Building your first SMART on FHIR app

Workshop Tutorial

Intro to Clinical Informatics (BMI 720)

Josh Herigon & Colby Uptegraft
Boston Children's Hospital

30 Sep 2020

Table of Contents

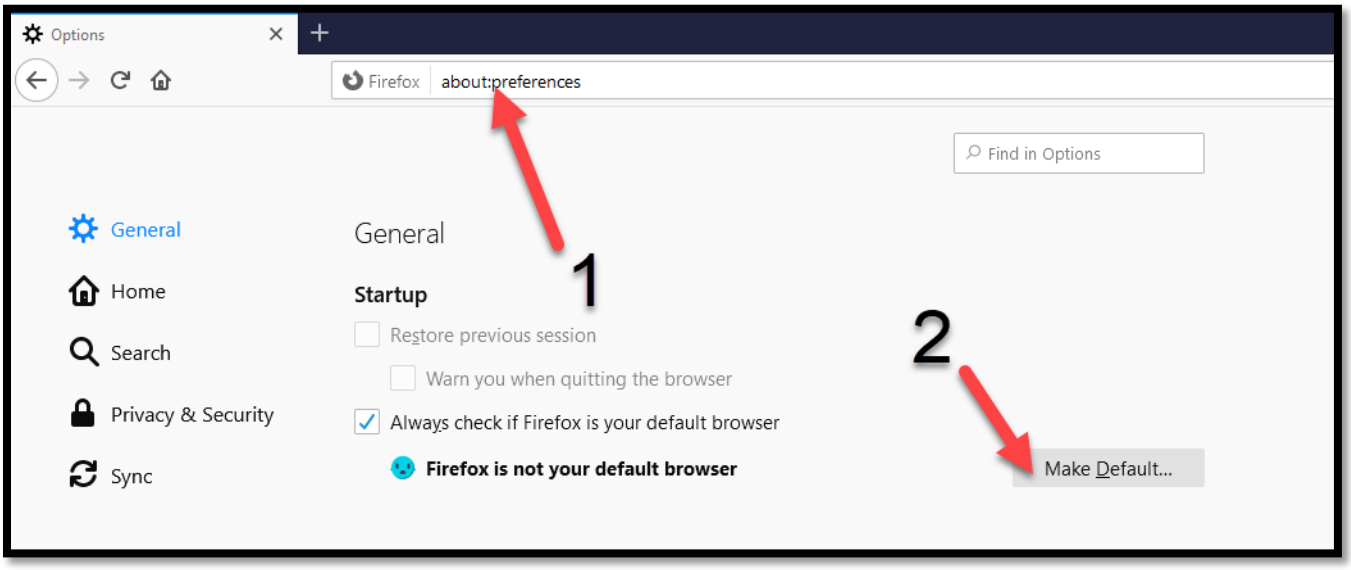
Required software / preparation

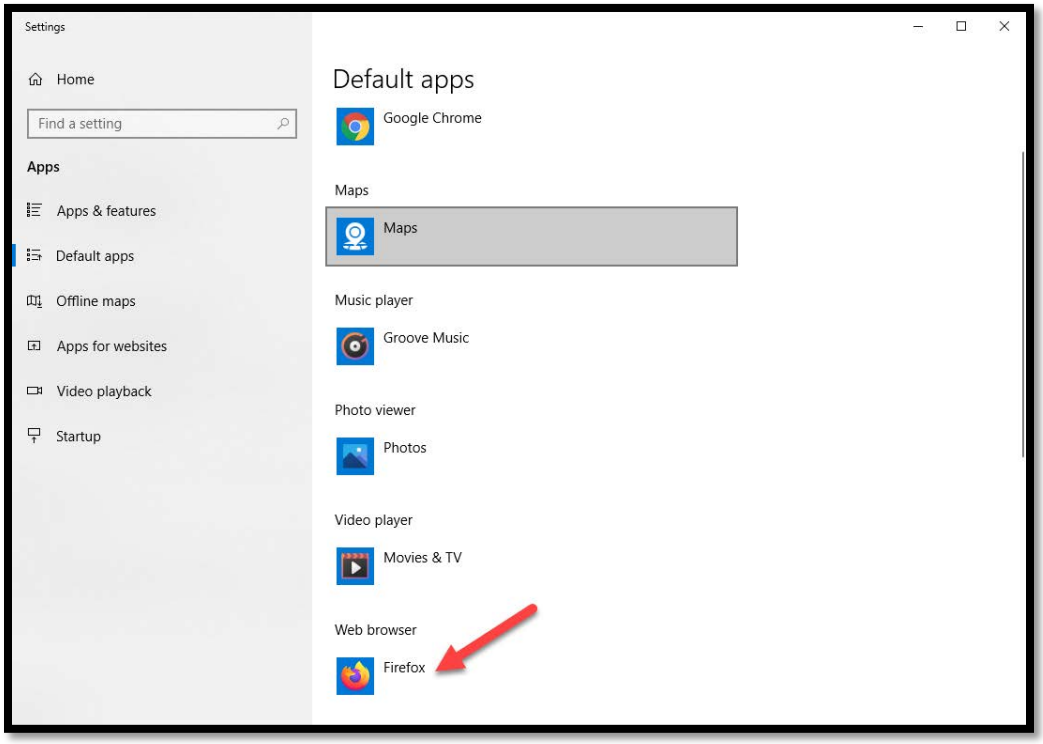
1) Microsoft Visual Studio Code or proficiency with similar web integrated development environment (IDE). You will need to run a live server and launch HTML files in Firefox.
    a. PC & Mac download (install with default options)
2) Firefox web browser
3) GitHub account
4) Git (v2.0.0+)
    a. PC
    b. Mac
5) Internet connection
6) Familiarity with HTML/CSS/JavaScript (helpful)
7) Zoom
8) Open windows for this document, Visual Studio Code, Firefox, and Zoom during live tutorial to follow along

The remaining steps will be accomplished during the live tutorial.

