ReactJS: Haciendo más fácil tu desarrollo Web

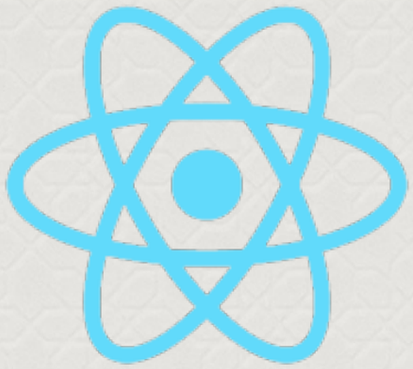
Pontificia Universidad Católica Madre y Maestra

Barcamp Santiago 2016, 19 de Noviembre del 2016

Ing. Jorge Núñez Siri

Ing. Jorge Núñez Siri

- Graduado de Ingeniería de Sistemas y Computación, honor Summa Cum Laude, en PUCMM, Santiago.
- Participación en Intercambio Académico en University of New Brunswick, Canadá, obteniendo conocimientos en áreas de Artificial Intelligence, Advance Networking, Systems Administration y Project Management.
- Participación en las competencias de la ACM International Collegiate Programming Contest (ACM-ICPC), en lenguajes como C++ y Java.
- Actualmente, soy Web Developer en Intellisys D. Corp., para Condé Nast, desarrollando en tecnologías como ReactJS y NodeJS.



React

- Es una librería de JavaScript declarativa, eficiente y flexible, que sirve para crear interfaces de usuarios.
- Es la vista de la arquitectura Modelo-Vista-Controlador.
- Creada por Facebook e Instagram en Marzo del 2013.
- Quinto repositorio más destacado en Github.

¿Quiénes usan React?

codecademy

NETFLIX

UBER

Atlassian

PayPal

DOM => Document Object Model

- DOM es la representación en memoria de cada elemento que contiene toda página HTML.

- El DOM provee una interfaz (API) para recorrer y modificar sus nodos de forma fácil:

```
var item = document.getElementById("li-example");
item.parentNode.removeChild(item);
```

- En este ejemplo "document" es la abstracción del nodo padre (root), "getElementById" y "removeChild" son dos de los métodos del API del HTML DOM.

Problemas del HTML DOM

- Lentitud a la hora de recorrer nodos de gran profundidad.
- Difícil de mantener provocando grandes riesgos de errores.
- Recorrido de forma imperativa.
- Ineficiente, lo que implica el tener que buscar exactamente los nodos que queremos cambiar.

