НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЛІНГВІСТИКИ ТА СОЦІАЛЬНИХ КОМУНІКАЦІЙ
КАФЕДРА АНГЛІЙСЬКОЇ ФІЛОЛОГІЇ І ПЕРЕКЛАДУ

ЗАТВЕРДЖУЮ
Завідувач кафедри
англійської філології і перекладу

_____Л. Буданова

Модульна контрольна робота № 1 (зразок)
з навчальної дисципліни «CAT-технології у перекладі»
Variant 1.

Translate the following text in Word Fast Anywhere (CAT-tool) under the university license. Create your own TM and Glossary. Make QA check (transcheck and spellcheck) of your translation.

The global object provides variables and functions that are available anywhere. By default, those that are built into the language or the environment.

In a browser it is named window, for Node.js it is global, for other environments it may have another name.

Recently, globalThis was added to the language, as a standardized name for a global object, that should be supported across all environments. It's supported in all major browsers.

We'll use window here, assuming that our environment is a browser. If your script may run in other environments, it's better to use globalThis instead.

In a browser, global functions and variables declared with var (not let/const!) become the property of the global object:

```
var gVar = 5;
alert(window.gVar); // 5 (became a property of the global object)
```

Function declarations have the same effect (statements with function keyword in the main code flow, not function expressions).

Please don't rely on that! This behavior exists for compatibility reasons. Modern scripts use JavaScript modules where such a thing doesn't happen.

That said, using global variables is generally discouraged. There should be as few global variables as possible. The code design where a function gets "input" variables and produces certain "outcome" is clearer, less prone to errors and easier to test than if it uses outer or global variables.