Delft University of Technology

CSE1500 Web & Database Technology

Final Exam

Date: 27/01/2020
Time Limit: 120 minutes

Instructions:

- This exam contains 22 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

Consider the following HTTP header:

```
Host: mytimetable.tudelft.nl
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:70.0)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Cookie: GUEST_LANGUAGE_ID=en_US; JSESSIONID=3C51754346465396565643
Cache-Control: max-age=0
```

Which of the following statements about it is TRUE?

A. It is a valid HTTP request header.

B. It is a valid HTTP response header.

C. It is an invalid HTTP request header, as the `Set-Cookie` attribute is missing.

D. It is an invalid HTTP response header, as the `Content` attribute is missing.

Question 2

Which of the following statements about HTTP/1.1 is TRUE?

A. HTTP/1.1 is a binary protocol.

B. HTTP/1.1 does not allow the reuse of the same open connection for several request-response cycles.

C. HTTP/1.1 supports bidirectional communication between client and server.

D. In HTTP/1.1 the client always initiates the conversation with the server.

Question 3

What does the header field `Cache-Control:max-age=100` mean if it is sent in an HTTP response? You can assume that the HTTP response was sent today.

A. The web cache through which the HTTP response passes, caches the resource sent in the HTTP response. For the next 100 minutes every HTTP request that targets that resource and reaches the cache is served the cached resource.

B. The web cache the HTTP response passes through should not cache the response as `max-age=100` indicates that the response is valid 100 seconds from the start of the Unix epoch (i.e., 00:00:00 UTC on 1 January 1970).

C. The web cache ignores this header field completely as `Cache-Control` is only valid when the `Expires` header field is present in the HTTP header as well.

D. The web cache through which the HTTP response passes, caches the resource sent in the HTTP response. If an HTTP request for the cached resource arrives at the cache more than two minutes after the response was cached, the web cache will revalidate the resource by querying the origin server

Question 4

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

A. The WebSocket protocol enables peer-to-peer communication between two browsers. Or in other words, two clients can connect directly to each other, without the need for a server.

B. Clients can initiate WebSocket connections and receive WebSocket connections.

C. Servers can initiate WebSocket connections with clients by sending a HTTP response with a `Connection:Upgrade` header field.

D. Clients can initiate WebSocket connections with servers by sending a HTTP request with a `Connection:Upgrade` header field.

Question 5

Which of the following statements about the HTTP methods `POST` and `PUT` is TRUE?

A. `POST` saves the HTTP request on the server at the path specified in the request.

B. `PUT` sends data from server to client; the client determines how to handle the data.

C. A `PUT` method request can lead to the creation of a new resource on the server.

D. A `POST` method response can lead to the creation of a new resource on the client.

Question 6

Consider the telnet exchange on the terminal. Which of the following statements about it is TRUE?

```
bash-3.2$ telnet tudelft.nl 80
Trying 131.180.77.82.256.0 ...
Connected to tudelft.nl.
Escape character is '^]'.
HEAD / HTTP/1.1
host:tudelft.nl

HTTP/1.1 301 Moved Permanently
Date: Wed, 22 Jan 2020 09:14:37 GMT
Server: Apache/2.4.6 (CentOS)
Location: https://www.tudelft.nl/
Content-Type: text/html; charset=iso-8859-1

Connection closed by foreign host.
```

A. The IP address shown here is not a valid IPv4 address.

B. The Domain Name System server resolution step is missing in this telnet exchange.

C. The `host` header field is invalid, as it does not contain the port number.

D. The `Content-Type` header field is invalid as it should contain the encoding (such as `gzip, deflate`) of the content.

Question 7

Which of the following statements about URLs is TRUE?

A. In the browser, we typically do not enter a port number when entering a URL into the address bar because the browser has a priority list of common port numbers and tries the URL with every one of those port numbers until a HTTP response is sent.

B. The `<host>` part of the URL can be omitted when making an HTTP request that does not contain a content body.

C. The `<frag>` part of the URL cannot be omitted when making an HTTP request over a persistent TCP connection.

D. The `path` part of the URL does not have to refer to a specific file on the server.

Question 8

Which of the following statements about HTML `<form>` elements is FALSE?

A. The `action` attribute defines where the data gets sent. Its value must be a valid relative or absolute URL.

B. If a form is sent using the `POST` method, the data is appended to the body of the HTTP request.

C. When the form is submitted using the `GET` method, the data is appended to the URL.

D. The `action` attribute defines how the form is accessed on the client: either as a HTTP POST request or an HTTP GET request.

