MOCHA ASSERTIONS

- **`assert.equal(actual, expected, [message])`**
  - This method checks for equality between the **`actual`** and **`expected`** values using type coercion. It means that if the types of **`actual`** and **`expected`** are different, JavaScript will attempt to convert one or both values to a common type before making the comparison.

```
assert.equal(5, '5'); // This will pass because '5' is coerced to a number before comparison
assert.equal(5, 5);    // This will pass as well
```

- **`assert.strictEqual(actual, expected, [message])`**
  - performs a strict equality comparison. It checks whether the **`actual`** and **`expected`** values are not only equal in value but also of the same data type, without any type conversion.

```
assert.strictEqual(5, '5'); // This will fail because '5' is a string and 5 is a number
assert.strictEqual(5, 5);    // This will pass
```

- **`assert.deepEqual(actual, expected, [message])`**:
  - Asserts that the **`actual`** and **`expected`** values are deeply equivalent.

```
const assert = require('assert');

describe('Array', function() {
  describe('#indexOf()', function() {
    it('should return -1 when the value is not present', function() {
      assert.deepEqual([1, 2, 3], [1, 2, 3]); // Passes
    });
  });
});
```

- **`assert.notEqual(actual, expected, [message])`**
- **`assert.notStrictEqual(actual, expected, [message])`**
- **`assert.notDeepEqual(actual, expected, [message])`**

- **`assert.isAbove(valueToCheck, valueToCompareAgainst, [message])`**:
  - Asserts that **`valueToCheck`** is strictly greater than **`valueToCompareAgainst`**.
- **`assert.isBelow(valueToCheck, valueToCompareAgainst, [message])`**:
  - Asserts that **`valueToCheck`** is strictly less than **`valueToCompareAgainst`**.

- **`assert.isAtLeast(valueToCheck, valueToCompareAgainst, [message])`**:
  - Asserts that **`valueToCheck`** is greater than or equal to **`valueToCompareAgainst`**.
- **`assert.isAtMost(valueToCheck, valueToCompareAgainst, [message])`**:
  - Asserts that **`valueToCheck`** is less than or equal to **`valueToCompareAgainst`**.

```
assert.isAbove(10, 5, '10 should be strictly greater than 5');

assert.isBelow(5, 10, '5 should be strictly less than 10');

assert.isAtLeast(10, 10, '10 should be greater than or equal to 10');

assert.isAtMost(5, 5, '5 should be less than or equal to 5');
```

- **`assert.isString(value, [message])`**: Asserts that **`value`** is a string.
- **`assert.isNumber(value, [message])`**: Asserts that **`value`** is a number.
- **`assert.isArray(value, [message])`**: Asserts that **`value`** is an array.
- **`assert.isObject(value, [message])`**: Asserts that **`value`** is an object.
- **`assert.isFunction(value, [message])`**: Asserts that **`value`** is a function.

```
assert.isString('hello', 'Value should be a string');

assert.isNumber(42, 'Value should be a number');

assert.isArray([1, 2, 3], 'Value should be an array');

assert.isObject({ name: 'John', age: 30 }, 'Value should be an object');

assert.isFunction(() => {}, 'Value should be a function');
```

- **`assert.include(container, value, [message])`**: Asserts that **`container`** includes **`value`**. Works for arrays, strings, and objects.
- **`assert.notInclude(container, value, [message])`**: Asserts that **`container`** does not include **`value`**. Works for arrays, strings, and objects.

```
assert.include([1, 2, 3], 2, 'Array should include value 2');

assert.notInclude([1, 2, 3], 4, 'Array should not include value 4');
```

- **`assert.isTrue(value, [message])`**: Asserts that the **`value`** is true
- **`assert.isFalse(value, [message])`**: Asserts that the **`value`** is false.
- **`assert.isOk(value, [message])`**: Asserts that the **`value`** is truthy.
- **`assert.isNotOk(value, [message])`**: Asserts that the **`value`** is falsy.
- **`assert.isUndefined(value, [message])`**: Asserts that the **`value`** is undefined.
- **`assert.isNotNull(value, [message])`**: Asserts that the **`value`** is not null.