Web browser settings

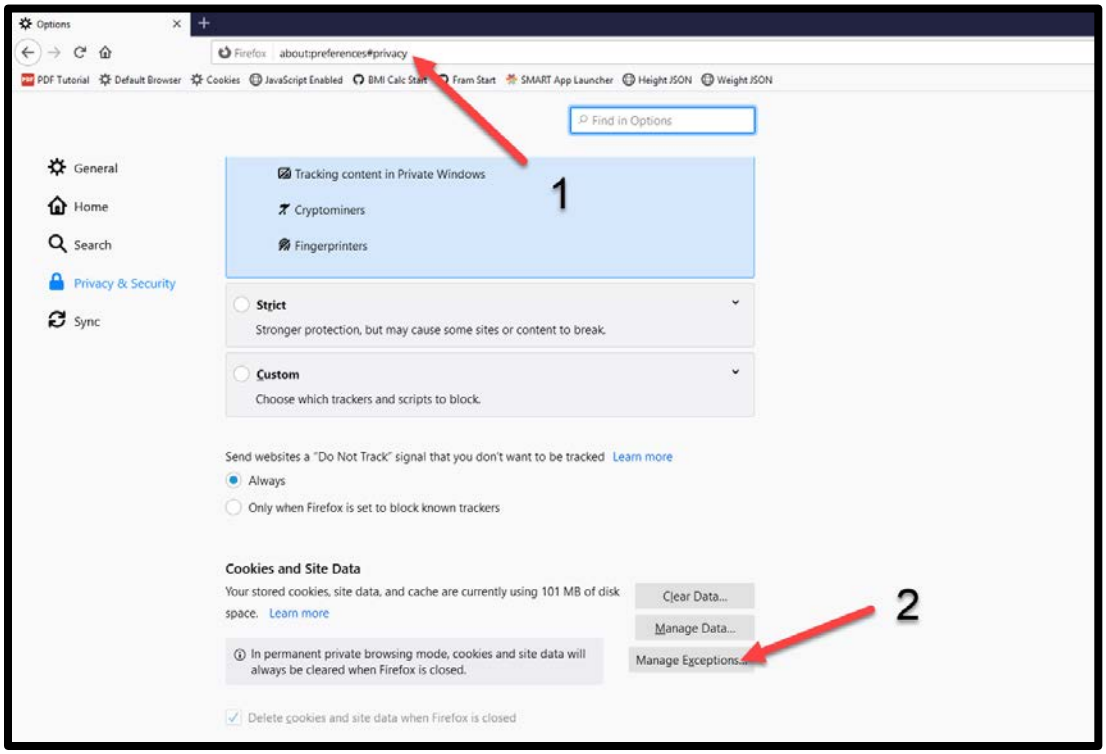
1) Make Firefox your default web browser.
    a. Go to about:preferences.
    b. Select *Make Default…*
    c. In windows, a new Settings window may appear. Select the app option under *Web Browser* and change to *Firefox*

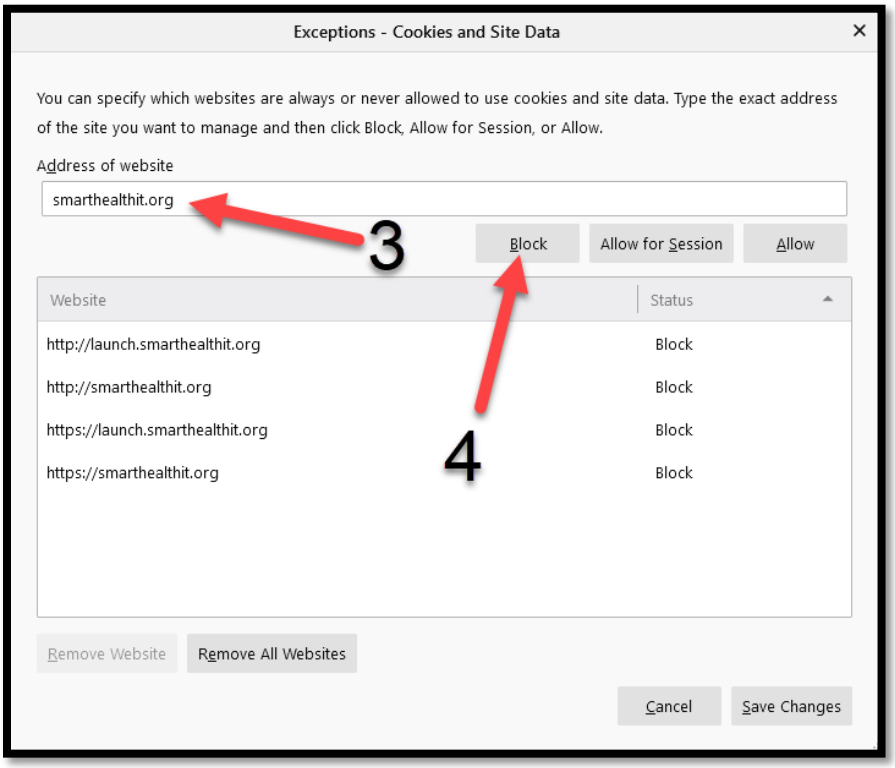




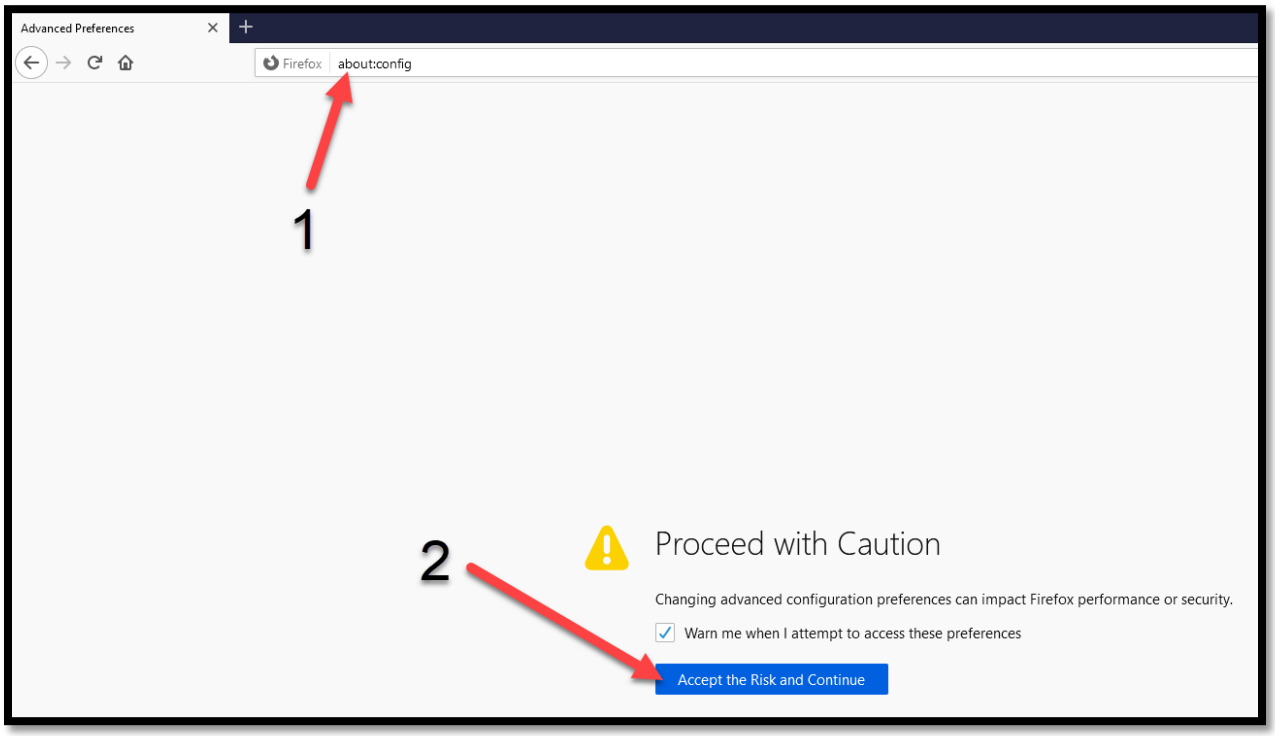
2) Block cookies for the SMART Health IT page in Firefox. Blocking cookies will allow us to test updates to our SMART app in the browser. Without blocking cookies, the SMART Health IT page will only use the starting version of your app.
    a. Go to: about:preferences#privacy
    b. Go to the second section, entitled, *Cookies and Site Data*

c. Select *Manage Exceptions…*
    i. Add the following URLs to the *Block* list:
        1. smarthealthit.org
        2. launch.smarthealthit.org
    ii. Click *Save Changes*