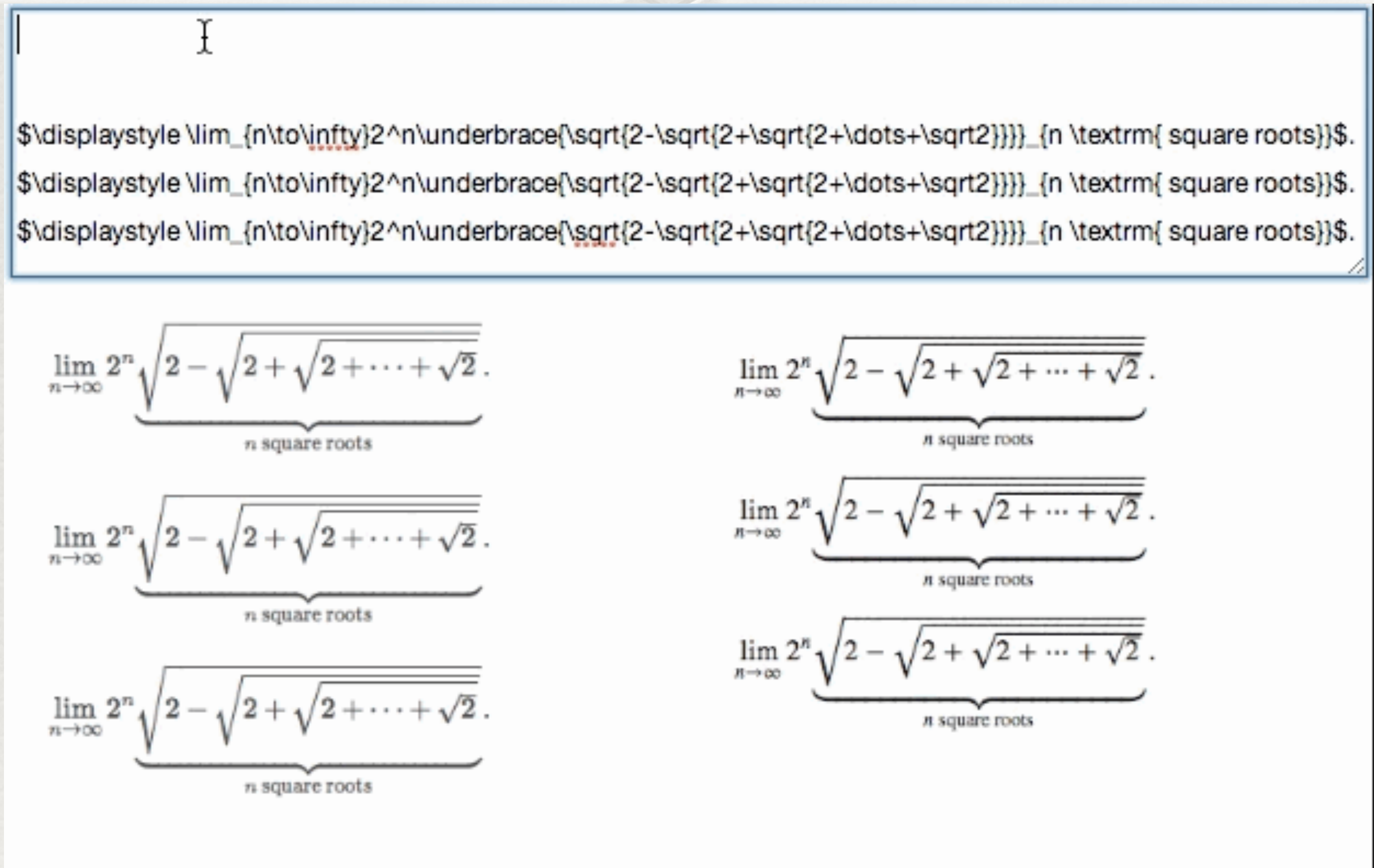
¿Por qué React?

KEEP
CALM
AND
REACT

Virtual DOM

- Es la copia simplificada de React del HTML DOM.
- Permite que React haga sus computaciones, sin interferir con las operaciones propias del DOM.
- Es creado como una estructura local de nodos.
- Cada vez que se quiere renderizar, React calcula las diferencias entre el DOM actual y el Virtual DOM, actualizando de manera eficiente el HTML DOM del explorador.

Virtual DOM



Khan Academy. Izquierda ReactJS. Derecha Backbone views

React Element

- Es una representación virtual, ligera, inmutable, y sin estado de un elemento del HTML DOM.
- Cualquier elemento HTML puede ser un ReactElement:

```
var root = React.createElement('div');
```

- JSX convierte las etiquetas HTML en ReactElements:

```
var root = <div />;
ReactDOM.render(root, document.getElementById('example'));
```

JSX => JavaScript XML

Es una extensión de JavaScript, que permite utilizar la sintaxis XML.

```
var Hello = React.createClass({
   render() {
     return <div>Hello {this.props.name}</div>;
   }
 });

ReactDOM.render(<Hello name="Jane" />, mountNode);
```

Desde JSX a JS

```
var Hello = React.createClass({
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name
    );
  }
});

ReactDOM.render(React.createElement(
                  Hello, { name: "Jane" }), mountNode);
```

Componentes

- Permiten dividir la interfaz de usuario en partes separadas, independientes y reusables.

- Mantienen el DOM en sincronización con la data.

- Pueden contener estados que definen como se ve la data de un componente a través del tiempo. Son mutables y a la hora de actualizarlos React se encarga de actualizar el Virtual DOM.