Question 9

Which of the following statements about the `position` property are true?

[1] If an element has the setting `position:relative`, the element's position is adjusted on the fly, other elements' positions are not affected.

[2] If an element has the setting `position:absolute`, the element is taken out of the normal element flow and no space is reserved for it.

[3] If an element has the setting `position:fixed`, the element's position is adjusted on the fly, other elements' positions are not affected.

[4] If an element has the setting `position:left`, the element is taken out of the normal element flow and attached to the left-hand side of the viewport.

A. Only 1 and 2.

B. Only 2 and 4.

C. Only 1, 3 and 4.

D. All statements are correct.

Question 10

Which of the following statements about Ajax are TRUE?

[1] Ajax is a JavaScript mechanism that enables the dynamic loading of content without having to reload the page manually.

[2] Ajax revolves around the use of the XMLHttpRequest object to communicate between client and servers.

[3] Using Ajax, servers can push data to clients.

[4] Ajax reduces the HTTP request/response cycle overhead as no longer a complete HTTP message is required for the communication between client and server.

A. Only 1 and 2.

B. Only 3 and 4.

C. All four statements are true.

D. All four statements are false.

Question 11

Which of the following statements about Node.js' middleware components is FALSE?

A. Middleware components can change the request and response objects.

B. Middleware components can end the request-response cycle.

C. Middleware components can execute code.

D. Middleware components can call any arbitrary middleware function in the middleware stack.

Question 12

What will be the result of executing this piece of JavaScript in the browser?

```
function printTopN(n){
    delimNeg = "------";
    var delim;
    let s = "LEADERBOARD\n";
    return function(leaderboardEntries){
        for(let i=0; i<n; i++){
            s += (i+1);
            s += ") ";
            s += leaderboardEntries[i].getFirstName();
            s += "\n";
        }
        return s;
    }
}

var delimPos = "+++++++";

const leaderboard = ["Daisy Duck", "Goofy", " Donald Duck", "Dagobert ", "Mickey"];

String.prototype.getFirstName = function(){
    //includes(searchString) determines whether
    //the calling string contains searchString.
    if(this.includes(" ")){
        return this.substring(0, this.indexOf(" "));
    }
    return this;
}

console.log( printTopN(3)(leaderboard) );
```

A. LEADERBOARD
1) Daisy
2) Goofy
3)

B. `TypeError: leaderboardEntries[i].getFirstName is not a function`

C. `ReferenceError: s is not defined`

D. `function printTopN()`

Question 13

Consider the JavaScript code of **the previous question** one more time. How many variables have block scope?

A. 1

B. 2

C. 3

D. 4

Question 14

What will be the result of executing the following piece of JavaScript in the browser?

```
var firstName = "Daisy";
var lastName = "Duck";
var age = "unknown";


function Person(f,l,a) {
    this.firstName = f;
    this.lastName = l;
    this.age = a;

    this["description"] = function () {
        return this.firstName+" "+this.lastName+" (age: "+this.age+")";
    };
};

var donald = new Person("Donald", "Duck", "unknown");
var description = donald.description;

console.log( description() );
console.log( donald.description() );

var dagobert = Person("Dagobert", "Duck", "oooold");

console.log( description() );
```

A. `Donald Duck (age: unknown)`
`Donald Duck (age: unknown)`
`Dagobert Duck (age: oooold)`

B. `Daisy Duck (age: unknown)`
`Donald Duck (age: unknown)`
`Dagobert Duck (age: oooold)`

C. `Daisy Duck (age: unknown)`
`Daisy Duck (age: unknown)`
`Daisy Duck (age: unknown)`

D. `ReferenceError: description is undefined.`

Question 15

Which of the following statements about JavaScript's prototype chaining principle is FALSE?

A. Each JavaScript object has a link to at least two prototype objects.

B. When trying to access a property of an object that the object does not have, the prototype chain is followed until a property with a matching name is found or the end of the chain is reached.

C. Objects whose prototypes have been changed (e.g. a property was added or altered) after the objects' creation, have access to those changes too.

D. If a property is defined as property of an object and as property of all the objects in its prototype chain, the object's property definition takes precedence.

