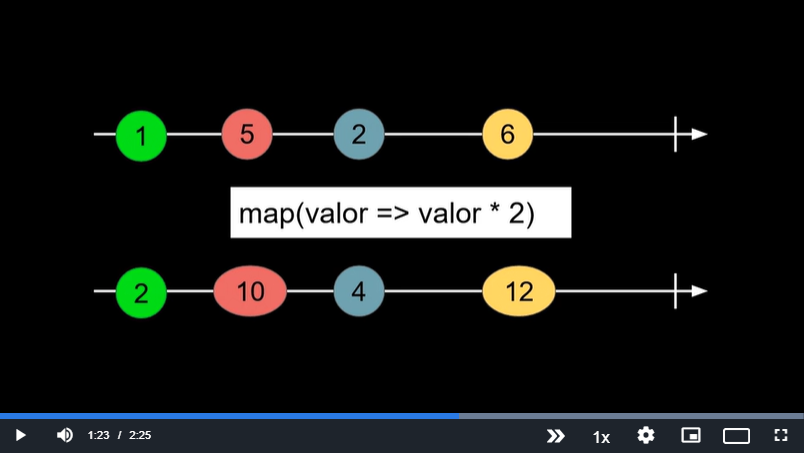
00:00] Até agora, começamos a explorar as vantagens de utilizar os operadores para controlar o fluxo de informações e uma ferramenta fundamental para entendê-los são os Diagramas de Marble, em inglês, bolinhas de gude.

[00:13] São com esses diagramas que a documentação oficial explica o funcionamento da maioria dos operadores. À primeira vista, eles podem parecer meio confusos, mas, se analisarmos elemento por elemento, vamos notar que eles não são um bicho de 7 cabeças.

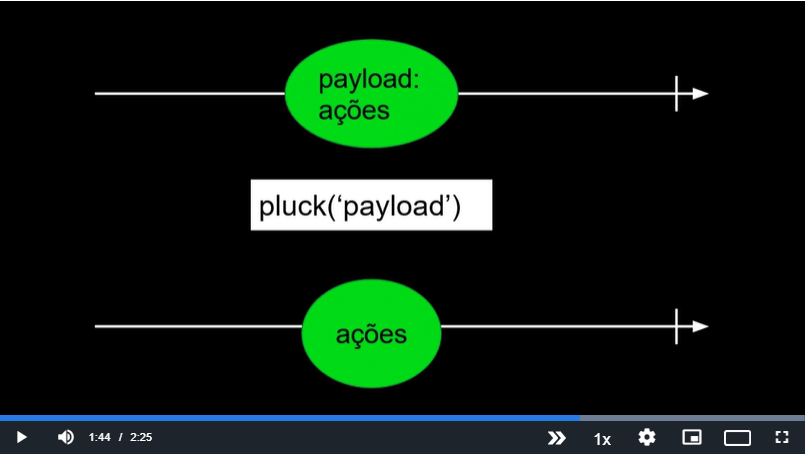
[00:29] Tudo começa com uma seta, que representa o fluxo de dados, também conhecido como string. O traço, no final dela, indica que o fluxo acabou ou foi interrompido. A bolinha de gude, que dá nome ao gráfico, representa um elemento desse fluxo . Podemos ter 1 ou N elementos num fluxo. A outra seta representa o fluxo do resultado da Ação que o operador realiza. Neste exemplo, é o map que irá dobrar o valor de cada elemento do fluxo que passar por ele: map(valor => valor\*2.

[01:01] Temos, no exemplo, a transformação do primeiro e a transformação dos demais. A posição das bolinhas também é muito importante porque representa o tempo que a ação foi feita.

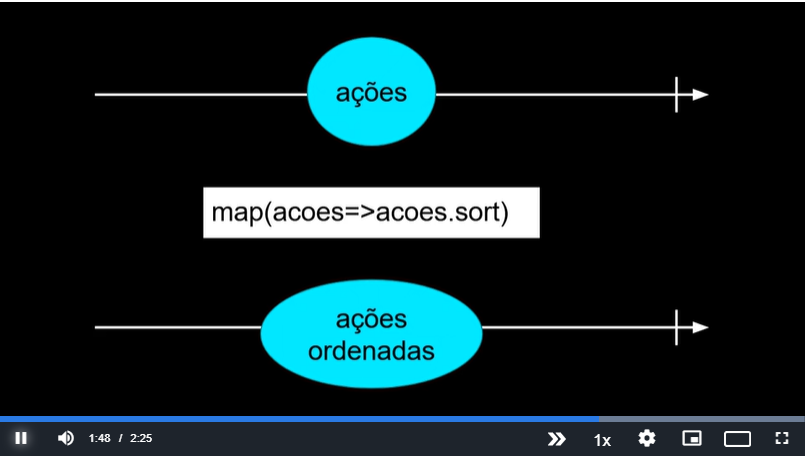
[01:14] Vamos estudar operadores que podem manipular esse tempo e, no caso, o map, executa a operação no mesmo momento em que a informação chega.



[01:24] Aplicando nossa implementação anterior, nós temos o HTTP que é um fluxo que possui apenas 1 elemento, no caso, um objeto que tem uma propriedade chamada payload, com array de Ações. Aplicando o operador pluck, pluck('payload'), nós extraímos o que está no atributo e, então, temos apenas o valor, que no caso, é um array de ações.



[01:46]Com esse array de ações, utilizamos o operador map do Observable para pegar esse array e realizamos um sort por código da ação: map(acoes=>acoes.sort). É importante lembrar que esse map não é o map do JavaScript, é o map do RxJS.



[02:02] Por fim, temos o fluxo que o async vai pegar com as ações já ordenadas. Conhecendo esse gráfico, você vai conseguir ler melhor as informações da documentação da biblioteca.

[02:15] E agora, vamos explorar mais situações onde o RxJS pode nos ajudar no nosso projeto.