WEB RESIT APRIL 16
QUESTION 1

Consider the following Node.js script. What is the main issue of this piece of code?

```
1   let http = require('http');
2   let url = require('url');
3
4   let httpResponder = function(req, res){
5       let parsedUrl = url.parse(req.url, true);
6
7       if( parsedUrl.pathname === "/hello"){
8           res.writeHead(404, {
9               "Content-Type": "text/html",
10              "Content-Length": "10"
11          });
12          let content = "Hello back!";
13          res.end(content);
14      }
15  };
16
17  let server = http.createServer(httpResponder).listen(5000);
```

The HTTP response is never sent.

The 404 status code triggers the script to send an empty HTTP response to the client.

➡ The HTTP response body will be truncated by the web browser.

The 404 status code needs to be paired with a "Type-Error" header field in order to yield a valid HTTP response.

QUESTION 2

How do Web caches employ the "If-Modified-Since" HTTP header field to work efficiently?

A Web cache sends an HTTP GET request to an origin server. In reply to this request, the origin server sends an HTTP response. If this response contains an "If-Modified-Since" header field with a value set to a date in the past, the Web cache does not store the response.

A Web cache only sends an HTTP request to an origin server if the Web resource has been modified by the origin server in the meantime.

➡ A Web cache sends an HTTP GET request to an origin server with the "If-Modified-Since" header field set. In reply to this request, the origin server sends an HTTP response. This response only contains a content body if the resource has changed after the date found in "If-Modified-Since".

A Web cache only sends an HTTP request to an origin server when the "Cache-Control" and the "If-Modified-Since" dates coincide.

QUESTION 3

Consider the following HTTP headers observed in a single HTTP message (no other headers have been observed). Which of the following statements about the HTTP message is TRUE?

```
Host: fastpages.fast.ai
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: _ga=GA1.2.77041210.1586941008; _gid=GA1.2.1408935603.1586941008
If-Modified-Since: Sun, 12 Apr 2020 08:11:38 GMT
Cache-Control: max-age=0
```

- These headers belong to a valid HTTP request.

- These headers belong to a valid HTTP response.

- These headers belong to an invalid HTTP request as the "Host" attribute needs to be employed together with the "Domain" attribute.

- These headers belong to an invalid HTTP response as the "Set-Cookie" attribute is missing.

QUESTION 4

Consider this written up telnet exchange in the terminal. Which of the following statements about it is TRUE?

```
 1  $ telnet github.com
 2  Trying 140.82.118.4...
 3  Connected to github.com.
 4  Escape character is '^]'.
 5  HEAD / HTTP/1.1 80
 6  host:github.com
 7
 8  HTTP/1.1 301 Moved Permanently
 9  Content-length: 0
10  Location: https://github.com/
11
12  Connection closed by foreign host.
```

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

➡ Line 5 is invalid: the HTTP protocol type and port number should not appear together.

The 301 response status code (line 8) is not "Moved Permanently" but instead should be "Found".

This telnet exchange is valid.

QUESTION 5

Which of the following statements describes long polling most accurately?

After the initial HTTP response, the client regularly sends an HTTP request to the server, the server in turn sends its HTTP response and if the response contains contains different content than before the client renders the updated content.

➡ After the initial HTTP response, the client sends an HTTP request and the server holds the request open until new content is available before sending its HTTP response. Once the response is sent, the client immediately sends another HTTP request that is kept open.

After the initial HTTP response, the server can send HTTP responses directly to the client when new content is available without having to wait for another HTTP request. The client renders the updated content.

The client's first HTTP request contains information about the WebSocket protocol update that is required for the server to send HTTP responses to the client without having to wait for another HTTP request.

QUESTION 6

Consider a web application which on the server-side logs all HTTP requests. On the server-side, the application does neither require login information from the users nor does it personalize HTTP responses (this means that every HTTP request with the same path, parameter and query components is treated the same). Now, based on these logs, logical sessions should be constructed, that is, series of HTTP requests that were coming most likely from the same user.

"Authorization" header field information

➡ "User-Agent" header field information

The unique IDs present in the path of each (fat) URL

"WWW-Authenticate" header field information

QUESTION 7

Which of the following statements about HTTP are TRUE?

1) The HTTP HEAD method can be used to determine whether a given URL refers to an existing web resource.

2) The HTTP header field "Last-Modified" is used as an HTTP request that informs the server of the client's latest version of a given web resource.

