







Frameworks

Tema 4

Paradigmas de programación

Imperativa vs Declarativa

Programación Imperativa

-  Secuencias de comandos que realizan acciones en su mayoría sin relación directa con el dominio del problema
-  Suelen estar acopladas a un contexto, concretamente a su estado interno
-  Es poco descriptivo del problema real que está resolviendo
-  Expresa claramente como se lleva a cabo la operación

Programación Orientada a objetos



Basada en la programación imperativa



Describen objetos del mundo real y cómo se relaciona con el resto a través de interfaces



Abstracción de datos y modularidad



Encapsula objetos que contienen funciones y variables



Herencia y poliformismo

Programación Declarativa

Cuando las expresiones que componen un programa cumplen la transparencia referencial

Transparencia Referencial

Una expresión **e** es transparente referencialmente, cuando para todos los programas **p**, cada ocurrencia de **e** en **p** puede ser reemplazada por el resultado de evaluar **e**, sin que afecte al resultado de cualquier **p**



Transparencia Referencial

WAT?!!








Programación Declarativa

Todas las funciones deben ser puras




Funciones Puras

-  El único output observable es el valor de retorno
-  La única dependencia de su output son los argumentos





Programación Declarativa

-  Cuando tu código describe que quieres hacer y no cómo hacerlo
-  SQL es un gran ejemplo. HTML....
-  Suelen estar desacopladas del contexto, con la consecuente reusabilidad
-  Pueden optimizarse mejor, dado que no se especifica que pasos a seguir
-  Es menos informativo a la hora de expresar las mecánicas de cómo se hace
-  Inmutabilidad
-  Expresiones declarativas expresan solo las relaciones lógicas, jerárquicas de sus partes




Programación Funcional

-  Funciones como ciudadanos de primer orden
-  Composición de funciones
-  Conjunto de utilidades para crear streams de datos (map, reduce, filter)

Programación Reactiva

-  Programación con streams de datos asíncronos
-  Funciones para combinar, mergear, crear, filtrar
-  Observer design pattern
-  Inmutabilidad

JavaScript

-  JavaScript es una mezcla de estos paradigmas
-  El programador tiene libertad para utilizar el paradigma que mas se adapte al problema a resolver
-  Diferentes frameworks se centraran en diferentes problemas

JavaScript

Historia de los frameworks en 1 minuto



VanillaJS



jQuery o spaghetti code



Backbone y MVC similares



Angular monolítico



React/Redux

JavaScript

Historia de los frameworks en 1 minuto



VanillaJS



jQuery o spaghetti code



Backbone y MVC similares







Angular monolítico



React/Redux

El problema: Acoplamiento

Las aplicaciones JS tienden a ser un caos

-  La lógica del interfaz se mezcla con
-  La lógica de control y validación de datos y con
-  La lógica de comunicación con el servidor y con
-  La lógica de reacción a eventos!

El problema: Acoplamiento

El peor tipo de código espagueti!

```
$.ajax({
  url: 'events/since',
  data: { timestamp: old_id },
  dataType: 'json',
  type: 'GET',
  global: false,
  cache: false,
  success: function(newEvents) {
    if (newEvents.events && newEvents.events.length != 0) {
      if (prepend_events) {
        if (page <= '1') {
          if ($('#div.new_events').length == 0) {
            $('#events').prepend('<div class="note new_events"><strong
class="new_event_count">'+newEvents.events.length+'</strong> New Events Are Available Click
here To View Them.</div>');

```

MVC

En los 70 se propuso una solución:

 Separar el código en 3 componentes:

Modelo: datos y lógica de negocio

Vista: presentación de los datos

Controlador: gestión de interacciones

 Limitar su comunicación

 Muy popular desde hace mucho tiempo!

