Introducción a React y Redux

Workshop v1.0

MEDALLIA

Quienes somos?

Pablo Carriqueo

- Senior Software Engineer @ Medallia
- Past: Globant (EA), startups
- FIUBA

Sebastian Torres

- Software Engineer II @ Medallia
- Electronica ⇒ Sistemas
- UTN

Qué hacemos?

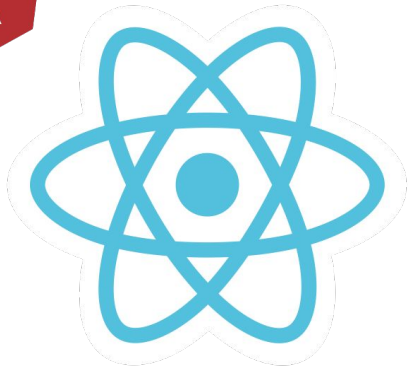
Much frontend.

Frontend applications

- Angular
- React

To support our product

- Distributed system
- Large scale
- Non-trivial solutions



Qué hacemos?

Foco en:

- Make things right
- Long term support
- Collaboration
- Best practices

De que se trata este workshop?

Hands-on experience

- Code
- Try new stuff
- Have fun!
- Next steps

AGENDA

AGENDA

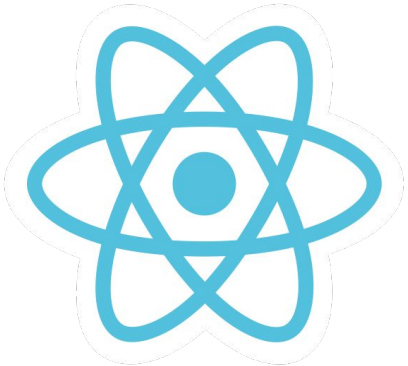
1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

AGENDA

1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

Que es React?

- Libreria
- UI basada en componentes
- Declarar vistas para cada estado de la aplicación
- Renderizar cambios
- No tiene opiniones respecto al modelo (sort of)



Fundamentals

...of a React Application

- Virtual DOM
- Componentes
  - State
  - Props
  - Render
    - ✓ ...otros componentes
- View: JS ó JSX

Component

Elementos clave

- **this.props**
  Parámetros para el componente

- **this.render()**
  Dibuja la vista. Se llama cuando cambia el estado.

- **this.state**
  Estado del componente. Cambiar con **this.setState(newState)**

WORK

Antes de empezar...

Asegúrense de tener

- Node, npm
- Git
- Repo del workshop

Tools

```
sudo apt-get install nodejs npm git
```

Repo

```
git clone https://github.com/storres93/UTN_Workshop.git
```

1.1. Nuestro primer componente

Hola mundo!

- HTML con un script
- Creamos un componente por código
- ES5: Javascript que entiende el browser

Recap

Qué aprendimos so far?

- “Hola mundo” en React
- Que necesita un componente?
- Solo vista, nada de modelo
- Si queremos una full app, tenemos que armar un proyecto

