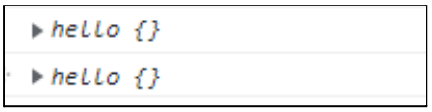
Putting it all together

```
var joe = {
   name: "Joe",
   hello: function(){
       console.log(this) ;
   }
}
new (joe.hello.bind(joe))();
```

Output:



You have to find the value of ‘this’ inside the execution context of the hello() function if it is called as (joe.hello.bind(joe))() using the new keyword.

This is how the output is determined:

1. joe.hello creates a reference to the hello() function in the joe object.
2. joe.hello.bind(joe) creates a new function with the same function definition as what joe.hello refers to.
3. This newly created function has the value of ‘this’ in its execution context set to the joe object.
4. Now, this newly created function is called as (joe.hello.bind(joe))(); using the new ‘keyword’.
5. Since this function uses the ‘new’ keyword, it’ll create a new object.
6. The ‘new’ keyword will link this new object with another object, which is a prototype of the function being called (the hello() function).
7. Then the function (joe.hello.bind(joe))() will be called with the value of ‘this’ set to the hello() function object.
8. Therefore console.log(this) prints the hello object function.
9. Also, new (joe.hello.bind(joe))(); also prints the same as we know when a constructor function returns nothing, then the function object is returned.