3) All information retrieved via the HTTP GET method can also be retrieved via the HTTP HEAD method.

4) The "Content-Length" HTTP header is used as an HTTP request header to inform the server which parts of a web resource a client wants to receive.

Only 1) and 3)

➡ Only 1)

Only 4)

Only 3) and 4)

QUESTION 8

Which of the following statements about URLs are TRUE?

1) When a URL starting with scheme http is entered into the browser's address bar, the browser uses port 80 by default.

2) When accessing the URL https://github.com/chauff/IN4325#course-description via the browser, the server sends the whole resource, not just the resource part that starts at fragment #course-description.

3) The URL https://duckduckgo.com/?q=chess&t=h_&ia=web contains an empty path and and several query components.

4) URLs with Unicode characters can be converted reversibly and uniquely to an ASCII string with Punycode encoding.

➡ All four statements are true.

Only 1) and 4)

Only 2) and 3) and 4)

Only 1) and 2) and 3)

QUESTION 9

Which of the following statements about web caches are TRUE?

1) Web caches have no impact on the processing power of origin servers.

2) Web caches function as the backup of the web: they keep a copy of every resource on the web.

3) Web caches rely on the "Last-Modified" HTTP header field to determine when a copy becomes invalid.

4) Web caches lead to increased distance delay.

➡ Only 1)

Only 4)

Only 2) and 4)

Only 1) and 3)

QUESTION 10

Consider the following HTML snippet. Which of the following renderings corresponds to the code shown here?

```
<html>
<head>
    <style>
        td {
            padding: 10px;
        }

        td:first-child {
            background: tomato;
        }

        td:last-child {
            background: gold;
        }

        td:first-of-type {
            background: black;
            color: white;
        }
    </style>
</head>
<body>
    <table>
        <tr>
            <th>Today's Tasks</th>
            <th>Tomorrow's Tasks</th>
        </tr>
        <tr>
            <td>WDT resit</td>
            <td>IDM exam</td>
        </tr>
        <tr>
            <td>Sports</td>
            <td>Sports</td>
        </tr>
    </table>
</body>
</html>
```

Note that the small colored squares are not part of the code but just a helper for you to identify the colors correctly.

Today's Tasks	Tomorrow's Tasks
WDT resit	IDM exam
Sports	Sports

Today's Tasks	Tomorrow's Tasks
WDT resit	IDM exam
Sports	Sports

Today's Tasks	Tomorrow's Tasks
WDT resit	IDM exam
Sports	Sports

Today's Tasks	Tomorrow's Tasks
WDT resit	IDM exam
Sports	Sports

QUESTION 11

Consider the HTML snippet below:

```
<html>
<head>
    <style>
        /* CSS rules go here */
    </style>
</head>
<body>
    <div id="description">
        <h1>Quartett</h1>
        <h2>
        </h2>
    </div>
    <div id="logo">
        <div class="card" id="c1">
            <span class="attribute">A1</span>
            <span class="attribute">A2</span>
            <span class="attribute">A3</span>
            <span class="attribute">A4</span>
            <span class="attribute">A5</span>
        </div>
        <div class="card" id="c0">
            <span class="attribute">A6</span>
            <span class="attribute">A7</span>
            <span class="attribute">A8</span>
            <span class="attribute">A9</span>
            <span class="attribute">A10</span>
        </div>
    </div>
    <div id="plant">
    </div>
</body>
</html>
```

Which of the following CSS rules leads to the following rendering:

```
#c1 :not(.attribute){
  color: gold;
}

#logo :not(.attribute){
  color: gold;
}

#c2 :not(.attribute){
  color: gold;
}

:not(.attribute){
  color: gold;
}
```

QUESTION 12

Consider the following HTML snippet:

```
<html>
<head>
    <style>
        /* CSS rules */
    </style>
</head>
<body>
    <div id="description">
        <h1>Quartett</h1>
        <h2>
        </h2>
    </div>
    <div id="logo">
        <div class="card" id="c1">
            <span class="attribute">A1</span>
            <span class="attribute">A2</span>
            <span class="attribute">A3</span>
            <span class="attribute">A4</span>
            <span class="attribute">A5</span>
        </div>
        <div class="card" id="c0">
            <span class="attribute">A6</span>
            <span class="attribute">A7</span>
            <span class="attribute">A8</span>
            <span class="attribute">A9</span>
            <span class="attribute">A10</span>
        </div>
    </div>
    <div id="plant">
    </div>
</body>
</html>
```

