Flux es un patrón de diseño que modela el flujo de datos de nuestra aplicación. Hasta ahora, el patrón modelo-vista-controlador (MVC) se había instaurado como una de las mejores alternativas a la hora de diseñar nuestras aplicaciones de Frontend. AngularJS 1, EmberJS y Backbone son buenos ejemplos.

Flux nace como alternativa a MVC. El problema que se encontró Facebook con MVC **reside en las interacciones entre la vista y el modelo**. Cuando estas interacciones crecen y distintas vistas requieren de distintos modelos, suscribirse a los cambios y actualizar estas no era algo trivial.

Con Flux los datos viajan en un solo sentido. Todas las vistas leen del mismo lugar y son capaces de suscribirse a los cambios. Si alguna vista requiere la actualización de un dato, no lo hace directamente. Para ello, necesita llamar a distintos métodos que se encargarán de actualizar el *estado actual* correctamente.

Flux se compone de cuatro elementos:

- **Action creators**: son funciones que formatean los mensajes de manera que el sistema los entienda
- **Dispatchers**: interpretan el mensaje y realizan cambios sobre el *store*
- **Store**: contiene los datos actuales. Es inmutable y cada cambio que realizan los dispatchers genera un nuevo estado
- **Views**: representan los datos del store y se suscriben a sus cambios. También pueden lanzar acciones para actualizar el store.

Si pensamos en React, estos conceptos son relativamente familiares. Las propiedades de un componente no pueden ser modificadas directamente. El componente debe de ejecutar *callbacks* en el padre para que estas se actualicen. Además, cada cambio en las propiedades provoca una renderización.

Como vemos, los principios son muy similares. Por ello Flux se integra especialmente bien con aplicaciones basadas en componentes web como React.