Aplicaciones de escritorio

Aplicaciones móviles

Una interpretación peculiar en los frameworks web

MVC

Hace pocos años se extendió en el cliente

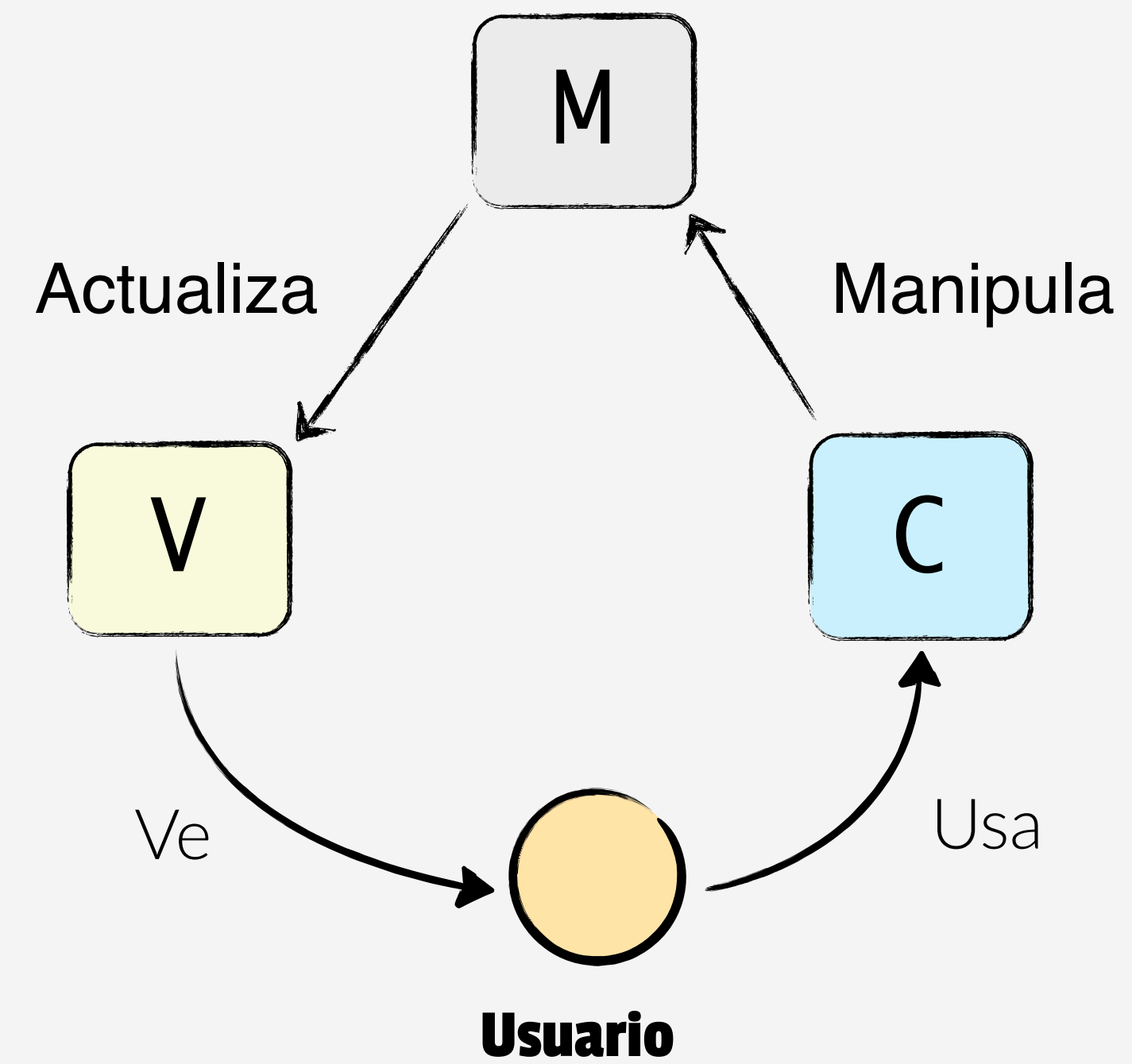
 Backbone.js

 Angular.js






 Ember.js

 ...

MVC



Desventajas

-  Oculta los datos, complejidad accidental
-  Difícil seguir el flujo, updates en cascada, difícil de depurar
-  No es predecible
-  Rendimiento mejorable en aplicaciones grandes
-  Manipulación del DOM es costosa y compleja

Paradigmas de programación

Imperativa → Declarativa

React

 Librería para construir **interfaces de usuario**

 No es un framework, es una librería para UI

 **UI = f(datos+estado)**

React

Es declarativo

React

```
class CourseList extends React.Component {
  render() {
    const { company } = this.props
    return (
      <div className="course-list">
        <h1>Lista de la compra de {company}</h1>
        <ul>
          <li>Curso de react</li>
          <li>Curso de ES6</li>
          <li>Curso de JSPro</li>
        </ul>
      </div>
    )
  }
}

// <CourseList company='redradix' />
```

React

```
function CourseList(props) {  
  const { company } = props  
  return (  
    <div className="course-list">  
      <h1>Lista de la compra de {company}</h1>  
      <ul>  
        <li>Curso de react</li>  
        <li>Curso de ES6</li>  
        <li>Curso de JSPro</li>  
      </ul>  
    </div>  
  )  
}  
  
// <CourseList company='redradix' />
```

Qué es el DOM?



Document Object Model



Representación en memoria del código HTML





HTML DOM es una API para recorrer y modificar los nodos del DOM

Enormes arboles cuyos nodos tienen que cambiar incesantemente para cumplir los requerimientos de las SPAs

- 1 **No** queremos escribir nosotros esa lógica
- 2 Queremos que se haga **eficientemente**

1 No queremos hacerlo nosotros: React es declarativo

 No implementamos las modificaciones en el DOM

 Solo describimos como queremos que nuestro componente se pinte

Virtual DOM

2 **Eficiente:** Virtual DOM

- ❓ Es una abstracción muy ligera del DOM
- ❓ Solo describimos como queremos que nuestro componente se pinte
- ❓ Es un conjunto de ReactElements

ReactElement

 Es una representación virtual de un elemento del DOM

 Ligera

 Inmutable

 Sin estado

 Virtual

 Viven en el DOM virtual. Son sus elementos básicos





ReactElement

```
let root = React.createElement('div') // esto es un reactElement  
ReactDOM.render(root, document.getElementById('container')) // Ahora ya es un elemento del DOM
```


ReactElement

```
let root = <div/> // esto es un reactElement  
ReactDOM.render(root, document.getElementById('container')) // Ahora ya es un elemento del DOM
```

