Delft University of Technology

CSE1500 Web & Database Technology

Resit: Midterm Exam

Date: 12/04/2019
Time Limit: 90 minutes

Instructions:

- This exam contains 15 pages (including this cover page) and 30 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:

```
HTTP/1.1 200 OK
Date: Mon, 01 Apr 2019 13:37:20 GMT
Content-Type: text/html; charset=utf-8
Cache-Control: no-store, no-cache, private
Expires: Sat, 15 Nov 2008 16:00:00 GMT
Set-Cookie: uuid2=5975858; Path=/; Domain=.adnxs.com
```

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 `Host` attribute is missing.

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

Question 2

One common representation of the network stack is the OSI model. In what order do TCP, IP and HTTP occur in it?

A.
```
IP
HTTP
TCP
..
Physical layer
```

B.
```
IP
TCP
HTTP
..
Physical layer
```

C.
```
HTTP
TCP
IP
..
Physical layer
```

D.
```
TCP
IP
HTTP
..
Physical layer
```

Question 3

What does *long polling* refer to in the context of HTTP?

A. Long polling emulates a *pull* mechanism that is missing in HTTP/1.1: it is implemented via WebSockets that require a protocol upgrade; after the upgrade, the client can pull data from the server at will.

B. Long polling emulates a *push* mechanism that is missing in HTTP/1.1: the client sends an HTTP request; the server holds the request open until new data is available before sending its HTTP response. Once the response is sent, the client immediately sends another HTTP request that is kept open.

C. Long polling emulates a *push* mechanism that is missing in HTTP/1.1: the client regularly sends an HTTP request to the server, the server in turn sends its HTTP response and if the response has changed compared to what is currently rendered in the client, the client renders the updated content.

D. Long polling emulates a *pull* mechanism that is missing in HTTP/1.1: it is layered over TCP and requires the use of XMLHttpRequest objects; the client initializes an XMLHttpRequest object for each connection to the server and can then pull data from the server at will.

Question 4

What does `Accept-Encoding:identity` indicate?

A. Sent in the request header. The client only understands content that is not compressed.

B. Sent in the response header. The client understands any type of encoding of the content the server chooses.

C. Sent in the request header. The client and server could not agree on an encoding; the client requests an alternative but identical server.

D. Sent in the response header. The server informs the client of the encoding of the content.

Question 5

Consider the following two files `foo.js` and `bar.js` (which are saved in the same directory):

```javascript
//foo.js
function increaseHealth(tokens){
    exports.health++;
}

function decreaseHealth(tokens){
    exports.health--;
    if(exports.health<0){
        exports.health = 0;
    }
}

function setName(name){
    exports.name = name;
}

exports.health = 0;
exports.name = "John Doe";
exports.increaseHealth = increaseHealth;
exports.decreaseHealth = decreaseHealth;
exports.setName = setName;
```

```javascript
//bar.js
var player1 = require('./foo');
var player2 = require('./foo');

player1.setName("Bob");
player2.setName("Alice");

player1.increaseHealth(10);
player2.increaseHealth(5);

console.log(player1.name + "/"+player1.health);
console.log(player2.name + "/"+player2.health);
```

What is the output on the console when running `node bar.js`?

A.
```
Bob/1
Alice/1
```

B. `Bob/10`
   `Alice/5`

C. `John Doe/0`
   `John Doe/0`

D. `Alice/2`
   `Alice/2`

Question 6

Which of the following statements about IP addresses are FALSE?

[1] The Internet maintains two principal namespaces: the domain name hierarchy and the IP address system.

[2] Version 4 IP addresses (IPv4) consist of 32 bits; they are divided into 4 blocks of 8 bits each.

[3] More than 100 billion unique IP addresses can be generated in the IPv4 address space.

[4] An IPv6 address consists of 128 bits.

A. Only 1 and 3.

B. Only 2 and 4.

C. Only 3.

D. Only 4.

Question 7

Consider the following URL: `https://www.bing.com/search?q=TU+Delft&qs=n`

In general, URLs can consist of up to nine parts. Which of the following parts is NOT present in the URL above?

