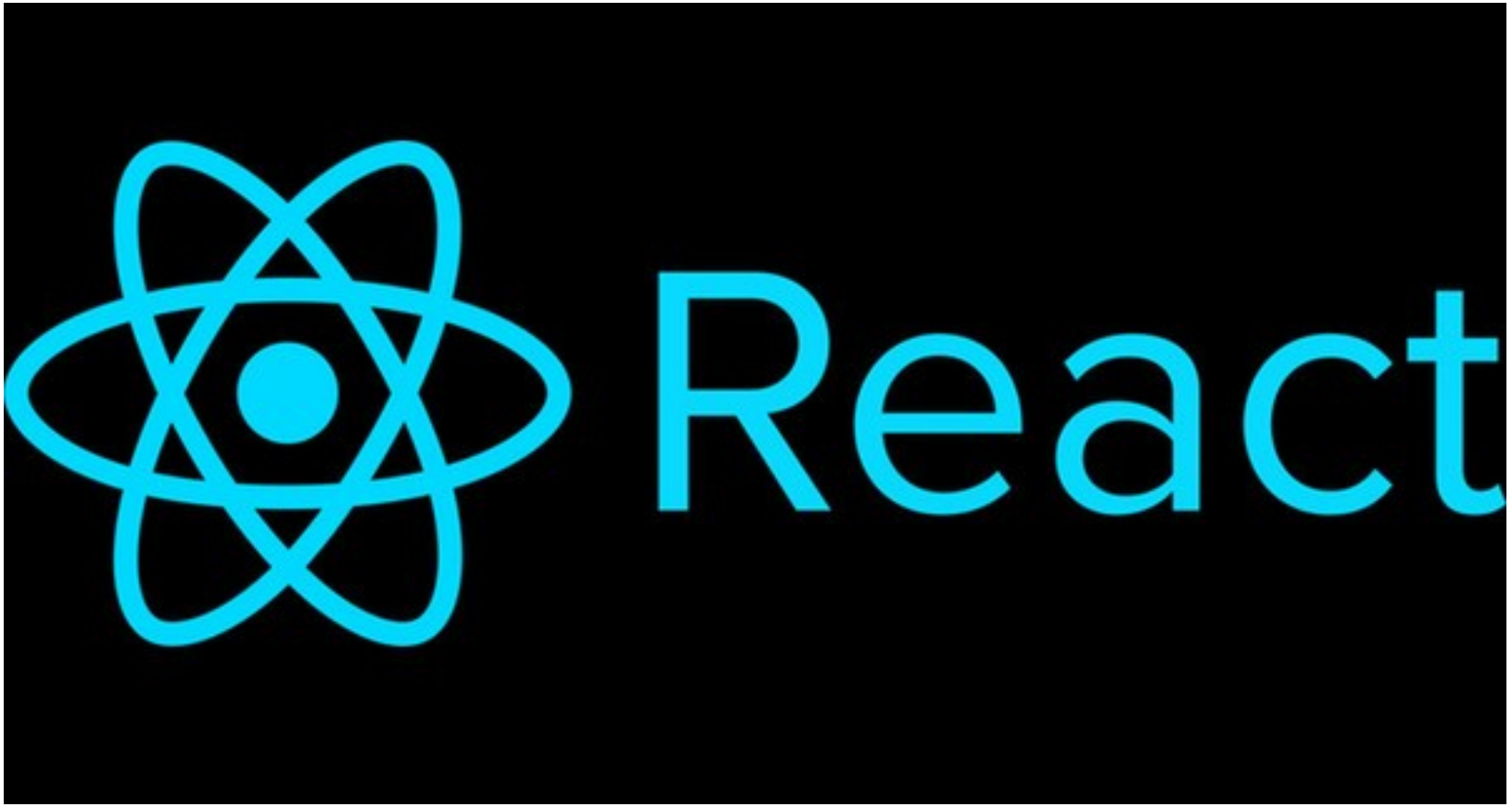
# “Programación con React.JS”




