THE UNIVERSITY OF SYDNEY

School of Computer Science

COMP5347: Web Application Development Week 5 Tutorial: Server-side Development (Node.js and Express.js)

Learning Objectives

- Understand basic express application structure
- Get familiar with a few view template engines
- Practice simple express application to handle client input and to prepare views

Part 1: Install Express module

Create a workspace folder (let's call it comp5347) to hold your Nodejs application on your computer. Create a new folder inside this workspace to hold your actual application (let's call it nodejs-labs).

Download the "node_modules.zip" from COMP5347 Canvas' site and extract the content from it. Then, put the "node_modules" folder directly inside the nodejs-labs folder.This prepared package contains modules (i.e., dependencies) that are required to work with Express.js framework which is used in this lab session.

To install required Express.js modules on your own PC, you should run the following commands (https://expressjs.com/en/starter/installing.html) from inside the project directory (nodejs-labs):

```
npm init
npm install express --save
npm install body-parser --save
npm install path --save
npm install ejs --save
npm install pug --save
```

Part 2: Start Eclipse IDE

Download week5.zip from COMP5347 Canvas' site and extract the content in a folder. The zip file contains a JavaScript file survey.js, a CSS file, two EJS templates and one PUG template.

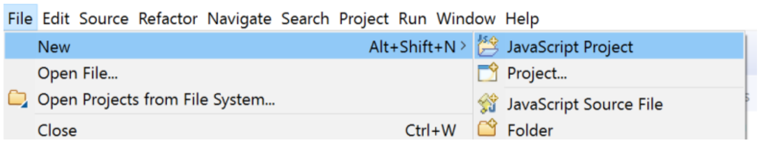
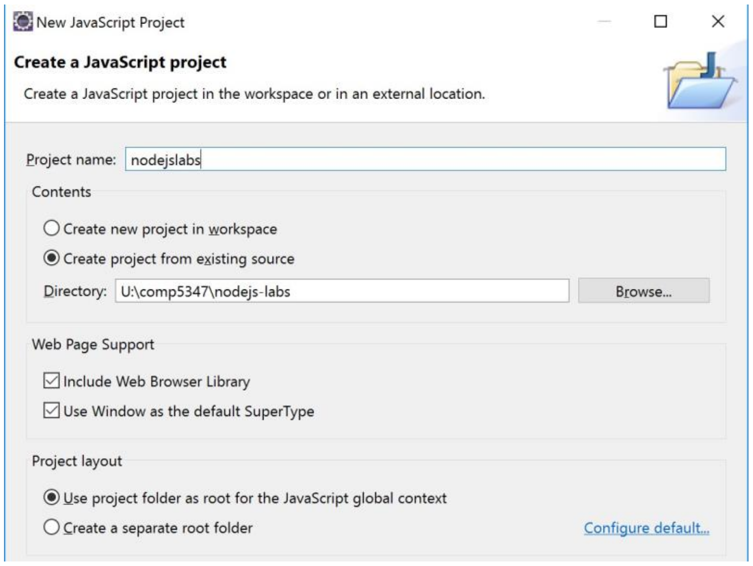

File Edit Source Refactor Navigate Search Project Run Window Help
New Alt+Shift+N >
Open File...
Open Projects from File System...
Close Ctrl+W
JavaScript Project
Project...
JavaScript Source File
Folder

Figure 1: Create a new JavaScript Project

Start Eclipse JEE Neon (the lab machine may have a few Eclipse versions installed, only the latest version Neon has integrated Node support, make sure you use the correct one). Point your workspace to the newly created workspace directory, e.g., U:\comp5347.

Click the File menu to create a new JavaScript Project. You need to specify a name and indicating that you will create project from existing source, which is the directory U:\comp5347\nodejs-labs you just created. Figure 1 and Figure 2 show the steps. You can leave the rest with default setting.


New JavaScript Project

Create a JavaScript project

Create a JavaScript project in the workspace or in an external location.

Project name: nodejslabs

Contents
- ○ Create new project in workspace
- ● Create project from existing source

Directory: U:\comp5347\nodejs-labs

Web Page Support
- ✓ Include Web Browser Library
- ✓ Use Window as the default SuperType

Project layout
- ● Use project folder as root for the JavaScript global context
- ○ Create a separate root folder

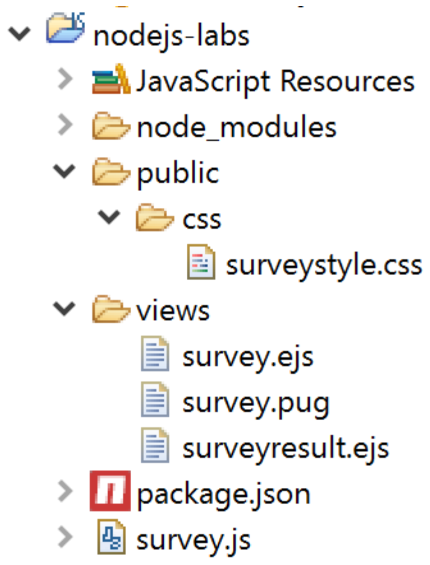
Figure 2: Specify project name and directory

Part 3: Create Application Structure

In your new project, create a few folders and sub folders as shown in Figure 3. Import those extracted files to respective folders as shown in Figure 3. The files represent a partially implemented survey application.

