Delft University of Technology

CSE1500 Web & Database Technology

Midterm Exam

Date: 13/12/2018
Time Limit: 120 minutes

Instructions:

- This exam contains 23 pages (including this cover page) and 40 multiple-choice questions. Check to see if any pages are missing.
- All questions are worth 1 point.
- The usage of books, notes, old exams, and other written resources is explicitly FORBIDDEN during the exam. The use of electronic aids such as smartphones, laptops, etcetera, is ALSO NOT allowed.
- There is only one right answer for each question.
- The order of the answers on your answer form is not always A-B-C-D.
- Be sure to fill in all header information on the answer form.
- Some questions refer to source code listed a few pages earlier. **Feel free to disassemble your copy of the exam, so that you can work comfortably.**

Good Luck!

Question 1

Which of the following statements about the WebSocket protocol is TRUE?

A. The WebSocket protocol requires the browser-level `XMLHttpRequest` object to function.

B. The WebSocket protocol relies on long polling to enable bidirectional communication between client and server.

C. Client and server agree to the WebSocket protocol as follows: the client initiates the protocol upgrade by sending a HTTP request with at least two headers: `Connection:Upgrade` and `Upgrade:websocket`.

D. It is sufficient for the client to initiate the protocol upgrade. The server has to accept the upgrade, it cannot reject it.

Question 2

What do persistent connections refer to in the context of HTTP?

A. A single TCP connection is used to send and receive multiple HTTP requests and responses.

B. A single HTTP connection is used to send and receive multiple IP requests and responses.

C. A single TCP connection is used to send and receive a single HTTP request and response.

D. A single HTTP connection is used to send and receive multiple TCP requests and responses.

Question 3

Which of the following statements about Web caches are generally TRUE?

[1] Web caches increase the load on origin servers that host unpopular resources.

[2] Web caches reduce network bottlenecks.

[3] Web caches make use of the `Expires` header field to determine when a fetched resource is no longer valid.

[4] Web caches are particularly beneficial (in terms of reduced latency) for web applications relying on fat URLs to track users.

A. Only [1].

B. Only [2].

C. Only [2] and [3].

D. Only [1] and [4].

Question 4

Which of the following statements about the hypertext transfer protocol (HTTP) is TRUE?

A. HTTP/1.1 is an unreliable protocol, i.e., data may appear out of order and damaged. HTTP/2 is a reliable protocol, i.e., data appears in order and undamaged.

B. An HTTP request has to contain the header field `Expires` to be valid (i.e., accepted and responded to by the server).

C. The `Accept-Encoding` header field is part of the request header; it informs the server which types of encryption the client is able to understand.

D. In 2018, the majority of HTTP messages exchanged over the Internet used HTTP/1.1.

Question 5

Which of the following is NOT an HTTP method?

A. `GET`

B. `PUT`

C. `POST`

D. `REMOVE`

Question 6

Consider the HTTP request message below. Which of the following statements about it is TRUE?

```
GET HTTP/1.1 /
Host: www.tudelft.nl
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:31.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
DNT: 1
Cookie: __utma=1.20923577936111.16111.19805.2;utmcmd=(none);
```

A. This HTTP request is invalid, as the client has to employ the `Set-Cookie` header to send its cookies to the server.

B. This HTTP request is invalid, as the syntax of the first line is incorrect. The correct syntax is the following: `GET / HTTP/1.1`

C. This HTTP request is invalid, as the `User-Agent` header field can only be set by the server.

D. This HTTP request is valid.

Question 7

If an HTTP request message contains the line below, what does this mean?

```
Authorization: Basic dXNlcjpwYXNzd2Q=
```

A. The client informs the server about the encryption types it supports (in this case only one: `Basic`) and its secret encryption key.

B. The server informs the client about the password to use to access the resource.

C. The client attempts to authenticate itself with a server, usually after the server responded with an `Unauthorized` **response to an earlier HTTP request for a particular resource.**

D. `Authorization` is not a valid HTTP header field; the whole line will be ignored by the entity processing the HTTP request.

Question 8

Which of the following statements about URLs is FALSE?

A. The last part of a URL (`#<frag>`) names a piece of a resource and is only used by the client.

B. The `<host>` part of a URL can be a domain name or a numeric IP address.

C. The `<path>` is the local path to the requested resource.

D. If no `<port>` **is provided in a URL, the default port number 0 is added to it.**

Question 9

What does status code `401` refer to?

A. Resource moved permanently.

B. Resource not found.

C. Internal server error.

D. Unauthorized access of resource.

Question 10

What does JavaScript's hoisting principle refer to?