ReactComponent

-  Es una representación virtual de un elemento del DOM
 -  Tienen un ciclo de vida
 -  Pueden tener estado
-  Cuando su estado cambia, el componente se reendea de nuevo

ReactComponent

```
class Clock extends Component {
  constructor(props, context) {
    super(props, context)
    this.state = {time: Date.now()}
  }
  updateClock() {
    this.setState({time: Date.now()})
  }
  componentDidMount: function() {
    this.interval = setInterval(this.updateClock.bind(this), 1000);
  },
  componentWillUnmount: function() {
    clearInterval(this.interval);
  },
  render() {
    const { time } = this.state
    <h1>{time}</h1>
  }
}
```

ReactComponent

```
let element = React.createElement(Clock)  
// let element = <Clock/>
```

Diff algorithm y reconciliación

Cuando hay un cambio en un `ReactComponent`, se crea un `ReactElement`, colaborando en la composición de un nuevo `virtualDOM`.

Se procede a hacer un diff entre ambos `virtualDOMs`. Dado que son `stateless`, es muy rápido.

Se determinan las operaciones mínimas a hacer sobre el DOM para aplicar los cambios.

¿Cómo hacer aplicaciones react?

- 1 Dividir la UI en componentes
- 2 Crear orden jerárquico
- 3 Construye un componente estático
- 4 Probar a modificar el modelo y volver a llamar a ReactDOM.render
- 5 Identificar la minima representación del estado de la UI
- 6 Identificar a quien corresponde cada parte del estado
- 7 Provocar un rerender

¿Cómo hacer aplicaciones react?

- 1 Dividir la UI en componentes

centralLog - Events

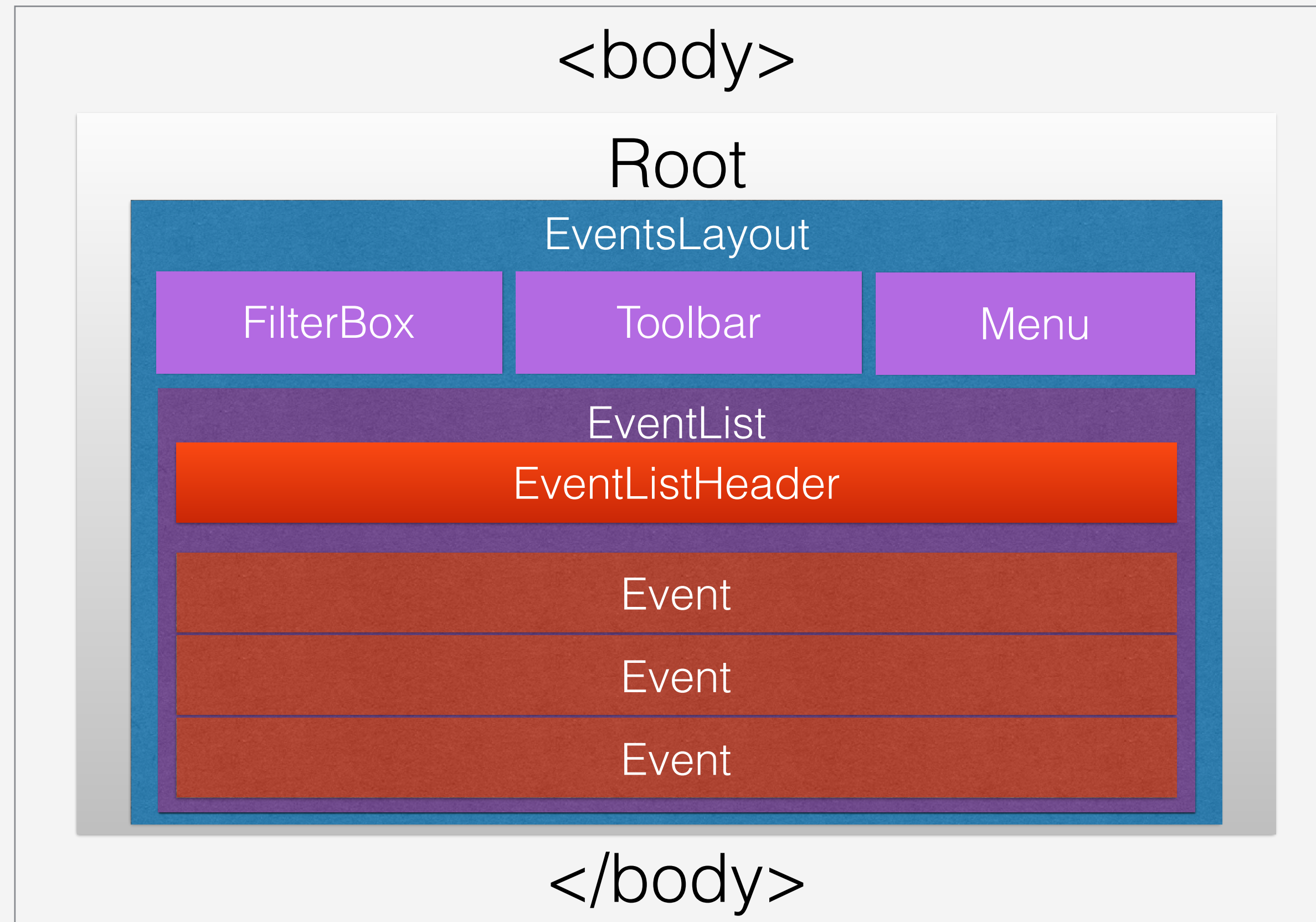
Anonymous (Guest)