Which CSS rule does NOT lead to the following rendering?

```
div div {
  color: gold;
}

body, div {
  color: gold;
}
div {
  color: gold;
}
div>div {
  color: gold;
}
```

QUESTION 13

Consider the following HTML snippet:

```
<html>

<head>
    <style>
        * {
            height: 100%;
            padding: 0;
            margin: 0;
        }

        div {
            width: 50%;
            height: 50%;
        }

        /* CSS rules */
    </style>
</head>

<body>
    <div id="A">
    </div>

    <div id="B">
    </div>

    <div id="C">
    </div>

    <div id="D">
    </div>
</body>

</html>
```

The addition of which CSS rules leads (*note: the colored squares in the answers are not part of the code but a helper for you to identify the colors correctly*) to the following rendering of the page?

```
#A {
    background-color: gold;
    float: right;
}

#B {
    background-color: tomato;
}

#C {
    background-color: cornflowerblue;
    float: right;
}

#D {
    background-color: black;
}
```

```
#A {
    background-color: gold;
}

#B {
    background-color: tomato;
    float: right;
}

#C {
    background-color: cornflowerblue;
}

#D {
    background-color: black;
    float: left;
}
```

```
#A {
    background-color: gold;
    float: left;
}

#B {
    background-color: tomato;
    float: left;
}

#C {
    background-color: cornflowerblue;
    float: left;
}

#D {
    background-color: black;
    float: left;
}
```

This rendering cannot be achieved with any of the listed CSS rules.

QUESTION 14

Consider the following HTML snippet. Approximately what percentage of time is the string "Hello World!" visible in the web browser?

```
<html>

<head>
    <style>
 div {
  position: absolute;
  width: 100%;
  height: 100%;

  animation: animate ease-out 5s infinite;
}

@keyframes animate {

  from { opacity: 0; }
  74% { opacity: 0;}
  75% { background-color: gold;  opacity: 0.6; }
  76% { background-color: gold;  opacity: 0.2; }
  80% { background-color: gold;  opacity: 0.95; }
  82% { opacity: 0; }
  92% { opacity: 0; }
  93% { background-color: gold;  opacity: 0.5; }
  94% { background-color: gold;  opacity: 0.2; }
  96% { background-color: gold;  opacity: 0.9; }
  to { opacity: 0; }
}
    </style>
</head>

<body>
    <div>Hello World!</div>
</body>

</html>
```

➡ 20%

The string is visible at all times.

50%

1%

QUESTION 15

After executing the JavaScript code snippet below, how many variables have a global scope?

```
1   {
2       let a = 5;
3   }
4
5   function outer(a) {
6       let b = 7;
7       function inner(c) {
8           a = 12;
9           b = 11;
10      }
11      inner(b);
12      d = 12;
13  }
14
15  outer(6);
```

0

➡ 1

2

4

QUESTION 16

After executing the JavaScript code snippet below, what is the output on the Web console?

```
1   var Game = function (n, p) {
2       this.name = n;
3       this.numPlayers = p;
4   };
5
6   let tennis = new Game("Tennis", 2);
7
8   tennis.setType = function (s) {
9       this.type = "[" + s + "]";
10  }
11
12  Game.prototype.setType = function (s) {
13      this.type = "/" + s + "/";
14  };
15
16  tennis.setType("outdoors");
17  console.log(tennis.type);
18
19  Game.prototype.setType = function (s) {
20      this.type = "(" + s + ")";
21  }
22  tennis.setType("outdoors");
23  console.log(tennis.type);
```

➡ [outdoors]
[outdoors]

/outdoors/
(outdoors)

/outdoors/
/outdoors/

Executing this code snippet leads to an error as an object's methods cannot be redefined once the object has been created.

