**Лабораторна робота №1**

**// #1**

var test = 'foo';

function testScope() {

var test = 'baz';

console.log(test);

}

testScope();\\На екрані з’являється 'baz', тому що виконується функція.

console.log(test);\\ З’являється 'foo' тому що 'baz' має свій скоуп.

**// #2**

var test = 'foo';

function testScope() {

test = 'baz';

console.log(test);

}

console.log(test); // З’являється 'foo' , тому що функція ще не виконується.

testScope(); // З’являється 'baz', тому що функція виконується і зміній присвоюється нове значення.

console.log(test); // З’являється 'baz' ще раз, тому що в функції змінній присвоїли нове значення.

**// #3**

var test;

test = 'foo';

var test = 'baz';

console.log(test); // З’являється 'baz', тому що ми перевизначили змінну test, закинувши в неї нове значення.

**// #4**

function foo() {

console.log(1);

}

var foo;

foo = function() {

console.log(2);

};

foo(); // З’являється 2, тому що ми перевизначили функцію та надали їй другий зміст.

**// #5**

var test = 'foo';

function foo() {

return test;

}

function baz(callback) {

return callback();

}

baz(foo); // З’являється 'foo' тому що ми вертаємо функцію foo() в функцію baz(callback).

**// #6**

var test = 'foo';

function foo() {

return test;

}

function baz(callback) {

var test = 'baz';

return callback();

}

baz(foo); // З’являється 'foo', тому що перезапис var test = 'baz' працює ну в усій області видимості.

**// #7**

var test = 'foo';

function foo() {

return test;

}

function baz(callback) {

this.test = 'baz';

var binded = callback.bind(this);

return binded();

}

baz(foo); // З’являється 'baz', тому що метод callback.bind(this) створює нову функцію та передає в неї this.test = 'baz'.

**// #8**

var test = 'foo';

var foo = function () {

return test;

};

var baz = function (callback) {

var test = 'baz';

return callback();

};

baz(foo); // З’являється 'foo', тому що в функцію baz = function (callback) передається функція foo = function () і ця ж сама функція не перезаписується.

**// #9**

var test = 'qwerty';

var a = 4;

function foo() {

if (a === 4) {

var test = 'bazinga';

}

console.log(test);

}

foo(); // З’являється 'bazinga', тому що перевірилась змінна a === 4 та перезаписується var test = 'bazinga'.

**// #10**

var x = 5;

(function () {

console.log(x);

var x = 10;

console.log(x);

})(); // З’являється undefined і 10, тому що var x; піднявся вгору в функції, а потім х=10.

**// #11**

(function() {

var a = b = 5;

})();

console.log(); // З’являється 5, тому що зміна а=b=5 може використовуватись в будь якій області, потім виконується функція і одразу виводиться 5, a локальна зміна а b будь-яка.

**// #12**

function test() {

console.log(a); // З’являється undefined, тому що зміна а ще не має значення.

console.log(foo()); // З’являється 2, тому що виконується функція function foo() яка повертає 2.

var a = 1;

function foo() {

return 2;

}

}

test();