Question 16

Given the two files `bar.js` and `foo.js`, what is the output on the console after running `node foo.js`?

```
//bar.js
module.exports = function(){
     return [
         {
              name: "Daisy",
              age: "unknown",
              location: "Duckburg"
         },
         {
              name: "Mickey",
              age: "unknown",
              location: "Mouseton"
         }
     ];
 }
```

```
//foo.js
var people1 = require('./bar');
console.log(people1()[1].name);

var people2 = require('./bar');
people2()[1].name = "Goofy";
people1()[1].location = "Columbus, Ohio";

console.log(people2()[1].name);
console.log(people1()[1].name);
```

A. Mickey
Mickey
Mickey

B. Mickey
Mickey
Goofy

C. Mickey
Goofy
Goofy

D. Daisy
Goofy
Goofy

Question 17

What is the output of the following piece of JavaScript?

```
function func(x) {
  var y = x + 2;
  z = 20;
  return function(y) {
    var z = y * 3;
    return function(z) {
      return x + y + z;
    };
  };
}
console.log(func(3)(4)("z"));
```

A. 7z

B. 34z

C. 18

D. None of the above.

Question 18

Which of the options show the correct order of events for third-party authentication?

A. Authorization Server sends access token to the Client.
B. Client accesses private resources from the Resource Server.
C. Resource Owner requests private resources from the Client.
D. Client sends private resources to the Resource Owner.
E. Client redirects the request to third-party Authentication Service.
F. Resource Owner authenticates herself with Authorization Server.

A. Order: C, E, F, A, B, D

B. Order: B, C, F, D, A, E

C. Order: E, F, A, B, C, D

D. None of the above orders is correct.

This question was not well-phrased. Everyone got the point for this question.

Question 19

Consider the server-side code below, running at `mysite.nl` on port 3000.

```
1  var express = require("express");
2  var http = require("http");
3  var credentials = require("./credentials");
4  var cookies = require("cookie-parser");
5  var sessions = require("express-session");
6  var app = express();
7
8  app.use(cookies(credentials.cookieSecret));
9  var sessionConfiguration = {
10     secret: credentials.cookieSecret,
11     resave: false,
12     saveUninitialized: true,
13 };
14 app.use(sessions(sessionConfiguration));
15 http.createServer(app).listen(3000);
16
17 app.get("/countMe", function (req, res) {
18     var session = req.session;
19     if (session.views) {
20         session.views++;
21         res.send("You have been here "
22            + session.views + " times (last visit: " + session.lastVisit + ")");
23         session.lastVisit = new Date().toLocaleDateString();
24     }
25     else {
26         session.views = 1;
27         session.lastVisit = new Date().toLocaleDateString();
28         res.send("This is your first visit!");
29     }
30 });
```

A user starts up her browser (which contains no cookies so far) and accesses `http://my.site.nl:3000/`, followed by `http://my.site.nl:3000/countMe`. She then clicks the browser's "Reload current page" button three times. How many times in the process does the server running at `my.site.nl:3000` send a cookie to the browser?

A. 0

B. 1

C. 4

D. 5

Question 20

The browser B currently has no stored cookies. The server sends the following four cookies to B (assume this is happening today):

```
Set-Cookie: bg=white; Expires=Fri, 01-Aug-2016 21:47:38 GMT; Path=/; Domain=tudelft.nl
Set-Cookie: pref=1; Path=/; Domain=tudelft.nl
Set-Cookie: dom=23; Expires=Thu, 01-Jan-2023 00:00:01 GMT; Path=/; Domain=tudelft.nl; HttpOnly
Set-Cookie: view=mobile; Path=/; Domain=tudelft.nl; secure
```

B crashes 10 minutes later and the user restarts B. How many cookies can the user access after the restart with client-side JavaScript?

A. 0

B. 1

C. 2

D. 3

Question 21

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

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

Question 22

Consider the following Node.js script:

```
1  var express = require("express");
2  var http = require("http");
3  var app = express();
4
5  app.use(function(req,res){
6    console.log(req.method + " => " +req.url);
7  });
8
9  app.use(function(req,res,next){
10   res.writeHead(200, {"Content-Type": "text/plain"});
11   res.end("Hello there!");
12   next();
13 });
14
15 app.use(function(req,res){
16   console.log("HTTP response completed.");
17 });
18
19 http.createServer(app).listen(3000);
```

What happens when a user accesses `http://localhost:3000` in the browser (assuming that the Node.js script was started on the same machine)?

A. On the server-side, the HTTP request is logged to the console as:
`GET -> /`.
The browser displays `Hello there!`.

B. On the server-side, the HTTP request is logged to the console as:
`GET -> /`.
The browser displays nothing.

C. On the server-side, the HTTP request is logged to the console as:
`GET -> http://localhost:3000/`
`HTTP response completed.`
The browser displays nothing.

