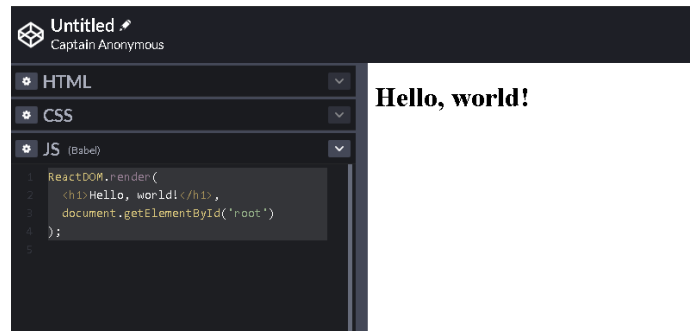


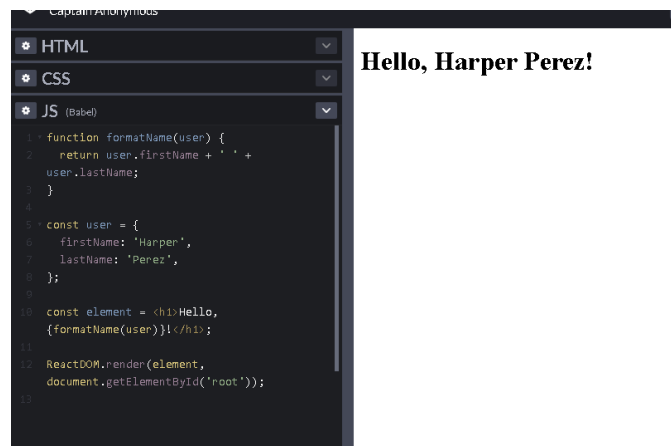
TUTORIAL REACT.JS

1. Hello world!.



2. Introducing JSX.

a. Declaración de variables y funciones javascript en JSX.



b. Representación de objetos.

```
const element = (  
  <h1 className="greeting">  
    Hello, world!  
  </h1>  
);
```

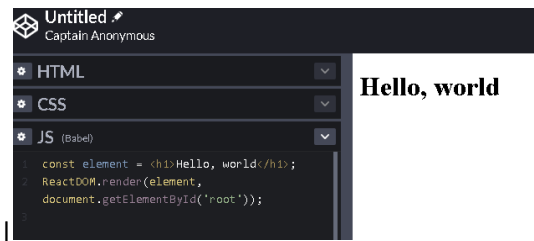
```
const element = React.createElement(  
  'h1',  
  {className: 'greeting'},  
  'Hello, world!'  
);
```

3. Elementos renderizados.

a. Los elementos son los bloques de construcción más pequeños de las aplicaciones React.

```
const element = <h1>Hello, world</h1>;
```

- b. Otra forma de utilizar los DOM con el ejemplo Hola mundo.

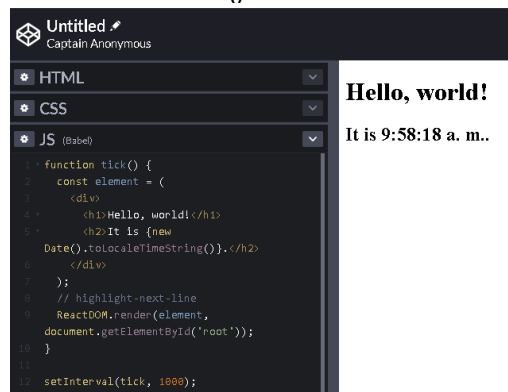


The screenshot shows a web browser window with the title 'Untitled' and 'Captain Anonymous'. The browser's developer tools are open, showing the 'JS (Babel)' tab. The code in the console is:

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element,
  document.getElementById('root'));
```

The browser displays 'Hello, world' in a large, bold, black font.

- c. La única manera de actualizar la interfaz de usuario es por medio de ReactDOM.render().

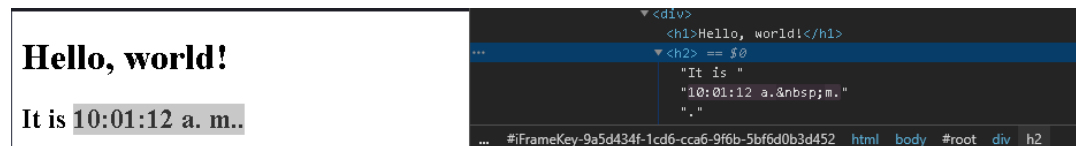


The screenshot shows a web browser window with the title 'Untitled' and 'Captain Anonymous'. The browser's developer tools are open, showing the 'JS (Babel)' tab. The code in the console is:

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new
        Date().toLocaleTimeString()}</h2>
    </div>
  );
  // highlight next line
  ReactDOM.render(element,
    document.getElementById('root'));
}
setInterval(tick, 1000);
```

The browser displays 'Hello, world!' in a large, bold, black font, and 'It is 9:58:18 a. m..' in a smaller, black font.

- d. React solo actualiza lo que es necesario.



The screenshot shows a web browser window with the title 'Untitled' and 'Captain Anonymous'. The browser's developer tools are open, showing the 'JS (Babel)' tab. The code in the console is:

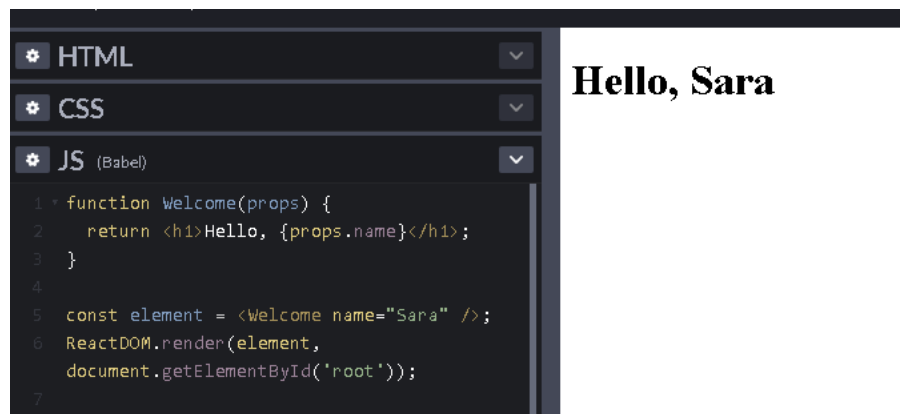
```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new
        Date().toLocaleTimeString()}</h2>
    </div>
  );
  // highlight next line
  ReactDOM.render(element,
    document.getElementById('root'));
}
setInterval(tick, 1000);
```

The browser displays 'Hello, world!' in a large, bold, black font, and 'It is 10:01:12 a. m..' in a smaller, black font. The React DevTools component inspector is open, showing the component tree with the following structure:

```
<div>
  <h1>Hello, world!</h1>
  <h2>It is "10:01:12 a. m.."</h2>
</div>
```

4. Componentes de funciones REACT.

- Llamamos ReactDOM.render() con el <Welcome name="Sara" /> elemento.
- React llama al Welcome componente con {name: 'Sara'}.
- Nuestro Welcome componente devuelve un <h1>Hello, Sara</h1> elemento como resultado.
- Reaccionar DOM actualiza eficientemente el DOM para que coincida <h1>Hello, Sara</h1>.



The screenshot shows a web browser window with the title 'Untitled' and 'Captain Anonymous'. The browser's developer tools are open, showing the 'JS (Babel)' tab. The code in the console is:

```
function welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(element,
  document.getElementById('root'));
```

The browser displays 'Hello, Sara' in a large, bold, black font.

- e. Se maneja la abstracción en todos sus niveles. Por ejemplo, podemos crear un App componente que se procese Welcome muchas veces:

```
Captain Anonymous
HTML
CSS
JS (Babel)
1 function Welcome(props) {
2   return <h1>Hello, {props.name}</h1>;
3 }
4
5 function App() {
6   return (
7     <div>
8       <Welcome name="Sara" />
9       <Welcome name="Cahal" />
10      <Welcome name="Edite" />
11    </div>
12  );
13 }
14
15 ReactDOM.render(<App />,
16   document.getElementById('root'));
```

