1. Выберите неправильное утверждение:

a) Все API NodeJS асинхронны

b) NodeJS однопоточный, но использует асинхронные функции для параллельного выполнения задач

c) Node.js реализует шаблон проектирования наблюдатель, используя класс EventEmitter.

**d) Все утверждения верны**

2. REPL расшифровывается как:

a) Read Earn Point Learn

b) Read Eval Point Loop

**c) Read Eval Print Loop**

3. Какая команда используется для старта REPL сессии?

**a) node**

b) node start

c) node repl

d) node run

4. Модули npm по-умолчанию устанавливаются

**a) локально**

b) глобально

5. Выберите правильный вариант загрузки встроенного модуля NodeJS:

a) var http = require('./http');

**b) var http = require('http');**

c) var http = require('./node\_modules/http');

6. Выберите неправильный вариант экспорта функции из модуля:

a) exports.sayHi = function () {

console.log('hello')

}

**b) exports = function () {**

**console.log('hello')**

**}**

c) module.exports.sayHi = function () {

console.log('hello')

}

d) module.exports = function () {

console.log('hello')

}

7. Какой вариант может быть использован как аналог \_\_dirname?

**a) process.cwd()**

b) console.cwd()

c) process.currentDir()

d) process.cwd

8. Что будет доступно у obj = require(‘./myModule’), если myModule выглядит так:

var x = 5;

var addX = function (value) {

return value + x;

}

module.exports.x = x;

module.exports.addX = addX;

**a) obj.x**

**b) obj.addX**

c) obj.exports

d) ничего

9. Что из перечисленного является глобальным объектом в NodeJS?

**a) process**

**b) buffer**

c) path

d) url

e) stream

10. Какой метод модуля fs используется для закрытия файла?

a) fs.closeFile(fd, callback);

**b) fs.close(fd, callback);**

c) fs.closePath(fd, callback);

d) ни один из перечисленных

11. Выберите вариант кода, позволяющий получить абсолютный путь к файлу:

a) os.resolve('main.js')

**b) path.resolve('main.js')**

c) fs.resolve('main.js')

12. Что вернет следующий код?

path.join('/sub1', 'sub2', 'sub3/sub4', 'sub', '..');

a) /sub1/sub2/sub3/sub4/sub/../

**b) /sub1/sub2/sub3/sub4/**

c) /sub1sub2/sub3/sub4sub/../

d) ../sub1/sub2/sub3/sub4/sub

13. В каком из вариантов кода длина буфера buf получена правильно?

**a) buf.length**

b) buf.size

c) buf.bytes

d) buf.length()

14. Что используется в NodeJS для установки обработчика события?

**a) eventEmitter.addListener(‘event\_name’);**

**b) eventEmitter.on(‘event\_name’);**

c) eventEmitter.addEventListener(‘event\_name’);

d) eventEmitter.when(‘event\_name’);

15. Выберите правильный вариант создания файла message.txt с текстом Hello World:

a) var fs = require('fs');

fs.write('message.txt', 'Hello World');

b) var fs = require('fs');

fs.writeFile('message.txt', {text:'Hello World'});

**c) var fs = require('fs');**

**fs.writeFile('message.txt', 'Hello World');**

16. Выберите верный вариант кода запуска сервера

a) http.createServer(handler).listen(3000, "127.0.0.1")

b) http.createServer(handler).listen(3000)

**c) оба варианта верны**

d) оба варианта неверны

17. Какой пример кода напечатает сам себя?

a) var fs = require('fs');

fs.readFile(\_\_filename, function (error, data) {

console.log(data);

})

**b) var fs = require('fs');**

**fs.readFile(\_\_filename, { encoding: 'utf-8' }, function (error, data) {**

**console.log(data);**

**})**

**c) var fs = require('fs');**

**fs.readFile(\_\_filename, function (error, data) {**

**console.log(data.toString());**

**})**

d) console.log(\_\_filename);

18. Выберите неправильный вариант обработки GET запроса:

**a) var server = http.createServer();**

**server.on('request', function (req, res) {**

**res.send('Some text');**

**})**

b) server.on('request', function (req, res) {

res.end('Some text');

})

c) server.on('request', function (req, res) {

res.writeHead(200, {'Content-Type' : 'text/plain'});

res.write('Some ');

res.write('text');

res.end();

})

19. Выберите вариант кода, который будет выводить в консоль параметры поисковой строки(query string params) в виде объекта?