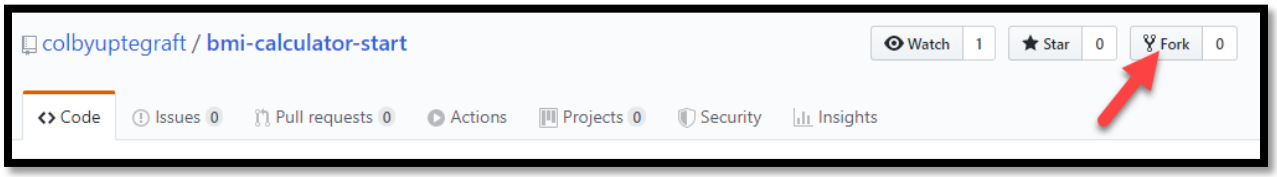


3) Ensure JavaScript is enabled in Firefox (should be enabled by default)
    a. Go to: about:config
    b. Click *Accept the Risk and Continue*
    c. Search for *Javascript.enabled*
    d. Make sure it says *true* in the middle. If not, select the *Toggle* arrows on the right to change it to *true*. For Mac users, double click the *true/false* under *Value* to change it.



Forking the GitHub repositories

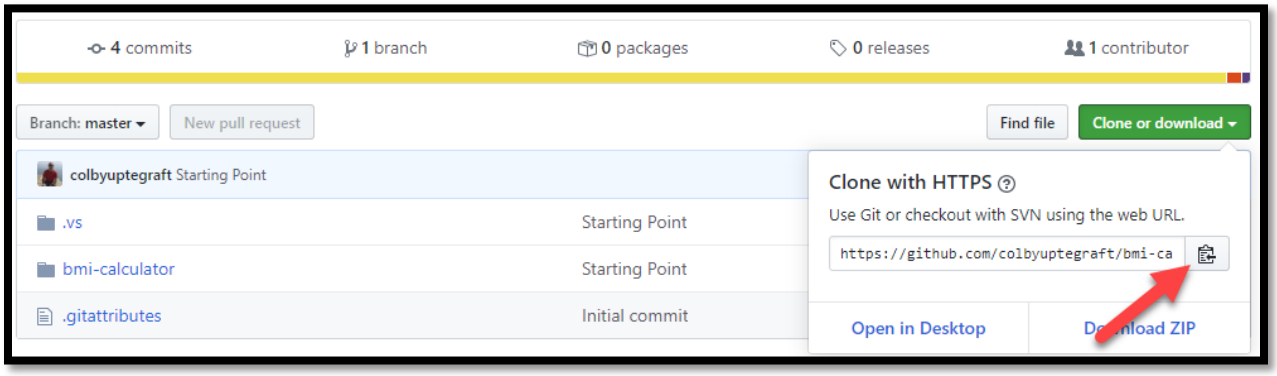
1) Go to https://github.com/colbyuptegraft/bmi-calculator-start
2) Fork this repository
    a. This should create your own version of the repository in your GitHub account



3) Go to your newly forked repository. Should be https://github.com/[yourGitHubUserName]/bmi-calculator-start

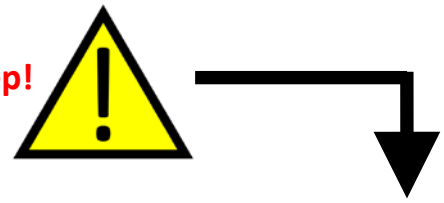
4) Click the 

button. Copy the URL that appears in the sub-menu.
   a. Should be: https://github.com/[yourGitHubUserName]/bmi-calculator-start.git

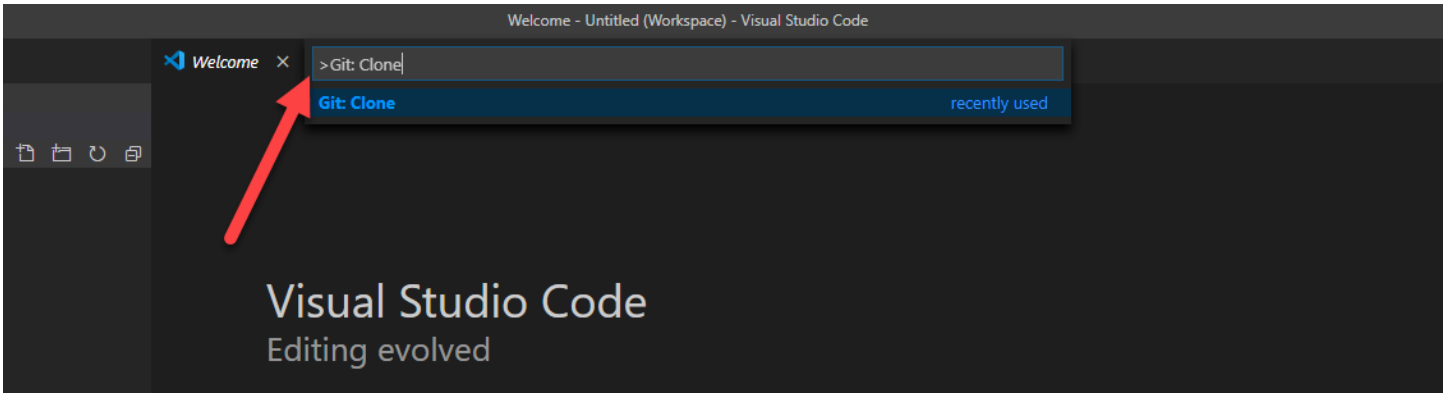


5) Open Visual Studio Code (or your own preferred IDE).

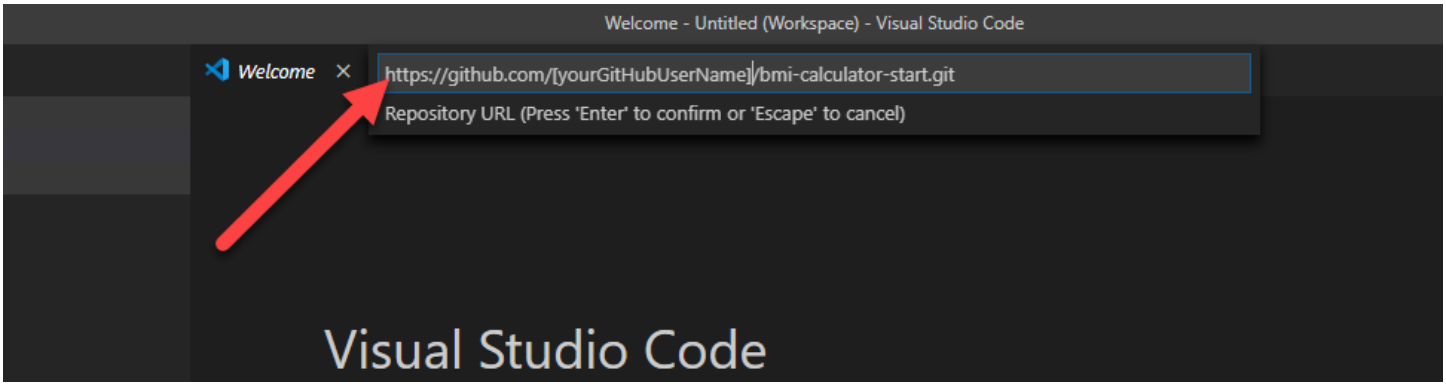
Make sure not to skip the next step!