D. On the server-side, the HTTP request is logged to the console as:
`GET -> http://localhost:3000/`.
The browser displays `Hello there!`.

Question 23

Which of the following statements about Node.js' module system are TRUE?

[1] Node.js has a file-based module system.

[2] Node.js runs each module in a separate scope and only returns the final value of `require`.

[3] The `module.exports` package runs synchronously.

[4] `module.exports` and `exports` can be used in exactly the same way.

A. Only 1.

B. Only 2 and 4.

C. Only 1 and 3.

D. All of the statements are true.

Question 24

Which type of attack occurs when an attacker convinces a user to send a request to a server with malicious input and the server echoes the input back to client?

A. Persistent XSS attack

B. Non-persistent XSS attack

C. Persistent injection attack

D. Non-persistent injection attack

Question 25

Consider the following Node.js script, saved in `bar.js`. What is the console output when running `node bar.js`?

```
1  var fs = require('fs');
2
3  //n.txt is a file in the same directory as bar.js;
4  //it contains a single number: 42
5  var file = "n.txt";
6
7  let reading = function(err, data) {
8      if (err) {
9          throw err;
10     }
11     return data;
12 };
13 fs.readFile(file, reading);
14 console.log(reading);
```

A. `[Function:reading]`

B. 42

C. The code runs without errors, but produces no visible output.

D. The code does not run. It contains an error: Node.js requires the callback in `fs.readFile` to be a function.

Question 26

Sanitizing user input protects from which of the following attacks?

A. Improper input validation

B. Broken authentication

C. Cross-site request forgery

D. Cross-site scripting

Question 27

Consider an attacker that can eavesdrop on a client's outbound traffic and read all HTTP requests. Which of the following statements is FALSE?

A. The attacker can store and replay the traffic to perform a denial of service attack.

B. The attacker can drop the client's traffic so the request never reaches the server.

C. The attacker can store and replay the cookies to hijack the client's session.

D. None of the above.

The following questions refer to a simple card game web application Quartett which consists of a splash screen and a game screen:

- The splash screen contains the name of the game, a short description of it, a large logo consisting of two playing cards, a `Play now!` button (a click on which leads to the game screen) and a visitor count.
- The game screen contains the name of the game; underneath it is a status bar where important information about the game is presented. Two cards are visible, each with a dinosaur picture and five *dinosaur attributes*. Each attribute consists of a name (e.g. weight, speed) and a value. The card shown on top is the one the user plays with. The bottom left part of the screen is occupied by a plant: it is initially small but grows each time the user wins (up to a certain maximum height) and shrinks each time (up to a certain minimum height) the user loses a game.

The game is a single-turn game between two players and works as follows: *Each player receives a random card from a stack of 20. That is the card shown on top. One of the two players (here: the player that joined the game as second player) selects one of the five attributes. The winner is the player whose card has a higher value for the picked attribute. Once the player makes the pick, the winner/loser is announced to both players and the plant of each players grows/shrinks accordingly.*

The web application is implemented in Node.js/Express and relies on WebSockets for the multiplayer aspect.

Splash screen.

Game screen.

Question 28

The server-side script of our card game consists of a single Node.js script: `app.js`. Let's take a look at the first half of the script:

```
1  var express = require("express");
2  var http = require("http");
3  var websocket = require("ws");
4  var port = process.argv[2];
5  var app = express();
6  var messages = require("./public/javascripts/messages");
7  var cookies = require("cookie-parser");
8
9  app.use(express.static(__dirname + "/public"));
10 app.set('views', __dirname + '/public');
11 app.set('view engine', 'ejs');
12
13 app.use(cookies("fdsfwr32sfe"));
14
15 let visitors10Days = 1;
16
17 app.use(function(request, response, next) {
18   console.log("%s\t%s\t%s", new Date(), request.method, request.url);
19   next();
20 });
21
22 app.get("/play", function(req, res) {
23   res.sendFile("game.html", { root: "./public" });
24 });
25
26 app.get("/s[tp]l?[ao][prs]?[th](ing)?a?*", function(req, res) {
27   if(req.signedCookies.visited == undefined){
28     res.cookie('visited','1',{signed: true, maxAge: 864000000});
29     //864000000 milliseconds == 10 days
30     visitors10Days++;
31   }
32   res.render('splash', { count: visitors10Days });
33 });
34
35 /* here comes the WebSockets part */
```