Tipos de componentes

Con estado:

```
var Example = React.createClass({
  getInitialState: function () {
    return {
      count: 0
    };
  },
  getDefaultProps: function () {
    return {
      name: 'example'
    };
  },
  render: function () {
    return (
      <h1>Hello World, {this.props.name}!</h1>
      <h3>{this.state.count}</h3>
    );
  }
});
```

Sin estado:

```
var Example = function (props) {
  return (
    <h1>Hello World, {props.name}!</h1>
  );
};

Example.defaultProps = {
  name: 'example'
};
```

Propiedades

- Son la forma inmutable de React de pasar data desde un componente padre a un componente hijo.

- Buscan crear comunicación entre los componentes.

- Existen los tipos de propiedades (propTypes), que permiten evitar los errores, y sirven como documentación de cada componente.

```
ListOfNumbers.propTypes = {
  className: React.PropTypes.string.isRequired,
  numbers: React.PropTypes.arrayOf(React.PropTypes.number)
};
```

Ciclo de vida de componentes

Inicial =>

- ✔ Initial
- ✔ GetDefaultProps
- ✔ GetInitialState
- ✔ ComponentWillMount
- ✔ Render
- ✔ ComponentDidMount

Actualización =>

- ✔ Updating State
- ✔ ShouldComponentUpdate
- ✔ ComponentWillUpdate
- ✔ Render
- ✔ ComponentDidUpdate

Resumen de características

- **Simple**: ¿Cómo se ven los elementos? en lugar de ¿cómo son creados los elementos?

- **Declarativo**: React sabe aquellas cosas que actualizan otras gracias al Virtual DOM, de manera que los cambios se propagan automáticamente, lo que hace el código más predecible y fácil de mantener.

- **Basado en componentes**: Todo en React es un componente. La interfaz de usuario es una encapsulación de varios componentes.

Ejemplo: Todo App @jnunez17

Aplicación de To-do que permite agregar y eliminar elementos únicos de una lista, utilizando "**create-react-app**".

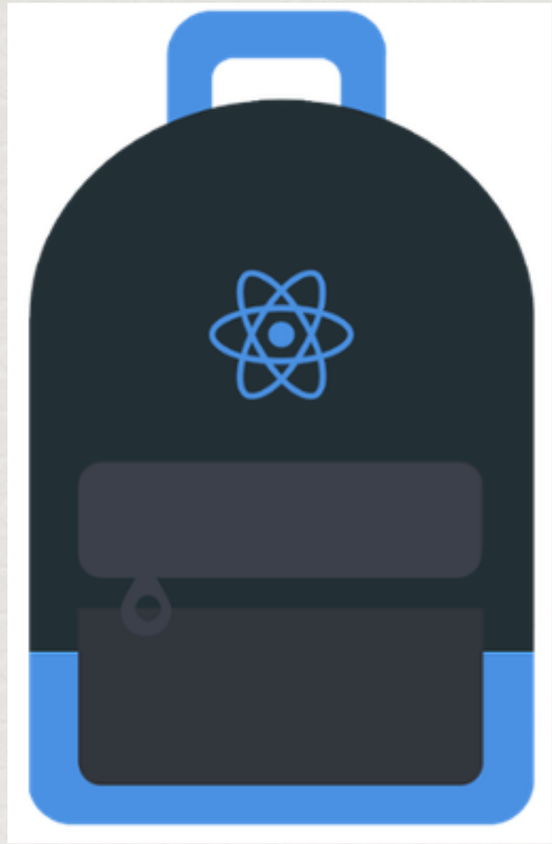
- Uso de componentes (con y sin estado), y sus propiedades.
- Enfoque de componentes contenedores y de presentación, que ofrece mejor separación de la aplicación.

```
- TODO APP
  - TODO TITLE
  - TODO LIST
      - TODO LIST ITEM #1
      - TODO LIST ITEM #2
        ...
      - TODO LIST ITEM #N
  - TODO FORM
```