**a) server.on('request', function (req, res) {**

**var parsedUrl = url.parse(req.url, true);**

**var params = parsedUrl.query;**

**console.log(params)**

**})**

b) server.on('request', function (req, res) {

var parsedUrl = url.parse(req.url, false);

var params = parsedUrl.query;

console.log(params)

})

c) server.on('request', function (req, res) {

var params = req.url.query;

console.log(params)

})

20. Выберите неправильный вариант создания GET запроса?

var options = {

host: 'localhost',

port: 3000,

method: 'GET',

path: '/',

};

a) var req = http.request(options, function (res) {

console.log('GET request')

});

req.end();

**b) var req = http.request(options, function (res) {**

**console.log('GET request')**

**});**

c) var req = http.get(options, function (res) {

console.log('GET request')

});

21. Выберите правильный вариант создания express-приложения:

**a) var express = require('express');**

**var app = express();**

**app.listen(3000)**

b) var express = require('express');

var app = express;

app.listen(3000)

c) var express = require('express');

var app = express.createServer();

app.listen(3000)

22. Какие функции может выполнять middleware-функция?

a) Выполнение любого кода.

b) Внесение изменений в объекты запросов и ответов

c) Вызов следующей функции промежуточной обработки из стека

d) Завершение цикла “запрос-ответ”

**e) Все перечисленное**

23. Какие аргументы принимает обработчик ошибок в express?

a) request, response, error

b) request, response, next

**c) request, response, next, error**

d) error

24. Что выведет в консоль данный код при отправке GET запроса по пути /test?

var express = require('express');

var app = express();

app.use(function (req, res, next) {

console.log('1');

next();

})

var subapp = express();

subapp.use(function (req, res) {

console.log('2');

})

subapp.get('/test', function () {

console.log('3');

})

app.use('/test', subapp);

app.listen(3000);

a) 1 2

b) 1 2 3

**c) 1 1 2**

d) 3

25. Что выведет в консоль данный код при отправке GET запроса по пути /id?

var express = require('express');

var app = express();

app.get('/:id', function (req, res, next) {

console.log('2');

next();

});

app.param('id', function (req, res, next, id) {

console.log('1');

next();

});

app.listen(3000);

a) 1

**b) 1 2**

c) 2 1

26. Что выведет в консоль данный код при отправке GET запроса по пути /test?

var express = require('express');

var app = express();

app.get('/test', function (req, res, next) {

console.log('1');

var err = new Error('test error');

next(err);

})

app.get('/test', function (req, res, next) {

console.log('2');

next();

})

app.use(function (req, res, next, err) {

console.log('Error')

})

app.listen(3000);

**a) 1 Error**

b) 1 2 Error

c) Error

d) Произойдет ошибка

27. Что выведет в консоль данный код при отправке GET запроса по пути ‘<http://localhost:3000/>’?

var express = require('express');

var app = express();

var router = express.Router();

app.use(router);

router.get('/', function (req, res) {

console.log('2');

})

app.use(function (req, res, next) {

console.log('1');

next();

})

app.listen(3000);

a) 21

**b) 2**

c) 12

d) произойдет ошибка

28. Дополните код для получения данных POST запроса в формате JSON:

var express = require('express');

var app = express();

var bodyParser = require('body-parser');

...ваш код...

app.listen(3000);

a) app.use(bodyParser(json));

app.post('/', function (req, res) {

console.log(req.body)

})

**b) app.use(bodyParser.json());**

**app.post('/', function (req, res) {**

**console.log(req.body)**

**})**

c) app.use(bodyParser);

app.post('/', function (req, res) {

console.log(req.body.json())

})

29. Какой код вернет страницу (используя шаблонизатор ejs)?