In this script, two routes are defined, one on line 22 (route to the game screen) and one on line 26 (route to the splash screen). Let's assume we start our server-side script on `localhost`, port 3000. We start our browser on the same machine. How many of the following URLs will return the splash screen (line 26)?

```
http://localhost:3000/splashing
http://localhost:3000/startingagame
http://localhost:3000/stopping
http://localhost:3000/spartan
```

A. 1

B. 2

C. 3

D. 4

Question 29

Consider the `app.js` script from Question 28 again. In order to ensure that users do not become addicted to this game, someone suggests to replace lines 26-33 with the following code snippet:

```
1  app.get("/s[tp]l?[ao][prs]?[th](ing)?a?*", function(req, res, next) {
2    if(req.signedCookies.visited == undefined){
3      res.cookie('visited','1',{signed: true, maxAge: 864000000});
4      res.cookie('counter','1',{signed: true, maxAge: 864000000});
5      visitors10Days++;
6    } else{
7      var c = req.signedCookies.counter + 1;
8      res.cookie('counter',c,{signed: true, maxAge: 864000000});
9    }
10   if(c>100){
11     next();
12   }
13   res.render('splash', { count: visitors10Days });
14 });
15
16 app.get("/s[tp]l?[ao][prs]?[th](ing)?a?*", function(req, res) {
17   res.setStatus(404);
18 });
```

What is its effect for a user that always accesses the game via the splash screen with the same browser on the same device?

A. A user whose `visited` cookie was set less than 10 days ago and who has accessed the splash screen at least 100 times since then, receives a 404 status code from now on until the browser invalidates the cookie with key `visited`.

B. This piece of code leads to no visible effect, as it is not possible for the cookie with key `counter` to reach a value of 100 or more, no matter how often a user accesses the splash screen.

C. As long as the user restarts her browser after every game, she will never receive the 404 status code.

D. As long as the user cleans her cookie storage every 24 hours she will observe no visible effects, no matter how often she accesses the splash screen.

Question 30

Having implemented the multiplayer part of our card game with WebSockets (in particular the `ws` module of Node.js), we now want to incorporate a chat option for our players: all players that are currently playing can chat with each other in a single chat room. There are no private chats: every chat message a player sends should be broadcasted to users that are currently playing. Is this possible to do with a WebSocket-based setup?

A. No, the WebSocket protocol has a strict requirement: a single message sent via the WebSocket protocol and received by the server can only be sent out once to a single client.

B. It is possible. Each user with an open WebSocket connection to the server can be sent the same message.

C. It is possible. The WebSocket protocol has a broadcasting channel to which all WebSocket connections currently active on a single server are subscribed to automatically.

D. No, the `ws` module has a security restriction that forbids messages from client A to be send to client B unless the scheme, domain, host and port of their protocol request matches.

Question 31

Consider a possible styling of one of our playing cards (the rendering of the card looks is shown in Figure 1):

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .card {
        padding: 10px;
        width: 300px;
        height: 350px;
        border-radius: 30px;
        background-color: lightgray;
        border: 5px solid gold;
      }

      .attribute {
        display: block;
        font-size: 28px;
        border-radius: 10px;
        padding: 1px;
      }

      span:nth-of-type(2n) {
        background-color: gold;
      }

      span:nth-of-type(2n + 1) {
        background-color: tomato;
      }
    </style>
  </head>
  <body>
    <div id="logo">
      <div class="card" id="c0">
        <div>Attributes:</div>
        <span class="attribute">height: 2 m</span>
        <span class="attribute">weight: 170 kg</span>
        <span class="attribute">speed: 10 km/h</span>
        <span class="attribute">strength: 3 stars</span>
        <span class="attribute">ferocity: 1 star</span>
      </div>
    </div>
  </body>
</html>
```

Figure 1: Playing card. The height/speed/ferocity rows have a tomato color, the weight/strength rows have a gold color.

We now want to style the `<div>Attributes:</div>` (line 33) element with a black background color and a white font. Which of the following options does NOT lead to the desired styling of the element?

A.
```
div > div + div {
    background-color: black;
    color: white;
}
```

B.
```
.card > div {
    background-color: black;
    color: white;
}
```

C.
```
div div div {
    background-color: black;
    color: white;
}
```

D.
```
#c0 div {
    background-color: black;
    color: white;
}
```

Question 32

Consider the HTML document of **the previous question** again. We now want to move the attributes and their values into data attributes and come up with the following HTML snippet that should replace lines 34 to 38.

