BackboneJS

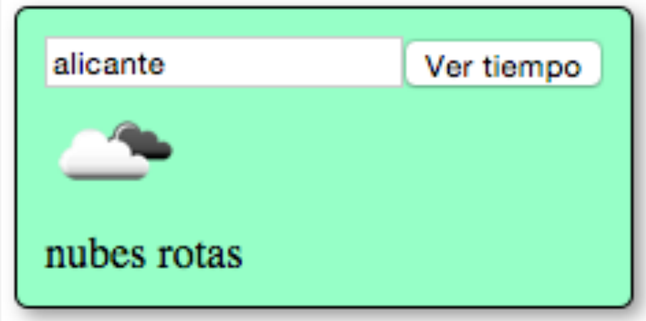
Sesión 5 - Backbone y ReactJS

Índice

El problema de las interfaces dinámicas

- Hay que estar constantemente redibujando la interfaz cuando cambian modelos/colecciones
- En Backbone `render()` por convenio **pinta todo el HTML**
- Como Backbone no tiene *data binding* tenemos que definir varios `renderXXX()` para actualizar las diversas partes de la interfaz

```
render: function() {
        this.$el.html('<input type="text" id="localidad">' +
           '<input type="button" value="Ver tiempo" id="ver_tiempo">' +
           '<div> <img id="icono" src=""></div>' +
           '<div id="descripcion"></div>');
},
renderData: function() {
        $('#icono').attr('src',this.model.get("icono_url"));
        $('#descripcion').html(this.model.get("descripcion"));
},
```

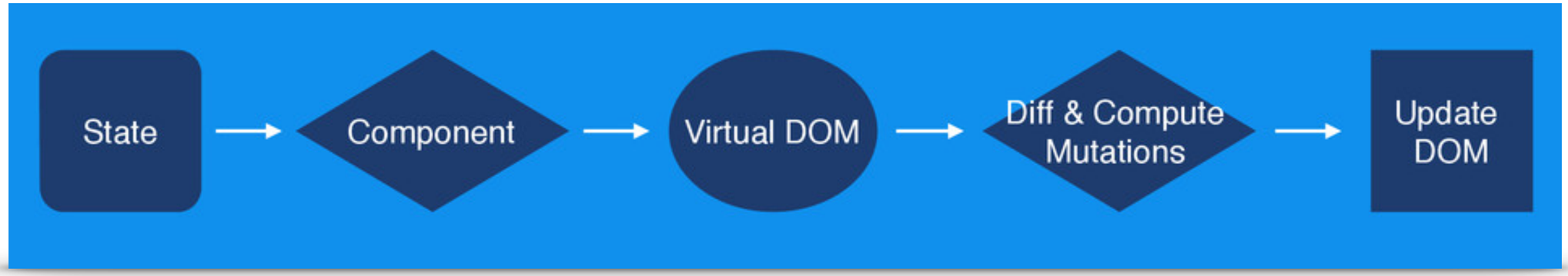


Solución radical

¿Por qué no llamar siempre a `render()` cada vez que cambie cualquier dato? así estaremos seguros de que la vista está actualizada, y el código se mantiene simple.