AGENDA

1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

Node.js

Javascript Runtime

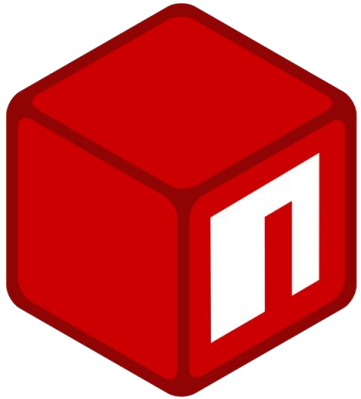
```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

- Javascript Runtime
- Async, Event driven
- Network applications
- JS fuera del browser
- https://www.nodejs.org

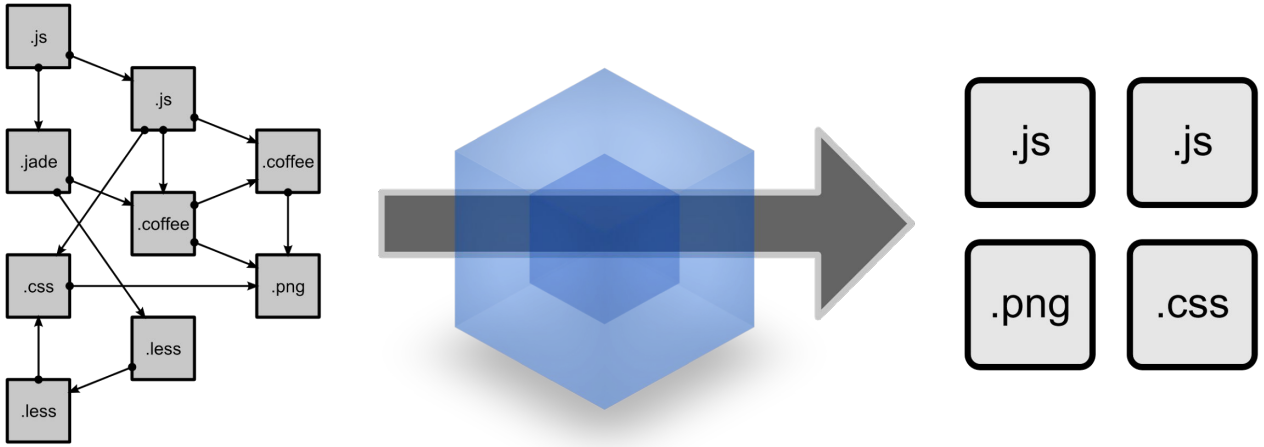
NPM

Node Package Manager

- Package Manager
- Already included in Node.js
- Software Registry (~.5M packages)
- Declarar e instalar dependencias
- Estándar de trabajo para JS
- https://www.npmjs.com



Webpack



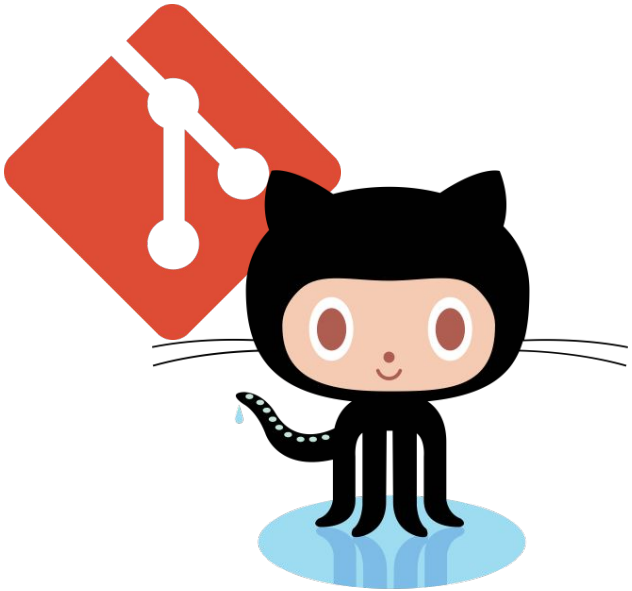
GIT

Because of reasons.

Características:

- Control de versiones
- Distribuido
- Rapido
- Eficiente

Vamos a usar GitHub.



Text editor

Choose your flavor

- Atom
- Sublime
- Webstorm o Idea
- VS Code
- Vim
- Notepad (?)
- ...pretty much anything.

Cómo arrancamos un proyecto?

(generalizando, claro)

Desde un repo

1. git clone [dir del repo]
2. npm install
3. npm start

Desde cero

1. git init
2. npm init
3. npm install --save [deps]
4. *<Crear index.js>*
5. npm start

WORK

Ejercicio 1.2

1. Agregar **webpack** como dependencia
2. Script para hacer build

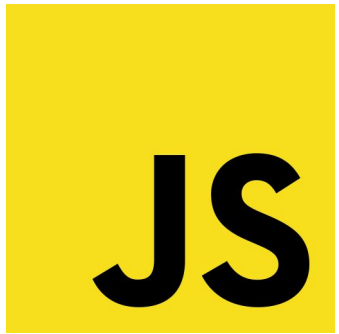
Ejercicio 1.3

1. Agregar como dependencias **react**, **react-dom**
2. Remover archivos traídos del cdn

ES6

Also called ECMAScript 2015