```
1 <span class="attribute" data-attr="height" data-val="2 m"></span>
2 <span class="attribute" data-attr="weight" data-val="170 kg"></span>
3 <span class="attribute" data-attr="speed" data-val="170 kg"></span>
4 <span class="attribute" data-attr="strength" data-val="3 stars"></span>
5 <span class="attribute" data-attr="ferocity" data-val="1 star"></span>
```

How does the `<span>` element have to be styled to make those attributes of the dinosaur visible on the playing cards as shown in Figure 1?

A.
```
span::after {
    content: attr(data-attr) ": " attr(data-val);
}
```

B.
```
span::after {
    content: val(data-attr) ": " val(data-val);
}
```

C.
```
span::after {
    attr: content(data-attr) ": " content(data-val);
}
```

D. This is not possible as the data attributes defined in this HTML snippet are invalid: HTML elements with data attributes cannot have a class assigned to them at the same time.

Question 33

Consider the HTML document of Question 31 one last time. What is the effect when lines 21-27 are replaced by the following code snippet (i.e., we are replacing `nth-of-type` with `nth-child`):

```
span:nth-child(2n) {
    background-color: gold;
}

span:nth-child(2n + 1) {
    background-color: tomato;
}
```

A. The card looks as shown in Figure 1.

B. The background color of each dinosaur attribute has changed compared to Figure 1: attributes that had a tomato background color before, now have a gold background color and vice versa.

C. All dinosaur attribute elements have the same background color: lightgray.

D. All dinosaur attribute elements have the same background color: white.

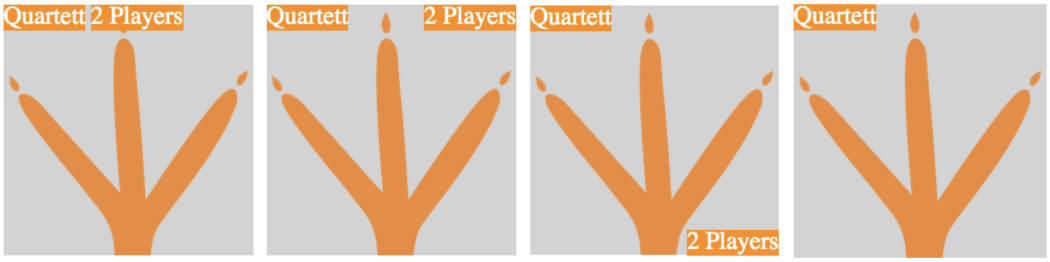
Question 34

Consider the following HTML document, which shows off the first idea of a logo for our card game:

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <style>
6          div {
7            /* image of an orange claw with a transparent background */
8            background-image: url("clawprint.png");
9            background-repeat: no-repeat;
10           background-size: cover; /* scales the background img to fit the div */
11           background-color: lightgray;
12           width: 200px;
13           height: 200px;
14         }
15
16         div::before,
17         div::after {
18           content: 'Quartett';
19           background-color: darkorange;
20           color: white;
21         }
22
23         div::after {
24             content: '2 Players';
25             float: right;
26         }
27     </style>
28 </head>
29
30 <body>
31     <div>
32     </div>
33 </body>
34 </html>
```

How does the rendering of the logo look like?

B is the correct answer.



A. **B.** **C.** **D.**

Question 35

Recall that on the game screen we have a small green plant that grows each time the user wins (up to a certain maximum height) and shrinks each time the user loses (up to a certain minimum height). The plant is represented by a `<div>` element in the game screen. The CSS styling of the element looks as follows:

```
#plant {
    background-size: contain;
    background-repeat: no-repeat;

    /* the path points to an image of a plant with a transparent background*/
    background-image: url("../images/plant.png");

    transform-origin: bottom center;
    transform: scale(1);

    /* the following rules concern the position of the plant on the game screen */
    position: absolute;
    left: 150px;
    bottom: 10px;
    height: 100px;
    width: 100px;
}
```

Since the plant "grows"/"shrinks" according to the outcome of the game, we use JavaScript to change the plant's size by changing the scaling factor. Which CSS rule do we have to include in this code snippet so that the plant does not "grow"/"shrink" instantaneously but takes roughly 2 seconds to reach its set height?

A. `animation-time: 2s;`

B. `transition: transform 2s;`

C. This can only be achieved through a combination of CSS rules: we first need to set up `@keyframes` from the current scale to the updated scale and then rely on the `animation-delay` property to set the delay to 2 seconds.

D. This cannot be achieved through one or more CSS rules alone, we need to rely on JavaScript's `setTimeout` function to gradually "grow"/"shrink" the plant (every few milliseconds we increase or decrease the scaling).