A. path

B. scheme

C. input parameters

D. fragment

Question 8

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

```
for (var i = 1; i <= 10; i++) {
    setTimeout(function() {
        console.log(i);
    }, 1000);
}
```

A. The numbers 1 to 10 are printed to the console immediately.

B. The numbers 1 to 10 are printed to the console; there is a delay of 1 second between each number being printed to the console.

C. After a ten second delay, the number 11 is printed to the console once.

D. After a one second delay, ten printouts of the number 11 appear on the console.

Question 9

What does JavaScript's hoisting principle refer to?

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

B. Function expressions are always executed before function declarations.

C. Expressions are processed before any code is executed.

D. Declarations are processed before any code is executed.

Question 10

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

```
var n = "Tom";
function animal(t,n){
    this.t = t; //type
    this.n = n; //name

    this.getInfo = function(){
        console.log(this);
    }
}
var jerry = new animal("mouse", "Jerry");
jerry.getInfo();

var infoFunc = jerry.getInfo;
infoFunc();
```

A. `ReferenceError: ‘this’ is not defined within function scope.`

B. `Object { t: "mouse", n: "Jerry", getInfo: getInfo() }`
`undefined`

C. `Object { t: "mouse", n: "Jerry", getInfo: getInfo() }`
`Object { t: "mouse", n: "Jerry", getInfo: getInfo() }`

D. `Object { t: "mouse", n: "Jerry", getInfo: getInfo() }`
`Window`

Question 11

Which of the following statements about JavaScript functions is TRUE?

A. Functions cannot be passed as parameters.

B. Functions cannot be returned from functions.

C. Functions cannot be assigned to object properties.

D. Functions are objects.

Question 12

Which of the following statements about Node.js' nature is TRUE?

A. Its event loop is multi-threaded. In addition, it maintains a single thread in order to process I/O requests.

B. Its event loop is single-threaded. In addition, it maintains a single thread in order to process I/O requests.

C. Its event loop is multi-threaded. In addition, it maintains a pool of threads in order to process I/O requests.

D. Its event loop is single-threaded. In addition, it maintains a pool of threads in order to process I/O requests.

Question 13

Consider the HTML document below. When opened in a browser, each click anywhere on the page is supposed to change the background color, alternating between red and blue.

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Click to switch colors</title>
5          <style>
6              html, body {
7                  height: 100%;
8                  width: 100%;
9              }
10             body {
11                 background-color: red;
12             }
13         </style>
14         <script>
15             var myColor = 'red';
16             window.onload = function() {
17                 //???
18             };
19
20             function func() {
21                 if(myColor == 'red'){
22                     myColor = 'blue';
23                 }
24                 else {
25                     myColor = 'red';
26                 }
27                 document.body.style.backgroundColor = myColor;
28             };
29         </script>
30     </head>
31     <body>
32     </body>
33 </html>
```

Someone accidentally deleted line 17. Which of the following code snippets (to fix line 17) will lead to the desired behaviour?

A. `document.getElementsByTagName("body")[0].onclick = func;`

B. `document.getElementsByClassName("body")[0].onclick = func;`

C. `$(body).onclick = func();`

D. `document.html.body.onclick(func);`

Question 14

Consider the HTML document of Question 13 one more time. Assume the issue of line 17 has been successfully fixed. What happens if the `onclick` property is replaced by the `onkeypress` property?

A. Every time a key is pressed on the keyboard the color changes. Mouse clicks no longer lead to a color change.

B. Every time a key is pressed on the keyboard the color changes. Every time the mouse is clicked the color changes.

C. No color changes are observed, no matter how the mouse and keyboard are used as `onkeypress` is not a valid event type.

D. No color changes are observed, no matter how the mouse and keyboard are used as the `onkeypress` event requires a particular key event as a function attribute.

Question 15

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

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

//n.txt is a file in the same directory as bar.js;
//it contains a single number: 42
var file = "n.txt";

let reading = function(err, data) {
    if (err) {
        throw err;
    }
    return data;
}();

fs.readFile(file, reading);
console.log(reading);
```

