DT00BN92 Development of Client-side Interactive Web Applications**,** Fall 2019

# Project 1

## You can collect up to 60 points for this project.

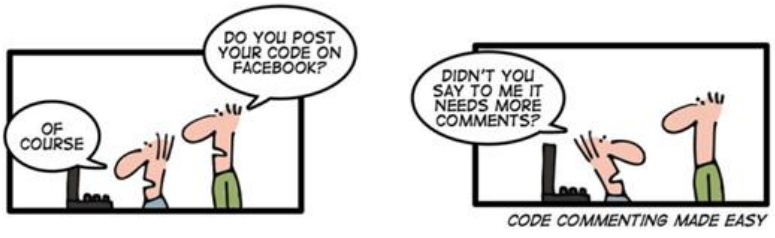
**This course work is individual.** Discussing the course tasks, assignments and specific issues with other course participants is allowed and even encouraged. However, you should be the only author of all the solutions you provide in this assignment. Team work, pair programming or copying solutions or program code from other persons is consider plagiarism and it will be handled following the Åbo Akademi protocol for such cases.

Requirements:

- Create a single html file (named *index.html*) having links to all sub-exercises in this project (exercise 1, 2 and 3).
- Complete this assignment using **exclusively** only HTML5, CSS3 and JavaScript. You have to use HTML-table-less design (no <table> element is allowed).
- You **cannot** use any CSS framework or an HTML template. Everything should be written from scratch in HTML5 and CSS3.
- Your code must be uploaded on GitHub. First, you create your repository using the following link: https://classroom.github.com/a/JoR3mOof. You must push your code to the *master* branch.
- **After** your first commit to the repo should enable the **"GitHub Pages"** option for the master branch from the *Settings*.
- Test that your page is visible from the link https://github.com/it-teaching-abo-akademi/csdewas-project-1-**YourGitUsername**
- Submit in Moodle as plain text the following info:
  - Your full name
  - Student ID at Åbo Akademi University
  - Link to the source code on GitHub https://github.com/it-teaching-abo-akademi/2018-interactive-web-apps-assignment-1-**YourGitUsername**
  - Link to the corresponding GitHub webpage: https://it-teaching-abo-akademi.github.io/2018-interactive-web-apps-assignment-1-**YourGitUsername**/

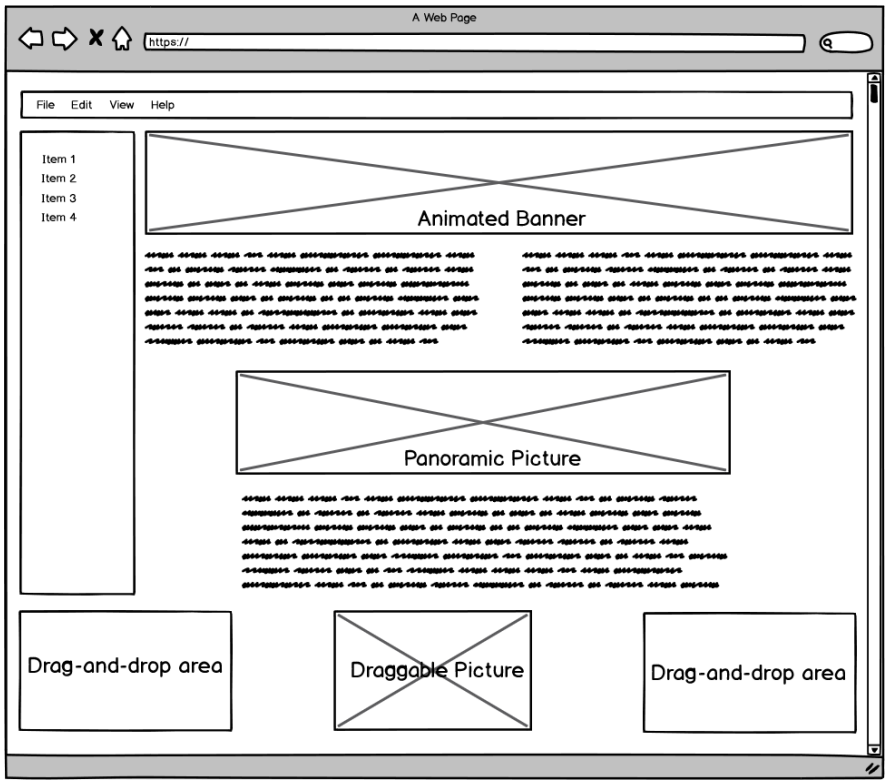
HINT:

- If you do not know which editor to use to write your code, try out Eclipse Web Tools or JetBrains WebStorm.
- The recommended browser is Chrome. Feel free to test your pages in different browsers.
- Comment your source code such that anyone reading your code can easily understand its logic.