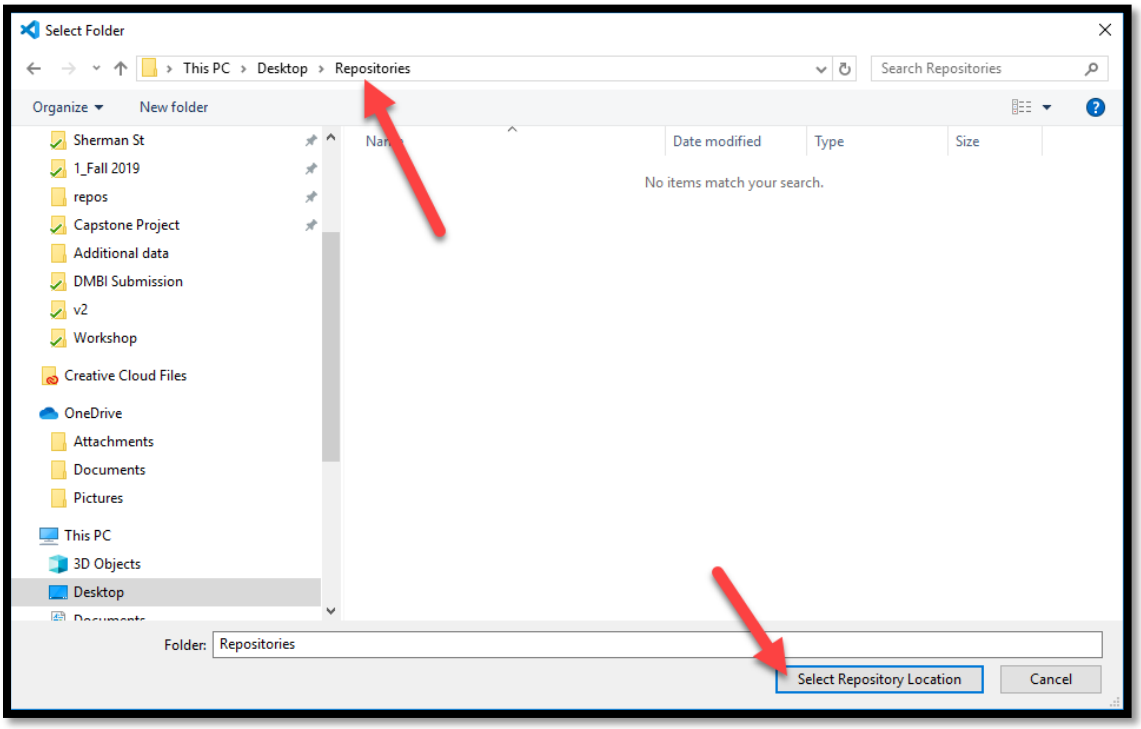
6) Type *Ctrl[Cmd] + Shift + P* to open the *Command Palette*. **Type *Git: Clone* in the *Command Palette* and hit *Enter*.**



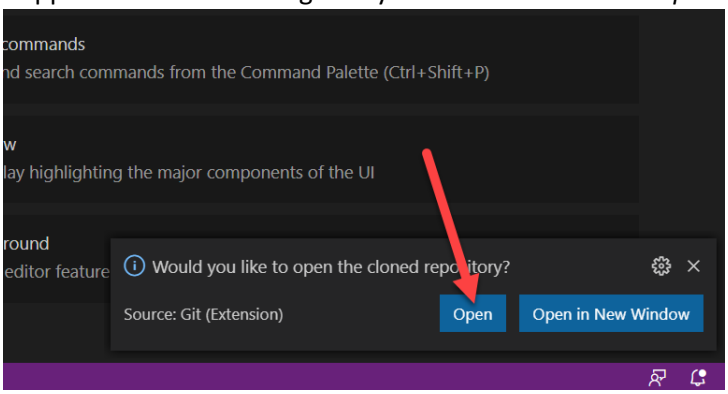
7) Paste the URL for the repository you just copied (https://github.com/[yourGitHubUserName]/bmi-calculator-start.git) and hit *Enter*.



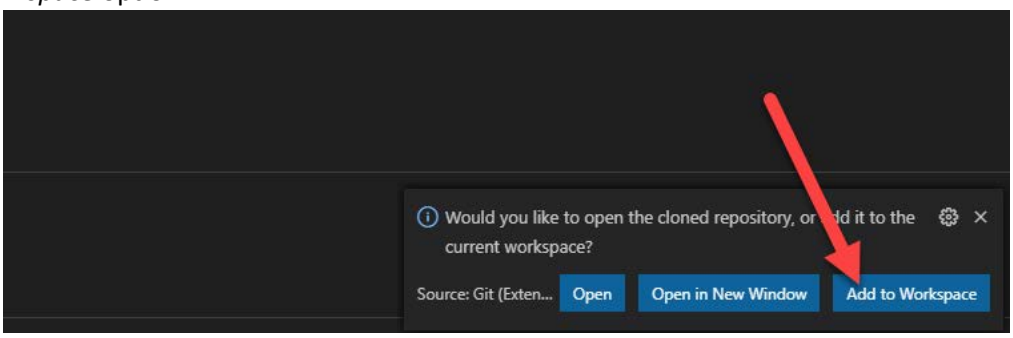
8) Select a location to save the repository or create a new folder. For the purposes of the tutorial, place it in an easy-to-find location, such as your desktop.



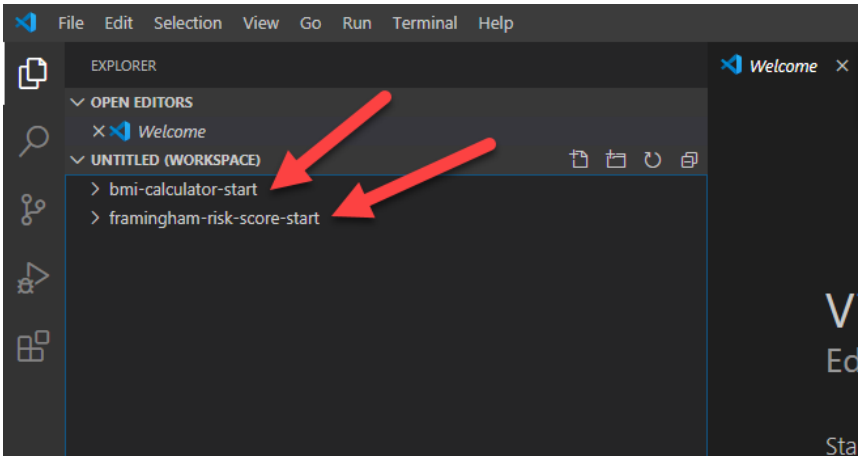
9) An info window should appear in the bottom right of your screen. Select the *Open* option.