A. `undefined`

B. 42

C. The code does not compile as `fs` is not a Node.js module.

D. The code does not compile as Node.js requires the callback to be a function.

Question 16

Consider the following Node.js script, saved in `bar.js`. The script is started on localhost with the following command: `node bar.js`.

```
var express = require("express");
var url = require("url");
var http = require("http");

var port = 3001;
var app = express();
http.createServer(app).listen(port);

var htmlPrefix = "<!DOCTYPE html><html><head><style>body { background-color: ";
var htmlSuffix = ";}</style></head><body></body></html>";

app.get("/*", function(req, res){
    var query = url.parse(req.url, true).query;
    var c = (query["c"] != undefined) ? query["c"] : "red";
    res.send(htmlPrefix + c + htmlSuffix);
})
app.get("/*", function(req, res){
    res.send("Not a valid route ...");
})
```

In a browser, the following four URLs are pasted into the address bar. How many times will the returned page be rendered red?

- `http://localhost:3001/blue`
- `http://localhost:3001/?c=blue`
- `http://localhost:3001/c=blue`
- `http://localhost:3001/c/blue`

A. 0

B. 2

C. 3

D. 4

Question 17

While in Question 16 the whole HTML document was served by the server-side script, now the goal is to only serve the random color via WebSockets: a client can connect through the WebSocket protocol to the server and request a randomly generated color an unlimited number of times. The server-side script looks as follows:

```
1  var express = require("express");
2  var url = require("url");
3  var http = require("http");
4  var websocket = require("ws");
5
6  var port = 3001;
7  var app = express();
8
9  var server = http.createServer(app).listen(port);
10
11 var wss = new websocket.Server({ server });
12
13 //??
14
15 //generate a random color
16 function getRandomRGB() {
17     var options = "ABCDEF0123456789";
18     let c = "#";
19     for (let i = 0; i < 6; i++) {
20       c += options[Math.floor(Math.random() * 16)];
21     }
22     return c;
23 }
```

What does line 13 have to look like for this script to achieve its goal?

A.
```
wss.on("connection", function(ws){
    if(ws.event == "message"){
        ws.send(getRandomRGB());
    }
});
```

B.
```
wss.on("message", function(ws){
    ws.send(getRandomRGB());
});
```

C.
```
wss.on("connection", function(ws){
    ws.send(getRandomRGB());
});
```

D.
```
wss.on("connection", function(ws){
    ws.on("message", function(msg){
        ws.send(getRandomRGB());
    });
});
```

Question 18

Which of the following statements about Ajax and WebSockets are TRUE?

[1] Ajax provides a low-latency and bi-directional connection between client and server.

[2] Ajax requires the use of XML as data exchange format.

[3] WebSockets require the use of the XMLHttpRequest object to request a protocol upgrade.

[4] In contrast to Ajax, WebSockets do not require entire HTTP messages to be sent between client and server (thus reducing the overhead).

A. Only 1.

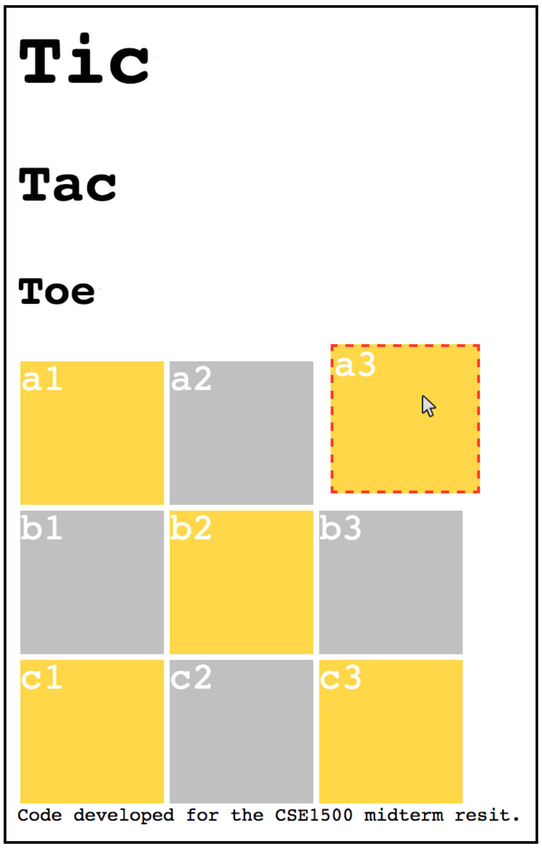
B. Only 2 and 4.

