Funciones en React - Bind

Dado que React, trabaja sobre JS, todos aquellos conceptos válidos, como clases, objetos, funciones, y el this, están disponibles y resultan fundamentales para la construcción de aplicaciones.

This

Como ***this*** se usa en React del mismo modo que se usa en Javascript, puede ser útil explicar por qué ***this*** se usa en todo. Todo se reduce al contexto. Cuando invocas una función, ***this*** determina qué objeto es el foco de la función. Aquí hay un ejemplo con una función en el alcance global:

```
var example = function() {
  console.log(this);
}
```

No resulta un ejemplo descabellado. Esto simplemente crea una variable cuyo valor es una función y registra el valor de ***this***. Al ejecutarla, en el ámbito global, se obtiene lo siguiente:

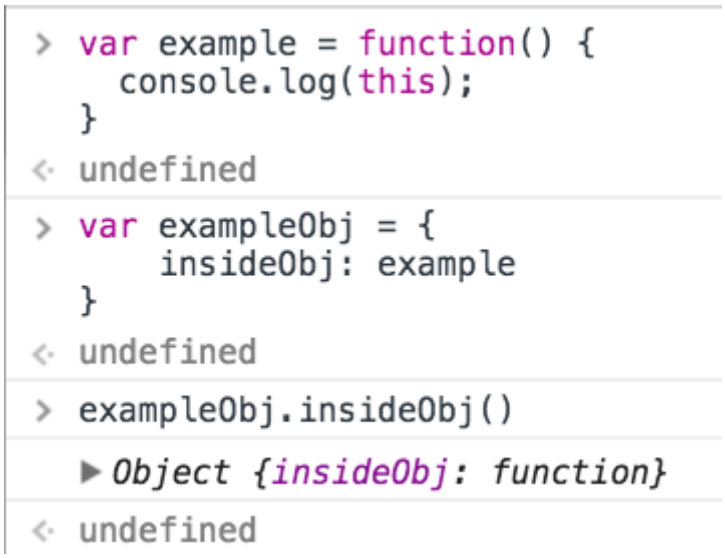
```
> var example = function() {
    console.log(this);
  }
<- undefined
> example()
  ▶ Window {stop: function, open: function, alert: function, confirm: function, prompt: function…}
<- undefined
```

Cuando se invoca la función *example ()*, devuelve el valor de ***this*** dentro del contexto, que en este caso es el objeto 'Window'. El objeto global en el que se ejecutan todas las funciones de Javascript a menos que se especifique lo contrario.

Otro ejemplo:

```
var example = function() {
  console.log(this);
}
var exampleObj = {
  insideObj: example
}
```

La función original *example()* , en lugar de invocarla en el ámbito global, es un valor dentro de un objeto. En Javascript, los objetos definen su propio alcance, entonces, ¿qué sucede cuando invoco esta función? La salida será:



El contexto para ***this*** ha cambiado porque fue invocado desde otro lugar. Este cambio de contexto puede ser difícil de seguir pero es muy importante y romperá su aplicación React si no recuerda configurarlo correctamente.

Entonces ahora veamos como lo manejamos en React.

Aquí tengo un componente de la aplicación que muestra un botón que al hacer clic llama a un método handleClick () que eventualmente cambiará el estado de la aplicación para rastrear la cantidad de veces que se ha hecho clic en el botón.

```
import React, { Component } from 'react';
class App extends Component {
  constructor(props) {
    super(props);
this.state = {
      numOfClicks: 0
    };
  }
handleClick() {
    console.log('From handleClick()', this);
  }
render() {
    console.log('From render()', this);
    return (
      <div>
        <button onClick={this.handleClick}>Click Me!</button>
        <p>Number of Times Clicked = {this.state.numOfClicks}</p>
      </div>
    )
  }
}
export default App;
```

La salida de los console.log():

```
From render() ▶App {props: Object, context: Object, refs: Object, updater: Object, state: Object…}
From handleClick() null
```