Go to

MODE LIVE

Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
✖	🔊	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
ⓘ	🔊	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
⚠	🔊	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
⚠	🔊	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
ⓘ	🔊	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
✖	🔊	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
⚠	🔊	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
✖	🔊	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
✖	🔊	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
✖	🔊	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
ⓘ	🔊	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
ⓘ	🔊	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
✖	🔊	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
⚠	🔊	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
ⓘ	🔊	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
⚠	🔊	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
⚠	🔊	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

Composición



Composición

centralLog - Events

Anonymous (Guest)

Go to

MODE LIVE

Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
<div></div>	<div></div>	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
<div></div>	<div></div>	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
<div></div>	<div></div>	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
<div></div>	<div></div>	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
<div></div>	<div></div>	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
<div></div>	<div></div>	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
<div></div>	<div></div>	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
<div></div>	<div></div>	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
<div></div>	<div></div>	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
<div></div>	<div></div>	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
<div></div>	<div></div>	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<EventsLayout />

Composición

centralLog - Events

Anonymous (Guest)

Go to

MODE
LIVE

Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
<div></div>	<div></div>	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
<div></div>	<div></div>	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
<div></div>	<div></div>	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
<div></div>	<div></div>	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
<div></div>	<div></div>	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
<div></div>	<div></div>	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
<div></div>	<div></div>	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
<div></div>	<div></div>	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
<div></div>	<div></div>	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
<div></div>	<div></div>	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
<div></div>	<div></div>	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<FilterBox />

Composición

centralLog - Events

Anonymous (Guest)

Go to

MODE LIVE

<Toolbar />

Composición

centralLog - Events

Anonymous (Guest)

Go to

MODE LIVE

Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
<div></div>	<div></div>	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
<div></div>	<div></div>	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
<div></div>	<div></div>	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
<div></div>	<div></div>	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
<div></div>	<div></div>	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
<div></div>	<div></div>	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
<div></div>	<div></div>	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
<div></div>	<div></div>	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
<div></div>	<div></div>	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
<div></div>	<div></div>	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
<div></div>	<div></div>	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
<div></div>	<div></div>	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
<div></div>	<div></div>	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
<div></div>	<div></div>	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
<div></div>	<div></div>	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<Menu visible={false} />

Composición

centralLog - Events

Anonymous (Guest)

▼

Go to

MODE LIVE

Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
✖	🔊	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
ⓘ	🔊	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
⚠	🔊	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
⚠	🔊	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
ⓘ	🔊	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
✖	🔊	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
⚠	🔊	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
✖	🔊	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
✖	🔊	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
✖	🔊	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
ⓘ	🔊	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
ⓘ	🔊	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
✖	🔊	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
⚠	🔊	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
ⓘ	🔊	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
⚠	🔊	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
⚠	🔊	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<EventList />

Composición

centralLog - Events

Anonymous (Guest)

Go to

MODE LIVE

<EventListHeader />

Composición

centralLog - Events

Anonymous (Guest)

Go to

MODE LIVE

Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
<div><div></div><div></div></div>	<div><div></div><div></div></div>	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<Event severity=“error” state=“acked” />

Composición

centralLog - Events							
▼		🔍		Go to		MODE LIVE	
Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
✖	🔊	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
ⓘ	🔊	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
⚠	🔊	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
⚠	🔊	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
ⓘ	🔊	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
✖	🔊	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
⚠	🔊	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
✖	🔊	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
✖	🔊	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
✖	🔊	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
ⓘ	🔊	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
ⓘ	🔊	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
✖	🔊	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
⚠	🔊	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
ⓘ	🔊	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
⚠	🔊	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
⚠	🔊	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<Event severity=“info” state=“acked” />

Composición

centralLog - Events							
▼		🔍				⌚ Go to	🔌 MODE LIVE
Sev	Ack	Time	Site	Module	Domain	Type	Message
04/08/2015							
✖	🔊	04/08/2015 09:53:05.000	Ryadh	FDS	BBDB5	R	test1_1
ⓘ	🔊	04/08/2015 09:53:05.000	Madrid	Fleet	BBDB5	N	test2_2
⚠	🔊	04/08/2015 09:53:04.000	Ryadh	Fleet	AB4FB	N	test1_1
⚠	🔊	04/08/2015 09:53:04.000	Madrid	Fleet	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	test1_1
✖	🔊	04/08/2015 09:53:03.000	Ryadh	Fleet	BBDB5	N	error enormous
ⓘ	🔊	04/08/2015 09:53:03.000	Ryadh	FDS	AM01M	N	test1_1
✖	🔊	04/08/2015 09:53:01.000	Ryadh	FDS	AM01M	R	test1_1
⚠	🔊	04/08/2015 09:53:01.000	Madrid	RTS2	AB5C	A	error enormous
✖	🔊	04/08/2015 09:53:00.000	Ryadh	Fleet	AM01M	R	test1_1
✖	🔊	04/08/2015 09:53:00.000	Madrid	Fleet	AM01M	A	test2_2
✖	🔊	04/08/2015 09:52:59.000	Madrid	FDS	AB4FB	A	error enormous
ⓘ	🔊	04/08/2015 09:52:59.000	Ryadh	Fleet	AM1AC	N	test1_1
ⓘ	🔊	04/08/2015 09:52:58.000	Ryadh	FDS	AB5C	A	test1_1
✖	🔊	04/08/2015 09:52:58.000	Madrid	FDS	AB4FB	M	error enormous
⚠	🔊	04/08/2015 09:52:58.000	Ryadh	Fleet	AB4FB	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	FDS	AM01M	N	error enormous
⚠	🔊	04/08/2015 09:52:56.000	Madrid	RTS2	AM1AC	N	test1_1
✖	🔊	04/08/2015 09:52:55.000	Madrid	Fleet	AB5C	N	test1_1
ⓘ	🔊	04/08/2015 09:52:54.000	Ryadh	RTS1	AB4FB	M	test1_1
⚠	🔊	04/08/2015 09:52:52.000	Ryadh	RTS2	AM1AC	R	test2_2
⚠	🔊	04/08/2015 07:53:03.000	Ryadh	RTS1	AB5C	N	test2_2