QUESTION 17

Consider the following JavaScript code snippet. It contains two missing lines (BLANK 1 and BLANK 2).

```
1   var locations = ["online", "offline"];
2
3   var game = {
4       name: "Tennis",
5       details:{
6           numPlayers: 2,
7           locations: ["indoors", "outdoors"],
8           getLocations: function(){
9               var self = this;
10              return self.locations;
11          }
12      }
13  };
14
15  var locationFunc = game.details.getLocations;
16
17  // BLANK 1
18  console.log( locationFunc())
19
20  // BLANK 2
21  console.log( locationFunc() )
```

What are the correct code snippets for the two blanks (*note: the blank line may remain empty to achieve this goal*) to achieve the following output (in this order)?

```
▸ Array [ "indoors", "outdoors" ]
▸ Array [ "online", "offline" ]
```

➡ locationFunc =
locationFunc.bind(game.details);
locationFunc =
game.details.getLocations;

locationFunc =
locationFunc.bind(game);
locationFunc = function(){return
self.locations};

[BLANK 1 remains empty]
locationFunc = function(){return
this.locations};

[BLANK 1 remains empty]
locationFunc = *locations;*

QUESTION 18

Consider the following web application: it is a simple stop watch with three buttons, each one counting down a different number of seconds. A click on a button starts the countdown, while the countdown lasts, the button is disabled. Once the countdown reaches zero, a beep sounds and the button becomes enabled again to start another countdown.

The code below misses the implementation of the start() function (here: line 16):

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <title>Stop watches</title>
5           <script>
6           var beep;
7             window.onload = function() {
8               let buttons = document.getElementsByTagName("button");
9               for(let i=0; i<buttons.length; i++){
10                  buttons[i].onclick = start;
11              }
12              beep = document.getElementById("beep");
13            };
14
15            function start() {
16              //JAVASCRIPT
17            }
18
19            var intervals = {};
20
21            function updateSeconds(button){
22                let num = parseInt(button.innerHTML);
23                num -= 1;
24                button.innerHTML = num+" seconds to ping";
25
26                if(num==0){
27                  beep.play();
28                  clearInterval(intervals[button.id]);
29                  button.innerHTML = button.id.substr(4)+" seconds to ping";
30                  button.disabled = false;
31                }
32            }
33          </script>
34      </head>
35      <body>
36          <audio src="https://www.soundjay.com/button/beep-01a.wav" autostart="false" width="0" height="0"
            id="beep""></audio>
37          <button id="stop5">5 seconds to ping</button>
38          <button id="stop60">60 seconds to ping</button>
39          <button id="stop300">300 seconds to ping</button>
40      </body>
41  </html>
```

Which of the following start() functions leads to the desired behaviour?

```
function start() {
    this.disabled = true;
    //last input parameter of setInterval provides the input argument for updateSeconds()
    intervals[this.id] = setInterval(updateSeconds, 1000, this);
  }
```

```
function start() {
    buttons[i].disabled = true;
    //last input parameter of setInterval provides the input argument for updateSeconds()
    intervals[buttons[i].id] = setInterval(updateSeconds(), 1000, buttons[i]);
  }
```

```
function start() {
    let buttons = document.getElementsByTagName("button");
    for(let i=0; i<buttons.length; i++){
        if(buttons[i].clicked==true){
            buttons[i].disabled = true;
            //last input parameter of setInterval provides the input argument for updateSeconds()
            intervals[buttons[i].id] = setInterval(updateSeconds(), 1000, buttons[i]);
        }
    }
  }
```

None of the code snippets achieves this behavior as the function setInterval() is never defined in this application.

QUESTION 19

Which of the following JavaScript snippets prints out 30 after execution?

1)

```
var f = ( function myfunc1(a){
    var c = 2 * a;
    return function myfunc2(b){
        return 3 * c;
    }
})(5);

console.log(f(4));
```

2)

```
var f = ( function myfunc1(a){
    var c = 2 * a;
    return function myfunc2(b){
        return 10 + c;
    }
})(10);

console.log(f());
```

3)

```
var f = function myfunc1(a){
    return function myfunc2(b){
        return 5 + a * b;
    }
};

console.log(f(5)(5));
```

4)

```
var f = function myfunc1(a){
    return function myfunc2(b){
        return 5 + a * b;
    }
};

console.log(f(5,5) )
```

- Only 1), 2) and 3)

All of them.

Only 2)

Only 1) and 4)

QUESTION 20

Consider the HTML snippet below:

```
<html>
    <head>
        <script>
            //JavaScript
        </script>
    </head>

    <body>
        <ul id="list">
            <li>Tennis</li>
            <li>Rugby</li>
            <li>Football</li>
        </ul>
    </body>
</html>
```

The list is by default rendered with bullet points. To change this we need a piece of JavaScript that turns the list into a numbered list of items:

1) Tennis
2) Rugby
3) Football

Which of the following JavaScript snippets achieves this goal? Assume that the JavaScript has been added to the <head> section of the HTML page.