**a) app.get('/', function (req, res) {**

**res.render('main.ejs', { title: 'main page'})**

**})**

b) app.get('/', function (req, res) {

res.send('main.ejs', { title: 'main page' })

})

c) app.get('/', function (req, res) {

res.exec('main.ejs', { title: 'main page' })

})

30. Выберите правильный вариант подключения к базе данных:

var mssql = require('mssql');

var config = {

driver: 'tedious',

user: 'demo\_user',

password: '12345',

server: 'localhost',

database: 'demo',

port: 1433,

pool: {

max: 10,

min: 0,

idleTimeoutMillis: 30000

}

}

**a) var connection = new mssql.Connection(config);**

**connection.connect(function(err) {**

**console.log('connected');**

**})**

b) var connection = mssql.Connection(config);

connection.connect(function(err) {

console.log('connected');

})

c) var connection = mssql.createConnection(config);

connection.connect(function(err) {

console.log('connected');

})

31. Выберите правильный вариант запроса к базе данных:

a)

var request = new mssql.Request(connection);

request.stream = true

request.query('SELECT \* FROM items', function(err, rows) {

console.log('request successful');

});

**b) var request = new mssql.Request(connection);**

**request.query('SELECT \* FROM items', function(err, rows) {**

**console.log('request successful');**

**});**

c) var request = new mssql.Query(connection);

request.query('SELECT \* FROM items', function(err, rows) {

console.log('request successful');

});

d) var request = new mssql.Request(connection);

request.execute('SELECT \* FROM items', function(err, rows) {

console.log('request successful');

});

32. Какой метод модуля node-mssql используется для возвращения соединения в пул соединений?

**a) PreparedStatement.unprepare()**

b) Request.unprepare()

c) Connection.release()

d) Connection.end()

33. Какая команда SQL используется для удаления из таблицы записей, удовлетволряющих указанному условию

a) DROP

**b) DELETE**

c) ALTER TABLE

34. Какой метод модуля node-mssql позволяет закрыть соединение ?

a) connection.end()

**b) connection.close()**

c) connection.unprepare()

d) connection.release()

35. Какие аргументы принимает callback-функция метода query node-mssql?

**a) error, rows**

b) fields, results

c) error, rows, affected\_rows

36. Имеется база данных test с таблицей items со следующими полями: id, name, description. Выберите правильный(е) вариант(ы) кода, который вернет элемент таблицы items по указанному id:

**a) var ps = new mssql.PreparedStatement(connection);**

**ps.input('id', mssql.Int);**

**ps.prepare('SELECT \* FROM items WHERE id=@id', function(err) {**

**ps.execute({id: 1}, function(err, rows) {**

**...**

**})**

**})**

b) var ps = new mssql.PreparedStatement(connection);

ps.prepare('SELECT \* FROM items WHERE id=@id', function(err) {

ps.execute({id: 1}, function(err, rows) {

...

})

})

c) var ps = new mssql.PreparedStatement();

ps.prepare('SELECT \* FROM items WHERE id=@id', function(err) {

ps.execute({id: 1}, function(err, rows) {

...

