Assessment of
Web Script Programming
2018-2019

Unit Coordinator	Dr Rich Boakes <rjb@port.ac.uk>
Issued	October 2018
Code	U21264 Web Script Programming
Purpose	This document describes ***all*** the work you must complete for the Web Script unit. There are three parts to the assessment.

Schedule & Deliverables

Item	Weighting	You will receive details of your challenge...	You will submit work by...	You should get feedback by...
Client-side Coursework	25%	2018-10-19	2018-10-23	2018-11-20
Supervised Work Session	25%	2019-01-07	2019-01-07	2019-02-04
Web Application	50%	Below	2019-05-03	2019-05-31

Notes

- This is individual work.
- If you need help you should...
    - Discuss the problem with your peers.
    - Then ask Rich, Jacek or Matt.
- We hope you enjoy these challenges!

Details

Client-Side Coursework

- The work you must undertake is defined as a series of unit-tests (reflecting the industry practice of test-driven development).
- Your task is to complete various functions and make the unit-tests pass.
- You shall develop your code in a single JavaScript file which will contain any and all necessary code.
- The coursework is strictly time limited: it will be issued during the lecture, and your code must be submitted within **two working days** later.

Supervised Work Session (Server-side)

- You will be provided with a **part-finished web app** with a client but no server yet.
- The work you need to undertake is again defined as a series of unit-tests.
- Make the unit-tests pass, thus enabling the app to work.
- The coursework is run as a supervised work session (SWS) in a computer lab.
    - i.e., you will attend a timetabled session that will last several hours,

      during which you will have to write the code to make the tests pass.
- You will be able to see all of the code before the SWS, except for the unit tests.

Web Application

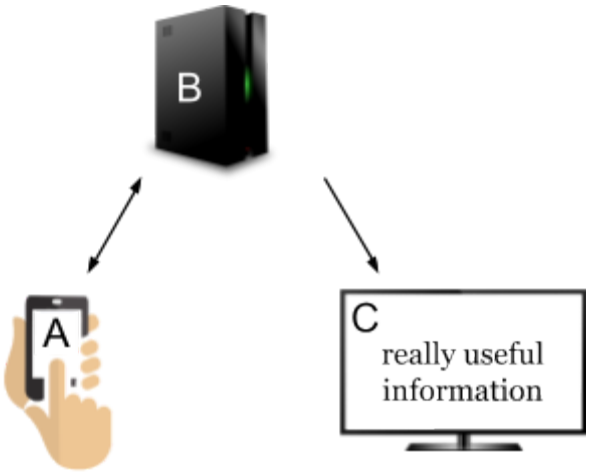
You will build an application from scratch, fulfilling these requirements:

1. The coursework needs to be submitted by deadline given above.
2. The application **must** start, and be reachable via HTTP on port 8080 when run on our local test machine.
3. The application must launch with **`npm start`**.
    - If your application requires a configuration step, this **must** be achieved via the command **`npm run setup`** - we will not undertake any other manual steps. The OS on which your setup script will run is not specified, so we recommend writing a small node utility to take care of any one-off configuration.
4. If the server does not start, we will assess the work based on the source code without the benefit of seeing it run.
5. You must include a **`README.md`** file that explains key features, how to use them, and details your design and implementation rationale.
6. You must include a **`REFLECTION.md`** where you reflect on the encountered designs and technologies and your learning.

Web Application Brief

Your challenge is to create a ***Configurable Dashboard for Unattended Displays***.

You are to specify and construct this configurable dashboard using a combination of HTML, CSS, and JavaScript, and any backing store you desire (e.g. a database).



The Unattended Display (C) does not have any inputs (keyboard, mouse, or touch), so any settings must be done remotely, via a server (B) which you must write. The settings tool must be another web page that ought to work on both smartphone (A) and desktop.

Example use cases include:

- the central area in Buckingham Building, where we have a flat panel for displaying information to staff and visitors. A combination of web pages, images, video, and data, could all be displayed. This might include short-lived notifications such as "there's free food in the kitchen" as well as long-term cyclic information, e.g. an advert for the placements office.
- on the door/wall outside a lecturer's office, there could be a small screen with helpful information for students and staff: e.g., current location/schedule, do-not-disturb, out-of-office, and urgent messages such as be-back-in-five.

Your task is to show (through your work) the extent to which you have met the learning outcomes for the unit.

The learning outcomes (as defined in the Unit Spec.) are:

1. Identify industry best practices in web application design
   (e.g. client, server and API layers).
2. Design a contemporary web application using industry best practices.
3. Critically evaluate the design and implementation of web applications.

Marking Scheme

The Coursework will be graded using academic judgement, with the following rubric used as a guide.

Topic	Description	Weighting
Functionality	How appropriate is the design (data, code, architecture)? Does it all work? How much does it do? How much is your own work as opposed to libraries?	10
Maintainability	Code style, comprehensibility and maintainability. This includes formatting, file structure, naming - everything that can help your work live on and be useful *after* it is graded, including how well the code and any documentation communicates any concepts necessary to understand the architecture and configuration of the system.	10
Usability	Ease-of-use of your system, including the use of event-driven input, background refresh, drag and drop, intuitive UI design, etc.	10
Accessibility	The appearance of your app, including use of CSS and relevant capabilities such that the product is suitable for a diverse audience.	5
Reflection	Marked for insight, analysis and evaluation of encountered designs and technologies.	5
Invention	We will award up to 10 bonus marks for unusual qualities, strengths, creativity and invention not otherwise prescribed here.	10