PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

St. George Campus

DECEMBER 2016 EXAMINATIONS

CSC 309H1F
Instructor:
Karen Reid
Duration: 3 hours

Examination Aids: None

Student Number: |_|_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): _____

First (Given) Name(s): _____

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This final examination consists of 8 questions on 15 pages. A mark of at least 29 out of 73 on this exam is required to pass this course. *When you receive the signal to start, please make sure that your copy of the examination is complete.*

Answers that contain a mixture of correct and incorrect or irrelevant statements will not receive full marks.

Good Luck!

MARKING GUIDE

1: _____/10
2: _____/ 8
3: _____/10
4: _____/ 6
5: _____/ 7
6: _____/ 8
7: _____/10
8: _____/14

TOTAL: _____/73

Question 1. [10 MARKS]

TRUE FALSE Web browsers are stateless: if a browser isn't displaying a page for a particular server it need not retain any state related to that server.

TRUE FALSE JavaScript is required for AJAX calls to work.

TRUE FALSE JavaScript is required to use JSON.

TRUE FALSE JQuery is primarily used in server code.

TRUE FALSE HTML5 local storage allows browser data to be shared between apps running on the same browser.

TRUE FALSE HTTPS should be used for all web pages served to authenticated users.

TRUE FALSE The DOM has the same structure as the HTML file.

TRUE FALSE Once a Web server returns a cookie to a browser, the cookie will be included in all future requests from the browser to the same server.

TRUE FALSE A Promise can be resolved multiple times, and the value of each resolution is immutable.

TRUE FALSE The Node event loop is multi-threaded so multiple requests can be handled simultaneously.

Question 2. [8 MARKS]

Part (a) [3 MARKS]

Give one example of a semantic HTML element and one example of a non-semantic HTML element. Why is it recommended to use semantic elements where sever possible?

Part (b) [2 MARKS]

HTTP is a stateless protocol. What does this mean? How do cookies and sessions solve this problem?

Part (c) [2 MARKS]

Explain how GET and POST differ in terms of how they send information, and what kind of information transfer they are used for.

Part (d) [1 MARK]

Describe a limitation of a GET request compared to a POST request.

Question 3. [10 MARKS]

Part (a) [3 MARKS] Name the three different ways in which CSS can be included in a web page.

Part (b) [2 MARKS]

One common way of using React is to add inline style attributes dynamically. Give one advantage and one disadvantage of this approach.

Part (c) [2 MARKS] Explain what "responsive web design" means.

Part (d) [1 MARK] How does the `@media` CSS query help to implement a responsive UI?

Part (e) [2 MARKS] Given the following HTML and associated CSS file, and assuming no errors:

```
<style>
    .apple {
        color: red;
    }
</style>

<h1 class='apple' style='color: green;'>Apple</h1>

<h2 id='pear'>Pear</h2>
```

```
#pear {
    color: red;
}
.apple {
    color: yellow;
}
h1 {
    color: gold;
}
h2 {
    color: brown;
}
```

The colour of the text Apple is _____

The colour of the text Pear is _____

Question 4. [6 MARKS]

Each of the three template languages we discussed in class has different design principles. Briefly explain the reasoning behind the design choice of each one.

Part (a) [2 MARKS] Mustache - HTML with variable replacement.

Part (b) [2 MARKS] PUG (formerly Jade) - A different syntax for specifying HTML elements. Embedded JavaScript is allowed.

Part (c) [2 MARKS] EJS - Embedded JavaScript inside HTML.

Question 5. [7 MARKS]

Part (a) [2 MARKS]

Describe a simple attack that could be executed if browsers did not implement the Same-Origin Policy.

Part (b) [3 MARKS]

Briefly describe 3 security properties of HTTPS that allow users to share confidential information over the Web.

Part (c) [2 MARKS]

Given the following pairs of potential attacks and countermeasures, state whether the countermeasure is *ineffective*, *somewhat effective*, or *very effective* against the attack, and justify your answer:

a) Cross-site Scripting : escaping user-supplied data when generating HTML

b) SQL injection : escaping user-supplied data when generating HTML

Question 6. [8 MARKS]

Part (a) [2 MARKS]

Explain what a callback is.

Part (b) [2 MARKS]

When running a program containing the following function, the user discovered that the `console.log` line printed `undefined`. Assuming that `Book.find` is operating correctly, explain why `allBooks` is undefined.

```
exports.findAll = function(req, res) {

    var allBooks;
    Book.find({}, function(err, allBooks) {
        if (err) throw err;
        res.send(allBooks);
    });
    console.log(allBooks)
};
```

Part (c) [4 MARKS]

Write the function `makeSequence` so that the code snippet below produces the output shown in the comments.

```
"use strict";

let byTwos = makeSequence(2);
let byFives = makeSequence(5);

let outputTwos = ""
let outputFives = ""
for(let i = 0; i < 5; i++) {
    outputTwos += parseInt(byTwos()) + " ";
    outputFives += parseInt(byFives()) + " ";
}
console.log(outputTwos);  // Prints: 2 4 6 8 10
console.log(outputFives); // Prints: 5 10 15 20 25
```

Question 7. [10 MARKS]

In Assignment 2 you implemented a REST API for a TA application.

Suppose that the response for `GET /applicants` is the following JSON format. `courses` has the list of courses for which the TA has applied.

```
{
  "tas": [
    {
      "stunum": "",
      "givenname": "",
      "familyname": "",
      "status": "",
      "year": "",
      "courses": [
         {
            "code": "",
            "rank": "",
            "experience": ""
          }
      ]
    }
  ]
}
```

We now want to add functionality to make offers to applicants. An applicant might receive zero to three offers. Each offer is for a distinct course. (We are ignoring the number of hours in an offer for this question.)