Documentación

- **Documentación oficial**:
  https://facebook.github.io/react/

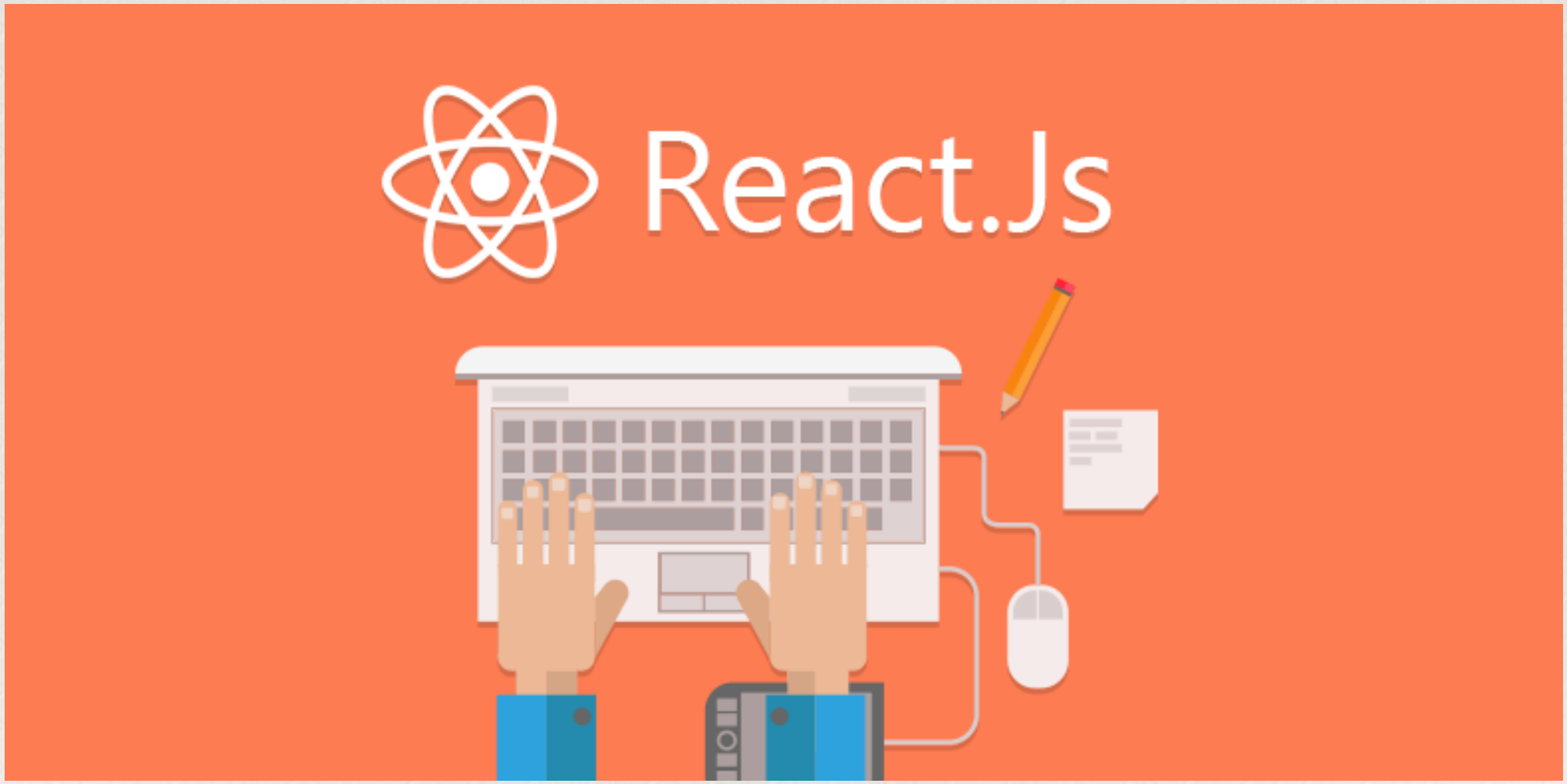
- **React fundamentals**:
  https://online.reacttraining.com/p/reactjsfundamentals



Extra: Para los curiosos...

- Aplicaciones isomórficas
- Webpack
- Flux / Redux
- React Router
- React Native

¡Muchas gracias! ¿Preguntas?



Ing. Jorge Núñez Siri

@jnunez17

@jorgenunezsiri

"Websites should look good from the inside and out."

–Paul Cookson