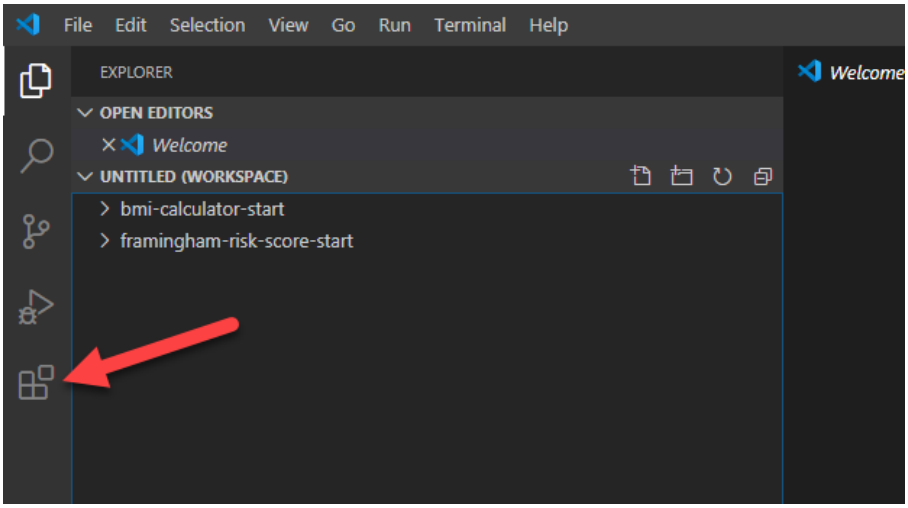
10) Repeat this process (steps 1-9) for the Framingham Risk Score repository: https://github.com/colbyuptegraft/framingham-risk-score-start. For step 8, place the second repository in the same location as the first.
    a. Note: Make sure your window is maximized. Otherwise, the window that appears may not show the *Add to Workspace* option.

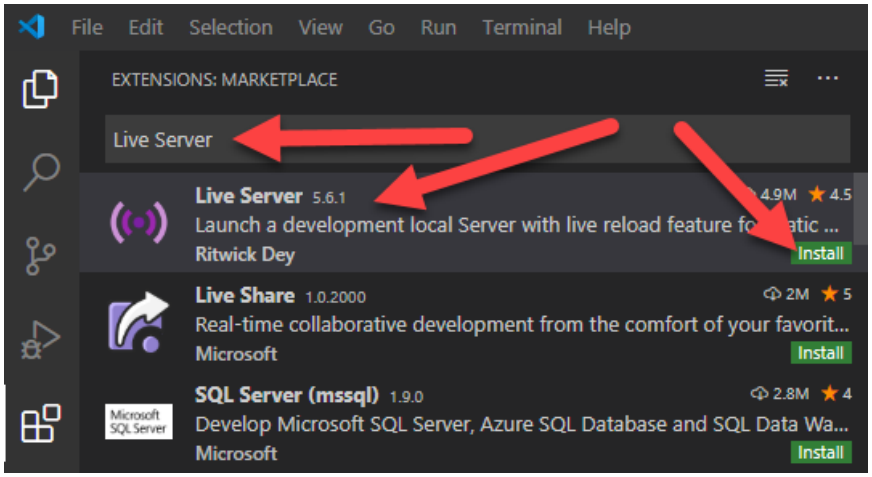


11) Both projects should now be in your Visual Studio Code workspace.



12) Install the *Live Server* extension.
    a. Click the *Extension* icon
    b. Type "Live Server" in the search box and select the top option by *Ritwick Dey*.
    c. Click *Install*





13) Go back and open the ***BMI Calculator*** *project* by clicking the  icon. All the files in each project have the same names. The three main files we'll be working with are,

a. index.html

i. The main HTML file for your application that structures the content of your application. This page is invoked at the conclusion of the SMART authorization workflow.

b. launch.html

i. The entry point to this application within this tutorial and within an actual production environment

c. example-smart-app.js

i. Follow bmi-calculator -> src -> js to find this file

ii. This is the only file in both apps that we'll be editing

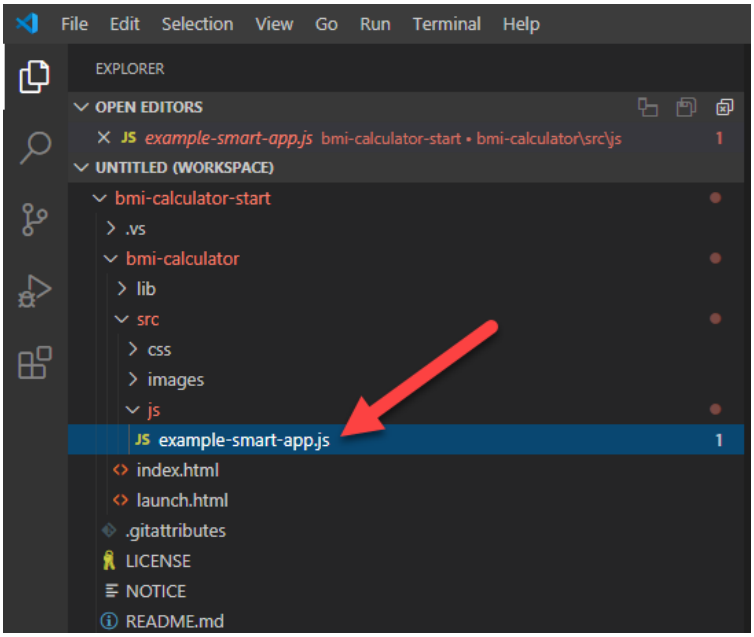
d. Other project files

i. bmi-calculator / framingham-risk-score

1. lib
    a. css
        i. cerner-smart-embeddable-lib-1.0.0.min.css
            1. A CSS file that you would need to embed this within Cerner's EHR, PowerChart
    b. js
        i. cerner-smart-embeddable-lib-1.0.0.min.js
            1. A JavaScript file that you would need to embed this app with Cerner's EHR, PowerChart
        ii. fhir-client-cerner-additions-1.0.0.js
            1. Additional JavaScript added to FHIR Client JavaScript library
        iii. fhir-client-v0.1.12.js
            1. An open-source JavaScript library to help make FHIR API calls and handle the SMART on FHIR authorization workflow.
2. src
    a. css
        i. example-smart-app.css
            1. Custom CSS styling for this app
        ii. relaxCSS.css
            1. CSS styling library
        iii. simple-grid.css
            1. CSS styling library
3. images
    a. logo.png
        i. Background picture for app

BMI Calculator

1) Open the *example-smart-app.js* file. This file will be the main file that we edit throughout this tutorial and is the meat of how we request FHIR resources.