Desde adentro, render() ***this*** está configurado para la aplicación (realmente lo que será una instancia de este componente de la aplicación), pero en este punto, ***this*** en la función es nulo. El problema aquí es que se necesita handleClick() para actualizar el estado de este componente y esto se hace con this.setState(). El ***this*** en this.setState() también se establecerá en nulo.

```
handleClick() {
    console.log('From handleClick()', this);
    this.setState({numOfClicks: this.state.numOfClicks + 1});
  }
```

```
From handleClick() null
▶Uncaught TypeError: Cannot read property 'setState' of null
    at handleClick (http://localhost:8080/index_bundle.js:13181:11)
    at Object.ReactErrorUtils.invokeGuardedCallback (http://localhost:8080/index_bundle.js:7330:16)
    at executeDispatch (http://localhost:8080/index_bundle.js:6854:21)
    at Object.executeDispatchesInOrder (http://localhost:8080/index_bundle.js:6877:5)
    at executeDispatchesAndRelease (http://localhost:8080/index_bundle.js:4433:22)
    at executeDispatchesAndReleaseTopLevel (http://localhost:8080/index_bundle.js:4444:10)
    at Array.forEach (native)
    at forEachAccumulated (http://localhost:8080/index_bundle.js:10872:9)
    at Object.processEventQueue (http://localhost:8080/index_bundle.js:4647:7)
    at runEventQueueInBatch (http://localhost:8080/index_bundle.js:22488:18)
```

De nuevo, el ***this*** al que se refiere *setState()* se establece en nulo. Dado que el estado es parte del componente de la aplicación, eso es a lo que ***this*** debe hacer referencia. Lo que debemos es cambiar a quien hace referencia ***this***.

El método **bind()**

Cada objeto JS tiene tres métodos integrados que pueden cambiar el contexto ***this*** para una función. Los primeros dos, .call () y .apply () son similares en que el primer argumento que les pasa es el nuevo contexto y luego pueden pasarles argumentos individuales o una matriz, respectivamente. Lo que también tienen en común es que la función a la que los llamas se ejecuta inmediatamente.

Eso no funcionará en este caso, ya que queremos que handleClick() solo se ejecute cuando se hace clic en el botón. El tercer método incorporado de Javascript puede hacer esto.

El método ***.bind()*** es similar a los otros dos en que le pasa el contexto al que desea vincular la función, pero no ejecuta la función de inmediato.

En su lugar, se devuelve una copia de la función con el contexto conmutado.

Esta copia se puede ejecutar cuando se desee. Como cuando se hace click en un botón.

Usando el Constructor

Dado que el contexto ***this*** de la aplicación es el que debe usar handleClick(), debemos usar el método *.bind ()* en el constructor para ese componente. Si esta línea se agrega al constructor, todo funcionará como se espera:

```
this.handleClick = this.handleClick.bind(this);
```

El primer this.handleClick se refiere al método handleClick(). Como esto se hace en el constructor, ***this*** se refiere a la aplicación. El segundo this.handleClick también hace referencia al método handleClick(), pero en este caso estamos llamando a .bind().

El ***this*** final es el contexto que estamos pasando a .bind() y se refiere al contexto de la aplicación.

Finalmente, estamos configurando el valor de handleClick() para igualar el resultado de llamar a .bind() en handleClick() y pasar .bind() al contexto de ***this*** que se refiere a la aplicación. Esto tendrá el resultado de hacer que el contexto para handleClick() sea ***App*** siempre que se llame, que es lo que queremos, ya que es también donde se encuentra el *estado* y eso es lo que estamos tratando de cambiar.

Usando Arow (=>)

Si se desea mantener los constructores ordenados, se puede usar una función de flecha ES6 para referirse a cualquier método y esto tendrá el mismo resultado.

```
constructor(props) {
    super(props);
this.state = {
      numOfClicks: 0
    };
  }
```

Pero ahora, agregué una función de flecha al props onClick del botón:

```
<button onClick={() => this.handleClick()}>Click Me!</button>
```

Los resultados son exactamente los mismos que antes. Las funciones de flecha no definen su propio contexto, por lo que ***this*** se establece en el contexto adjunto.