- Estandar desde 2015
- Adopción en browsers no es tan rápida como quisiéramos
- Solución? Transpile to ES5 (Babel)



ES6

Algunos elementos que vamos a usar

- **Arrow functions**
  Shorter syntax, **this** bound to object, not function.

- **const, let**
  Read-only references (or not). Replaces **var**.

- **class**
  Define members of an object (prototype). Keywords: **static**, **constructor()**, **extends**, **super**.

Babel

The future of javascript, today!

- Javascript transpiler
- Features agrupadas en módulos
- Permite usar features no implementadas en browsers
- https://babeljs.io/

WORK

Ejercicio 1.4

1. Agregar como dependencias: **babel**, **babel-core**, **babel-preset-es2015**
2. Crear **webpack.config.js**, agregar Babel al toolchain
3. Mover el componente de Hello World a su propio archivo, e importarlo.

Ejercicio 1.5

1. Setup Hot Module Replacement
2. Setup **.babelrc** para aceptar la sintaxis de React, y HMR
3. Script para watch (usar **webpack-dev-server**)

JSX

...o por qué tenés un HTML en tu JS

- Lenguaje para declarar interfases, como XML
- Se compila a JS
- Statically typed
- Object oriented
- React permite su uso dentro de archivos JS
  - Todos los tags html tienen su equivalente en JSX
  - En render() podes devolver un JSX

JSX

...o por qué tenés un HTML en tu JS

Ejemplo:

```
function Item(props) {
  return <li>{props.message}</li>;
}

function TodoList() {
  const todos = ['finish doc', 'submit pr', 'nag dan to review'];
  return (
    <ul>
      {todos.map((message) => <Item key={message} message={message} />)}
    </ul>
  );
}
```

Ejercicio 1.6

1. Pasar parte visual de nuestro componente a JSX
2. Agregar clickHandler, disparar un alert

1.7. To Do List

Agregamos un ToDoCreator, que tenga:

- Un textinput para escribir el .text de un nuevo todo
- Un boton, para agregar el todo a la lista

Agregamos un componente ToDoContainer

- Que tenga el array de todos,
- Que contenga al ToDoCreator, y ToDoList; y les pase las propiedades que necesiten.

1.7. To Do List

- Crear componente TodoList
  - Que tenga una propiedad **todos**, que sea un array
  - Que haga render del array de **todos**
  - Para cada item, hacer render de una propiedad (por ejemplo: .text). Se puede usar un
- Para cada ítem, crear un componente **ToDoItem**, que muestre:
  - El texto del todo (.text)
  - Si fue completado o no (.checked)

1.8. Managing State

Estado para el input: que limpie el input cuando creas un todo nuevo
Estado para el container: setear el array de todos por estado

- Usar this.setState va a forzar un render, que antes haciamos con this.forceUpdate (dado que siempre usabamos el mismo array)

Sass

CSS Preprocessor

- Otras opciones: Less, PostCSS
- Facilitar la forma de escribir estilos
  - Variables (para colores, tamaños)
  - Declarar estilos como una jerarquía de clases
  - Mixins

1.9. Adding styles

- Agregar sass al toolchain: **css-loader**, **node-sass**, **style-loader**, **sass-loader**
- Escribir estilos (go crazy)
- Algunos ejemplos:
  - Centrar el todo
  - Estilar el botón para agregar todos
  - Tachar un todo ya realizado

AGENDA

1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

Testing

...o cómo sabemos que lo que hicimos está bien.

- Unit tests
- Functional tests