A. Function expressions are always executed before function declarations.

B. Declarations are processed before any code is executed.

C. Function and variable expressions can be used before they are declared.

D. Expressions are processed before any code is executed.

Question 11

After executing the JavaScript snippet below in the browser, what is the output on the Web Console?

```
1  var gameID = 1;
2
3  function gameSetup(p1,p2,t){
4      this.gameID = gameID++;
5      this.player1 = p1;
6      this.player2 = p2;
7      this.time = t;
8      this.score1 = 0;
9      this.score2 = 0;
10
11     this.getStats = function(){
12         return {
13             score1: this.score1,
14             score2: this.score2
15         }
16     }
17 }
18
19 var g1 = new gameSetup("A", "B", "10 seconds");
20 g1.score1 = 10;
21
22 var g2 = g1.getStats();
23 var g3 = g1.getStats;
24 var g4 = g3();
25
26 console.log(g2);
27 console.log(g4);
```

A.
```
Object { score1: 10, score2: 0 }
Object { score1: undefined, score2: undefined }
```

B.
```
Object { score1: 10, score2: 0 }
Object { score1: 10, score2: 0 }
```

C.
```
Object { score1: 10, score2: 0 }
Object { score1: 0, score2: 0 }
```

D.
```
Object { score1: undefined, score2: undefined }
Object { score1: undefined, score2: undefined }
```

Question 12

Which of the following statements about JavaScript functions is FALSE?

A. Functions can be passed as parameters.

B. Functions can be returned from functions.

C. Functions can be assigned to object properties.

D. Functions are not objects.

Question 13

After executing this JavaScript snippet in the browser, what is the Web Console output?

```
function m(fn,ar){
    let result = [];
    for(let i=0; i!=ar.length; i++){
        result[i] = fn(ar[i]);
    }
    return result;
}
console.log( m(function(x){return 2*x;},[1,4,6,7]) );
```

A. `[]`

B. `[ undefined, undefined, undefined, undefined ]`

C. `[ 2, 8, 12, 14 ]`

D. `FuncError: fn is not a valid argument of m`

Question 14

Which of the following statements about JavaScript's prototype-based and module design pattern are TRUE?

[1] In the prototype-based design pattern, objects do not share functions.

[2] The prototype-based design pattern exposes only the necessary members to the public.

[3] In the module design pattern, methods added on the fly to an already created object cannot access emulated private members.

[4] With both design patterns encapsulation is achieved.

A. Only [1].

B. Only [3].

C. Only [2] and [4].

D. Only [1], [2] and [4].

Question 15

Consider these two files, `foo.js` and `bar.js`. What is the console output of `node bar.js`?

```
//foo.js
module.exports = function() {
    return {
        pi: 3.1415,
        one: 1,
        login: "root",
        password: "root"
    }
};
```

```
//bar.js
var constants1 = require('./foo')();
constants1["password"] = "admin";
var constants2 = require('./foo')();

console.log(constants2["password"]);

var constants3 = require('./foo')();
constants2["pi"] = 3;

console.log(constants3["pi"]);
```

A. `TypeError: require(...) is not a function`

B.
```
undefined
undefined
```

C.
```
admin
root
```

D.
```
root
3.1415
```

Question 16

Which of the following statements about middleware components are FALSE?

[1] Middleware components have three parameters.

[2] Middleware components can execute code.

[3] Middleware components can change the request and response objects.

[4] Middleware components can start the request-response cycle.

[5] Middleware components can call any middleware function in the middleware stack.

A. Only [2].

B. Only [3].

C. Only [1] and [3].

D. Only [4] and [5].

Question 17

Which of the following statements correctly describe the roles in the OAuth 2.0 authorization framework?

[1] The resource owner grants access to a protected resource.

[2] The resource server hosts the protected resources, and is capable of accepting and responding to protected resource requests using access tokens.

[3] The client is an application making protected resource requests on behalf of the authorization server.

[4] The authorization server issues access tokens to the resource server after successfully authenticating the client.

A. Only [1] and [2].

B. Only [2] and [3].

C. Only [3] and [4].

D. All statements are correct.

Question 18

Which of the following statements about cookies are TRUE?

[1] Transient cookies remain intact after the browser is closed.

[2] A cookie consists up to seven components, only one of which is required.

[3] Cookies are sent from server to client using the `Cookie` HTTP response header field.

[4] Cookies are deleted from the server through the use of the `Delete-Cookie` HTTP request header field.