Hello, Sara
Hello, Cahal
Hello, Edite

- f. Extracting Componets: Permite extraer elementos de un componente y permitiendo mejor manejo y accesibilidad a los mismos.

```
function Comment(props) {
  return (
    <div className="Comment">
      <div className="UserInfo">
        <img className="Avatar"
          src={props.author.avatarUrl}
          alt={props.author.name}
        />
        <div className="UserInfo-name">
          {props.author.name}
        </div>
      </div>
      <div className="Comment-text">
        {props.text}
      </div>
      <div className="Comment-date">
        {formatDate(props.date)}
      </div>
    </div>
  );
}

function formatDate(date) {
  return date.toLocaleDateString();
}

function Avatar(props) {
  return (
    <img
      className="Avatar"
      src={props.user.avatarUrl}
      alt={props.user.name}
    />
  );
}

function UserInfo(props) {
  return (
    <div className="UserInfo">
      <Avatar user={props.user} />
      <div className="UserInfo-name">
        {props.user.name}</div>
    </div>
  );
}

function Comment(props) {
  return (
    <div className="Comment">
      <UserInfo user={props.author} />
      <div className="Comment-text">
        {props.text}</div>
      <div className="Comment-date">
        {formatDate(props.date)}
      </div>
    </div>
  );
}

const comment = {
  date: new Date(),
  text: 'I hope you enjoy learning React!',
  author: {
    name: 'Hello Kitty',
    avatarUrl: 'https://placekitten.com/g/64/64',
  },
};

ReactDOM.render(
  <Comment
    date={comment.date}
    text={comment.text}
    author={comment.author}
  />,
  document.getElementById('root')
);
```

Se extrajo el elemento avatar (objeto), texto y como fecha. Permitiendo mayor flexibilidad y bajo acoplamiento en el código. Además de una futura reutilización del código.

5.

a. Estado y ciclo de vida.

```
function Clock(props) {
  return (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {props.date.toLocaleTimeString()}</h2>
    </div>
  );
}

function tick() {
  ReactDOM.render(
    <Clock date={new Date()} />,
    document.getElementById('root')
  );
}

setInterval(tick, 1000);
```

```
function Clock(props) {
  return (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is
{props.date.toLocaleTimeString()}</h2>
    </div>
  );
}

function tick() {
  ReactDOM.render(
    <Clock date={new Date()} />,
    document.getElementById('root')
  );
}

setInterval(tick, 1000);
```

b. Convertir una función en clase

1. Cree una clase ES6 , con el mismo nombre, que se extienda React.Component.
2. Agregue un único método vacío al llamado render().
3. Mueva el cuerpo de la función al render()método.
4. Reemplazar props con this.props en el render()cuerpo.
5. Elimine la declaración de función vacía restante.

```
HTML
CSS
JS (Babel)
1 class Clock extends React.Component {
2   render() {
3     return (
4       <div>
5         <h1>Hello, world!</h1>
6         <h2>It is
7         {this.props.date.toLocaleTimeString()}</h2>
8       </div>
9     );
10  }
11 }
12 function tick() {
13   ReactDOM.render(
14     <Clock date={new Date()} />,
15     document.getElementById('root')
16   );
17 }
18
19 setInterval(tick, 1000);
```

Hello, world!

It is 11:49:17 a. m..

c. Agregar un estado local en una clase

```
1 class Clock extends React.Component {
2   constructor(props) {
3     super(props);
4     this.state = {date: new Date()};
5   }
6
7   render() {
8     return (
9       <div>
10        <h1>Hello, world!</h1>
11        <h2>It is
12        {this.state.date.toLocaleTimeString()}.
13        </h2>
14      </div>
15    );
16  }
17 }
18 ReactDOM.render(
19   <Clock />,
20   document.getElementById('root')
21 );
```

Hello, world!

It is 11:52:14 a. m..

1. Reemplazar this.props.date con this.state.date en el render() método:
2. Agregue un constructor de clase que asigne la inicial this.state:
3. Retire el date accesorio del <Clock /> elemento:

6. Agregar métodos de ciclo de vida a una clase

- a. El componentDidMount() método se ejecuta después de que la salida del componente se haya procesado en el DOM. Este es un buen lugar para configurar un temporizador:
- b.