en día no es viable para los usuarios hacer una recarga en la página para obtener información de manera de tiempo real sin que el usuario deba interactuar con un botón para obtener esa información.

¿En qué momento es correcto utilizar extensiones reactivas?

* Eventos de interfaz de usuario.
* Cuando es necesario notificar sobre cambios en un objeto.
* Comunicaciones por sockets.
* Cuando necesitamos trabajar con flujos de información.

Los pilares de la programación reactiva son los Observables Subscribers Operators.

**Observables:**

* Son la fuente de información.
* Pueden emitir múltiples valores, sólo uno o ninguno.
* Pueden emitir errores.
* Pueden ser infinitos o finitos. (completarse)
* Pueden ser síncronos o asíncronos.

**Subscribers**

* Se subscriben a un observable, es decir, estar pendiente de lo que realizar el observable.
* Consumen / observan la data del observable.
* Pueden recibir los errores y eventos del observable.
* Desconocen todo lo que se encuentra detrás del observable.

**Operators**

* Usados para transformar Observables (map, group, scan ).
* Usados para filtrar observables (filter,, distinct, skip, debounce).
* Usados para combinar observables.
* Usados para crear nuevos observables.

**Video #08: ¿Qué es ReactiveX ?,¿Como funciona?**

ReactiveX funciona con las tres siguientes bases, el patrón observador, el patrón iterador y la programación funcional.

**Observer Pattern**: es un patrón de diseño de software que define una dependencia del tipo de uno a muchos entre objetos, de manera que cuando uno de los objetos cambia de estado, notifica este el cambio a todos los dependientes.

**Iterador Pattern:** En POO, el patrón iterador define una interfaz que declara los métodos necesarios para acceder secuencialmente a un grupo de objetos de una colección.

**Programación funcional:** Es básicamente, crear un conjunto de funciones que tengan un objetivo específico. Es decir, si tengo una función que reciba ‘A’ y retorna ‘A+1’, siempre que yo llame esa función retornara ‘A+1’.

Sin efectos secundarios, sin mutar la data.

**Video #09: ¿Cómo leer los diagramas de canicas?**

Para identificar un observable se usa el signo de dólar.