React

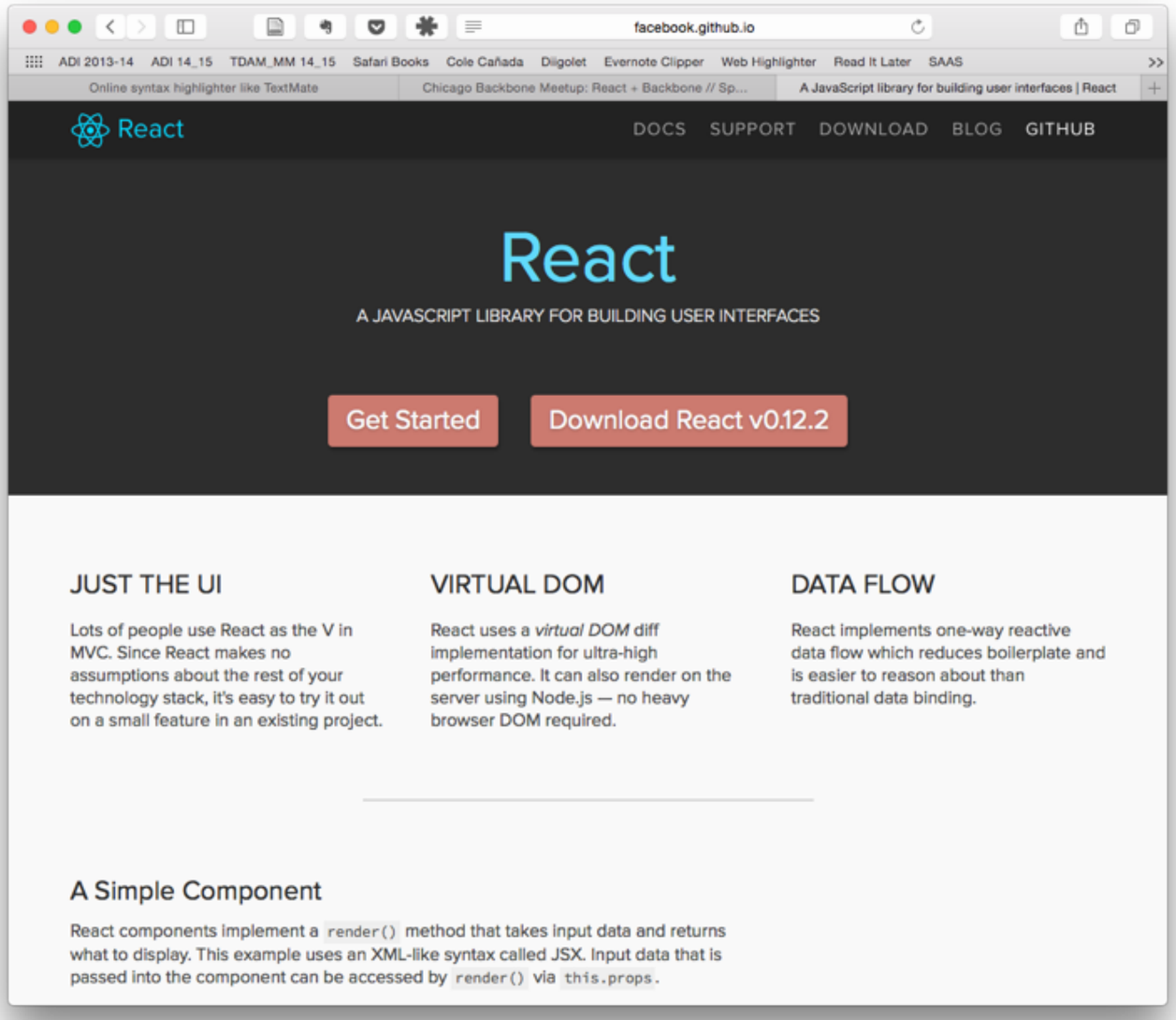
- Hace una especie de `diff` entre la nueva vista y la vista actual, y solo actualiza lo necesario
- Trabaja internamente con un “DOM virtual”, sobre el que hace las operaciones, y solo al final actualiza el DOM real



¿De dónde sale React?

- Desarrollado en Facebook
- Usado en producción en multitud de sitios "grandes": Facebook, Instagram, AirBnb, Khan Academy,…

http://facebook.github.io/react/



React & Backbone

- React **no es un framework MVC**, solo sirve para hacer interfaces
- Podemos sustituir las vistas de Backbone por componentes de React, y seguir usando modelos y colecciones de Backbone

Vista React vs. Vista Backbone

```
var Libro = React.createClass({
    render: function() {
        return (
          <div className="libro">
              <b>{this.props.children}</b>, por <em>{this.props.autor}</em>
          </div>
        );
    }
});

React.render(
        <Libro autor="George R.R. Martin">Tormenta de espadas</Libro>,
        document.getElementById('example')
);
```

```
var LibroView = Backbone.View.extend({
    tagName: 'div', className: 'libro'
    render: function() {
        this.el.innerHTML = '<b>' + this.model.get("titulo")
                          + '</b>, por <em>’ + this.model.get("autor") + '</em>'
        return this
    }
});
var miLibroView = new LibroView({model: libro1});
```