```
window.onload = function(){
    var list = document.getElementById("list");
    list.style.listStyle="none";
    var items = list.getElementsByTagName("li");
    for (var i = 0; i < items.length; ++i) {
        items[i].innerHTML = (i+1)+") "+items[i].innerHTML;
    }
}
```

```
window.onload = function(){
    var list = document.getElementsByTagName("ul")[0];
    list.style.listStyle="none";
    var items = list.getElementByTagName("li");
    for (var i = 0; i < items.length; ++i) {
        items[i].innerHTML = (i+1)+") "+items[i].innerHTML;
    }
}
```

```
let module = function(){
    var list = document.getElementById("ul");
    list.style.listStyle="none";
    var items = body.getElementsByTagName("ul li");
    for (var i = 0; i < items.length; ++i) {
        items[i].innerHTML = (i+1)+") "+items[i].innerHTML;
    }
}();
```

None of the JavaScript code snippets listed here achieve this goal: an unordered list cannot be turned into a numbered list

QUESTION 21

What is the console output when executing the Node.js script below?

```
1   var fs = require("fs");
2
3   var loginAttempts = 0;
4
5   function f(done){
6       fs.readFile('n.txt', function(err, fileContents){
7           var n = parseInt(fileContents); //n.txt contains 100
8           n += 1;
9           done(loginAttempts++);
10      });
11  }
12
13  f(function(n){
14      console.log(n);
15  });
```

➡ 0

100

101

1

QUESTION 22

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

1) A file is its own module.

2) A file accesses its own module definition through the _self variable.

3) Modules are cached after they have been loaded N times in an application, where N is a threshold fixed by the runtime environment.

4) Node.js has several core modules compiled into the binary which load automatically when Node.js starts.

Only 1) and 2)

➡ Only 2) and 3)

Only 3) and 4)

Only 4) and 1)

QUESTION 23

Consider the following Node.js script which is started on the local machine:

```
1   var express = require("express");
2   var http = require("http");
3
4   var m1 = require("./StatModule");
5   var m2 = require("./StatModule");
6
7   const port = 3000;
8   var app = express();
9   http.createServer(app).listen(port);
10  console.log(m1.getCounter());
```

with StatModule.js looking as follows:

```
1   var Stats = function(){
2       this.counters = new Array(100).fill(0); //array with 100 elements initialized to 0
3       this.increment(0);
4   }
5
6   Stats.prototype.isValidID = function(id){
7       if(Number.isInteger(id) && id>=0 && id<=this.counters.length)
8           return true;
9
10      return false;
11  }
12
13  Stats.prototype.increment = function(id){
14      if( this.isValidID(id))
15          this.counters[id]++;
16  }
17
18  Stats.prototype.decrement = function(id){
19      if(this.isValidID(id) && this.counters[id]>0){
20          this.counters[id]--;
21      }
22  }
23
24  Stats.prototype.getCounter = function(id){
25      if(this.isValidID(id))
26          return this.counters[id];
27      return undefined;
28  }
29
30  module.exports = Stats;
```

What is logged on the console after the script's execution?

- TypeError: m1.getCounter is not a function

0

1

undefined

QUESTION 24

Consider the Node.js script below. It is started on the local machine.

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

var app = express();
const port = 3010;
http.createServer(app).listen(port);

function parseGrade(u){
    var query = url.parse(u, true).query;
    var grade = (query["grade"]!=undefined) ? query["grade"] : "0";
    var nGrade = Number(grade);
    return nGrade;
}

app.get("/feedback",
    function(req, res, next){
        if( parseGrade(req.url)>5.5){
            next();
        }
        res.send("Not so good");
    },
    function(req, res, next){
        if( parseGrade(req.url)>7.5){
        }
        res.send("Nice!");
    },
    function(req, res){
        res.send("Very good");
        next();
    }
)

app.get("/feedback", function(req, res){
    res.send("-")
});
```

The following four URLs (in this order) are accessed:

```
http://localhost:3010/feedback?grade=5.5
http://localhost:3010/feedback?grade=help
http://localhost:3010/feedback?grade=15
http://localhost:3010/feedback
```

Which of the following responses will you see in the browser (in this order)?