A. Only [1].

B. Only [2].

C. Only [1] and [3].

D. Only [2] and [4].

Question 19

Which of the following statements about cookie flags are TRUE?

[1] The `Path` field determines for which paths on the issuing server the cookie is applicable to.

[2] A signed cookie enables the server to issue an encrypted value to the client, that cannot be decrypted by the client.

[3] If a cookie's `HTTPOnly` flag is set, the user cannot view or change the cookie in the browser.

[4] For third-party cookies, setting the `Domain` to the first-party domain is a requirement.

A. Only [1].

B. Only [2].

C. Only [1] and [3].

D. Only [2] and [4].

Question 20

A server-side application uses sessions instead of cookies to track users. What is the most common approach to determine the end of a session?

A. The server makes an HTTP request to the client every x seconds to determine whether the client is still online. If the client is not online, the session ends.

B. The cookie the client sends with the final HTTP request to the application contains a special `Session-Ending` cookie field to indicate the end of the session.

C. The client makes the final HTTP request to the application without returning its session cookie, indicating the end of the session.

D. If more than a fixed number of minutes have passed without another HTTP request from the client, the server ends the session.

Question 21

Consider a web application (e.g. a web store) with many user accounts. Which threat is the web application susceptible to if one can manipulate the URL of an account page to access all accounts?

A. Unvalidated redirect.

B. Reflected XSS.

C. CSRF.

D. Insecure direct object reference.

Question 22

One common defense against CSRF attacks is the use of CSRF tokens. What is the idea behind this defense?

A. Create a secret identifier (the CSRF token) on the client and share it with the server through the use of the HTTP request header field `SET-CSRF`. The server stores the token after the first request from the client and only handles all subsequent requests if they contain the same token.

B. An authorization server creates a secret identifier (the CSRF token) that is sent to both the client and that server that want to establish a secure session. Both the client and server check whether the received HTTP message contains the correct token. If not, the HTTP message is ignored.

C. Create a secret identifier (the CSRF token) on the server and send it to the client in a transient cookie with HTTPOnly and Secure flags set. Any HTTP request from the client that does not return the cookie to the server is ignored.

D. Create a secret identifier (the CSRF token) on the server, share it with the client in a hidden form field and require the client to return it when making a request. The server only handles the request if the provided token is the same as the one it sent to the client earlier.

Question 23

Which of the following techniques is not suitable for an attacker to attempt in order to inject malicious data into a web application?

A. Manipulation of hidden form fields.

B. Manipulation of data-* attributes.

C. Manipulation of cookies.

D. Manipulation of HTTP request headers.

The following questions relate to the **We Asked 100 People ...** (**WA100P**) web application, available at `http://145.94.164.39:3000/`. The app is a simple two-player game, consisting of a splash screen and a game screen (Figure 1).

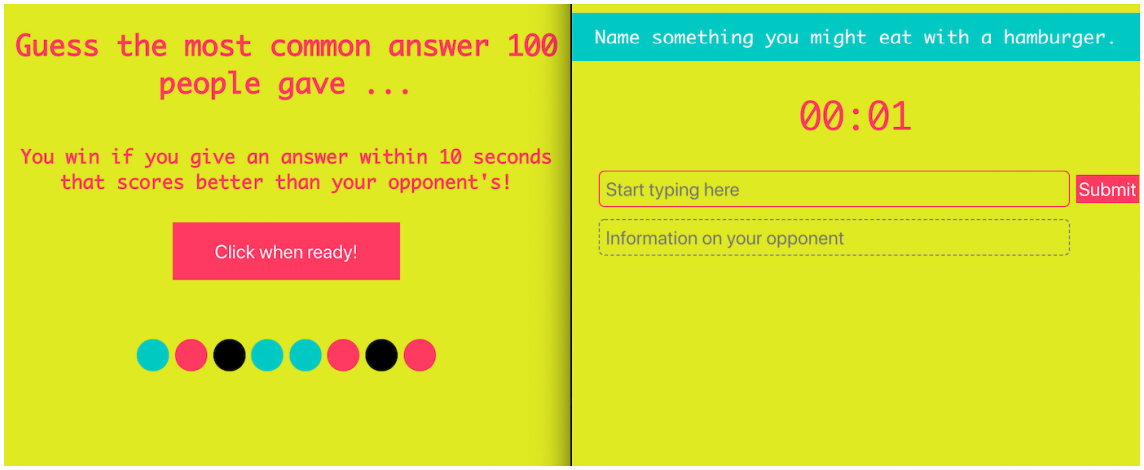

Figure 1: Renderings of the splash (left) and game (right) screen.

