



Globalne zmienne

<code>__filename</code>	ścieżka bezwzględna aktualnie wykonywanego skryptu.
<code>__dirname</code>	nazwa katalogu, w którym znajduje się aktualnie wykonywany skrypt (ścieżka bezwzględna).
<code>console</code>	referencja do konsoli.
<code>module</code>	referencja bieżącego modułu.
<code>global</code>	referencja globalnej przestrzeni nazw.
<code>process</code>	obiekt globalny, reprezentujący proces.
<code>Buffer</code>	typ globalny do przechowywania danych binarnych.

Liczniki czasu

<code>setTimeout(callback, delay[, ...args])</code>	ustawia opóźnienie wykonywanego skryptu.
<code>clearTimeout(t)</code>	czyści ustawienia opóźnienia dla obiektu typu <code>Timeout</code> .
<code>setInterval(callback, delay[, ...args])</code>	ustawia interwał wykonywania danej czynności.
<code>clearInterval(t)</code>	czyści interwał ustawiony komendą <code>setInterval()</code> .
<code>setImmediate(callback[, ...args])</code>	wykonuje daną operację asynchronicznie.
<code>clearImmediate(immediateObject)</code>	anuluje zadanie stworzone przez <code>setImmediate()</code> .

Moduły

<code>const loadedModule = require('./exampleModule')</code>	wczytuje moduł <code>exampleModule</code> .
<code>const loadedModule = require('nazwaModulu')</code>	wczytuje moduł zainstalowany z npm lub wbudowany w node
<code>loadedModule.require('./exampleModule')</code>	wczytuje moduł <code>exampleModule</code> .
<code>loadedModule.id</code>	identyfikator modułu.
<code>loadedModule.filename</code>	nazwa pliku modułu.
<code>loadedModule.loaded</code>	wskazuje, czy moduł jest załadowany.
<code>loadedModule.parent</code>	referencja do modułu, który jako pierwszy zażądał modułu <code>module</code>
<code>loadedModule.children</code>	zwraca moduły, które zostały wczytane w zasięgu modułu <code>module</code> .

Util

<code>util.format(format[, ...args])</code>	zwraca sformatowany ciąg tekstowy, działa jak <code>printf</code> .
<code>util.inspect(object, [opts])</code>	zwraca reprezentację obiektu w celu sprawdzenia go i debugowania.

URL

<code>url.parse(urlStr, [parseQueryString, [slashesDenoteHost]])</code>	zwraca obiekt URL.
<code>url.format(urlObj)</code>	zwraca sparsowany obiekt URL jako ciąg tekstowy.
<code>url.resolve(from, to)</code>	rozwiązuje docelowy URL w stosunku do bazowego.

Testowanie

<code>assert(warunek[, informacja])</code>	sprawdza, czy dany warunek się zgadza, a jeśli tak, to zwraca informację; alias <code>assert.ok()</code> .
<pre>const spodziewanaWartosc = 3 assert(spodziewanaWartosc === 3, 'zgadza się')</pre>	
<code>assert.equal(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza, czy wartość sprawdzana jest równa oczekiwanej, a jeśli tak, to zwraca informację.
<code>assert.notEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza, czy wartość sprawdzana nie jest równa oczekiwanej, a jeśli tak, to zwraca informację.
<code>assert.deepEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza dogłębnie, czy dane tablice lub obiekty są sobie równe.
<pre>const assert = require('assert') const obj1 = { a: { b: 1 } } const obj2 = { a: { b: 2 } } const obj3 = { a: { b: 1 } }</pre>	

<pre>} const obj4 = Object.create(obj1) assert.deepEqual(obj1, obj1) // OK, wykaże, że obiekt jest identyczny assert.deepEqual(obj1, obj2) // AssertionError: { a: { b: 1 } } deepEqual { a: { b: 2 } } // wartości a i b są różne assert.deepEqual(obj1, obj3) // OK, wykaże, że obiekt jest identyczny assert.deepEqual(obj1, obj4) // AssertionError: { a: { b: 1 } } deepEqual {}</pre>	
<code>assert.notDeepEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza dogłębnie, czy dane tablice lub obiekty nie są sobie równe; najlepiej używać do obiektów i tablic.
<code>assert.strictEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza, czy wartość sprawdzana jest identyczna z oczekiwaną (wartość i typ muszą się zgadzać), a jeśli tak, to zwraca informację; najlepiej używać do zmiennych.
<code>assert.notStrictEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza, czy wartość sprawdzana jest identyczna z oczekiwaną (wartość i typ muszą się zgadzać), a jeśli tak, to zwraca informację; najlepiej używać do zmiennych.
<code>assert.deepStrictEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza dogłębnie, czy dane tablice lub obiekty są identyczne (czyli muszą być równe co do wartości i typu); najlepiej używać do obiektów i tablic.
<code>assert.notDeepStrictEqual(wartoscSprawdzana, wartoscOczekiwana[, informacja])</code>	sprawdza dogłębnie, czy dane tablice lub obiekty nie są identyczne (czyli nie mogą być równe co do wartości i typu); najlepiej używać do obiektów i tablic.
<code>assert.throws(blok[, bład, informacja])</code>	sprawdza, czy podana funkcja zwraca błąd. Opcjonalnie możemy określić rodzaj błędu, podając go jako drugi parametr. Po spełnieniu warunku zostanie wyświetlony komunikat zawarty w parametrze informacja.
<pre>assert.throws(() => { throw new Error('Zła wartość') }, Error)</pre>	

Bufor (Buffer)

Buffer został stworzony do działania na danych binarnych, jest oparty na standardzie `TypedArrays`, który opisuje tablice liczb całkowitych.

Bufora używa się do obsługi strumienia danych.

<code>Buffer.alloc(rozmiar[, wypelnienie[, kodowanie]])</code>	tworzy nowy bufor o danym rozmiarze, wypełnia go danymi — <code>wypelnienie</code> — i konwertuje go z konkretnego kodowania na tablicę liczb całkowitych.
--	--

```
const buf = Buffer.alloc(5)

// Zwraca: <Buffer 00 00 00 00 00>
console.log(buf)

const buf = Buffer.alloc(5, 'a')

// Zwraca: <Buffer 61 61 61 61 61>
console.log(buf)

const buf = Buffer.alloc(11, 'aGVsbG8gd29ybGQ=', 'base64')

// Zwraca: <Buffer 68 65 6c 6c 6f 20 77 6f 72 6c 64>
console.log(buf)
```

<code>Buffer.byteLength(ciągZnaków[, kodowanie])</code>	zwraca długość bufora wypełnionego danymi ciągami znaków, przekonwertowanego z danego kodowania.
<code>Buffer.compare(buffer1, buffer2)</code>	porównuje, czy dwa bufory są sobie równe.

```
const buf1 = Buffer.from('1234')
const buf2 = Buffer.from('0123')
const arr = [buf1, buf2]

// Zwraca: [<Buffer 30 31 32 33>, <Buffer 31 32 33 34>]
```

*Dalsza część książki dostępna w wersji
pełnej.*

