Cycle JS

David Pérez

Índice

- ¿Qué es Cycle JS?
- ¿Qué es programación funcional?
- ¿Qué es programación reactiva?
- Cycle JS
- Cycle Core
- Cycle DOM
- Patrón MVI
- Componentes
- Drivers
- Ejemplos
- Explicación proyecto

¿Qué es Cycle JS?

- Cycle JS es un “framework” de JavaScript que nos permite realizar programación reactiva y funcional.
- Composición Cycle js:
  - xJS: 71,1%
  - virtual-dom: 9,76%
  - @cycle/dom: 4,87%
  - src: 2,47%
  - @cycle/core: 0,64%
  - misc: 10,77%

Programación funcional

- La programación funcional es un paradigma en el que el estado se modifica mediante funciones evitando el uso de una programación imperativa y favoreciendo que el estado sea inmutable.
  - High Order Functions
  - Pureza
    - Inmutabilidad
    - Transparencia referencial
    - Evaluación perezosa
    - Side Effects
  - Recursividad

High Order Functions

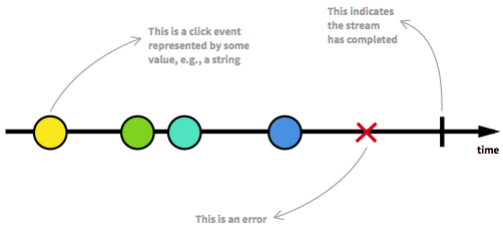
- Son funciones que reciben funciones como parámetros. Un claro ejemplo de High order function es map() el cual recibe como parámetros una lista y una función y aplica dicha función a todos los elementos de la lista.

Pureza

- Inmutabilidad
  - Un objeto no puede cambiar su estado tras su creación.
- Transparencia referencial
  - Una expresión siempre se evalúa al mismo resultado en un mismo contexto.
- Evaluacion perezosa
  - Es una estrategia de evaluación que retrasa el cálculo de una expresión hasta que su valor sea necesario y que evita repetir la evaluación en caso de ser necesaria en posteriores ocasiones.
- Side Effects
  - Modificaciones en el estado (cambiar valor de variable, escribir algo en pantalla…)

Programación Reactiva

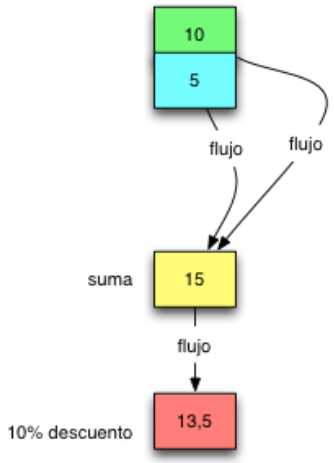
- La programación reactiva es programar con flujos de datos asíncronos (Observable en Cycle JS).



- Todo puede ser un flujo de datos: eventos, variables….

Programación Reactiva

- Con estos flujos de datos conseguimos que unas cosas dependan de otras de tal forma que si cambia un valor en un punto cambia en todo lo que depende de este.



Programación Reactiva

- A estos flujos de datos se les pueden aplicar funciones funcionales que nos permiten crear, filtrar y combinar todos estos flujos de datos.
- Al utilizar cualquier función esta genera otro flujo de datos, de manera que la primera no se haya modificado. Inmutabilidad
- Ejemplo:

  Flujo de datos que representa cada clic en un botón:

  ```
  ----1------1-------1------1----->
  ```

  Puedo recorrer el flujo de datos aplicando una función tantas veces como elementos haya con la función map().

Cycle js

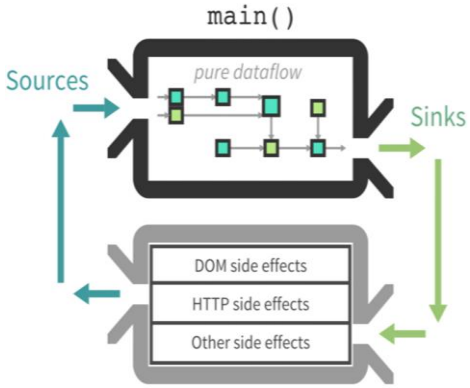
- Cycle Js se compone de Core y Dom.
- Cycle Core: es en una abstracción en la que tu aplicación es una función pura main(sources).
- Cycle Dom: es un conjunto de funciones.

Cycle Core

Es una función run(main, drivers) la cual genera un ciclo entre la función main y los drivers.

La función main() toma como parámetros efectos de lectura (sources) y retorna efectos de escritura (sinks).

Drivers es un objeto cuyas claves son nombres y cuyos valores son funciones.



Cycle Dom

- Se compone de varias funciones:

  Las mas importantes son:

  - makeDOMDriver(): toma como parámetro un contenedor donde en el DOM donde el driver operara y puede recibir también un objeto con distintas propiedades. Retorna una colección de Observables.
  - makeHTMLDriver(): toma como parámetros un Observable de elementos del Virtual DOM y devulve un Observable de strings renderizados.
  - h(): atajo a virtual-hyperscript para crear VTrees en las vistas
    - `h(selector, propiedades, hijos)`
    - ```
      h( ‘ div.clase ’, { style: { color: red } } , [
            h( ‘p’ , ’Hola’ )
      ])
      ```
      ```

Patrón MVI

Este patrón hace referencia a MODEL, VIEW, INTENT, sirve para refactorizar la función main().

- Model: su función es manejar el estado recibe Observables que son acciones y devuelve el estado.
- View: renderiza el estado, recibe el estado y devuelve un Onservable que representa los efectos de escritura del DOM.
- Intent: interpreta los eventos que produce el usuario como acciones.

Componentes

Un componente es una pieza reutilizable de código.

Es una función que recibe al igual que main() efectos de lectura y puede recibir propiedades y que devuelve efectos de escritura.

Usar un componente es tan sencillo como llamar a una función.

Drivers

- Son funciones que reciben efectos de escritura y que realizan alguna modificación en el estado o tiene alguna interaccion.
- Los mas utilizados son DOM driver y HTTP driver aunque hay bastantes mas y puedes crear uno para un cierto propósito.
  - DOM driver: es un proxy que interactúa entre el usuario y el navegador, este muestra la web al usuario y capta las acciones realizadas por el usuario.
  - HTTP driver: sirve para realizar peticiones AJAX. Es una función que recibe la petición y devuelve una respuesta.

Ejemplos

- Hola Mundo: http://jsbin.com/luxahayuzi/edit?js,output
- Contador: http://jsbin.com/vinojeqite/edit?js,output