- splash: this is the entry page of the game. It shows the game's rule (*Guess the most common answer 100 people gave. You win if you give an answer within 10 seconds that scores better than your opponent's!*) and contains a button to start the game (*Click when ready!*). This click leads to the game screen.
- game: the player waits for an opponent (as soon as two players pressed the *Click when ready!* button, they are paired) and as soon as two players are paired in a game the question appears at the top of the game screen. The timer starts (10 seconds). The player now has 10 seconds to type an answer and submit it. The player then receives her answer's score (e.g. 22 people answered *French fries*) as well as her opponent's. If

a player does not manage to submit an answer within the time limit, the end score is −1. The player achieving a higher score won. In order to play another game, the player has to reload the game screen.

Question 24

The **WA100P** app developers decide to make the splash screen more exciting by adding a simple animation: the color of the eight "dots" below the *Click when ready!* button should randomly change their color once a second. In order to achieve this, they write the following piece of JavaScript (stored in `animations.js`):

```
var colors = ['#ec576b', '#4ec4c1', '#000000'];

for (var i = 1; i <= 8; i++) {
    //repeatedly calls a function with a fixed time delay between each call
    //(delay provided in milliseconds)
    setInterval(function() {

        //Math.floor: returns the largest integer <= a given number.
        //Math.random: returns a float between 0 (inclusive) and 1 (exclusive)
        let color = Math.floor(Math.random() * 3);

        //changing the CSS background-color property of an element
        document.getElementById("b" + i).style.backgroundColor = colors[color];
    }, 1000);
}
```

The splash screen's HTML looks as follows:

```
<!DOCTYPE html>
<html>
    <head>
        <title>We asked 100 people ...</title>
        <link rel="stylesheet" href="stylesheets/splash.css">
    </head>
    <body>
        <main>
                <h1>Guess the most common answer 100 people gave ...</h1>
                <h2>You win if you give an answer
                within 10 seconds that scores better than your opponent's!</h2>

                <form action="/play" method="get">
                        <button type="submit">Click when ready!</button>
                </form>
        </main>
        <div id="dots">
            <div id="b1" class="dot"></div>
            <div id="b2" class="dot"></div>
            <div id="b3" class="dot"></div>
            <div id="b4" class="dot"></div>
            <div id="b5" class="dot"></div>
            <div id="b6" class="dot"></div>
            <div id="b7" class="dot"></div>
            <div id="b8" class="dot"></div>
        </div>
        <script src="javascripts/animations.js"></script>
    </body>
</html>
```

What is the result of employing this piece of JavaScript?

A. No animation is visible, the browser's Web Console outputs `TypeError:document.getElementById("b9") is null.`

B. No animation is visible, the browser's Web Console outputs `FuncError:setInterval is expecting a callback function as argument.`

C. One second after the splash screen loads, all dots will turn black (color code `#000000`) and remain black.

D. The code works as intended: the eight dots on the splash screen change their color randomly once a second.

Question 25

One of the developers is preparing the game for the next version — it should support a variable number of players (instead of two), enable a group of players to play multiple matches (instead of just one) and allow for a minimum threshold the winning player has to cross in terms of final score. The developer has heard about the module pattern as a way to make code maintenance easier in the long term and goes about implementing a configuration object that, once the configurations are set, can only be read from. Which of the following code snippets shows a valid module pattern for this problem?

A.

```
var configModule = function(p=4, s=10, m=2){

    var numPlayers = p;
    var winningScore = s;
    var numMatches = m;

    return {
        var myObj = {};
        myObj.getNumPlayers = function(){
            return numPlayers;
        };
        myObj.getNumMatches = function(){
            return numMatches;
        }
        myObj.getWinningScore = function(){
            return winningScore;
        };
    }
};

console.log(configModule.getNumPlayers());//should print "4"
console.log(configModule.numPlayers);//should print "undefined"
```

B.

```
var configModule = function(p, s, m){

    var numPlayers = p;
    var winningScore = s;
    var numMatches = m;

    return {
        this.getNumPlayers: function(){
            return numPlayers
        },
        this.getNumMatches: function(){
            return numMatches;
        },
        this.getWinningScore: function(){
            return winningScore
        }
    }
}(4,10,2);

console.log(configModule.getNumPlayers());//should print "4"
console.log(configModule.numPlayers);//should print "undefined"
```