Part 4: Start Server

Right click survey.js on the Project Explorer panel and select from the drop down menu to "Run As" "Node.js" application. You should see a console message saying "survey app listening on port 3000".



```
nodejs-labs
  > JavaScript Resources
  > node_modules
  v public
    v css
        surveystyle.css
  v views
      survey.ejs
      survey.pug
      surveyresult.ejs
  > package.json
  > survey.js
```

Figure 3: Folder and file structure

Start Chrome and point it to "localhost:3000". You should get a screen similar to Figure
4. This is the start screen of the simple survey application. It contains a form with two groups of radio buttons. It allows a user to specify gender and preference for next mobile phone.

Make a selection and click the Submit Vote button. You will see a screen like Figure 5. It shows a template of the survey result. No actual result is displayed because

- the result handling logic is missing from survey.js

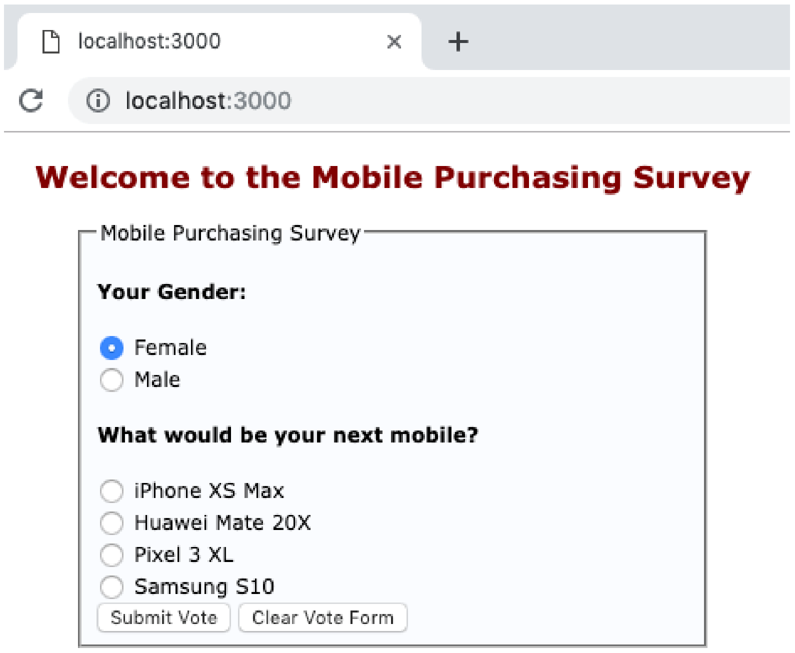
- the associated view surveyResult.ejp does not have relevant elements to display the results.

Part 5: Understand the code

The main application logic is implemented in survey.js. It initializes two global variables: products and surveyResults. The products variable points to an array of strings representing mobile phone models surveyed. The surveyResults variable points to an object of two properties: fp and mp. Property fp stores the votes by female users for each model as an array of 4 number. The array is initialized to 0. Property mp stores the votes by male users for each model as another array. It is initialized to 0 as well.

survey.js also contains two route methods. One maps to HTTP GET to url **'/'**, the other maps to HTTP POST to url '/survey'. Each with a function implemented to respond request send to the corresponding url.

When we start the server and point Chrome browser to 'localhost:3000'. This sends a GET request to the application's root: '/'. The route method simply renders a view



Welcome to the Mobile Purchasing Survey

Mobile Purchasing Survey

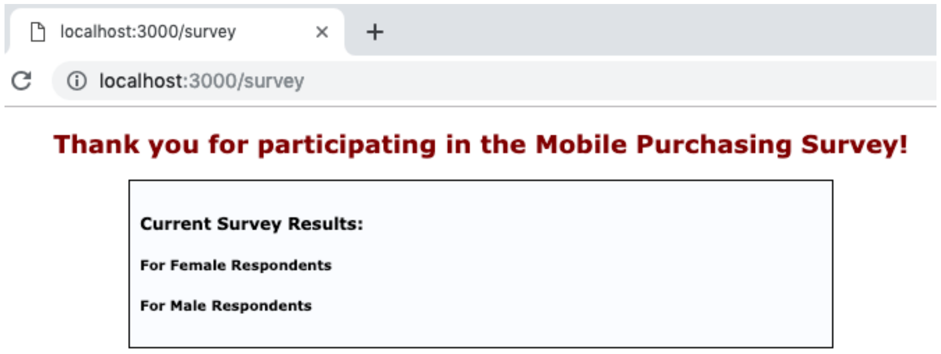
Your Gender:

[✓ Female]
Male

What would be your next mobile?

iPhone XS Max
Huawei Mate 20X
Pixel 3 XL
Samsung S10

Figure 4: Survey form


localhost:3000/survey

Thank you for participating in the Mobile Purchasing Survey!