- Not so good
  Not so good
  Nice!
  Not so good

  Not so good
  -
  Very good
  -

  -
  -
  Very good
  -

  Not so good
  Not so good
  Not so good
  Not so good

QUESTION 25

Consider the following Node.js script:

```
1   var express = require("express");
2   var http = require("http");
3   var credentials = require("./session-credentials");
4
5   var sessions = require("express-session");
6   var cookies = require("cookie-parser");
7
8   var app = express();
9   console.log(credentials.cookieSecret);
10  app.use(cookies(credentials.cookieSecret));
11  app.use(sessions(credentials.cookieSecret));
12
13  http.createServer(app).listen(3022);
14
15  app.get("/countMe", function(req, res){
16      var session = req.session;
17      if(session.views){
18          session.views++;
19          res.send("You have been here "+session.views+" times. ");
20          session.lastVisit = new Date().toLocaleDateString();
21      }
22      else {
23          session.views = 1;
24          session.lastVisit = new Date().toLocaleDateString();
25          res.send("This is your first visit!");
26      }
27  })
```

The script is started on the local machine and a user now executes the following actions (on the same machine, in a timeframe of 5 minutes).

1. The user opens Firefox and accesses http://localhost:3022/countMe.
2. The user shuts down Firefox and then starts it again.
3. The user opens Firefox and accesses http://localhost:3022/countMe.
4. The user's browser remains open.

How many different sessions does the Node.js server maintain at the end of these actions?

0, as the server-side script never explicitly establishes a session.

0, as the middleware components cookie-parser and express-session are included in the wrong order in this script.

➡ 2, as the session timeout (usually set to 20 minutes) has not been reached.

1, as the user has used a single browser on a single physical device.

QUESTION 26

All four statements are true.

➡ Only 1), 2) and 4)

Only 2) and 4)

Only 3) and 4)

QUESTION 27

Consider the following Node.js script:

```
1   var express = require("express");
2   var http = require("http");
3   var credentials = require("./session-credentials");
4
5   var sessions = require("express-session");
6   var cookies = require("cookie-parser");
7
8   var app = express();
9   console.log(credentials.cookieSecret);
10  app.use(cookies(credentials.cookieSecret));
11  app.use(sessions(credentials.cookieSecret));
12
13  http.createServer(app).listen(3024);
14
15  app.get("/counts?([MU][e]s?)+", function(req, res){
16      var session = req.session;
17      if(session.views){
18          session.views++;
19          res.send("You have been here "+session.views+" times. ");
20          session.lastVisit = new Date().toLocaleDateString();
21      }
22      else {
23          session.views = 1;
24          session.lastVisit = new Date().toLocaleDateString();
25          res.send("This is your first visit!");
26      }
27  })
```

and these four routes:

1) http://localhost:3024/countUs
2) http://localhost:3024/countMeMeMes
3) http://localhost:3024/countsMUes
4) http://localhost:3024/countssss

Which of these 4 routes will match the route defined in app.get()?

Only 1) and 2)

➡ Only 2)

Only 3) and 4)

None of the routes match the route defined in app.get().

QUESTION 28

Consider the two files, foo.js and bar.js:

foo.js

```
1   module.exports = function() {
2       return {
3           name: "Tennis",
4           numPlayers: 2,
5           locations: ["indoors", "outdoors"],
6           getName: function(){return this.name;}
7       }
8   };
```

bar.js

```
1   var game1 = require("./foo")();
2   game1.name = "Tennis Clash";
3   var game2 = require("./foo")();
4   game2.name = "Super Mario";
5
6   game2.getName = function(){
7       return "["+this.name+"]";
8   }
9
10  console.log(game1.getName());
```

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

Tennis

➡ Tennis Clash

[Tennis]

[Tennis Clash]

QUESTION 29

Which of the following statements about Middleware components are TRUE?

1) Middleware components have three parameters.

2) Middleware components cannot change the request and response objects.

3) Middleware components can end the request-response cycle.

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

➡ Only 1) and 3)

All statements are true.

Only 3) and 4)

Only 1) and 2)

QUESTION 30

Browser B currently has no stored cookies. The user opens B and visits the TU Delft homepage (https://www.tudelft.nl). In the HTTP response, the following four cookies are sent:

```
Set-Cookie: bg=white; Expires=Fri, 01-Apr-2022 21:47:38 GMT; Path=/; Domain=weer.nl; HttpOnly
Set-Cookie: pref=1; Path=/; Domain=tudelft.nl
Set-Cookie: dom=23; Expires=Thu, 01-Jan-1970 00:00:01 GMT; Path=/;
Domain=chemistry.tudelft.nl; HttpOnly
Set-Cookie: view=mobile; Path=/; Domain=tudelft.nl; secure
```

How many of the cookies will be rejected (i.e. not stored in either permanent or temporary storage) by the browser?