C.

```
var configModule = function(p=4, s=10, m=2){

    var numPlayers = p;
    var winningScore = s;
    var numMatches = m;

    function getNumPlayers(){
        return numPlayers;
    }

    function getWinningScore(){
        return winningScore;
    }

    var myObj = {};
    myObj.getNumPlayers = getNumPlayers;
    myObj.getNumMatches = function(){
        return numMatches;
    }
    myObj.getWinningScore = getWinningScore;
    return myObj;
};

console.log(configModule.getNumPlayers());//should print "4"
console.log(configModule.numPlayers);//should print "undefined"
```

D.

```
var configModule = function(p, s, m){

    var numPlayers = p;
    var winningScore = s;
    var numMatches = m;

    function getNumPlayers(){
        return numPlayers;
    }

    function getWinningScore(){
        return winningScore;
    }

    var myObj = {};
    myObj.getNumPlayers = getNumPlayers;
    myObj.getNumMatches = function(){
        return numMatches;
    }
    myObj.getWinningScore = getWinningScore;
    return myObj;
}(4,10,2);

console.log(configModule.getNumPlayers());//should print "4"
console.log(configModule.numPlayers);//should print "undefined"
```

Question 26

In this next version, the multiple matches should be played on the same game screen. To make clear to the player that the different text input elements refer to different questions, the development team decides to use different sounds (played for each typed letter) as an auditory help. In order to test the usability of the idea, a quick prototype is implemented by a developer. Unfortunately, no version control was used and our developer accidentally deleted a portion of the code again (line 9).

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Prototype: multiple input forms</title>
5          <script>
6            window.onload = function() {
7              let answers = document.getElementById("answers");
8              for(let i=0; i<3; i++){
9                  ?????
10             }
11           };
12
13           let audios = [new Audio("http://www.soundjay.com/button/beep-07.wav"),
14             new Audio("http://www.soundjay.com/button/beep-06.wav"),
15             new Audio("http://www.soundjay.com/button/beep-05.wav")];
16
17           function playSound() {
18             //substr() returns portion of the str.,
19             //starting at the specified index
20             let index = this.id.substr(1);
21             audios[index].play();
22           }
23         </script>
24     </head>
25
26     <body>
27         <section id="answers">
28             <button type="submit" id="submit">Submit</button>
29         </section>
30     </body>
31 </html>
```

What should the missing code look like to lead to the desired functionality?

A.

```
1  let input = document.createElement('input');
2  input.type = "text";
3  input.id = "a"+i;
4  input.placeholder = "Start-typing";
5  answers.appendChild(input);
6  input.onkeypress = playSound;
```

B.

```
1  let input = document.appendElement('input');
2  input.type = "text";
3  input.id = "a"+i;
4  input.placeholder = "Start-typing";
5  input.onkeypress = playSound();
```

C.

```
1  let input = document.createElement('input');
2  input.type = "text";
3  input.id = "a"+i;
4  input.placeholder = "Start-typing";
5  answers.append(input);
6  input.onkeypress = function(){
7      playSound: playSound(this)
8  };
```

D.

```
let input = document.appendElement('input');
input.type = "text";
input.id = "a"+i;
input.placeholder = "Start-typing";
input.onkeypress = playSound(this);
```

Question 27

Consider the HTML code from Question 24 again. One of the developers decides to change the value of the method attribute in line 13 from `get` to `post`.

Now, the application no longer works: a click on the button no longer starts the game. The developer pulls up the relevant server-side script:

```
1  var express = require("express");
2  var http = require("http");
3  var websocket = require("ws");
4
5  var app = express();
6  app.use(express.static(__dirname + "/public"));
7
8  app.get("/play", function(req, res) {
9      res.sendFile("game.html", {root: "./public"});
10 });
11
12 app.get("*", function (req, res) {
13     res.sendFile("splash.html", {root: "./public"});
14 });
15
16 /* ... */
17 server.listen(3000);
```

What change in the server-side code will make this code work again?

A. No matter the server-side code changes, the code no longer works as `post` is not a valid setting of the method attribute.

B. In line 5, `var app = express()` has to be replaced by `var app = express({method: "post"})` .

C. In line 8, `app.get` has to be replaced by `app.post` .

D. In line 12, `app.get` has to be replaced by `app.use` .

Question 28

In order to attract more players, our developer team decides to show the most important game statistics on the splash screen: the number of unique players and the total number of matches played. These two statistics are maintained on the server-side and should be rendered on the splash screen. Our developers argue about two options: the use of Ajax or the use of templates (in particular EJS) for this purpose. Which option should they choose and why?

A. Ajax, because in contrast to EJS, no additional JavaScript libraries have to be loaded on the client-side.

B. Ajax, because it requires fewer HTTP requests to arrive at the final rendered splash screen than the use of EJS.

C. EJS, because it provides more secure transport options than Ajax.

D. EJS, because more older browsers will render the splash screen correctly compared to the Ajax option.

Question 29

The game is a success (despite all the bugs the developers battle) and another developer is added to the team. The developer starts digging through the server-side code and is

particularly intrigued by the `game.js` script, which takes care of the game elements and game logic:

```
/* every game has two players, identified by their WebSocket */
var game = function(id) {
    this.id = id;
    this.playerA = null;
    this.playerB = null;
    this.question = null;
    this.answerA = null;
    this.answerB = null;
    this.answers = {};
};

game.prototype.setQuestion = function() {
    let me = this;
    require("fs").readdir("./questions", function(err, files){
        let len = files.length;
        let randIndex = Math.floor(len * Math.random());
        let randomFile = files[randIndex];
        require("fs").readFile("./questions/"+randomFile, function(err, content){
            let lines = content.toString().split("\n");
            me.question = lines[0];
            let counter = 0;
            lines.forEach(function(line){
                if(counter++ > 0){
                    me.answers[line.substring(3)] = line.substring(0,2);
                }
            });
        });
    });
};

game.prototype.scoreAnswer = function(a){
    if( this.answers[a] == undefined){
        return -1;
    }
    return this.answers[a];
};

game.prototype.hasTwoConnectedPlayers = function() {
    return (this.playerA != null && this.playerB != null);
};

game.prototype.addPlayer = function(p) {

    if (this.playerA == null) {
        this.playerA = p;
        return "A";
    }
    else {
        this.playerB = p;
        return "B";
    }
};

module.exports = game;
```

How many callbacks are set up in this code?

A. 0

B. 2

C. 3

D. 4

Question 30

Consider the `game.js` script of Question 29 again. Our developer tries to figure out the format of the `question` files that hold the questions/answers/scores within the `questions` folder. Which of the following example question files has the correct format?

A.
```
Name a reason a person might wake up at 2 in the morning.

12Hungry
16Bad dream
```

B.
```
Name a reason a person might wake up at 2 in the morning.

Hungry 12
Bad dream 16
```

C.
```
Name a reason a person might wake up at 2 in the morning.
Hungry 12
Bad dream 16
```

D.
```
Name a reason a person might wake up at 2 in the morning.
12 Hungry
16 Bad dream
```

Question 31

Consider the `game.js` script of Question 29 again. Our developer is not happy with the `setQuestion` function (it looks messy!) and wants to replace it with the following piece of code:

```
1  game.prototype.setQuestion = function() {
2      let me = this;
3      let files = require("fs").readdirSync("./questions");
4      let len = files.length;
5      let randIndex = Math.floor(len * Math.random());
6      let randomFile = files[randIndex];
7
8      let content = require("fs").readFileSync("./questions/"+randomFile);
9      let lines = content.toString().split("\n");
10     me.question = lines[0];
11     let counter = 0;
12
13     lines.forEach(function(line){
14         if(counter++ > 0){
15             me.answers[line.substring(3)] = line.substring(0,2);
16         }
17     });
18 };
```

What will be the effect of this code snippet?

A. The server will continue working as before, players will have exactly the same game experience as before.

B. The server will continue working as before, though whenever a new game object is initialized the server will react more slowly to incoming HTTP requests.

C. The server is no longer able to serve questions to players: at the end of this code snippet, both the question and `answers` property of the `game` object will be `undefined`.

D. The server is no longer able to serve questions to players: the use of `*Sync` methods requires `this` to be returned from `setQuestion`.

Question 32

One of the senior developers in the team is wary of the junior ones screwing around with

the question bank (which here is the folder holding all the questions). The senior decides to set up a little script to be notified of any changes made to the folder holding the game's questions:

```
const fs = require('fs');
const net = require('net');