## Exercise 1: Responsive Layout, Animations and Interactivity (20 points)

Your task is to build a **responsive** web page according to the following sketch:




- You are free to assign values to unspecified measurements as long as your page follows the layout specified in the above figure. The page does not have to be viewable in one screen, i.e., it is acceptable for the page to be scrollable.
- You are free to choose any photo to be dragged from the "Photo to Drag" box. The photo can be only dropped on the "Drag&Drop" areas. The photo can then be dragged and dropped between the two areas back and forth. For this, you need to use JavaScript as discussed in the slides.
- You are free to choose any photo for the "Panoramic Photo"
- You should implement some CSS3 based animation(s) on the animated banner. The animation(s) should play forever (infinitely). The type of animation does not matter.

Additionally, you should make the page responsive:

- The layout of your page should adjust when the size of the browser window changes. The page should look good on different devices: desktops (width>992px), tablets (width in [768px, 992px]), and phones (width <768 px).
- You can use https://www.w3schools.com/css/css_rwd_intro.asp as a source for tutorials and examples for responsive web pages. Ask google for more if needed!

For this exercise, name your HTML file ***exc1.html*** and CSS file ***exc1.css***.

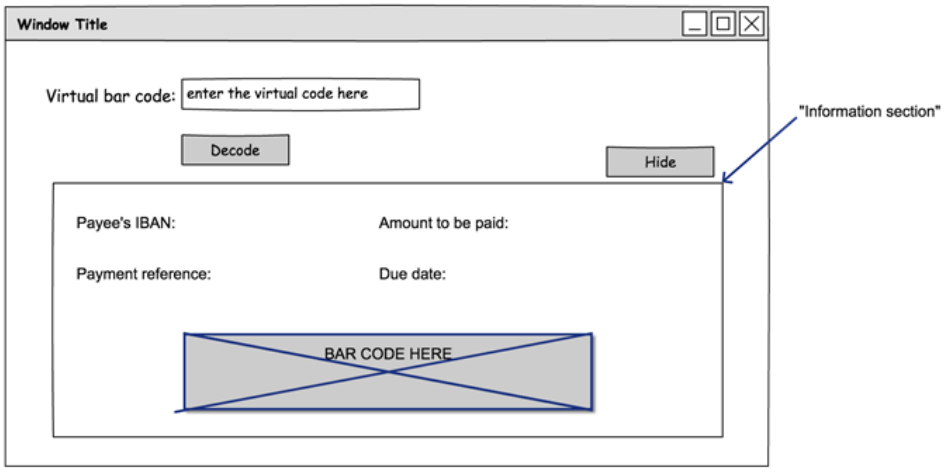
## Exercise 2: Decoding invoice information in JavaScript (20 points)

When receiving an invoice by e-mail you can sometime get as plain text a “virtual bar code” (a long string of integers) you can copy and then past into your online banking portal to speed-up the process of entering the invoice information when paying it.

This virtual bar code is a representation of the barcode usually present at the bottom of invoices. The full specification of the barcodes on invoices is available from The Federation of Finnish Financial Services in the following document.

**Step 1:** Read the document and understand the way invoice information is encoded into a bank bar code.

**Step 2:** Implement a **responsive web page** according to the following sketch (the HTML page should be named ***exc2.html***):




Requirements:

- Users should be able to enter virtual bar codes corresponding to both version 4 or version 5 of the Bank Bar Code
- When entering the virtual bar code into the form (<input type="text">) the colour of background of the text field should change to light grey
    - This should be done using unobtrusive style by attaching event handler functions to the form field
- When pushing the decode button, the virtual bar code should be decoded and the extracted information should be displayed in the information section bellow. The following information should be extracted and displayed:
    - Payee’s bank account number (IBAN format, without the country code)
    - Amount in Euro
    - Reference of the Payment
    - Due date
- If the virtual bar code cannot be decoded (e.g., due to incorrect format) an error message should be displayed.
- In addition, a bar code should be generated from the virtual bar code and be displayed instead of the “BAR CODE HERE” box on the page sketch.

    - You can use the following JavaScript library to generate your Code 128, character set C, bar code: https://github.com/lindell/JsBarcode (MIT licence)
- When pushing the “Hide button” the “information section” should be hidden, and the content of the button should be updated to “Show”. When pushing again this button, the “information section” should appear again.

To test your code, you can use the following virtual bar code (extracted from the FINANSSIALA document):

- Version 4:

  Account No. Sp. 79 4405 2020 0360 82
  Total: 4,883.15
  Reference number: 86851 62596 19897
  Due Date: 12.06.2010
  Virtual code: 4794405202003608200488315000000008685162596198971006 12

  Account No. Sampo 16 8000 1400 0502 67
  Total: 935.85
  Reference number: 78 77767 96566 28687
  Due Date: None
  Virtual code: 4168000140005026700093585000000787776796566286870000 00

- Version 5:

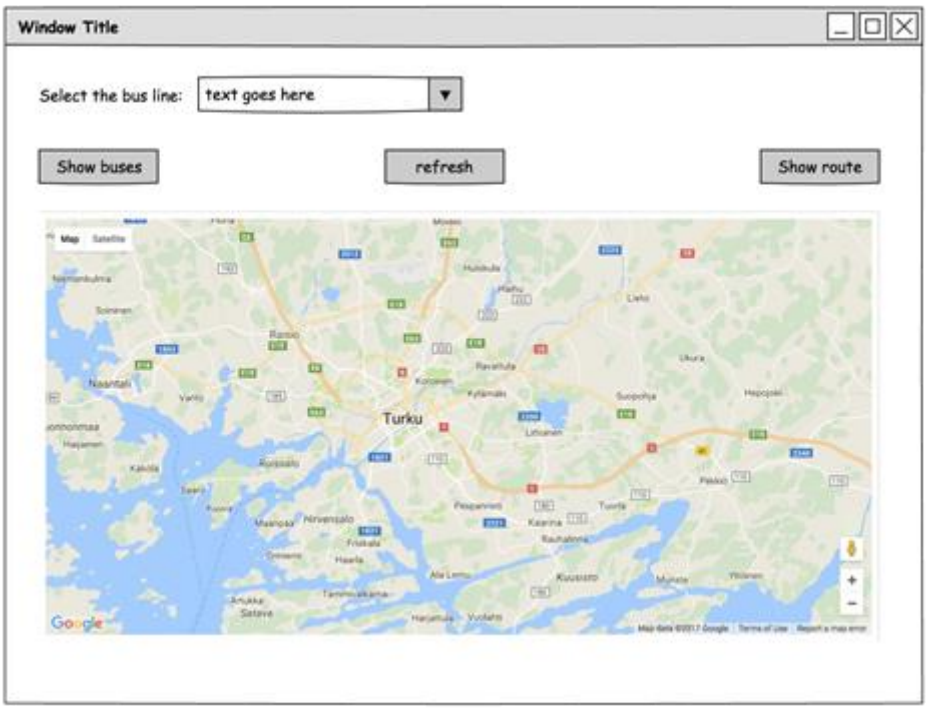
  Account No. Op. 02 5000 4640 0013 02
  Total: 693,80
  Reference number: RF02 6987 5672 0834
  Due Date: 24.7.2011
  Virtual code: 5025000464000130200069380020000000006987567208341107 24

  Account No. Handelsbanken 73 3131 3001 0000 58
  Total: 0,00
  Reference number: RF10 8686 24
  Due Date: 9.8.2013
  Virtual code: 5733131300100005800000000100000000000000000868624130809

## Exercise 3: Föli buses and routes (20 points)

You need to implement a **responsive web page** displaying the real-time location of buses in Turku and their corresponding routes.

You should implement your document according to the following sketch:




For this exercise, you are free to use any mapping JS library you like, but it is recommended that you use something like OpenLayers or Leaflet. (We will be using OpenLayers in the labs.)

Google Maps Platform is also an option, but it requires API-keys, which the others do not need. If you choose to use Google Maps Platform, you will need to create the API key yourself, and make sure the Google Maps JavaScript API is enabled for your key. *(The free daily quota provided by google map will be far enough for your application.)*

**Requirements:**

- The list of bus routes in the drop-down list should be created based on the data provided by the 'routes' API:
  - http://data.foli.fi/doc/gtfs/v0/routes-en
- When clicking the "Show route" button you should draw on the map the route for the bus line selected from the drop-down list
  - Have a look at how to draw simple polylines with OpenLayers here: https://openlayers.org/en/latest/apidoc/module-ol_geom_LineString-LineString.html
- When clicking on the "Show bus" button you should indicate on the map the current location of all buses operating on the route (bus line) selected from the drop-down list
  - The API (http://data.foli.fi/doc/siri/v0/vm-en) is made such that the returned data contains the position of all buses. You will need to parse the data to extract only the

buses corresponding to the selected route (checking the values for 'publishedlinename'). (To visualise the server response in a human readable way use following url: http://data.foli.fi/siri/vm/pretty )

- When clicking on the "Refresh" button, the location of all buses operating on the selected route is refreshed.

NOTES:

- You should implement the fetching of the data using the current date, not a hardcoded date, since the link will not be valid after a certain period, which can be shorter than the grading.
- Read carefully the documentation of the Föli API. Check the last line at the bottom of http://data.foli.fi/doc/gtfs/v0/index-en
- You might want to first check few examples explaining the JavaScript **`JSON.parse()`** function:
  - https://docs.microsoft.com/en-us/scripting/javascript/reference/json-parse-function-javascript
  - https://www.w3schools.com/js/js_json_parse.asp
- The Föli APIs are described on the following page: http://data.foli.fi/doc/index-en
  - You will need to exploit the following APIs:
    - The real-time VM - Vehicle Monitoring data: http://data.foli.fi/doc/siri/v0/vm-en
    - TSJL – GTFS static planning data http://data.foli.fi/doc/gtfs/v0/index-en
    - read carefully the section "Acquiring trip's path's coordinates" on the page http://data.foli.fi/doc/gtfs/v0/shapes-en