Part (a) [2 MARKS]

Now when we want to retrieve an applicant, we also want to include the offers that have been made to the applicant. There are two main options when deciding how to add offers to the above JSON response:

- Add a boolean field to each element of `courses` called `givenOffer`.
- Add a field called `offers` to the TA object. `offers` will be an array of course codes for which the applicant has received an offer.

Which approach would you recommend and why?

Part (b) [4 MARKS]

Design a RESTful route to add an offer for one applicant for a particular course. An applicant is identified by `stunum`. A course is identified by its `code`. Briefly explain your choice of method, resource, arguments and response.

Part (c) [4 MARKS]

Design a RESTful route to retrieve all offers. The response will be used to display a table of offers with the following columns:

```
Course code, Given Name, Family Name, Status
```

Briefly explain your choice of method, resource, arguments and response.

Question 8. [14 MARKS]

A very small web app displays a list of movies, and allows the user to filter them by genre (category).

The directory contains the following files. (The normal directory structure has been collapsed to simplify.)

```
prompt$ ls
assets/  index.html  movies.js  movies.json  package.json  app.js

prompt$ ls assets
showMovies.js   styles.css
```

Contents of `app.js`

```
var express = require('express');
var movies = require('./movies');
var bodyParser = require('body-parser');

var app = express();
app.use(express.static(__dirname + '/assets'));
app.use(express.static(__dirname + '/'));

app.use( bodyParser.json() );
app.use(bodyParser.urlencoded({
    extended: true
}));

app.get('/', function(req, res) {
    res.sendfile('index.html');
});

app.get('/movies', movies.findAll);
app.post('/filter', movies.findBy);

app.listen(3000);
```

Contents of `index.html`

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="styles.css?v=1.0"
          type="text/css">
    <script src="https://ajax.googleapis.com/ajax/
    libs/jquery/3.1.0/jquery.min.js"></script>
    <script src="showMovies.js"></script>
    <title>Current movies</title>
</head>
<body>
    <h1>Current movies</h1>

    <ol id="mlist">  </ol>

    <form action="filter" id="filter" method="post">
        <select id="genre" name="genre">
            <option value="All">All</option>
            <option value="Action">Action</option>
            <option value="Family">Family</option>
            <option value="Comedy">Comedy</option>
            <option value="Drama">Drama</option>
        </select>
        <input id="genreButton" type="submit" value="Submit">
    </form>
</body>
</html>
```

Part (a) [1 MARK]

Circle the files in the directory listings above that are executed or used on the client (browser).

Part (b) [2 MARKS]

Is it possible to implement this app so that only HTML and CSS are sent to the browser? (This might require changes to the files shown above.) Circle the correct answer and explain it. (No marks if explanation is missing or incorrect.)

YES NO

Question 8. (CONTINUED)

Part (c) [6 MARKS]

Given the following code from `assets/showMovies.js`, fill in the blanks in the ajax call, and complete the `buildList` function.

```
$(document).ready(function() {
    $("#filter").submit(function (e) {
       e.preventDefault();
        console.log($('form').serialize())
        $.ajax({


            url: ____________________________,


            type: ____________________________,

            data: $('form').serialize(),
            success: function result (data) {
                let movies = JSON.parse(data);
                buildList(movies);
            }
        });
    });

    getMovies();
});

function getMovies() {
    $.get('movies', function (data) {
        let movies = JSON.parse(data);
        buildList(movies);
    });
}

// Inserts each movie in the array of movies into the DOM element with id mlist
function buildList(movies) {
```

`movies.json` format:

```
{
    "current" : [
        {
            "title": "Doctor Strange",
            "genre": "Action"
        }
    ]
}
```

Question 8. (CONTINUED)

Part (d) [1 MARK]

How many callback functions appear in the code shown in part c)?

Part (e) [4 MARKS]

Complete the implementation of `findBy` below.

```
var fs = require('fs');
var movieObj;
fs.readFile('movies.json', 'utf-8', function(err, data) {
    if(err) throw err;
    movieObj = JSON.parse(data);
});

exports.findAll = function(req, res) {
    res.send(JSON.stringify(movieObj.current));
};
```

`movies.json` format (repeated from above):

```
{
    "current" : [
        {
            "title": "Doctor Strange",
            "genre": "Action"
        }
    ]
}
```

```
// findBy handles a POST request, and sends a response containing a JSON array
// of movies that match the genre provided as a POST argument.
exports.findBy = function(req, res) {
    var genre = req.body.genre;
```

Total Marks = 73

This page can be used if you need additional space for your answers.

This page can be used if you need additional space for your answers, but only if you don't tear it off.

Note: You may detach this page for easier reference.

Basic JavaScript

```
JSON.parse()
JSON.stringify()

document.getElementById(string)
document.getElementsByTagName(string)
document.createElement(string)
element.innerHTML
element.apend(element)
element.empty()
alert(value)

array.length()
array.push()
array.splice()
object.toJSON()

event.preventDefault()
```

JQuery

```
$(selector).append()
$(selector).html()
$(selector).empty()
$(selector).parent()
$(selector).insertAfter()
$(selector).on()
```

Express route handling

```
req.send()
req.get()
req.params()
req.query()
req.body()
req.route()
```