})

})

37. Выберите правильный вариант создания подписанного cookie, доступного только на стороне сервера:

**a) app.use(cookieParser(‘secret’));**

**res.cookie('name', 'value', {signed: true, httpOnly: true})**

b) app.use(cookieParser());

res.cookie('name', 'value', {signed: true, httpOnly: true, secret: ‘secret’})

c) app.use(cookieParser());

res.signedCookie('name', 'value', {secret: ‘secret’})

38. Какой вариант кода выведет в консоль id сессии?

var session = require('express-session');

app.use(session({

secret: 'secret',

saveUninitialized: true,

resave: false

}));

a) app.use(function(req, res) {

var sid = req.sessionID();

console.log(sid);

})

b) app.use(function(req, res) {

var sid = res.session.id;

console.log(sid);

})

**c) app.use(function(req, res) {**

**var sid = req.sessionID;**

**console.log(sid);**

**})**

**d) app.use(function(req, res) {**

**var sid = req.session.id;**

**console.log(sid);**

**})**

39. Выберите правильный вариант сохранения сессии в хранилище(store):

var express = require('express');

var app = express();

var session = require('express-session');

var MSSQLStore = require('connect-mssl')(session);

var options = {

driver: 'tedious',

user: 'demo\_user',

password: '12345',

server: 'localhost',

database: 'sessions,

port: 1433,

pool: {

max: 10,

min: 0,

idleTimeoutMillis: 30000

}

};

**a) var sessionStore = new MSSQLStore(options);**

**app.use(session({**

**secret: 'secret',**

**saveUninitialized: true,**

**resave: true,**

**store: sessionStore**

**}));**

b) var sessionStore = MSSQLStore(options);

app.use(session({

secret: 'secret',

saveUninitialized: true,

resave: true,

store: options,

store: sessionStore

}));

40. Выберите неправильное утверждение:

a) REST – архитектура программного обеспечения, основанная на веб стандартах и использующая протокол HTTP для обмена данными

b) Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола.

c) В архитектуре REST Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL.

**d) все утверждения верны**

e) все утверждения неверны

41. Что из перечисленного не является принипом REST?

a) Клиент-серверная архитектурa

b) Используются стандартные методы HTTP

c) Сервер не хранит состояние

**d) Сервер хранит состояние**

42. Какие форматы данных HTTP запросов допустимы в RESTful API?

**a) JSON**

**b) XML**

c) text/html

d) все из перечисленного

43. Какой HTTP метод используется для обновления данных ресурса?

**a) PUT**

b) POST

c) PATCH

d) GET

44. Socket.io используется:

a) на сервере

b) на клиенте

**c) на сервере и на клиенте**

45. Выберите правильный вариант подключения socket.io к серверу express:

a) var app = require('express');

var server = require('http').Server(app);

var io = require('socket.io')(server);

server.listen(3000);

b) var app = require('express')();

var server = require('http').Server();

var io = require('socket.io')(server);

server.listen(3000);

**c) var app = require('express')();**

**var server = require('http').Server(app);**

**var io = require('socket.io')(server);**

**server.listen(3000);**

46. Какой метод используется в socket.io для генерации события?

a) on

**b) emit**

c) event

d) createEvent

47. Выберите неправильные варианты обработки подключения клиента

**a) io.on('connect', function(socket){  
 console.log('connected')  
}**

**b) io.on('connected', function(socket){  
 console.log('connected')  
}**

c) io.on('connection', function(socket){  
 console.log('connected')  
}

d) все варианты неправильные

e) все варианты правильные

48. Выберите правильный вариант обработки события socket:

a) socket.on('hello from client', function (data) {

console.log(data);

})

b) socket.on('message', function (data) {

console.log(data);

})

c) socket.on('client\_message', function (data) {

console.log(data);

})

**d) все варианты правильные**

e) нет правильного ответа

49. Выберите правильный вариант кода, который отправит событие всем подключенным клиентам:

a) io.emit('some\_event');

b) io.sockets.emit('some\_event');

**c) оба варианта правильны**

d) оба варианта неправильны

50. Выберите правильный вариант создания пространства имен(namespace) в socket.io:

**a) var nsp = io.of('/my-namespace');**

b) var nsp = io.on('/my-namespace');

c) var nsp = socket.of('/my-namespace');