Índice

Componentes React

- Encapsulan el **lenguaje de marcado** junto con la lógica de presentación y el **manejo de eventos**

```
<script src="react.js"></script>
```

```
<div id="componente"></div>
<script>
  var comp = React.createElement('h1', {id:"saludo"}, '¡Hola React!');
  React.render(comp, document.getElementById('componente'));
</script>
```

Sintaxis JSX

- La anterior sintaxis se va haciendo más incómoda conforme se complica el HTML
- **JSX**: mezclar HTML (en realidad XML) con JS
  - **No es** poner fragmentos de HTML entre comillas
  - El HTML está **directamente** insertado en el JS.
- JSX no es JS, por tanto requiere de un `type` distinto y hay que compilarlo

<script src="JSXTransformer.js"></script>

```
<script src="JSXTransformer.js"></script>
```

```
<div id="componente"></div>
<script type="text/jsx">
    var saludo = "Hola";
    React.render(<h1 id="saludo">¡{saludo} React!</h1>,
                 document.getElementById('componente'));
</script>
```

Clase componente

- Normalmente **crearemos una clase** para encapsular el componente

```
<script type="text/jsx">
var Libro = React.createClass({
    render: function() {
        return (
          <div className="libro">
              <b>{this.props.children}</b>, por <em>{this.props.autor}</em>
          </div>
        );
    }
});

React.render(
        <Libro autor="George R.R. Martin">Tormenta de espadas</Libro>,
        document.getElementById('example')
);
</script>
```

Redibujado eficiente de componentes

- Podemos forzar el redibujado de varias formas. Se hará de forma eficiente automáticamente
  - cambiando props
  - llamando otra vez a Render, en el mismo punto del DOM
  - llamando a forceUpdate()

```
var libro = React.render(
    <Libro autor="George R.R. Martin">Tormenta de espadas</Libro>,
    document.getElementById('example')
);
setTimeout(function() {
    React.addons.Perf.start();
    libro.setProps({children:"Danza de dragones"});
    React.addons.Perf.stop();
    React.addons.Perf.printDOM();
}, 1000);
```

(index)	data-reactid	type	args
0	".2.0"	"set textContent"	"{"textContent":"Danza de dragones"…

Índice

Composición de componentes

- Normalmente tendremos **componentes dentro de componentes**

```
var Portada = React.createClass({
    render: function() {
        return (<img src={this.props.url}/>);
    }
});

var Libro = React.createClass({
    render: function() {
        return (
          <div className="libro">
              <Portada url={this.props.portada}/> <br/>
              <b>{this.props.children}</b>, por <em>{this.props.autor}</em>
          </div>
        );
    }
});
```

Composición de componentes (II)

- Nos faltan los **datos** y ejecutar el **render**

```
var datosLibro = {
    titulo: "Juego de tronos",
    autor: "George R.R. Martin",
    portada:”http://img1.wikia.nocookie.net/…”
};

var libro = React.render(
  <Libro autor={datosLibro.autor} portada={datosLibro.portada}>
   {datosLibro.titulo}</Libro>,
  document.getElementById('libro')
);
```

Encapsular una lista de componentes

- El **render del padre debe ir creando los hijos de manera dinámica**

```
var ListaLibros = React.createClass({
        render: function() {
            var libros = this.props.data.map(function(libro) {
               return (
                  <Libro autor={libro.autor}>
                     {libro.titulo}
                  </Libro>
               );
            });

            return (
                <div className="listaLibros">
                   {libros}
                </div>
            );
        }
    });
```

Encapsular una lista de componentes (II)

```
var Libro = React.createClass({
        render: function() {
            return (
              <div className="libro">
                  <b>{this.props.children}</b>, por <em>{this.props.autor}</em>
              </div>
            );
        }
    });

    React.render(
            <ListaLibros data={datos}></ListaLibros>,
            document.getElementById('ejemplo')
    );
```

