OS	ブラウザ	バージョン	ループ内try	try内ループ
Windows	Opera	11.00	435	373
Windows	Safari	5.0.3	33	25

［参考］ECMA-262 仕様
12.6.3 for 文（The for Statement）
12.14 try 文（The try statement）

4.2.18 | スコープにおける処理の実行

with文は、そのブロック内を引数で与えたスコープで実行する。たとえば、下記コードのwithブロック内では、document.bodyは、「body」と書くだけで実行できる。

スコープを強制できる点は手軽だが、重たい処理になる。しかも、with(a){ var b; }などとしたときに、withブロック内ではaオブジェクトのスコープにアクセスできるが、同時にvar bはwithの外のスコープに露出するため、非常にわかりにくいコードになる。速度を別にしてもお勧めしない。

with あり ★

```
with(document) {
    for (var i = 0; i < 100000; i++) {
        body
    }
}
```

with なし ★★★★★

```
for (var i = 0; i < 100000; i++) {
    document.body
}
```

パフォーマンス比較表 ［計測値の単位：ミリ秒］

OS	ブラウザ	バージョン	with	withなし
iOS	iPad	4_2_1	1065	33
iOS	iPhone	4_2_1	1343	46
Linux	Android	2.1(IS03)	1271	220
Linux	Android	2.1(SO01B)	1207	203
Linux	Android	2.2(001HT)	1416	162
Linux	Android	2.2(SC02B)	2422	354
Mac	Chrome	8.0.552.215	128	6
Mac	Opera	10.63	143	87
Mac	Opera	11.00	151	77

OS	ブラウザ	バージョン	with	withなし
Mac	Safari	5.0.3	48	4
Windows	Chrome	8.0.552.215	143	15
Windows	Firefox	3.6.12	483	25
Windows	Firefox	4.0b7	373	16
Windows	IE	7.0	305	406
Windows	IE	8.0	183	273
Windows	Opera	10.70	284	107
Windows	Opera	11.00	108	55
Windows	Safari	5.0.3	62	4

［参考］ECMA-262 仕様
12.10 with 文（The with Statement）

4.2.19 | スコープ内外の変数へのアクセス速度

関数内のソースが実行されるとき、関数内にある変数は同じスコープにあり、すぐにアクセスできる。グローバル変数は関数の外にあるため、スコープチェーンは遠くなり、それをたどってアクセスするぶん遅くなる。

実は、このテストはその2つの変数へのアクセス速度の比較のはずだったが、適切ではない。なぜなら、グローバル変数にwindow.aを使ってしまったために、DOM操作の負荷が大きく加わっているからだ。結果はスコープによる遅延をはるかに超えてwindow.aへのアクセスのケースが非常に遅いものとなった。しかし、関数内からグローバルwindowへアクセスすることは多いので、あえて載せておく。ちなみに、もし、関数内からwindowへのアクセスが頻発する場合は、いったんwindowをローカル変数winへ代入してから、そのローカル変数winを使うと、ChromeやOperaなどでは数倍速くなる。

グローバル変数 window.a へのアクセス ★

```
function () {
    for (var i = 0; i < 1000000; i++) {
        window.a = 'abc';
    }
}
```

ローカル内変数 a へのアクセス ★★★★

```
function () {
    var a = '';
    for (var i = 0; i < 1000000; i++) {
        a = 'abc';
    }
}
```