<Event severity=“warning” state=“active” />

¿Cómo hacer aplicaciones react?

2 Crear orden jerárquico

¿Cómo hacer aplicaciones react?

```
<EventsLayout>  
  <FilterBox/>  
  <Toolbar/>  
  <Menu/>  
  <EventList>  
    <EventListHeader/>  
    <Event>  
    <Event>  
    <Event>  
  </EventList>  
</EventsLayout>
```

¿Cómo hacer aplicaciones react?

3

Construye un componente estático

¿Cómo hacer aplicaciones react?

```
class Event extends Component {  
  render() {  
    const { severity, ack, description } = this.props  
    return (  
      <tr>  
        <td>{severity}</td>  
        <td>{ack}</td>  
        <td>{description}</td>  
      </tr>  
    )  
  }  
}
```

¿Cómo hacer aplicaciones react?

```
class EventList extends Component {  
  render() {  
    const events = this.props.events.map(e => <Event key={e.id} {...e}/>)  
    return (  
      <div>  
        <EventsHeader/>  
        <ul>  
          {events}  
        </ul>  
      </div>  
    )  
  }  
}
```

¿Cómo hacer aplicaciones react?

```
const events = [{  
  severity: 'WARN',  
  ack: true,  
  description: 'Problem rendering'  
}, {  
  severity: 'ERROR',  
  ack: false,  
  description: 'Ack invalid'  
}]
```

```
ReactDOM.render(<EventList events={events}/>,  
document.getElementById('container'))
```


¿Cómo hacer aplicaciones react?

4

Probar a modificar el modelo y volver a llamar a ReactDOM.render

¿Cómo hacer aplicaciones react?

Eso es one way data flow

¿Cómo hacer aplicaciones react?




- 5 Identificar la mínima representación del estado de la UI

¿Cómo hacer aplicaciones react?

- ❓ Lista de eventos
- ❓ Estado de cada evento

¿Cómo hacer aplicaciones react?

Seguir tres reglas básicas

-  Si se pasa como prop, no pertenece al estado
-  Si no se modifica, no pertenece al estado
-  Si es un valor computado, no pertenece al estado

¿Cómo hacer aplicaciones react?

 Lista de eventos

 Estado de cada evento

¿Cómo hacer aplicaciones react?

- 6 Identificar a quien corresponde cada parte del estado

¿Cómo hacer aplicaciones react?

```
<EventsLayout>  
  <FilterBox/>  
  <Toolbar/>  
  <Menu/>  
  <EventList>  
    <EventListHeader/>  
    <Event>  
    <Event>  
    <Event>  
  </EventList>  
</EventsLayout>
```


¿Cómo hacer aplicaciones react?

```
class Event extends Component {  
  constructor(props) {  
    super(props)  
    this.state = {ack: props.ack}  
  }  
  render() {  
    const { severity, description } = this.props  
    const { ack } = this.state  
    return (  
      <tr>  
        <td>{severity}</td>  
        <td>{ack}</td>  
        <td>{description}</td>  
      </tr>  
    )  
  }  
}
```

¿Cómo hacer aplicaciones react?

7 Provocar un rerender

¿Cómo hacer aplicaciones react?

```
class Event extends Component {
  constructor(props) {
    super(props)
    this.state = {ack: props.ack}
    this.handleClick = this.handleClick.bind(this)
  }
  handleClick() {
    this.setState({ack: !this.state.ack})
  }
  render() {
    const { severity, description } = this.props
    const { ack } = this.state
    return (
      <tr>
        <td>{severity}</td>
        <td onClick={this.handleClick}>{ack}</td>
        <td>{description}</td>
      </tr>
    )
  }
}
```

Ejercicio: eventos y state

 Vamos a trastear con eventos y con estado interno del componente

 Vamos a hacer un componente con un botón y un contador de clicks

Es un conjunto de patrones y buenas prácticas

Composicion frente a herencia

```
<App>  
  <Header>  
    <Navigation> ... </Navigation>  
  </Header>  
</App>
```