0

1

➡ 2

3

QUESTION 31

Browser B currently has no stored cookies. The server sends the following four cookies to B:

```
Set-Cookie: bg=white; Expires=Fri, 01-Apr-2018 21:47:38 GMT; Path=/; Domain=tudelft.nl; HttpOnly
Set-Cookie: pref=1; Path=/; Domain=tudelft.nl
Set-Cookie: dom=23; Expires=Thu, 01-Jan-1970 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 and when prompted opts for the browser's session restore feature (i.e. all tabs were saved and are restored to the state just before the crash). How many cookies can the user access after the restart with client-side JavaScript?

- 0
- 1
- ➡ 2
- 3

QUESTION 32

Which of the following roles of the OAuth 2.0 authorization framework are correctly described?

1) The client is the entity that grants access to a protected resource.

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

3) The resource owner makes protected resource requests on behalf of the client and with its authorization.

4) The authorization server issues access tokens to the client after successfully authenticating the resource owner and obtaining authorization.

- ➡ Only 2) and 4)
- All of them.
- Only 1) and 3)
- Only 2)

QUESTION 33

Consider the following code (*note: the colored square is not part of the code but just a helper for you to identify the color correctly*):

```
1   <html>
2   <body style="background-color: tomato; height: 100%; width: 100%" onclick="F()">
3
4   <script>
5
6       let colors = ["Gold", "Crimson", "DeepPink", "LightGreen", "Moccasin", "Teal",
        "Wheat", "Peru", "PowderBlue", "Orchid", "Indigo", "SlateBlue"];
7       function F(){
8           if( document.cookie.includes("cook=")!==true){
9
10              console.log("no cookie ...");
11              let myColor = document.body.style.backgroundColor;
12              let r = Math.floor(Math.random()*colors.length);
13
14              document.body.style.backgroundColor = colors[r];
15              document.cookie = "cook="+colors[r]+";max-age=1";
16          }
17      }
18  </script>
19  </body>
20  </html>
```

Assume that this code is stored in some file and opened in a modern browser. Which of the following statements is true?

- Every click on the browser tab will change the background color as long as each click is at least 1 second apart.

- The background color will always be "tomato" as the onclick attribute has no effect on the <body> element.

- After the first click on the browser tab, the background color will change and remain in this color, independent of the number of clicks that follow.

- The background color will always be "tomato" as document.cookie is used incorrectly in line 8: its key/value pairs cannot be accessed through methods of the String object.

QUESTION 34

Which of the following approaches is most likely to secure an application against cross-site request forgery?

Use of state-of-the-art encryption algorithms to store the data on the server.

➡ Use of reauthentication and CAPTCHA mechanisms.

Avoid the use of direct object references; the use of objects should include an authorization subroutine.

Validate all user input and escape generated output.

QUESTION 35

Which of the following approaches is not suitable for an attacker to attempt the the injection of malicious data into a server-side application?

URL parameter manipulation

Hidden HTML form field manipulation

Cookie manipulation

➡ Appcache manipulation

QUESTION 36

You are tasked with writing a web application that contains a session management component. You decide to use as session ID the concatenation of the current POSIX time (i.e. the number of seconds elapsed since 00:00:00 UTC January 1, 1970) and a counter starting at 100 that is incremented by a random integer between 1 and 100 each time a new user requests a session at the same POSIX timestamp. Which main vulnerability arises in this scenario?

Reflected XSS

Denial-of-service attack

➡ Session hijacking

CSRF token hijacking

QUESTION 37

Web portals may secure themselves against CSRF attacks using CSRF tokens. Those tokens often appear as hidden fields in HTML forms:

```
1  <form action="/transfer.do" method="post">
2      <input type="hidden" name="CSRFToken" value="OT...A">
3      ...
4  </form>
```

Which of the following characteristics should a CSRF token have?

1. The token should be verifiable on the server-side.
2. The token should be unique per user session.
3. The token should allow re-authentication through CAPTCHAs.
4. The token value should be the base64-encoding of the session ID.

Only 1), 2) and 4)

➡ Only 1) and 2)

Only 3) and 4)

Only 1)