

Assert.js v1.1.0 cheatsheet - <https://github.com/Serrin/assert.js>

Import	Javascript
Import the assert function	<pre>import assert from "./assert.js"; globalThis.assert = assert;</pre>
Import the assert function as defaultExport	<pre>import defaultExport from "./assert.js"; globalThis.assert = defaultExport;</pre>
Dynamic import	<pre>const assert = await import("./assert.js"); globalThis.assert = assert;</pre>

Category	Assertions
Constants	<code>assert.VERSION;</code>
Errors	<code>assert.AssertionError([message], [options]);</code>
Basic	<code>assert(condition, [message: string Error]);</code> <code>assert.ok(condition, [message: string Error]);</code> <code>assert.notOk(condition, [message: string Error]);</code> <code>assert.fail([message: string Error]);</code>
Equality	<code>assert.equal(actual, expected, [message: string Error]);</code> <code>assert.strictEqual(actual, expected, [message: string Error]);</code> <code>assert.deepEqual(actual, expected, [message: string Error]);</code> <code>assert.notEqual(actual, expected, [message: string Error]);</code> <code>assert.notStrictEqual(actual, expected, [message: string Error]);</code> <code>assert.notDeepEqual(actual, expected, [message: string Error]);</code>
Exceptions	<code>assert.throws(fn, [ErrorType string RegExp], [message: string Error]): Promise;</code> <code>await assert.rejects(asyncFnOrPromise, [ErrorType string RegExp], [message: string Error]);</code> <code>await assert.doesNotReject(asyncFnOrPromise, [ErrorType string RegExp], [message: string Error]);</code>
Boolean	<code>assert.isTrue(value, [message: string Error]);</code> <code>assert.isFalse(value, [message: string Error]);</code>
String	<code>assert.match(string, regexp, [message: string Error]);</code> <code>assert.doesNotMatch(string, regexp, [message: string Error]);</code> <code>assert.stringContains(actual, substring, [message: string Error]);</code> <code>assert.stringNotContains(actual, substring, [message: string Error]);</code>
Comparison	<code>assert.lt(value1, value2, [message: string Error]);</code> <code>assert.lte(value1, value2, [message: string Error]);</code> <code>assert.gt(value1, value2, [message: string Error]);</code> <code>assert.gte(value1, value2, [message: string Error]);</code> <code>assert.inRange(value, min, max, [message: string Error]);</code> <code>assert.notInRange(value, min, max, [message: string Error]);</code>
Objects	<code>assert.includes(container, options: {keyOrValue, [value] }, [message: string Error]);</code> <code>assert.doesNotInclude(container, options: {keyOrValue, [value] }, [message: string Error]);</code> <code>assert.isEmpty(value, [message: string Error]);</code> <code>assert.isNotEmpty(value, [message: string Error]);</code>

Type	<pre> 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.isNotNullable(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.isDefined(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]); </pre>
Testrunner	<pre> 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 </pre>