Assert.js v1.0.4 cheatsheet - https://github.com/Serrin/assert.js

Category	Assertions
Constants	`assert.VERSION;`
Errors	`assert.AssertionError`

Basic	`assert(condition, [message: string | Error]);`
	`assert.ok(condition, [message: string | Error]);`
	`assert.notOk(condition, [message: string | Error]);`
	`assert.fail([message: string | Error]);`

Equality	`assert.equal(actual, expected, [message: string | Error]);`
	`assert.strictEqual(actual, expected, [message: string | Error]);`
	`assert.deepEqual(actual, expected, [message: string | Error]);`
	
	`assert.notEqual(actual, expected, [message: string | Error]);`
	`assert.notStrictEqual(actual, expected, [message: string | Error]);`
	`assert.notDeepEqual(actual, expected, [message: string | Error]);`

Exceptions	`assert.throws(fn, [ErrorType | string | RegExp], [message: string | Error]): Promise;`
	`await assert.rejects(asyncFnOrPromise, [ErrorType | string| RegExp], [message: string | Error]);`
	`await assert.doesNotReject(asyncFnOrPromise, [ErrorType | string | RegExp], [message: string| Error]);`

Boolean	`assert.isTrue(value, [message: string | Error]);` `assert.isFalse(value, [message: string | Error]);`

String	`assert.match(string, regexp, [message: string | Error]);`
	`assert.doesNotMatch(string, regexp, [message: string | Error]);`
	
	`assert.stringContains(actual, substring, [message: string | Error]);`
	`assert.stringNotContains(actual, substring, [message: string | Error]);`

Comparison	`assert.lt(value1, value2, [message: string | Error]);`
	`assert.lte(value1, value2, [message: string | Error]);`
	`assert.gt(value1, value2, [message: string | Error]);`
	`assert.gte(value1, value2, [message: string | Error]);`
	
	`assert.inRange(value, min, max, [message: string | Error]);`
	`assert.notInRange(value, min, max, [message: string | Error]);`

Objects	`assert.includes(container, options: {keyOrValue, [value] }, [message: string | Error]);`
	`assert.doesNotInclude(container, options: {keyOrValue, [value] }, [message: string | Error]);`
	
	`assert.isEmpty(value, [message: string | Error]);`
	`assert.isNotEmpty(value, [message: string | Error]);`

Type	assert.is(value, expectedType: string | function | Array<string | function>, [message: string | Error]);
	assert.isNot(value, expectedType: string | function | Array<string | function>, [message: string | Error]);
	
	assert.isPrimitive(value, [message: string | Error]);
	assert.isNotPrimitive(value, [message: string | Error]);
	
	assert.isNullish(value, [message: string | Error]);
	assert.isNotNullish(value, [message: string | Error]);
	
	assert.isNaN(value, [message: string | Error]);
	assert.isNotNaN(value, [message: string | Error]);
	
	assert.isNull(value, [message: string | Error]);
	assert.isUndefined(value, [message: string | Error]);
	assert.isString(value, [message: string | Error]);
	assert.isNumber(value, [message: string | Error]);
	assert.isBigInt(value, [message: string | Error]);
	assert.isBoolean(value, [message: string | Error]);
	assert.isSymbol(value, [message: string | Error]);
	assert.isFunction(value, [message: string | Error]);
	assert.isObject(value, [message: string | Error]);
	
	assert.isNotNull(value, [message: string | Error]);
	assert.isNotUndefined(value, [message: string | Error]);
	assert.isNotString(value, [message: string | Error]);
	assert.isNotNumber(value, [message: string | Error]);
	assert.isNotBigInt(value, [message: string | Error]);
	assert.isNotBoolean(value, [message: string | Error]);
	assert.isNotSymbol(value, [message: string | Error]);
	assert.isNotFunction(value, [message: string | Error]);
	assert.isNotObject(value, [message: string | Error]);

Testrunner	type TestResult<T> =
	  | {ok: true, value: T, block: Function, name: string}
	  | {ok: false, error: Error, block: Function, name: string};
	
	assert.testSync(block): TestResult
	await assert.testAsync(block): TestResult
	assert.testCheck(result: TestResult): result.ok is true