React

```
// app.jsx
import Header from './Header.jsx';
export default class App extends React.Component {
  render() {
    return <Header />;
  }
}
// Header.jsx
import Navigation from './Navigation.jsx';
export default class Header extends React.Component {
  render() {
    return <header><Navigation /></header>;
  }
}
// Navigation.jsx
export default class Navigation extends React.Component {
  render() {
    return (<nav> ... </nav>);
  }
}
```


React

```
// app.jsx
import Header from './Header.jsx';
export default class App extends React.Component {
  render() {
    return (
      <Header>
        <Navigation/>
      </Header>
    )
  }
}

// Header.jsx
import Navigation from './Navigation.jsx';
export default class Header extends React.Component {
  render() {
    return <header>{this.props.children}</header>;
  }
}
```

React

```
// app.jsx
import Header from './Header.jsx';
export default class App extends React.Component {
  render() {
    const title = <h1>Title</h1>
    return (
      <Header title={title}>
        <Navigation/>
      </Header>
    )
  }
}

// Header.jsx
import Navigation from './Navigation.jsx';
export default class Header extends React.Component {
  render() {
    return (
      <header>
        {this.props.title}
        {this.props.children}
      </header>;
    )
  }
}
```

Higher Order Components (HOCs)

React

```
const enhanceComponent = (Component) =>
  class Enhance extends React.Component {
    render() {
      return (
        <Component
          {...this.state}
          {...this.props}
        />
      )
    }
  };

export default enhanceComponent;
```

React

```
var OriginalComponent = () => <p>Text</p>;

class App extends React.Component {
  render() {
    return React.createElement(enhanceComponent(OriginalComponent));
  }
};
```

React

```
const omitProp = (Component, prop) => {  
  class Enhance extends React.Component {  
    render() {  
      const {prop, ...rest} = this.props  
      return (  
        <Component  
          {...this.state}  
          {...rest}  
        />  
      )  
    }  
  };  
}  
  
export default omitProp;
```

```
const Greet = (props) => <p>Hello {props.name}</p>;  
const NoNameGreet = omitProp(Greet, 'name')  
  
class App extends React.Component {  
  render() {  
    return <NoNameGreet name={'Mike'}>  
  }  
};
```

React

Dependency Injection

React

```
class Text extends React.Component {  
  render() {  
    const {text} = this.props  
    return (  
      <p>{text}</p>  
    )  
  }  
};  
  
export default Text;
```

React

```
import i18n from 'i18n'

const i18nize = (Component, prop) => {
  class Enhance extends React.Component {
    render() {
      return (
        <Component
          {...this.state}
          {...this.props}
          {...i18n}
        />
      )
    }
  };
}

export default omitProp;
```

React

```
var context = { i18n };
class App extends React.Component {
  getChildContext() {
    return context;
  }...
};
App.childContextTypes = {
  i18n: PropTypes.object
};

class I18nize extends React.Component {
  render() {
    var i18n = this.context.i18n;
    ...
  }
}
I18nize = {
  i18n: React.PropTypes.object
};
```

```
export default {  
  data: {},  
  get(key) {  
    return this.data[key];  
  },  
  register(key, value) {  
    this.data[key] = value;  
  }  
}
```

Ejercicio: wire

Como harías un utilidad wire que dado un componente y un array de valores alojados en el contexto, devuelva un componente con esas valores inyectados como props?

React

One way data flow

React

```
class Switcher extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = { flag: false };  
    this._onButtonClick = e => this.setState({ flag: !this.state.flag });  
  }  
  render() {  
    return (  
      <button onClick={ this._onButtonClick }>  
        { this.state.flag ? 'lights on' : 'lights off' }  
      </button>  
    );  
  }  
};
```

Componente con estado interno → estado desconocido por el resto

React

```
class Switcher extends React.Component {
  constructor(props) {
    super(props);
    this.state = { flag: false };
    this._onButtonClick = e => {
      this.setState({ flag: !this.state.flag })
      this.props.onChange(this.state.flag);
    }
  }
  render() {
    return (
      <button onClick={ this._onButtonClick }>
        { this.state.flag ? 'lights on' : 'lights off' }
      </button>
    );
  }
};
```

```
var Store = {
  _flag: false,
  set: function(value) {
    this._flag = value;
  },
  get: function() {
    return this._flag;
  }
};

class App extends React.Component {
  render() {
    return <Switcher onChange={
      Store.set.bind(Store)
    } />;
  }
};
```

React

Input de usuario ➡ Switcher ➡ Store

React

Introducimos sincronizacion con el servidor

```
// ... en el componente App
<Switcher
  value={ Store.get() }
  onChange={ Store.set.bind(Store) } />

// ... en el componente Switcher
constructor(props) {
  super(props);
  this.state = { flag: this.props.value };
}
```

Input de usuario



Switcher



Store



Backend

React

Oneway data flow maneja un unico estado

```
var Store = {  
  _handlers: [],  
  _flag: '',  
  onChange: function(handler) {  
    this._handlers.push(handler);  
  },  
  set: function(value) {  
    this._flag = value;  
    this._handlers.forEach(handler => handler())  
  },  
  get: function() {  
    return this._flag;  
  }  
};
```

