Ramda js es una de las librerías que esta adquiriendo más tracción en el universo de JavaScript . Se trata de una librería orientada a la programación funcional como ya lo son Lodash o Underscore . ¿Qué es lo que hace a Ramda js diferente?. Para entenderlo hay que construir un ejemplo concreto con Lodash . Vamos a usar un array con alumnos y calcular la media de todas las notas que sean menores de 5 .

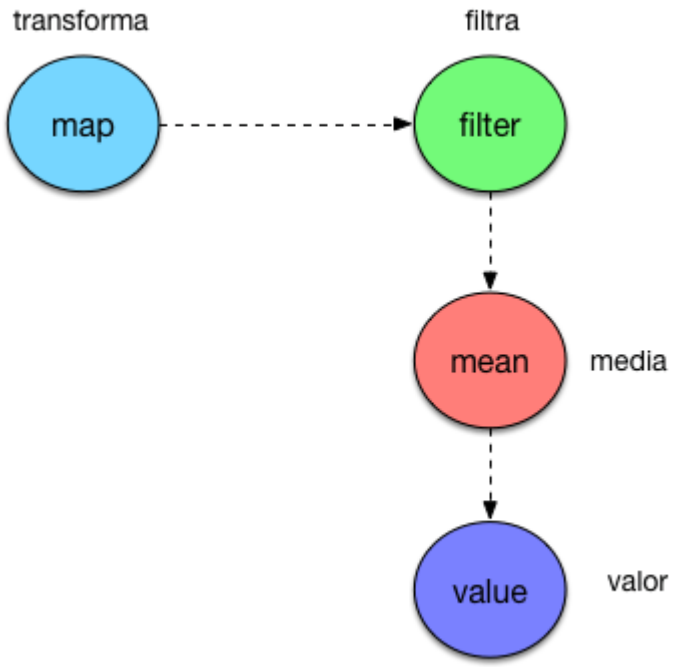
```
var lista = [{
        nombre: "pepe",
        nota: 1
    }, {
        nombre: "ana",
        nota: 5
    }, {
        nombre: "gema",
        nota: 7
    }, {
        nombre: "angel",
        nota: 3
    }];

    var resultado = _.chain(lista)
        .map(item=>item.nota)
        .filter(item=>item<5)
        .mean()
        .value();


    console.log(resultado);
```

Se trata de una operación natural, primero usamos map para quedarnos únicamente con las

notas de cada alumno. Acto seguido filtramos las que son menores de 5 con filter. Una vez filtradas usamos el método mean y calculamos la media. Finalmente solicitamos el valor de la media calculada y lo imprimimos en la consola , nos imprimirá 2.



Estamos acostumbrados a esta sintaxis fluida a la hora de trabajar con programación funcional. ¿Cómo enfoca Ramda?

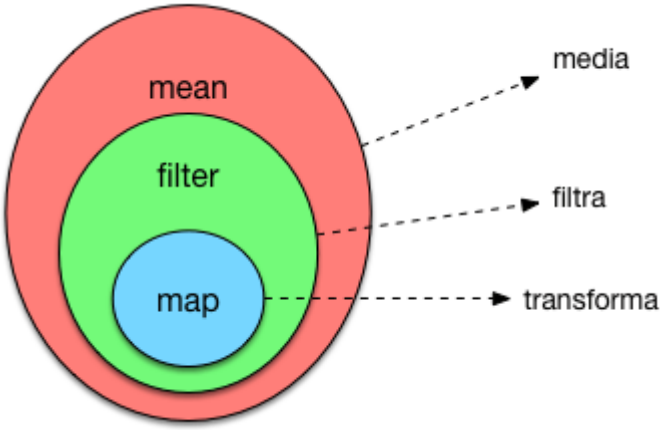
Usando Ramda js

Rambda js es una librería de JavaScript orientada a la programación funcional . Lo que destaca de Ramba js es su enfoque. Vamos a ver el mismo ejemplo anterior con Ramda.

```
var
resultado2=R.mean(R.filter(item=>item<5,R.map(item=>item.nota,lista)))
```

;

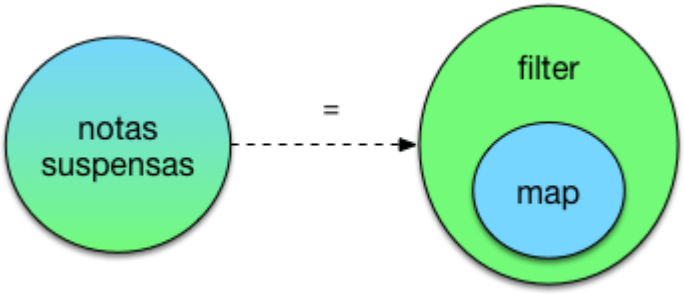
El código es diferente y más compacto , pero también más difícil de entender en un primer momento. Sin embargo es una forma de expresarse más natural y más matemática.
 Primero se aplica map sobre la lista , después sobre el resultado filter y finalmente mean.



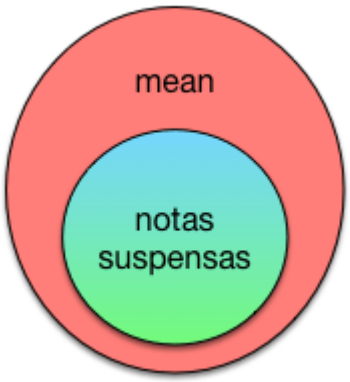
En matemáticas definiríamos el funcionamiento como mean(filter(map())). Al principio no convence mucho . Pero la programación funcional hace mucho hincapié en la reutilización y creación de nuevas funciones. Es en esta parte donde Ramda brilla. Supongamos que queremos generar una función que calcule las notas suspensas:

```
 var
notasSuspensas=lista=>R.filter(item=>item<5,R.map(item=>item.nota,list
a));
var media= R.mean(notasSuspensas(lista));
```

Es muy sencillo ,acabamos de crear una nueva función notasSuspensas a partir de map y filter



Luego hemos usado esa nueva función para calcular la media:



Ramda js nos permite trabajar una forma mucho más natural con los conceptos de programación funcional en javascript. Eso sí tendremos que realizar un esfuerzo a la hora de adaptarnos a la nueva sintaxis.

Otros artículos relacionados: JavaScript currying y funciones parciales. JavaScript pure functions y su uso ,JavaScript map y su implementación