Principales matchers de Jasmine

1. not

Invert the matcher following this `expect`

Example

```
expect(thing).not.toEqual(realThing);
```

2. toBeCloseTo(expected, precision)

`expect` the actual value to be within a specified precision of the expected value.

Parameters:

Name	Type	Attributes	Default	Description
`expected`	Object			The expected value to compare against.
`precision`	Number	<optional>	2	The number of decimal points to check.

Example

```
expect(number).toBeCloseTo(42.213, 3); // Comprueba si number
redondeado a 3 decimales coincide con 42.213
```

3. toBeDefined()

`expect` the actual value to be defined. (Not `undefined`)

Example

```
expect(result).toBeDefined();
```

4. toBeFalse()

`expect` the actual value to be `false`.

Example

```
expect(result).toBeFalse();
```

5. toBeTrue()

`expect` the actual value to be `true`.

Example

```
expect(result).toBeTrue();
```

6. toBeGreaterThan(expected)

`expect` the actual value to be greater than the expected value.

Parameters:

Name	Type	Description

`expected`	Number	The value to compare against.

Example

```
expect(result).toBeGreaterThan(3);
```

7. toBeGreaterThanOrEqual(expected)

`expect` the actual value to be greater than or equal to the expected value.

Parameters:

Name	Type	Description
`expected`	Number	The expected value to compare against.

Example

```
expect(result).toBeGreaterThanOrEqual(25);
```

8. toBeInstanceOf(expected)

`expect` the actual to be an instance of the expected class

Parameters:

Name	Type	Description
`expected`	Object	The class or constructor function to check for

Example

```
expect('foo').toBeInstanceOf(String);

expect(3).toBeInstanceOf(Number);
```

```
expect(new Error()).toBeInstanceOf(Error);

expect(bonoloto()).toBeInstanceOf(Array);
```

9. toBeLessThan(expected)

`expect` the actual value to be less than the expected value.

Parameters:

Name	Type	Description
`expected`	Number	The expected value to compare against.

Example

```
expect(result).toBeLessThan(0);
```

10. toBeLessThanOrEqual(expected)

`expect` the actual value to be less than or equal to the expected value.

Parameters:

Name	Type	Description
`expected`	Number	The expected value to compare against.

Example

```
expect(result).toBeLessThanOrEqual(123);
```

11. toBeNaN()

`expect` the actual value to be `NaN` (Not a Number).

Example

```
expect(thing).toBeNaN();
```

12. toBeNull()

`expect` the actual value to be `null`.

Example

```
expect(result).toBeNull();
```

13. toBeUndefined()

`expect` the actual value to be `undefined`.

Example

```
expect(result).toBeUndefined()
```

14. toContain(expected)

`expect` the actual value to contain a specific value.

Parameters:

Name	Type	Description

`expected`	**Object**	**The value to look for.**

Example

```
expect(array).toContain(anElement);

expect(string).toContain(substring);
```

15. toEqual(expected)

`expect` **the actual value to be equal to the expected, using deep equality comparison.**

Parameters:

Name	Type	Description
`expected`	**Object**	**Expected value**

Example

```
expect(bigObject).toEqual({"foo": ['bar', 'baz']});
```

16. toHaveSize(expected)

`expect` **the actual size to be equal to the expected, using array-like length or object keys size.**

Parameters:

Name	Type	Description

`expected`	Object	Expected size

Example

```
array = [1,2];

expect(array).toHaveSize(2);
```

17. toThrowError(message)

`expect` a function to `throw` an `Error`.

Ejemplo

Pondremos como ejemplo una clase Jarra que tiene un setter llamado cantidad el cual lanza un error cuando le pasamos un valor negativo:

```
class Jarra {
    constructor(capacidad, cantidad) {
        this._capacidad = capacidad;
        this._cantidad = cantidad;
    }

   set cantidad(c) {
        if (c < 0) throw new Error('La cantidad debe ser un número positivo');
        this._cantidad = Math.min(c, this._capacidad);
    }
}
```

Desde Jasmine podemos testear que efectivamente se lance el error con un código similar al siguiente:

```
expect(function() { jarra1.cantidad = -5
}).toThrowError('La cantidad debe ser un número
positivo');
```