Question 36

In order to ensure that our web application is always running on the server, we create a Node.js script `watcher.js` that checks every 10 seconds whether HTTP responses are being sent out by our web application server. The Node.js script to achieve this looks as follows:

```
1  var http = require("http");
2
3  console.log("Watcher started!");
4
5  //setInterval: repeatedly calls a function with a fixed time delay
6  //(in milliseconds) between each call
7  setInterval(function(){
8      //http.get makes a GET request to the specified server;
9      //the callback consumes the response data
10     http.get({
11         hostname: 'localhost',
12         port: 3000,
13         path: '/splash',
14         agent: false
15     }, function(res){
16         console.log("Server is working.");
17     }).on('error', function(err){ //in case of an 'error' event, execute this callback
18         console.log("SERVER DOES NOT WORK!");
19     });
20     console.log("Checking now whether our card game web server is alive");
21 }, 10000);
22
23 console.log("Watcher is set up!");
```

What is the output on the console after having started our script with the command `node watcher.js` on a machine where the web application server does NOT run on and having stopped it after 15 seconds?

A.
```
Watcher started!
Watcher is set up!
Checking now whether our card game web server is alive
SERVER DOES NOT WORK!
```

B.
```
Watcher started!
Server is working.
Checking now whether our card game web server is alive
Watcher is set up!
```

C.
```
Watcher is set up!
Watcher started!
SERVER DOES NOT WORK!
Checking now whether our card game web server is alive
```

D.
```
Watcher started!
Watcher is set up!
```

Question 37

Digging into Quartett's code base we find the information of the cards themselves to be contained in the following JavaScript object:

```
var dinosaurCards = [
    {
        image: "images/dino1.png",
        height: "1.50 m",
        weight: function(){
            return (Number.parseFloat(this.height) * 3)+" kg";
        },
        speed: "25 km/h",
        strength: function(){
            return (Number.parseFloat(this.height) *
             Number.parseFloat(this.weight())) +" stars";
        }
    },
    {
        image: "images/dino1.png",
        height: "1.76 m",
        weight: function(){
            return (Number.parseFloat(this.height) * 3)+" kg";
        },
        speed: "15 km/h",
        strength: function(){
            return (Number.parseFloat(this.height) *
             Number.parseFloat(this.weight())) +" stars";
        }
    }
];
```

In order to transmit this object across the network we convert it into a string with the following line of code: `var stringToSend = JSON.stringify(dinosaurCards)`. How does `stringToSend` look like? *Note: to make the string fit the width of this page, we have introduced line breaks where necessary. Ignore the line breaks.*

A. `"[{\"image\":\"images/dino1.png\",\"height\":\"1.50 m\",\"weight\":\"4.5 kg\",\"speed\":\"25 km/h\",\"strength\":\"6.75 stars\"},{\"image\":\"images/dino1.png\",\"height\":\"1.76 m\",\"weight\":\"5.28 kg\",\"speed\":\"15 km/h\",\"strength\":\"9.2928 stars\"}]"`

B. `"{\"image\":\"images/dino1.png\",\"height\":\"1.50 m\",\"speed\":\"25 km/h\"}"`

C. `"[{\"image\":\"images/dino1.png\",\"height\":\"1.50 m\",\"speed\":\"25 km/h\"},{\"image\":\"images/dino1.png\",\"height\":\"1.76 m\",\"speed\":\"15 km/h\"}]"`

D. `"{\"image\":\"images/dino1.png\",\"height\":\"1.50 m\",\"weight\":\"4.5 kg\",\"speed\":\"25 km/h\",\"strength\":\"6.75 stars\"}"`

Question 38

The game screen of our card game application looks as follows:

```
1  <!DOCTYPE HTML>
2  <html>
3      <head>
4          <link href="stylesheets/style.css" rel="stylesheet">
5      </head>
6      <body>
7          <div id="description">
8              <h1>Quartett</h1>
9              <h2 style="color:␣Tomato">
10             </h2>
11         </div>
12
13         <div id="logo">
14             <div class="card" id="c1">
15                 <span class="attribute"></span>
16                 <span class="attribute"></span>
17                 <span class="attribute"></span>
18                 <span class="attribute"></span>
19                 <span class="attribute"></span>
20             </div>
21             <div class="card" id="c0">
22                 <span class="attribute"></span>
23                 <span class="attribute"></span>
24                 <span class="attribute"></span>
25                 <span class="attribute"></span>
26                 <span class="attribute"></span>
27             </div>
28         </div>
29         <div id="plant">
30         </div>
31         <script src="javascripts/dinosaurs.js"></script>
32         <script src="javascripts/messages.js"></script>
33         <script src="javascripts/interactions.js"></script>
34     </body>
35 </html>
```