2) On line 16, find the *smart.patient.api.fetchAll* function. This function uses the fhir.js API to retrieve resources for the current patient in context. On line 17, find *type:*. This is your first assignment. Replace the commented-out text with the type of FHIR resource we'll be requesting for this app. Hint: This app only uses height and weight to calculate the BMI. Height and weight are both _____ event resources in the FHIR standard. Choices include:
    a. 'Observation'
    b. 'Specimen'
    c. 'ImagingStudy'
    d. 'Person'

ANSWER

```
16  var obv = smart.patient.api.fetchAll({
17      type: /* [Insert FHIR resource type here] */,
18      query: {
19          code: {
20              $or: [
21                  'http://loinc.org|8302-2',     // Height
22                  'http://loinc.org|3137-7',     // Height [measured]
23                  'http://loinc.org|3138-5',     // Height [stated]
24                  'http://loinc.org|8308-9',     // Height [standing]
25                  'http://loinc.org|8306-3',     // Height [lying]
26                  'http://loinc.org|8301-4',     // Height [estimated]
27
28                  'http://loinc.org|29463-7',    // Weight
29                  'http://loinc.org|3141-9',     // Weight
30                  'http://loinc.org|18833-4',    // Weight
31                  'http://loinc.org|3142-7',     // Weight [stated]
32                  'http://loinc.org|75292-3',    // Weight [usual]
33                  'http://loinc.org|8335-2',     // Weight [estimated]
34                  'http://loinc.org|8351-9',     // Weight [without clothes]
35              ]
36          }
37      }
38  });
```

3) Next, we need to select the applicable LOINC codes. As you can see, there're many different LOINC codes for height and weight, and these may vary by institution and context. For the sake of this application, we've included a subset of both that cover height and weight for the set of sample patients provided above (Lines 21-34). The *$or* operator modifies the *fetchAll* query to return the resources that match any of the following LOINC codes (8302-2 or 3137-7 or 3138-5, etc).

```
var obv = smart.patient.api.fetchAll({
    type: /* [Insert FHIR resource type here] */,
    query: {
        code: {
            $or: [
              'http://loinc.org|8302-2',     // Height
              'http://loinc.org|3137-7',     // Height [measured]
              'http://loinc.org|3138-5',     // Height [stated]
              'http://loinc.org|8308-9',     // Height [standing]
              'http://loinc.org|8306-3',     // Height [lying]
              'http://loinc.org|8301-4',     // Height [estimated]

              'http://loinc.org|29463-7',    // Weight
              'http://loinc.org|3141-9',     // Weight
              'http://loinc.org|18833-4',    // Weight
              'http://loinc.org|3142-7',     // Weight [stated]
              'http://loinc.org|75292-3',    // Weight [usual]
              'http://loinc.org|8335-2',     // Weight [estimated]
              'http://loinc.org|8351-9',     // Weight [without clothes]
            ]
        }
    }
});
```

   a. If you need to look up additional LOINC codes, you can create an account on their website and download their documentation.
   b. You can also look up the LOINC codes used for particular patients within SMART Health IT:
      i. Height
      ii. Weight

Response Body

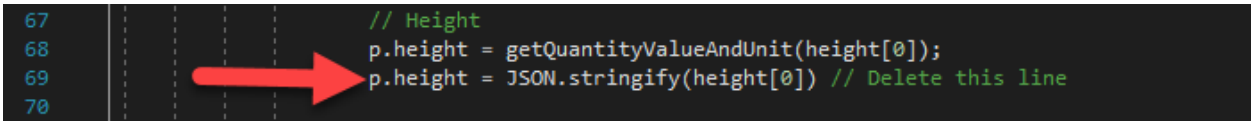

```
{
  "resourceType": "Observation",
  "id": "790ab88d-6a87-4785-b7dd-42a694580f13",
  "meta": {
    "versionId": "1",
    "lastUpdated": "2018-05-07T13:34:01.721-04:00",
    "tag": [
      {
        "system": "https://smarthealthit.org/tags",
        "code": "synthea-8-2017"
      }
    ]
  },
  "status": "final",
  "category": {
    "coding": [
      {
        "system": "http://hl7.org/fhir/observation-category",
        "code": "vital-signs"
      }
    ]
  },
  "code": {
    "coding": [
      {
        "system": "http://loinc.org",
        "code": "8302-2",
        "display": "Body Height"
      }
    ],
    "text": "Body Height"
  },
  "subject": {
    "reference": "Patient/57b85682-ce42-4187-a593-7864248a9484"
  },
```

4) The *byCodes* function, created on line 44 and used on lines 55-56, is from the FHIR Client JavaScript library. It searches resources by a given code, LOINC codes in this context, and returns an array of JSON objects of all resources with that code. In lines 55-56, copy/paste the remaining LOINC codes from lines 21-34.


```
var obv = smart.patient.api.fetchAll({
    type: /* [Insert FHIR resource type here] */,
    query: {
        code: {
            $or: [
                'http://loinc.or  8302-2',     // Height
                'http://loinc.or  3137-7',     // Height [measured]
                'http://loinc.or  3138-5',     // Height [stated]
                'http://loinc.or  8308-9',     // Height [standing]
                'http://loinc.or  8306-3',     // Height [lying]
                'http://loinc.or  8301-4',     // Height [estimated]

                'http://loinc.or  29463-7',    // Weight
                'http://loinc.or  3141-9',     // Weight
                'http://loinc.or  18833-4',    // Weight
                'http://loinc.or  3142-7',     // Weight [stated]
                'http://loinc.or  75292-3',    // Weight [usual]
                'http://loinc.or  8335-2',     // Weight [estimated]
                'http://loinc.or  8351-9',     // Weight [without clothes]
            ]
        }
    }
});

$.when(pt, obv).fail(onError);

$.when(pt, obv).done(function (patient, obv) {

    var byCodes = smart.byCodes(obv, 'code');
    var gender = patient.gender;
    var fname = '';
    var lname = '';

    if (typeof patient.name[0] !== 'undefined') {
        fname = patient.name[0].given.join(' ');
        lname = patient.name[0].family.join(' ');
    }

    // Create arrays of JSON objects
    var height = byCodes('8302-2', '3137-7', /* [Copy/paste remaining LOINC codes for height here] */);
    var weight = byCodes('29463-7', '3141-9', /* [Copy/paste remaining LOINC codes for weight here] */);
```

ANSWER

5) Now, find line 69. Don't delete this line yet! We're going to see what *Height* looks like as a JSON object. The current code is set to return the first item in the *Height Array* that we created above as the *String* version of a JSON object. Let's run our app to see!

