1. How many callbacks queues in Node.js and what are they? Give examples to explain how different callbacks are enqueued in different queues.

There are 8 callback queues in Node.js. They are:

* 1. Timer callback queue
     + callbacks are scheduled by setTimeout() and setInterval()
  2. I/O callback queue
     + Execute Input and Output callback like file reading and writing.
  3. Idle/prepare callback queue
     + This queue is only used internally by NodeJS
  4. Poll callback queue
  5. Check callback queue
     + Callback from setImmediate()
  6. Close callback queue
     + Callbacks from the close event like a close socket, close connections.
  7. Next tick callback queue
     + Callbacks coming from process.nextTick()
  8. Promise callback queue
     + Callbacks coming from promise’s resolve and reject

1. What's the output in the code below? Try to understand how it works as we explained in the class

console.log('start');

setTimeout(() => console.log('timeout 1'), 0);

setTimeout(() => {

    console.log('timeout 2')

    process.nextTick(() => console.log('nextTick 3'));

}, 0);

setTimeout(() => console.log('timeout 3'), 0);

new Promise((resolve, reject) => resolve('Hello')).then(() => console.log('Promise...1'));

new Promise((resolve, reject) => resolve('Hello')).then(() => console.log('Promise...2'));

process.nextTick(() =>  console.log('nextTick 1'));

process.nextTick(() =>  console.log('nextTick 2'));

console.log('end');

The output of the code is

start

end

nextTick 1

nextTick 2

Promise…1

Promise…2

timeout 1

timeout 2

nextTick3

timeout 3

1. What's the output of the code below? Try to understand how it works as we explained in the class.

const fs = require('fs');

fs.readFile('hello.txt', () => {

    console.log('readFile....');

});

setTimeout(() => console.log('timeout'), 0);

setImmediate(() => console.log('Immediate'));

This code’s output can vary.

Timeout

Immediate

readFile…

1. What's the output of the code below when running in Node.js? If for the same code running in browser, what's the output? Why is the output different?

var message = 'Hello World';

function logMessage() {

    console.log(this.message);

}

logMessage();

The output of the code in Node.js is ‘undefined’ and in the browser is ‘Hello World’.  
The ‘this’ object in the browser is Windows. In NodeJS, the ‘this’ object is empty because all js files are wrapped with a function object.

1. What will happen when execute the code below? If there's error, what's the error and what cause the error? If no error, what's the output in the console? Assume pattern1.js and app.js are in the same folder

// pattern1.js

module.exports.getName = function () {

    console.log('Josh Edward');

};

// app.js

const getName = require('./pattern1');

getName();

output will be “getName is not a function”.

The problem is function is exported as getName attribute.

To fix this, we can call ‘getName.getName()’ from app.js. Or change module.export = function(){}.

1. What will happen when execute the code below? If there's error, what's the error and what cause the error? If no error, what's the output in the console? Assume pattern1.js and app.js are in the same folder

// pattern1.js

exports.getFirstname = function () {

    console.log('Josh');

};

exports = {

    getLastname: function(){

        console.log('Edward');

    }

}

module.exports = function (){

    console.log('Josh Edward');

}

// app.js

const getName = require('./pattern2');

getName();

This program will print ‘Josh Edward’.

The reason for this, only module.exports will return as function for app.js

1. What will happen when execute the code below? If there's error, what's the error and what cause the error? If no error, what's the output in the console? Assume pattern1.js and app.js are in the same folder

// pattern1.js

exports.getFirstname = function () {

    console.log('Josh');

};

exports = {

    getLastname: function(){

        console.log('Edward');

    }

}

module.exports.getFullname = function (){

    console.log('Josh Edward');

}

// app.js

const {getFullname} = require('./pattern2');

getFullname();

  This program will print ‘Josh Edward’.

The reason for this, only module.exports will return an object with the getFullname attribute which is an function.

Review IIFE, bind, apply, call methods first, what's the value of result in the console?

const result = (function(exports, module){

    exports = module.exports;

    exports.firstname = 'John';

    module.exports.lastname = 'Smith';

    exports = {

        getFullName: function(){

            console.log('John Smith')

        }

    }

    return module.exports;

}).apply(null, [null, {exports: {}}]);

console.log(result);

The result value will be {fistname:’John’,lastname:’Smith’}.