Índice

Estado e interactividad

```
var LikeButton = React.createClass({
      getInitialState: function() {
        return {meGusta: false};
      },
      handleClick: function(event) {
        this.setState({meGusta: !this.state.meGusta});
      },
      render: function() {
        var text = this.state.meGusta ? '¡Me gusta!' : 'No me gusta';
        return (
          <button onClick={this.handleClick}>
            {text}
          </button>
        );
      }
});

React.render(
      <LikeButton />,
      document.getElementById('ejemplo')
);
```

Referenciar los componentes desde código

- Atributo **ref**

```
var Formulario = React.createClass({
       verValor : function() {
        console.log(document.getElementById('nombre').value);
        console.log(this.refs.nombre.getDOMNode().value);
       },

       render: function() {
          return (
           <form>
             <label htmlFor="nombre">Login:</label><input type="text" id="nombre" ref="nombre"/> <br/>
             <input type="button" value="Ver valor" onClick={this.verValor}/>
           </form>
          );
       }
    });

React.render(<Formulario/>, document.getElementById('ejemplo'));
```

Índice

React & Backbone

- React implementa una forma de **mixins**, que nos permiten compartir código Javascript entre múltiples componentes, sin tener que repetirlo.
- Hay implementaciones de terceros de *mixins* para **combinar Backbone y React**.
  - Tener un componente React asociado a 1 o varios modelos/colecciones
  - Vamos a usar aquí una llamada backbone-react-component.
- El mixin sirve de "pegamento" entre componentes React y modelos y/o colecciones de Backbone.
  - Si tenemos un componente React asociado a una colección y esta cambia, el mixin disparará el re-renderizado.

Componente con un modelo asociado

- Atributos del modelo accesibles como propiedades de `state`

```
<script type="text/jsx">
    var LibroComp = React.createClass({
        mixins: [Backbone.React.Component.mixin],
        render: function() {
          return (
            <div class="libro">
              <b>{this.state.titulo}</b>, por <em>{this.state.autor}</em>
            </div>
          );
        }
    });
    var libro1 = new LibroModel({titulo:"Crónicas marcianas", autor: "Ray Bradbury"});
    React.render(<LibroComp model={libro1}></LibroComp>,
                 document.getElementById('un_libro'));
</script>
```

Componente con un modelo asociado (II)

- El modelo también es accesible con `getModel()`

```
...
render: function() {
  var m = this.getModel();
  return (
    <div class="libro">
      <b>{m.get('titulo')}</b>, por <em>{m.get('autor')}</em>
    </div>
  );
}
...
```

Componente con una colección asociada (I)

- Definición de colección y modelos que contiene

```
<script type="text/javascript">
    var LibroModel = Backbone.Model.extend({});
    var Biblioteca = Backbone.Collection.extend({
        model: LibroModel
    });
    var miBiblio = new Biblioteca([
        new LibroModel({titulo: "Juego de tronos", autor: "George R.R. Martin"}),
        new LibroModel({titulo: "El mundo del río", autor: "Philip J. Farmer"})
    ]);
</script>
```

Componente con una colección asociada (II)

- Componente "**padre**"

```
<script type="text/jsx">
    var ListaLibros = React.createClass({
        mixins: [Backbone.React.Component.mixin],
        render: function() {
            var libros = this.getCollection().map(function(libro) {
                return (
                    <Libro key={libro.cid} model={libro}/>
                );
            });

            return (
              <div className="listaLibros">
                 {libros}
              </div>
            );
        }
```

Componente con una colección asociada (II)

- Componente "**hijo**"

```
var Libro = React.createClass({
    mixins: [Backbone.React.Component.mixin],
    render: function() {
            return (
              <div className="libro">
                  <b>{this.state.autor}</b>, por <em>{this.state.titulo}</em>
              </div>
            );
    }
 });

React.render(
    <ListaLibros collection={miBiblio}></ListaLibros>,
    document.getElementById('example')
);
```

¿Preguntas?