C. Only 4.

D. None of the statements are true.

Question 19

Consider the following Tic Tac Toe game interface (the black border around the interface marks the page boundary; you can ignore it for the purposes of the following CSS questions):



Code developed for the CSE1500 midterm resit.

The game interface consists of nine tiles (a1 to c3); hovering with the mouse over a tile (a3 in the screenshot) moves the tile slightly to the top-right of its original position (the tile also receives a red border). Once the hovering stops, the tile takes up its original position and appearance again.

The corresponding HTML document (with part of the CSS missing) looks as follows:

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Tic Tac Toe</title>
        <style>
            :root {
                --tile-color1: silver;
                --tile-color2: gold;
                --hover-color: red;
                --body-background: white;
            }

            html {
                font-family: monospace;
            }

            body {
                background-color: var(--body-background);
            }

            div.c1 {
                background-color: var(--tile-color2)
            }
            div.c2 {
                background-color: var(--tile-color1);
            }

            footer {
                clear: both;
                font-size: 50%;
            }

            .tile {
                border: 1px solid var(--body-background);
            }

            /*
             MISSING CSS
            */
        </style>
    </head>
    <body>
        <header>
            <h1>Tic</h1>
            <h2>Tac</h2>
            <h3>Toe</h3>
        </header>
        <main>
            <div id="board">
                <div class="tile c1" data-id="a1"></div>
                <div class="tile c2" data-id="a2"></div>
                <div class="tile c1" data-id="a3"></div>

                <div class="tile c2 newline" data-id="b1"></div>
                <div class="tile c1" data-id="b2"></div>
                <div class="tile c2" data-id="b3"></div>

                <div class="tile c1 newline" data-id="c1"></div>
                <div class="tile c2" data-id="c2"></div>
                <div class="tile c1" data-id="c3"></div>
            </div>
        </main>
        <footer>
            Code developed for the CSE1500 midterm resit.
        </footer>
    </body>
</html>
```

Which of the following CSS snippets (added at the end of the current CSS rules) achieves the desired *layout of the tiles*?

A.
```
.tile {
        float: right;
        width: 50px;
        height: 50px;
}
.newline {
        clear: left;
}
```

B.
```
.tile {
        float: left;
        width: 50px;
        height: 50px;
}
.newline {
        clear: both;
}
```

C.
```
.tile {
        float: none;
        width: 50px;
        height: 50px;
}
.newline {
        clear: left;
}
```

D.
```
#tile {
        float: right;
        width: 50px;
        height: 50px;
}
#newline {
        float: left;
}
```

Question 20

Consider the Tic Tac Toe interface description of Question 19 again. Assume the layout of the tiles is correct. Which of the following CSS snippets (added at the end of the current CSS rules) achieves the desired *hovering effect*?

A.
```
.tile:hover{
    border: 1px dashed var(--hover-color);
    position:relative;
    bottom: 5px;
    left: 5px;
}
```

B.
```
.tile:hover{
    border: 1px dashed var(--hover-color);
    position:absolute;
    bottom: -5px;
    left: -5px;
}
```

C.
```
#tile:hover{
     border: 1px dashed var(--hover-color);
     position:absolute;
     bottom: 5px;
     left: 5px;
}
```

D.
```
.tile:hover{
     border: 1px dashed var(--hover-color);
     position:relative;
     bottom: -5px;
     left: -5px;
}
```

Question 21

Consider the Tic Tac Toe interface description of Question 19 again. Assume the layout of the tiles and the hovering effect work as intended. Which of the following CSS snippets (added at the end of the current CSS rules) will result in the labeling of each tile (i.e. the top left tile is labeled a1, the bottom right c3 and so on).

A.
```
.tile::after {
     color: white;
     content: attr(data-id);
}
```

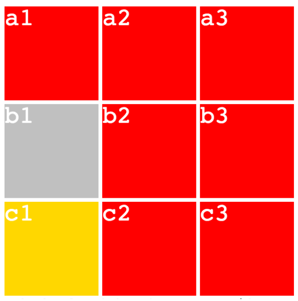
B.
```
.tile {
     color: white;
     content: attr(data-id);
}
```

C.
```
#tile::after {
     color: white;
     attr: data-id;
}
```

D.
```
.tile::top-left {
     color: white;
     attr: data(data-id);
}
```

Question 22

Consider the Tic Tac Toe interface description of Question 19 again. Assume the layout of the tiles, the hovering effect and the tile labeling work as intended. Which of the following CSS snippets (added at the end of the current CSS rules) will color the board as follows (i.e. b1 and c1 retain their original color and the remaining tiles are red):



A.
```
:not(.newline){
     background-color: red;
}
```

B.
```
main,*:not(.newline){
     background-color: red;
}
```

C.
```
main:not(.newline){
     background-color: red;
}
```

D.
```
main :not(.newline){
     background-color: red;
}
```

Question 23

Consider the Tic Tac Toe interface of Question 19 again. What happens when the following CSS rule is added at the end of the currently existing CSS rules:

```
* { visibility: hidden}
```

A. This CSS rule has no effect (the game interface does not change) as `visibility` is not a valid CSS property.

B. This CSS rule has no effect (the game interface does not change) as we have not defined a single HTML document with class or id `*`.

C. Only the HTML elements defined within `<header></header>` and `<footer></footer>` remain visible.

D. The entire web page will be blank.

Question 24

Which of the following statements about cookies are TRUE?

[1] Transient cookies remain intact after the browser is closed, they are stored on disk.

[2] The HttpOnly flag requires cookies to be sent via HTTP instead of HTTPS.

[3] The Signed flag allows the client and server to exchange cookies that are encrypted.

[4] A cookie's Domain field determines the domain the cookie is associated with.

A. Only 1 and 3.

B. Only 2 and 4.

C. Only 3.

D. Only 4.

Question 25

Which of the following statements about third-party cookies are TRUE?

[1] Third-party cookies pose a security threat to the browser as they enable the execution of code transmitted via the cookie's CONTENT field.

[2] Third-party cookies and first-party cookies are stored in different cookie storage spaces in the browser.

[3] Third-party cookies belong to the same domain as shown in the browser's address bar.

[4] Third-party cookies are set via the `Set-Cookie` response header.

A. Only 1, 2 and 4.
B. Only 2 and 4.
C. Only 2.
D. Only 4.

Question 26

A session-based system authenticates a user to a Web application to provide access to one or more restricted resources. To *increase security* for the end user, an authentication token should ..

[1] .. be sent via FTP.
[2] .. be stored in a non-persistent cookie.
[3] .. be stored in a persistent cookie.
[4] .. be base-64 encoded.

A. Only 2.
B. Only 4.
C. Only 1 and 2.
D. Only 1 and 3.

Question 27

Which of the following statements best describes a CSRF attack?

A. Untrusted data is sent to an interpreter as part of a command.
B. A web application takes untrusted data and sends it to a client without input validation.
C. A user's login and password information is sent unencrypted over the network.
D. A client, authenticated to a particular domain, sends a forged HTTP request which includes the session cookie.

Question 28

What does an attacker require to succeed in a reflected XSS attack?

A. A vulnerable web service that takes parts of the URLs it receives and generates output without validating the received data.
B. A network eavesdropping device (software or hardware) in order to read all outgoing HTTP requests.
C. A web application that is not relying on HTTPS, but HTTP instead.
D. The attacker needs to be able to *drop* HTTP requests.

Question 29

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 `module.exports`.
[3] The `require` module runs asynchronously.
[4] `module.exports` and `exports` can be used in exactly the same way.

A. Only 1 and 2.
B. Only 2 and 3.

C. Only 3 and 4.

D. All of the statements are true.

Question 30

Consider the following two routes defined in a Node.js/Express application:

```
app.get('/tud(elft)?*cse150+',function(req, res){
    res.send("Yes!");
});
app.get('*', function(req,res){
    res.send("No!");
});
```

Let's assume these routes are part of a server-side script started on localhost, port 3001. How many of the following URLs will lead to the `Yes!` response?

```
http://localhost:3001/tudelftscse
http://localhost:3001/tudelftscse1500
http://localhost:3001/tudcse15000
http://localhost:3001/tudelftuniversitycse15
```

A. 1

B. 2

C. 3

D. 4