  - Selenium (and the likes)
  - Shallow rendering
  - Where is the industry headed
- Integration tests

1.10. Agregar unit tests

1. Agregar mocha y enzyme a nuestro toolchain
2. Agregar script en npm para correr tests
3. Agregar tests A TODO. Go crazy.

4. Recap: que testeamos?

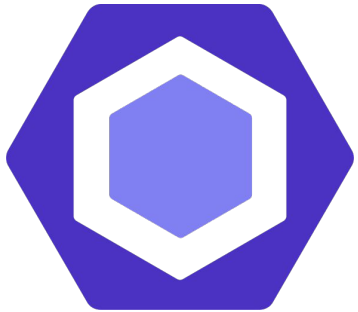
Code coverage

- After recap: cuanto les quedó sin testear?
- Hablemos de Istambul
- Git hooks para comprobar antes de hacer merge
- Cuanto seria un buen %?

ESLint

Linting

- Antes JSHint
- ESLint (as current standard)
  - Extensible
  - Configurable
  - ...podés usar reglas de otras personas
- Git hooks



1.11. Agregar linting

- Agregar ESLint al proyecto
- Agregar un script para correrlo con npm
- Si queres, agregar un hook.
- Si queres, usa la config de ESLint de un lugar que te guste (por ejemplo: airbnb)

Questions?

Special announcement

AGENDA

1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

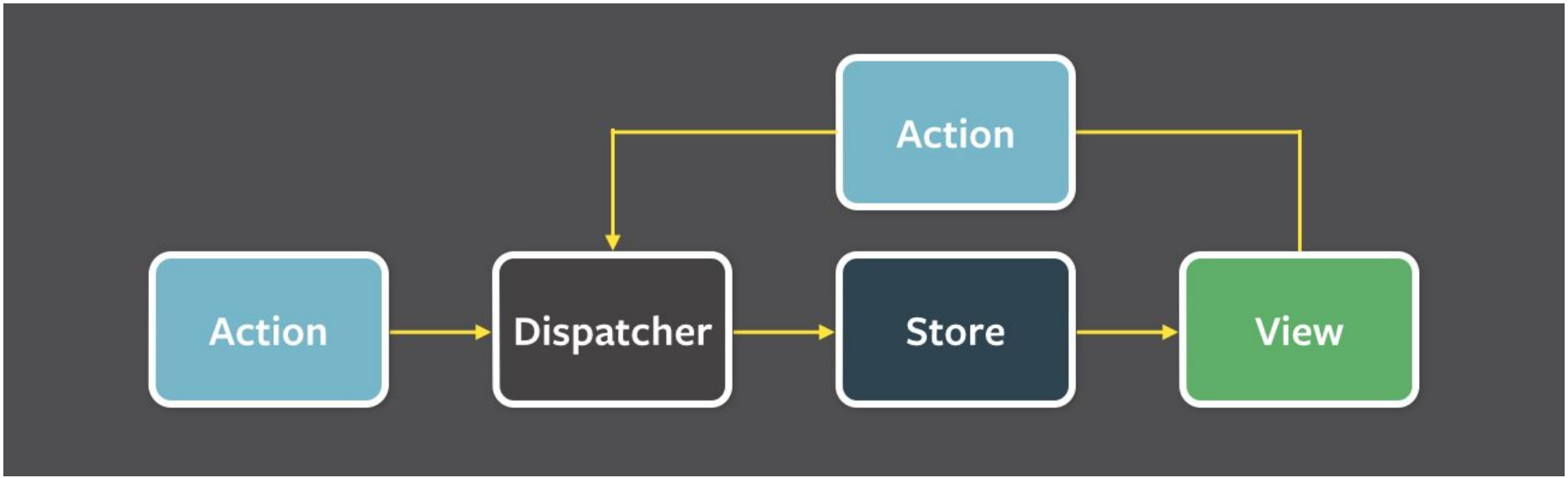
Managing State

We have a view, now what?

- Problema común dentro del desarrollo de aplicaciones
- React no es opinionado...
- ...pero Facebook si.

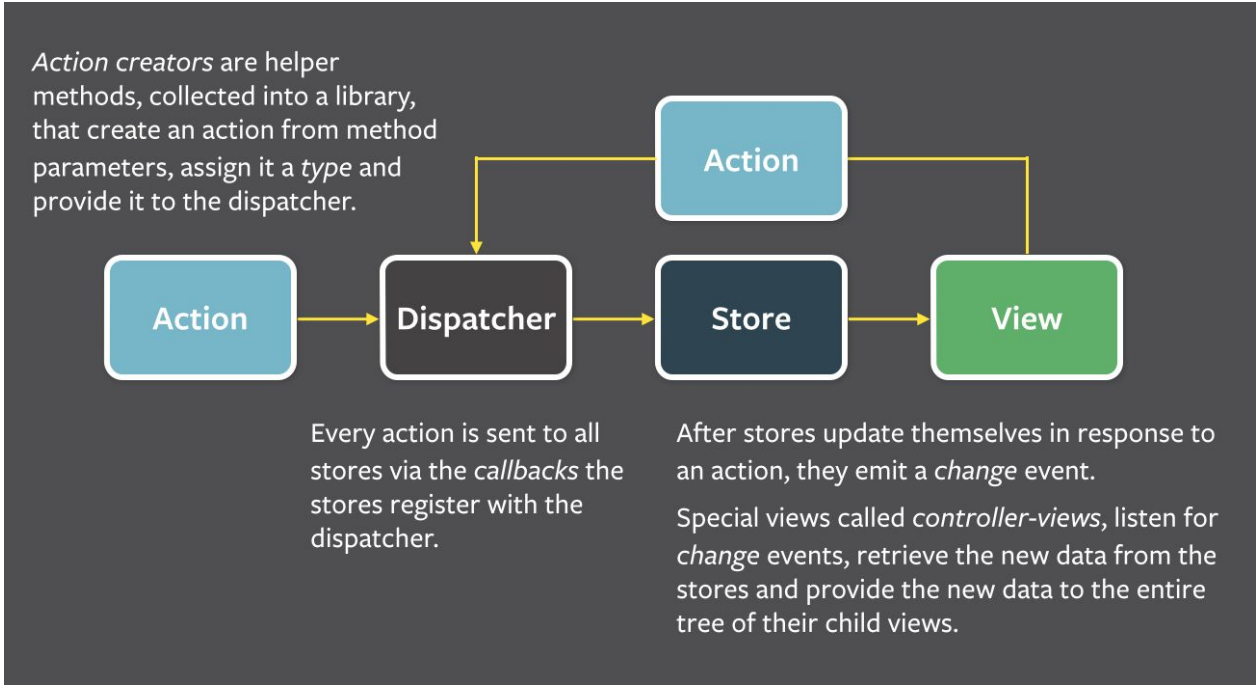
Flux

Patrón de arquitectura para aplicaciones



Flux

...explained.



Redux

Managing state in our application

- Inspired by Flux
- Single store
- Single state (and read only)
- Funciona con cualquier librería (no solo React)

Redux

Three principles

- **Single source of truth**
  Single state, stored in a single store
- **Read-only state**
  State only changes by emitting actions
- **Changes made with pure functions**
  Called reducers

Redux

Perfect companion for React

- Promueve el uso de **Visual** and **Container** components
  - Visual: just render, no logic
  - Container: fire actions, map data

- Lógica ya resuelta por react-redux
  - Cambios predecibles en el state
  - Containers con codigo simple (declaraciones de mapeos)

Redux

How to

Para instalar