Rogelio Ferreira Escutia

# Convenciones de escritura de código

# PascalCase

- **Se recomienda este tipo de convención para la escritura de código (si no, marcará Warnings en React).**
- **Cada letra de cada palabra que aparezca será mayúscula.**
- **Generalmente se usan para definir los nombres de las funciones, clases y objetos.**

- PascalCase: `LongFunctionName()`

"PascalCase", https://techterms.com/definition/pascalcase, mayo 2020

# Convenciones

- **Otro tipo de convenciones utilizadas por los desarrolladores:**

```
camelCase          =>   firstName
PascalCase         =>   FirstName
SnakeCase          =>   first_name
KebabCase          =>   first-name
UpperCase + SnakeCase  =>   FIRST_NAME
lowercase          =>   firstname
```

"La importancia de las convenciones de codificación", https://geeks.ms/jorge/2019/03/24/la-importancia-de-las-convenciones-de-codificacion-pascalcase-camelcase-snake_case-y-kebab-case/, mayo 2020

# Creación del primer Componente

# Creación del primer Componente

- **Dentro de nuestro archivo "App.js" escribiremos el primer componente que se encargará de imprimir el "Hola Mundo":**

```
function HolaMundo(){
  return(
    <div id="hola">Hola Crayola!!!</div>
  )
}
```

# Hola Mundo desde un Componente

# Hola Mundo con Componente

- **Imprimir el “Hola Mundo” desde la función principal (App) y llamando al componente HolaMundo:**

```
function HolaMundo(){
  return(
    <div id="hola">Hola Crayola!!!</div>
  )
}

function App() {
  return (
    <div><HolaMundo/></div>
  );
}
```

# Hola Mundo desde un Componente con paso de mensaje

# Paso de mensaje

- **Imprimir el "Hola Mundo" desde la función principal (App) y llamando al componente HolaMundo, donde se envía como parámetro el mensaje:**

```
function HolaMundo(x){
  return(
    <div id="hola">{x.mensaje}</div>
  )
}

function App() {
  return (
    <div><HolaMundo mensaje="Hola Crayola" /></div>
  );
}
```

# Paso de mensaje

- **Se envían 2 parámetros (cliente y descripción) para que sean impresos por el componente:**

```
function HolaMundo(x){
  return(
    <>
      <div id="texto">Nombre:</div>
      <div id="hola">{x.nombre}</div>
      <div id="texto">Descripción:</div>
      <div id="hola">{x.descripcion}</div>
    </>
  )
}

function App() {
  return (
    <div>
      <HolaMundo nombre="Rogelio" descripcion="cliente" />
      </div>
  );
}
```

# Reutilizando un Componente

# Reutilizando un Componente

- **Se utiliza el componente “HolaMundo” 3 veces, cada una con mensajes diferentes:**

```
function HolaMundo(x){
  return(
    <div id="hola">{x.mensaje}</div>
  )
}

function App() {
  return (
    <div>
      <HolaMundo mensaje="Hola Crayola!!!" />
      <HolaMundo mensaje="Otro mensaje" />
      <HolaMundo mensaje="Y algo más" />
      </div>
  );
}
```

# Componentes como Clases

# Componentes como Clases

- **En vez de crear un componente com Función, lo hacemos como clases y utilizamos los conceptos de POO:**

```
class HolaMundo extends React.Component {
  render() {
    return (
      <div id="hola">
        <h3>{this.props.nombre}</h3>
        {this.props.descripcion}
      </div>
    )
  }
}

function App() {
  return (
    <div>
      Este es mi componente:
      <HolaMundo nombre="Rogelio " descripcion="cliente" />
      </div>
  );
}
```