Los componentes se renderizarán cada vez que la store modifique sus valores

React

```
class App extends React.Component {  
  constructor(props) {  
    super(props);  
    Store.onChange(this.forceUpdate.bind(this));  
  }  
  render() {  
    return (  
      <div>  
        <Switcher  
          value={ Store.get() }  
          onChange={ Store.set.bind(Store) } />  
      </div>  
    );  
  }  
};
```

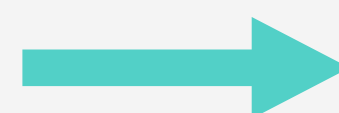
```
class Switcher extends React.Component {  
  constructor(props) {  
    super(props);  
    this._onButtonClick = e => {  
      this.props.onChange(!this.props.value);  
    }  
  }  
  render() {  
    return (  
      <button onClick={ this._onButtonClick }>  
        { this.props.value ? 'lights on' : 'lights off' }  
      </button>  
    );  
  }  
};
```

```
function Switcher(props) {  
  function handleClick() {  
    props.onChange(!props.value)  
  }  
  return (  
    <button onClick={handleClick}>  
      { props.value ? 'lights on' : 'lights off' }  
    </button>  
  );  
}
```

Input de usuario



Switcher



Store



Backend




React

Redux

React

Contenedor de estado predecible

Motivaciones

-  Tenemos que manejar mas estado que nunca
-  Con MVC perdemos el control de cuando, por qué y como hemos llegado al estado actual
-  Complejidad creciente: optimistic updates, server side rendering

Motivaciones




Material radiactivo

 Mutaciones al estado

 Asíncronia

Redux

Principios básicos

-  Single source of truth
-  Estado de solo lectura
-  Cambios a través de funciones puras

React

Patron reducer

Redux

```
var initialState = 0;

var sum = [1,2,3,4,5].reduce(add, initialState); // => 15

function add(a, b) {
  // `a` sera el estado previo
  // `b` sera el elemento actual de la lista
  return a + b;
}

// cuando llamamos a `.reduce`, `add` se llama recursivamente.
// add(0, 1)  => 1
// add(1, 2)  => 3
// add(3, 3)  => 6
// add(6, 4)  => 10
// add(10, 5) => 15
```

Redux

```
var initialState = 0;

function reducer(initialState, action) {
  switch (action.type) {
    case 'add':
      return initialState + action.payload;
  }
}
```

Redux

 Vamos a implementar redux

Ejercicio: createStore

La store es el corazon de redux

Ejercicio: createStore

La store es el corazon de redux. Se compone de:

- 1 El estado de la aplicacion
- 2 Un metodo dispatch, usado para enviar acciones a la store
- 3 Un metodo listen, usado para escuchar cambios en la store
- 4 Un reducer, una funcion pura usada para cambiar el estado de la aplicacion a partir de una accion
- 5 Un metodo getState, que devuelva el estado de la aplicacion

Ejercicio: CombineReducers

Necesitamos poder combinar los distintos reducers de nuestra aplicacion en uno unico

Ejercicio: CombineReducers

Necesitamos poder combinar los distintos reducers de nuestra aplicacion en uno unico

```
const state = {  
  session: {},  
  contador: 0  
}  
  
const appReducer = combineReducers({  
  session: sessionReducer,  
  counter: counterReducer  
})
```

Ejercicio: CombineReducers

Necesitamos poder combinar los distintos reducers de nuestra aplicacion en uno unico

```
function combineReducers(reducers) {  
  ...  
  return newReducer  
}
```

Redux: Middlewares

Los middlewares ofrecen una capa de abstraccion entre el envio de una accion y el momento en el que llega al reducer

Provee un punto de anclaje para extensiones fuera del framework

Ejercicio: CombineReducers

Vamos a crear nuestro primer middleware. Un logger de acciones

▼ ADD_TODO
❏ dispatching: <i>Object {type: "ADD_TODO", text: "Use Redux"}</i>
next state: ► <i>Object {visibilityFilter: "SHOW_ALL", todos: Array[1]}</i>
▼ ADD_TODO
❏ dispatching: <i>Object {type: "ADD_TODO", text: "Learn about middleware"}</i>
next state: ► <i>Object {visibilityFilter: "SHOW_ALL", todos: Array[2]}</i>
▼ COMPLETE_TODO
❏ dispatching: <i>Object {type: "COMPLETE_TODO", index: 0}</i>
next state: ► <i>Object {visibilityFilter: "SHOW_ALL", todos: Array[2]}</i>
▼ SET_VISIBILITY_FILTER
❏ dispatching: <i>Object {type: "SET_VISIBILITY_FILTER", filter: "SHOW_COMPLETED"}</i>
next state: ► <i>Object {visibilityFilter: "SHOW_COMPLETED", todos: Array[2]}</i>

Ejercicio: CombineReducers

Dada una accion...

```
store.dispatch(addTodo('Use Redux'))
```

Ejercicio: CombineReducers

... nuestra primera aproximacion seria hacer el log manualmente

```
let action = addTodo('Use Redux')  
  
console.log('dispatching', action)  
store.dispatch(action)  
console.log('next state', store.getState())
```

Ejercicio: CombineReducers

Podríamos encapsularlo en una funcion

```
function dispatchAndLog(store, action) {  
  console.log('dispatching', action)  
  store.dispatch(action)  
  console.log('next state', store.getState())  
}
```