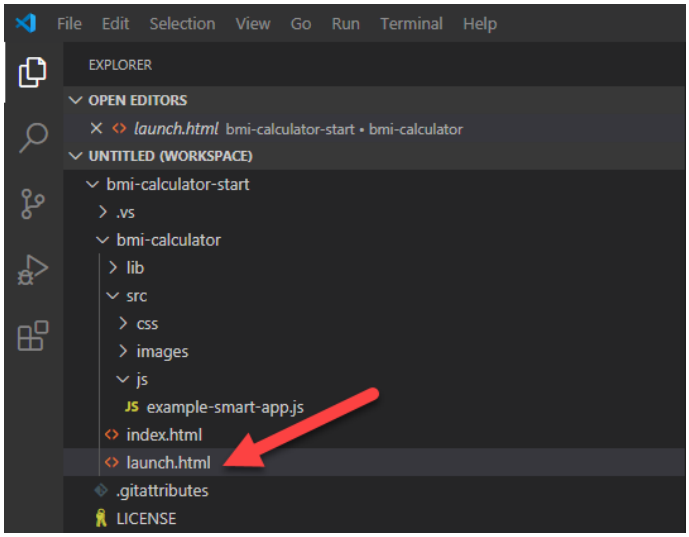


```
// Height
p.height = getQuantityValueAndUnit(height[0]);
p.height = JSON.stringify(height[0]) // Delete this line
```

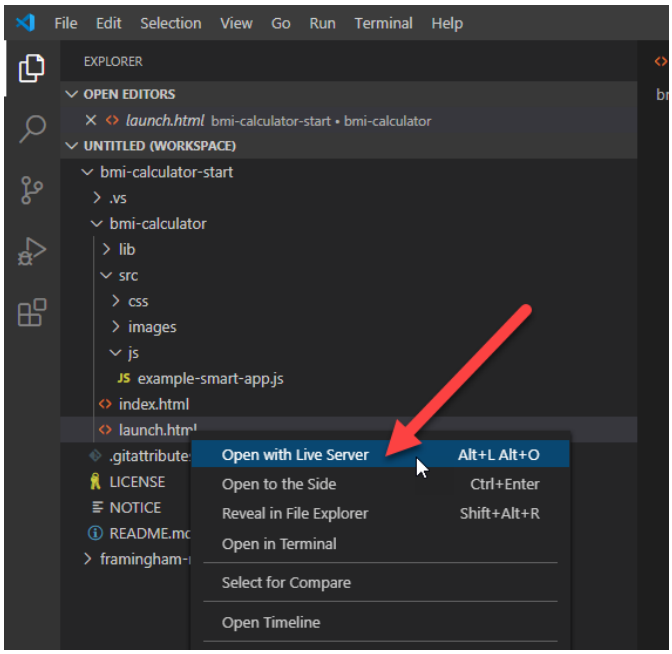
Testing your app

1) Save all your open files in Visual Studio Code.
2) If you've already launched your app at least once, close the two browser tabs initiated with that launch, ie the browser tabs automatically opened with *steps 3 & 7.h* below. Otherwise, ignore this step.

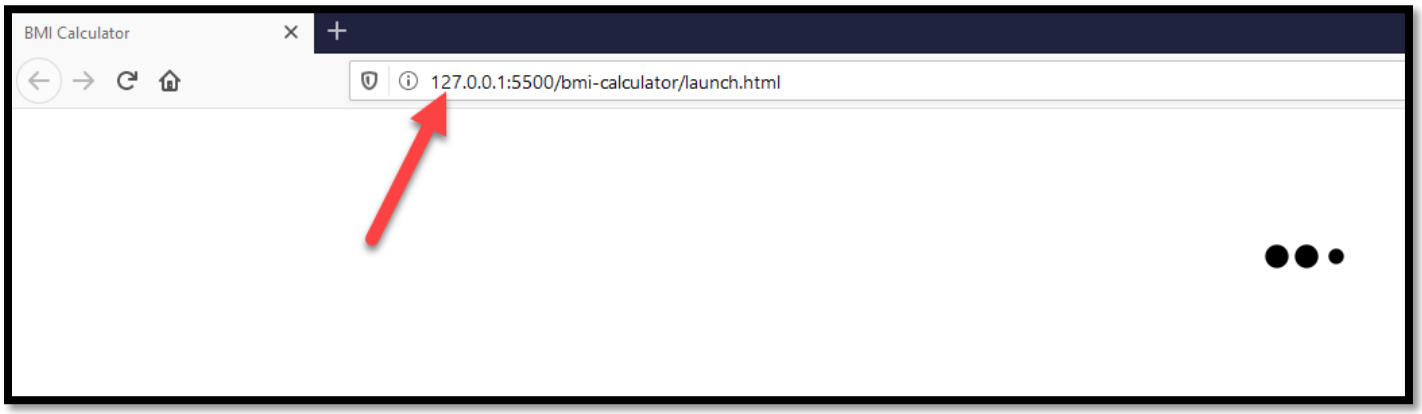
3) Open the *launch.html* file from within Visual Studio Code.



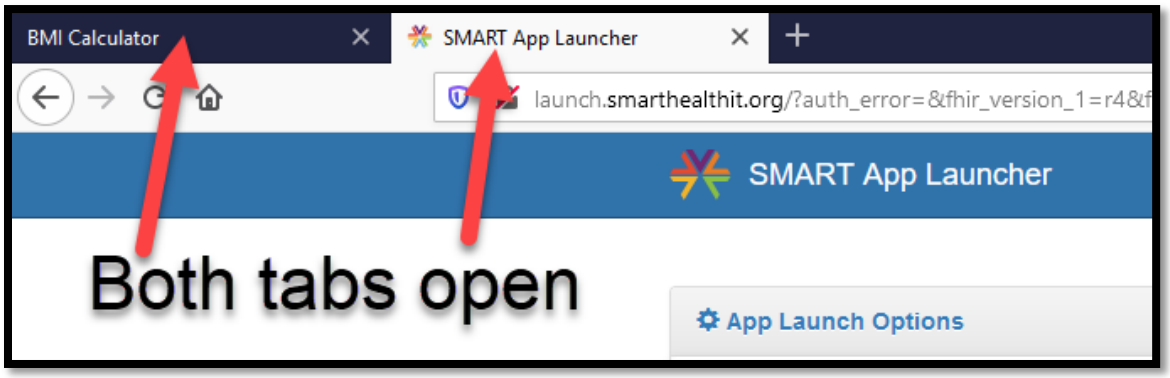
4) Right click on the file and select *Open with Live Server*. This should open a new tab in your default browser. You should also see an animation of three dots fading in and out.



