JavaScript Test

Write a `Calculator` class that matches this specification

```
describe( "Calculator", function(){
 var calculator;

 beforeEach( function(){
  calculator = new Calculator();
 } );

 it( "adds 1 and 2", function(){
  expect( calculator.add( 1, 2 ) ).to.equal( 3 );
 } );

 it( "subtracts 2 from 9", function(){
  expect( calculator.subtract( 9, 2 ) ).to.equal( 7 );
 } );

 it( "multiplies 4 and 3", function(){
  expect( calculator.multiply( 4, 3 ) ).to.equal( 12 );
 } );

 it( "divides 10 by 2", function(){
  expect( calculator.divide( 10, 2 ) ).to.equal( 5 );
 } );

 it( "does not divide by 0", function(){
  expect( calculator.divide( 5, 0 ) ).to.equal( NaN );
 } );
} );
```

Write a `ScientificCalculator` class that matches this specification

```
describe( "ScientificCalculator", function(){
 var calculator;

 beforeEach( function(){
  calculator = new ScientificCalculator();
 } );

 it( "extends Calculator", function(){
  expect( calculator ).to.be.instanceOf( Calculator );
  expect( calculator ).to.be.instanceOf( ScientificCalculator );
 } );

 it( "returns the sine of PI / 2", function(){
  expect( calculator.sin( Math.PI / 2 ) ).to.equal( 1 );
 } );

 it( "returns the cosine of PI", function(){
  expect( calculator.cos( Math.PI ) ).to.equal( -1 );
 } );

 it( "returns the tangent of 0", function(){
  expect( calculator.tan( 0 ) ).to.equal( 0 );
 } );

 it( "returns the logarithm of 1", function(){
  expect( calculator.log( 1 ) ).to.equal( 0 );
 } );
} );
```

Write a `withExponents` functional mixin that matches this specification

```
describe( "withExponents", function(){
 var calculator;

 beforeEach( function(){
  calculator = new Calculator();
  withExponents.call( calculator );
 } );

 it( "returns 2^3", function(){
  expect( calculator.pow( 2, 3 ) ).to.equal( 8 );
 } );

 it( "multiplies 2^3 and 2^4", function(){
  expect( calculator.multiplyExp( [ 2, 3 ], [ 2, 4 ] ) ).to.equal( 128 );
 } );

 it( "divides 2^3 by 2^5", function(){
  expect( calculator.divideExp( [ 2, 3 ], [ 2, 5 ] ) ).to.equal( 0.25 );
 } );
} );
```

Write a `delay` function that matches this specification

```
describe( "delay", function(){
 var calculator = new Calculator();

 it( "returns a promise", function(){
  var willAdd = delay( 100, calculator, 'add', [ 1, 1 ] );

  expect( willAdd ).to.be.instanceOf( Promise );
  expect( willAdd ).to.be.fulfilled;
 } );

 it( "delays execution", function(){
  expect( delay( 1000, calculator, 'add', [ 10, 5 ] ) ).to.eventually.equal( 15 );
  expect( delay( 500, calculator, 'subtract', [ 9, 5 ] ) ).to.eventually.equal( 4 );
 } );

 it( "cannot execute functions that do not exist", function(){
  expect( delay( 1000, calculator, 'sqrt', [ 2, 2 ] ) ).to.be.rejected;
 } );
} );
```