Current Survey Results:

For Female Respondents

For Male Respondents

Figure 5: Survey result template

template survey.ejs with the products data. This would display the original survey form. The mobile phone model names are obtained from products array. A loop structure is used in survey.ejs to display the model name and the associated radio button:

```
<% for (var i = 0; i<products.length;i++){%>
<label>
    <input type="radio" name="vote" value= <%= i%> />
    <%= products[i]%>
</label> <br />
<% } %>
```

After we click the Submit Vote button, a POST request is sent to URL '/survey'. This is configured in the form element's method and action attributes. The corresponding route method again simply renders a view template surveyResult.ejs with products and surveyresults data. This template is partially implemented.

Part 6: Switching View Templates

Stop the application by clicking the stop icon on the console window. Update the render method for GET URL '/' to:

```
res.render('survey.pug',{products:products})
```

Start the server again by running survey.js as "Node.js application". Reload the root page 'localhost:3000' from Chrome, you should see the similar survey form.

The PUG template uses white space indent to represent nested HTML elements. It also provides simple ways for loop structure. For instance, the follow code displays model name using data from products variable:

```
each product,idx in products
    input(type="radio" name="vote" value= idx)
    label=product
    br
```

It has the same effect as the for loop in the ejs template. The new survey.js file has a mixture of PUG and EJS templates. Express is able to load the correct template engine based on the file extension.

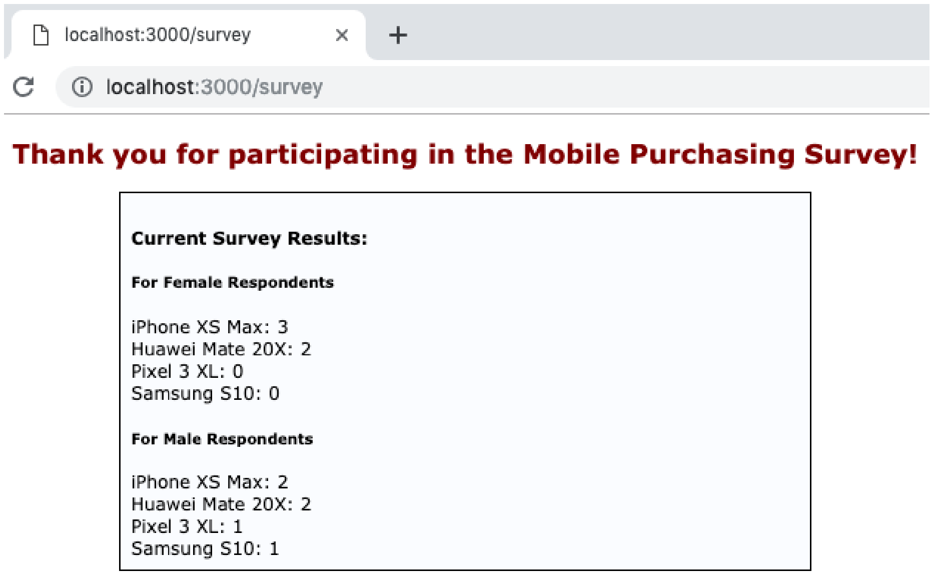
Part 7: Implement the vote taking logic

Now update the route method handling form submit to

- obtain user input gender and vote from the POST request.
- update variable surveyresults based on current user input.

The form data is parsed by middleware functions defined in body-parser module. The data is populated in req.body. You can check the logged console message to see what is contained in req.body.

You also need to update the view template to display the survey result. Figure 6 shows an example survey result. You may choose to update the given template surveyresult.ejs or write your own PUG template. If you write your own PUG template, make sure you refer to it in application code when rendering the result.



Thank you for participating in the Mobile Purchasing Survey!

Current Survey Results:

For Female Respondents

iPhone XS Max: 3
Huawei Mate 20X: 2
Pixel 3 XL: 0
Samsung S10: 0

For Male Respondents

iPhone XS Max: 2
Huawei Mate 20X: 2
Pixel 3 XL: 1
Samsung S10: 1

Figure 6: Survey result sample after several submissions