When digging through the code of `interactions.js` (line 33) we find that the player that has to decide which card attribute to pick executes the following code snippet in her browser:

```
1  for (let j = 0; j < 5; j++) {
2      document.querySelectorAll("#c0 span")[j].onclick = function func() {
3          let clickedSpan = this;
4          clickedSpan.style.border = "1px solid red";
5          setTimeout(function () {
6              clickedSpan.style.border = "";
7          }, 1000);
8
9          for (let k = 0; k < 5; k++) {
10             let el = document.querySelectorAll("#c0 span")[k];
11             el.onclick = function () { };
12         }
13     };
14 }
```

What effect does this code snippet have?

A. The dinosaur attribute that was first clicked receives a red border for a short amount of time. Subsequent clicks have no visible effect.

B. Each click on a dinosaur attribute leads to a red border around the clicked attribute for a short amount of time. Thus, if the user clicks quickly (e.g. two clicks in under one second), it is possible to observe two or more red borders around different attributes at the same time. The size of the border does not change, no matter how often an attribute is clicked.

C. Clicking on any of the dinosaur attributes has no visible effect.

D. Each click on a dinosaur attribute increases the size of the red border by `1 px` until a second has passed. Once the second is passed, the border styling is reset to no border.

Question 39

Further digging into `interactions.js` reveals this function, which seems to be used to parse messages received from the server:

```
/*Notes (from MDN):
 *String.prototype.includes(search) determines whether one string
 *is found in another string, returning true or false as appropriate.
 *
 *String.prototype.indexOf(search) returns the index within the calling String object
 *of the first occurrence of the search value. Returns -1 if the value is not found.
 *
 *String.prototype.lastIndexOf(search) returns the index within the calling String
 *object of the last occurrence of the search value. Returns -1 if not found.
 *
 *String.protoytpe.substr(index, len) returns a portion of the string, starting at
 *(and including the character at) index; len contains the number of of characters
 *to extract. If no len argument is provided the entire substring starting at index
 *is returned.
 *
 *Number.parseFloat(s) parses a floating point number from argument s. If parseFloat
 *encounters a character other than a plus sign (+), minus sign (-), numeral (0 9 ),
 *decimal point (.), or exponent (e or E), it returns the value up to that character,
 *ignoring the invalid character and all characters following it.
 */

function extractAttributeValue(s) {
  let res = ["", "", ""];

  let ss = s;
  if (ss.includes("[")) {
    ss = ss.substr(ss.indexOf("[") + 1);
    ss = ss.substr(0, ss.indexOf("]"));
  }

  res[0] = ss.substr(0, ss.indexOf(":"));
  res[1] = Number.parseFloat(ss.substr(ss.indexOf(":") + 2));
  res[2] = ss.substr(ss.lastIndexOf(" "));

  return res;
}
```

How does the returned array of `extractAttributeValue("[height:120.5 m]")` look like?

A. `[ "height", 20.5, " m" ]`

B. `[ "height", 120.5, "m" ]`

C. `[ "eight", "120.5 m", "" ]`

D. `[ "eight", 20, " m" ]`

Question 40

Taking a final look at `interactions.js`, we find the following code snippet:

```
function setPlantScale(n){
  document.cookie="scaling="+n;
}
```

It seems like client-side cookies are used to track the height of the plant across games.

Given this knowledge, how should a function that retrieves the current scaling of the plant from the client-side cookie look like? Returned should be a single number. *Note: for the purposes of this question, you can assume that* `setPlantScale()` *was called before, i.e. the cookie has been set.*

A.

```
function getPlantScale(){
  let cc = document.cookie.split('; ');
  for(let i=0; i < cc.length; i++) {
      if(cc[i].includes("scaling")){
        return cc[i].split("=")[1];
      }
  }
}
```

B.

```
function getPlantScale(){
  return document.cookie["scaling"][1];
}
```

C.

```
function getPlantScale(){
  let cc = document.cookie.split(', ');
  return cc["scaling"][1];
}
```

D.

```
function getPlantScale(){
  if(document.cookie.length==1)
    return document.cookie.split("=")[1];
  else {
    return document.cookie["scaling"].split("=")[1];
  }
}
```