//command line arguments: file to watch and port number
const folder = process.argv[2];
const port = process.argv[3];

var server = net.createServer(function (connection) {

    connection.write("Watching " + folder +" for changes\n");

    var watcher = fs.watch(folder, function () {
        connection.write("Folder " + folder + " has changed: " + Date.now() + "\n");

        watcher.close();
    });
});

server.listen(port, function () {
    console.log("Keeping watch on the question bank ...");
});
```

The server-side script (stored in `watching.js`) is started from the same directory as the folder `questions` is in with the following command: `node watching.js questions 3000`. The senior goes home, opens a terminal and connects: `telnet 145.94.164.39 3000`. What will the senior observe now?

A. The senior will not be notified of any changes that are happening to the questions folder.

B. The first time a change is made to the questions folder, the senior is notified. Of any subsequent changes the senior is not notified.

C. The senior will be notified of all changes made to the questions folder.

D. The senior will be notified when files are added/deleted but not of changes made to the *contents* of files.

Question 33

One of the most requested features of the game is authentication: players want to save their progress and achievements. Our developer team considers the use of third-party authentication, and in particular the OAuth 2.0 framework. As TU Delft happens to have an OAuth server, the developers want to make use of it to obtain authentication for their players from TU Delft. What role will **WA100P** play in this scenario?

A. Resource owner

B. Resource server

C. Client

D. Access token server

Question 34

Our newly-added developer from Question 29 is back. The developer has been assigned to review the changes made by another recently appointed developer to `app.js` — the main application file. It now looks as follows:

```
1  /* Node.js/Expression application setup lines
2   * (removed for brevity)
3   */
4  var websocket = require("ws");
5  /*
6   * app is our Express object
7   */
8  let server = http.createServer(app);
9  const wss = new websocket.Server({ server });
10
11 let websockets = {};//key: websocket id, value: game object
12
13 let gameCounter = 0;
14
15 //Question 19 contains the code of game.js
16 var currentGame = new game(gameCounter++);
17 currentGame.setQuestion();
18
19 var connectionID = 0;//unique ID per websocket
20
21 wss.on("connection", function connection(ws) {
22
23     let con = ws;
24     con.id = connectionID++;
25
26     //player A (joined the game first) and player B (joined as second)
27     let playerType = currentGame.addPlayer(con);
28     websockets[con.id] = currentGame;
29
30     if(playerType == "B"){
31         //msg looks as follows {type: "QUESTION", data: null}
32         let msg = messages.QUESTION;
33         msg.data = currentGame.question;
34         con.send(JSON.stringify(msg));
35         currentGame.playerA.send(JSON.stringify(msg));
36
37         currentGame = new game(gameCounter++);
38     }
39
40     con.on("message", function(message) {
41
42         let inMsg = JSON.parse(message);
43         let gameObj = websockets[con.id];
44
45         //inMsg looks as follows {type: "MYWORD", data: [answer to score]};
46         if(inMsg.type == "MYWORD"){
47             let score = gameObj.scoreAnswer(inMsg.data);
48             //outMsg1: {type: "MYSCORE", data: null}
49             let outMsg1 = messages.MYSCORE;
50             outMsg1.data = score;
51             con.send(JSON.stringify(outMsg1));
52
53             //outMsg2: {type: "OPWORDSCORE", data: null}
54             let outMsg2 = messages.OPWORDSCORE;
55             outMsg2.data = inMsg.data + " ("+score+")";
56             if(gameObj.playerA !== con) {
57                 gameObj.playerA.send(JSON.stringify(outMsg2));
58             }
59             else {
60                 gameObj.playerB.send(JSON.stringify(outMsg2));
61             }
62         }
63     });
64 });
65 server.listen(3000);
```

What will our developer say about the implementation of players' communication via Web-

Sockets?

A. The code will not work as envisioned, as `message` (line 40) is not actually an event defined by the WebSocket object `ws`.

B. The code will work as envisioned, the WebSocket communication is set up correctly.

C. The code will not work as envisioned, as a JSON string is sent across the network (lines 34, 51, 57, 60) instead of a JavaScript object.

D. The code will not work as envisioned, as after the first two players have clicked on the splash screen's *Click when ready!* button, no additional game objects are created.

Question 35

Consider the HTML of the splash screen in Question 24 once more. Its style is determined by `splash.css`:

```
1  @import 'style.css';
2
3  h1, h2 {
4      margin-left: auto;
5      margin-right: auto;
6      margin-top: 20px;
7      padding: 20px;
8      color:var(--pink);
9      text-align: center;
10 }
11
12 h1 { font-size: 300%; }
13 h2 { font-size: 200%; }
14
15 button {
16     background-color: var(--pink);
17     color: #fff;
18     font-size: 200%;
19     border: none;
20     padding: 30px;
21     width: 40%;
22     margin-left: 30%;
23     margin-right: 30%
24 }
25
26 button:hover {
27     background-color:var(--aqua);
28     transition: background-color 2s ease;
29     text-decoration: underline overline;
30 }
31
32 #dots {
33     text-align: center;
34     padding-top: 100px;
35 }
36
37 .dot {
38     height: 55px;
39     width: 55px;
40     border-radius: 50%;
41     display: inline-block;
42 }
43
44 #b1, #b4, #b5 { background-color: var(--aqua); }
45
46 #b2, #b3, #b6, #b7, #b8 { background-color: var(--pink); }
```

which in turn imports `style.css`:

```
:root {
  --lime: #e5e338;
  --aqua: #4ec4c1;
  --pink: #ec576b;
  --black: #000;
}
html, body, main, section {
  margin: 0px;
  padding: 0px;
}
body {
  background-color: var(--lime);
  font-family: monaco,Bitstream Vera Sans Mono,monospace;
}
```

What happens to the rendering of the splash screen (depicted in Figure 1) if the following lines would be added to the end of `splash.css`:

```
main:not(.dot) {
    background-color: white;
}
```

A.

B.

C.

D.

Question 36

Consider `splash.css` from Question 35 once more. What would happen to the rendering of the splash screen (depicted in Figure 1) if the following lines were added to the end of `splash.css`:

```
main h1 {
    background-color: black;
}

main,h2{
    background-color: white;
}
```

A. 

B. 

C. 

D. 

Question 37

Consider `splash.css` from Question 35 a last time. What would happen to the rendering of the splash screen (Figure 1) if the following lines were added to the end of `splash.css`:

```
.dot {
    position: absolute;
}
```

A. A single dot remains visible under the *Click when ready!* button.

B. The dots remain in their position as shown in Figure 1.

C. All dots disappear; they are no longer visible below the *Click when ready!* button.

D. All dots will appear at the top left corner of the viewport.

Question 38

One of the developers has an idea to lighten up the game screen (Figure 1): the text box showing *Information on your opponent* should be blinking in order to highlight the fact that information is being updated here. Which CSS snippet achieves this?

A.

```
input[type=text]:-moz-read-only {
    color: grey;
    border: 2px dashed grey;
    animation: blinking 2s ease infinite;
}

input[type=text]:read-only {
    color: grey;
    border: 2px dashed grey;
    animation: blinking 2s ease infinite;
}

@keyframes blinking {
    0% {
        visibility: display;
    }
    50%, 100% {
      visibility: hidden;
    }
}
```

B.

```
input[type=text]:-moz-read-only {
    color: grey;
    border: 2px dashed grey;
    transition: blinking 2s infinite;
}

input[type=text]:read-only {
    color: grey;
    border: 2px dashed grey;
    transition: blinking 2s infinite;
}

@transition blinking {
    from {
        visibility: display;
    }
    50%, to {
      visibility: hidden;
    }
}
```

C.

```
input[type=text]:-moz-read-only {
    color: grey;
    border: 2px dashed grey;
    animation: 2s ease infinite;
    @keyframes blinking {
        0% {
            visibility: display;
        }
        50%, 100% {
          visibility: hidden;
        }
    }
}

input[type=text]:read-only {
    color: grey;
    border: 2px dashed grey;
    animation: 2s ease infinite;
    @keyframes blinking {
        0% {
            visibility: display;
        }
        50%, 100% {
          visibility: hidden;
        }
    }
}
```

D. None of the above code snippets achieves this goal.

Question 39

Having decided on an authentication scheme to establish "game sessions", our developer team now needs to decide on how best to make use of sessions. Which of the following statements about sessions is TRUE?

A. Sessions are more secure than cookies, as all information about the session resides on the client and none on the server.

B. As sessions do not rely on outdated cookie technology, but instead on OAuth 2.0 key/value pairs, they can only be established with modern browsers acting as clients.

C. Sessions are only secure if the session identifiers are unique and unguessable.

D. Sessions are a defense against third-party cookies: once a session is established between client and server, third-party cookies are no longer accepted by the client during that time.

Question 40

Our developer team wants to implement a special Christmas treat: players accessing the splash screen (`christmas-splash.html`) on particular URL routes will receive its Christmas edition:

To achieve this, the following route string pattern is used:

```
app.get("/*ho(ho)?(ho)?", function (req, res) {
    res.sendFile("christmas-splash.html", {root: "./public"});
});
```

How many of the following URL requests will be served the Christmas edition splash screen?

`http://localhost:3000/christmas`
`http://localhost:3000/hohohoho`
`http://localhost:3000/christmas-hoho`
`http://localhost:3000/kerstmishohohoho`
`http://localhost:3000/hohohooo`

The printed version of the exam contained a mistake in the route: it said `splash.html` instead of `christmas-splash.html`.

A. 1

B. 2

C. 3

D. 4