Ejercicio: CombineReducers

Podríamos encapsularlo en una función...

```
function dispatchAndLog(store, action) {  
  console.log('dispatching', action)  
  store.dispatch(action)  
  console.log('next state', store.getState())  
}
```

...y usarlo en todos los sitios en vez por store.dispatch

Ejercicio: CombineReducers

Monkeypatching! Reemplazar dispatch en la instancia de store.

```
let next = store.dispatch
store.dispatch = function dispatchAndLog(action) {
  console.log('dispatching', action)
  let result = next(action)
  console.log('next state', store.getState())
  return result
}
```

Ejercicio: CombineReducers

Y si tenemos mas de un middleware?

```
dispatchAndLog()  
dispatchAndCrashReporting()
```

Ejercicio: CombineReducers

Primera aproximacion al middleware: next

```
function patchStoreToAddLogging(store) {  
  let next = store.dispatch  
  store.dispatch = function dispatchAndLog(action) {  
    console.log('dispatching', action)  
    let result = next(action)  
    console.log('next state', store.getState())  
    return result  
  }  
}
```

Ejercicio: CombineReducers

Sacamos a una funcionalidad comun del propio framework

```
function applyMiddlewareByMonkeypatching(store, middlewares) {  
  middlewares = middlewares.slice()  
  middlewares.reverse()  
  
  // Sobrescribimos dispatch en cada iteracion.  
  middlewares.forEach(middleware =>  
    store.dispatch = middleware(store)  
  )  
}}
```

```
applyMiddlewareByMonkeypatching(store, [ logger, crashReporter ])
```

Ejercicio: CombineReducers

Sacamos a una funcionalidad comun del propio framework

```
function logger(store) {  
  let next = store.dispatch  
  
  return function dispatchAndLog(action) {  
    console.log('dispatching', action)  
    let result = next(action)  
    console.log('next state', store.getState())  
    return result  
  }  
}
```

Ejercicio: CombineReducers

Eliminemos el monkeyPatching.

Ejercicio: CombineReducers

Aqui viene el salto importante.

Tenemos el middleware encadenado

Como podemos hacer para no sobrescribir dispatch

Ejercicio: CombineReducers

Aqui viene el salto importante.

Tenemos el middleware encadenado

Como podemos hacer para no sobrescribir dispatch -> pasemos next como parametro.

Ejercicio: CombineReducers

Añadimos un nivel de indireccion mas para clausurar next.

```
function logger(store) {  
  return function wrapDispatchToAddLogging(next) {  
    return function dispatchAndLog(action) {  
      console.log('dispatching', action)  
      let result = next(action)  
      console.log('next state', store.getState())  
      return result  
    }  
  }  
}
```

Ejercicio: CombineReducers

Respira...

....

A ver que tal ahora

Ejercicio: CombineReducers

Las arrow functions ayudan con la curryficacion.

```
const logger = store => next => action => {  
  console.log('dispatching', action)  
  let result = next(action)  
  console.log('next state', store.getState())  
  return result  
}
```

Ejercicio: CombineReducers

Asi es como son los middlewares de redux

```
const logger = store => next => action => {  
  console.log('dispatching', action)  
  let result = next(action)  
  console.log('next state', store.getState())  
  return result  
}
```

Ejercicio: CombineReducers

Y así como se aplican

```
import { createStore, combineReducers, applyMiddleware } from 'redux'

let reducer = combineReducers(reducers)
let store = createStore(
  reducer,
  // applyMiddleware() indica a createStore como manejar los middlewares
  applyMiddleware(logger, crashReporter)
)
```

Ejercicio: applyMiddleware

Vamos a crear esa funcion de utilidad, applyMiddleware

Patron smart and dumb components

Redux

Dumb	Smart
No conoce sobre la aplicacion	Conoce la aplicacion y el estado
No hace data fetching	Hace data fetching
Su fin es visualizacion	Su fin es la interaccion
Rendea html	No rendea html
Reusable	Normalmente no es reusable
Se comunica solo con su padre a traves de props	Pasa estado y datos a sus hijos
Maneja el estado	No maneja el estado

Ejercicio: CombineReducers

Dumb component

```
class Event extends Component {  
  render() {  
    const { severity, ack, description } = this.props  
    return (  
      <tr>  
        <td>{severity}</td>  
        <td>{ack}</td>  
        <td>{description}</td>  
      </tr>  
    )  
  }  
}
```

Ejercicio: CombineReducers

Smart component

```
class EventContainer extends Component {  
  render() {  
    return <Event {...props}/>  
  }  
}  
  
function mapStateToProps(state) {  
  return {  
    severity: state.currentEvent.severity  
    acked: state.currentEvent.severity  
    description: state.currentEvent.severity  
  }  
}  
  
const mapDispatchToProps = {  
  ack: actions.ack  
}  
  
connect(mapStateToProps, mapDispatchToProps)(EventContainer)
```

Patron observer: connect y provider

Ejercicio: CombineReducers

Poner la store disponible a todos los smart components

```
let store = createStore(reducer)

render(
  <Provider store={store}>
    <EventContainer />
  </Provider>,
  document.getElementById('root')
)
```