```
npm install --save redux react-redux
npm install --save-dev redux-devtools
```

Luego, instalar el plugin **Redux Devtools** para su browser.

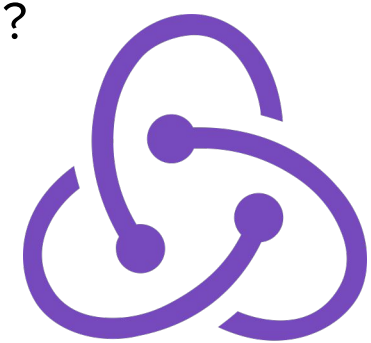
Redux

How to

1. **Modelar el estado de la aplicación**
   Tenemos toda la información que necesitamos?

2. **Modelar acciones**
   Incluimos toda la información necesaria en el payload?

3. **Reducers**
   Cómo impactan esas acciones en el estado?



WORK

Ejercicios

- Modelar el estado de nuestra aplicación
- Acciones
- Reducers

Async Redux

Because what about servers?

- Calling an API has two crucial moments
  - When the request is made
  - When the request returns (success or fail)

- Dispatch actions to reflect these:

```
{ type: 'FETCH_POSTS_REQUEST' }
{ type: 'FETCH_POSTS_FAILURE', error: 'Oops' }
{ type: 'FETCH_POSTS_SUCCESS', response: { ... } }
```

Async Redux

Using redux-thunk

Async action creators

- Return a function instead of a plain object
- When an action is a function, it gets handled by redux-thunk
- The function doesn’t have to be pure

Async Redux

Ejemplo de async action creator

```
export function fetchPosts(subreddit) {

  return function(dispatch) {
    dispatch(requestPosts(subreddit))

    return fetch(`https://www.reddit.com/r/${subreddit}.json`)
      .then(
        response => response.json(),
        error => console.log('An error occured.', error)
      )
      .then(json =>
        dispatch(receivePosts(subreddit, json))
      )
  }
}
```

WORK

Ejercicio

- Crear acciones adicionales
- Crear reducers para las respuestas de los requests
- Async action creators

Probablemente necesitemos...

- Un server mock para pegarle
- Algo para hacer fetch

AGENDA

1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

Mejores prácticas

Sobre componentes en React

- Go stateless whenever you can
- Use visual and container components
  - Visual should render its props
  - Components should wire properties, and dispatch actions
- Bind this (either in constructor or by using arrow functions)
- Avoid passing new closures to subcomponents
- Use propTypes, and defaultProps for non-required props

Mejores prácticas

Sobre state y data flow

- Use Redux.
- ...or another Flux implementation.
- Keep your state as flat as possible
  - Easier to split into reducers later
  - Use normalizr if your APIs have deeply nested objects
  - Use Immutable.js if you want to enforce a read-only state
- In general, your state is composed of: UI, and data model.
  - ...so plan your reducers accordingly.

Mejores prácticas

Sobre folder structure

- Structure should reproduce your app
  - ...so, don't use a single components folder.
  - You can nest components folders
- Keep API calls separate
  - ...in a single place, if possible
- Keep working on your structure!
  - Don't settle for a subpar solution.

Mejores prácticas

Que tan lejos estamos de un dia normal @ Medallia?

Tech

- Lint everywhere
  - Enforce standards through rules
- Extensive testing
  - Unit (tons of)
  - Functional (many)
  - Integration (some)
- Automate everything!

Mejores prácticas

Que tan lejos estamos de un dia normal @ Medallia?

Process

- Specs
- Planning
- Code reviews
- Metodologia (Agile!)
- Act on feedback

AGENDA

1. Primeros pasos
2. Trabajando con un proyecto
3. Testing goodies
4. Redux
5. Mejores prácticas
6. Q&A

Questions?

Pasantías

Thank you!

The Graveyard.

A donde van a morir las slides que no usamos.

MEDALLIA

Routing

...if you're reading this, we have time

- Util para Single Page Applications
- Separar nuestra aplicación en pseudo-páginas (sigue siendo SPA)
- Tomar variables de la ruta
- React-router, pero hay otras opciones