# Botones

# Botones

- **Agregar a un componente un botón, que al oprimirlo envie un mensaje a la pantalla (alert):**

```
class HolaMundo extends React.Component {
  render() {
      return (
        <div>
          <button onClick={function cambio(){alert("Bienvenido!!!")}}>Botón</button>
        </div>
      )
  }
}

function App() {
  return (
    <div>
      Página Principal:
      <HolaMundo/>
      </div>
  );
}
```

# Estado de un componente

# Estado

- **Un componente puede tener un estado interno (state), el cual puede ser cambiado de manera dinámica.**

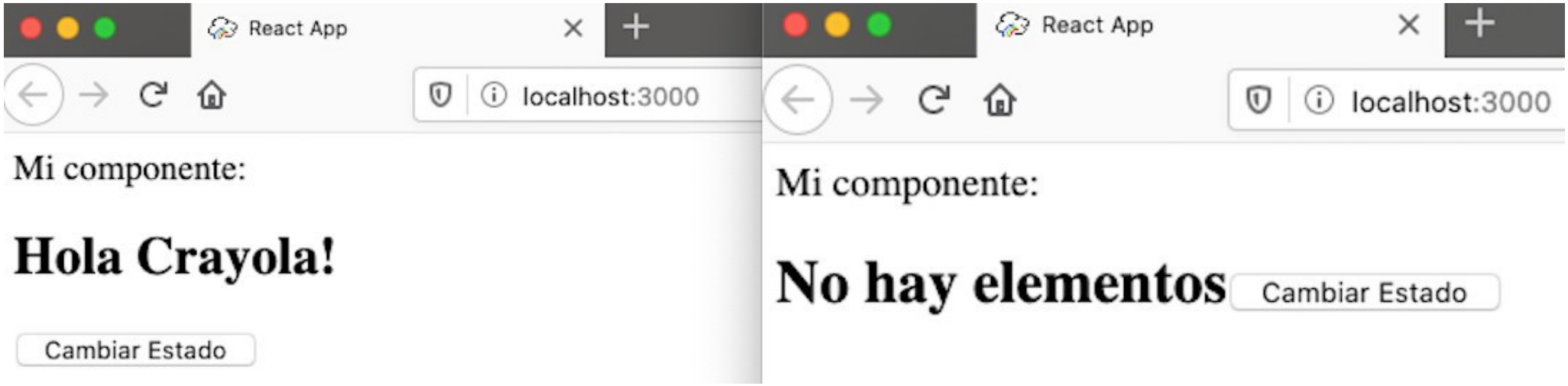
- **Esto es muy útil para usar y reutilizar componentes de acuerdo a los eventos que se presenten en el sistema.**

# Estado

```
class HolaMundo extends React.Component {
  state = {
    mostrar: true
  }
  CambiarEstado = () => {
    this.setState({mostrar: !this.state.mostrar})
  }
  render() {
    if (this.state.mostrar) {
      return (
        <div id="hola">
          <h2>{this.props.texto}</h2>
          <button onClick={this.CambiarEstado}>Cambiar Estado</button>
        </div>
      )
    } else {
      return <h2>
        No hay elementos
        <button onClick={this.CambiarEstado}>Cambiar Estado</button>
      </h2>
    }
  }
}
function App() {
  return (
    <div>
      Mi componente:
      <HolaMundo texto="Hola Crayola!" />
      </div>
  );
}
```

# Estado

- **Pantalla antes y después de hacer click en un botón:**



# *Rogelio Ferreira Escutia*


***Instituto Tecnológico de Morelia***
***Departamento de Sistemas y Computación***

***Correo:*** ***rogelio@itmorelia.edu.mx***
***rogeplus@gmail.com***

***Página Web:*** ***http://sagitario.itmorelia.edu.mx/~rogelio/***
***http://www.xumarhu.net/***

***Twitter:*** ***http://twitter.com/rogeplus***
***Facebook:*** ***http://www.facebook.com/groups/xumarhu.net/***