5) Copy the URL for this tab.
    a. Should be similar to: http://127.0.0.1:5500/bmi-calculator/launch.html
    b. This is the 'App Launch URL'



6) In a different browser tab, go to: http://launch.smarthealthit.org/
    a. This is the *SMART App Launcher* page where we'll launch our SMART apps

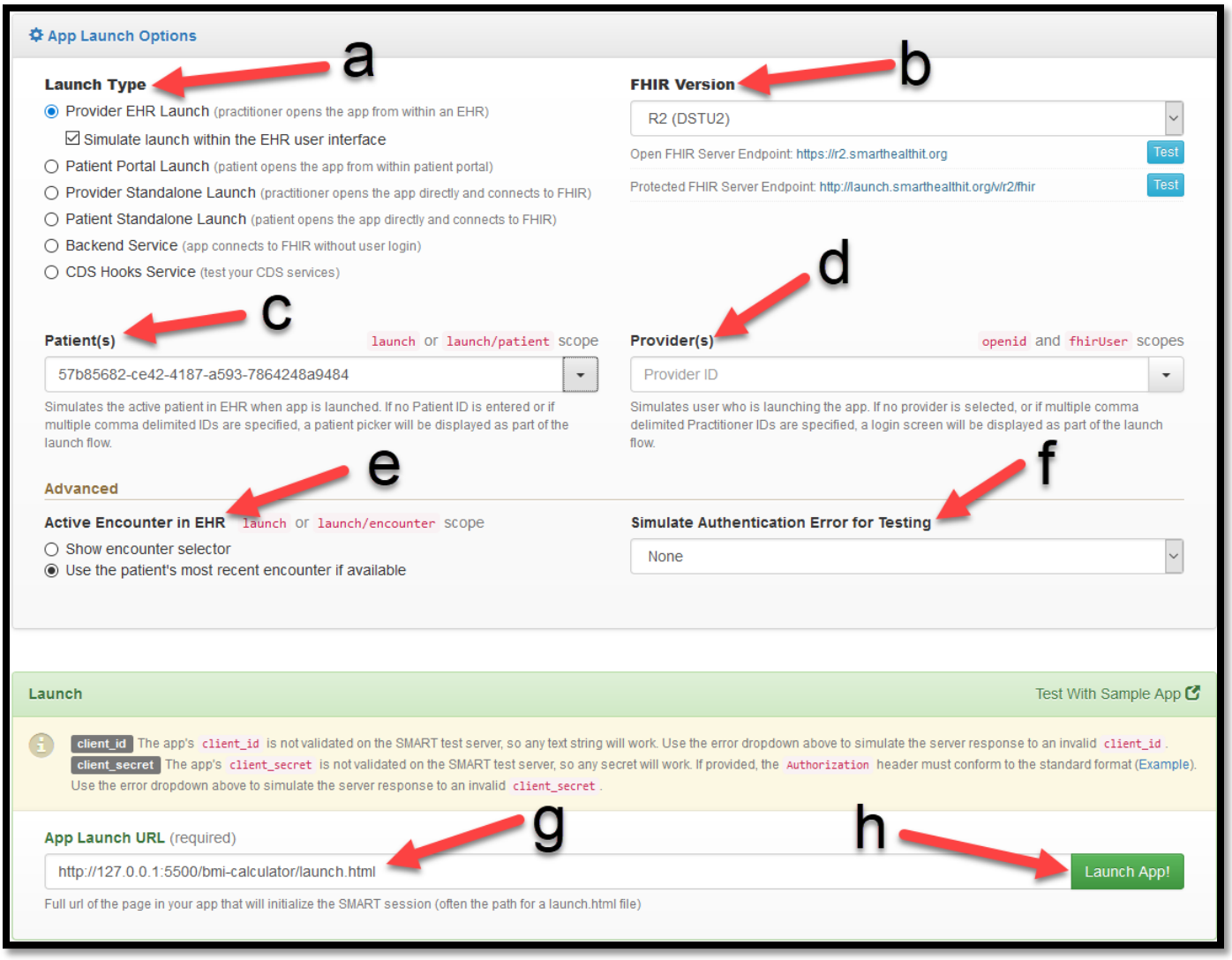


7) Configure the following *App Launch Options*
    a. **Launch Type**
        i. Select *Provider EHR Launch*
            1. Check *Simulate launch within the EHR user interface*
    b. **FHIR Version**
        i. Select *R2 (DSTU2)*
    c. **Patient(s)**
        i. Copy/paste the ID(s) for select patient(s). If you select multiple, put a common between IDs.
            1. For *BMI Calculator*
                a. With *Height* & *Weight*
                    i. Mr. Ronnie Gutmann
                        1. ID: 57b85682-ce42-4187-a593-7864248a9484
                    ii. Daniel X. Adams
                        1. ID: smart-1288992
                    iii. Ruth C. Black
                        1. ID: smart-665677
                b. Without *Height* &/or *Weight*
                    i. Mr. Carey Krajcik
                        1. ID: cb44df11-064a-4132-aeda-aeca5a7d2f5f

ii. Mr. Andreas Ratke
1. ID: 8ab4791f-0790-44f3-97c0-88f14c9329da

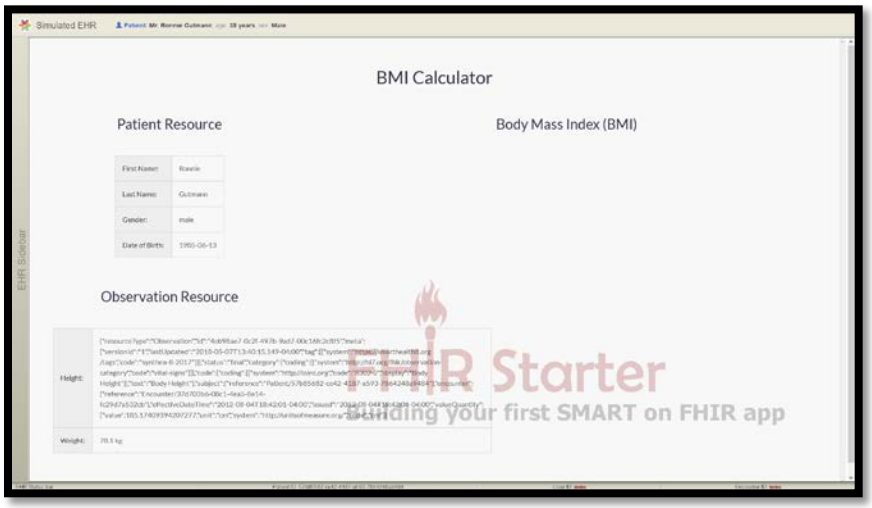
2. For *Framingham Risk Score*
   a. With all data fields
      i. Ruth C. Black
         1. ID: smart-665677
      ii. Carol G. Allen
         1. ID: smart-1577780
      iii. Daniel X. Adams (Technically, too old)
         1. ID: smart-1288992
   b. Missing at least one of the required fields
      i. Cristi Bernhard
         1. ID: 7478ade9-a06d-45ed-b179-1af1b13da49f
      ii. Dan Diabetes
         1. ID: DDME
      iii. Betty N. Davis
         1. ID: smart-1642068
   c. Missing *Smoking Status* &/or *Medications*
      i. Ewa Lowe
         1. ID: c9452102-96dd-46e0-8857-1ac14e65c214
      ii. Mrs. Slyvia Emard
         1. ID: a13c822d-b782-415e-8a1a-9590e8fdd3a7
      iii. Damon Shanahan
         1. ID: fb24ec64-ed6b-404a-a4dd-43f1ee69d67e

d. **Provider(s)**
   i. Leave blank
e. **Active Encounter in EHR**
   i. Select *Use the patient's most recent encounter if available*
f. **Simulate Authentication Error for Testing**
   i. Select *None*
g. **App Launch URL**
   i. Paste URL you copied above
      1. Should be similar to: http://127.0.0.1:5500/bmi-calculator/launch.html



h. Select 

i. You should briefly see the animation of three dots fading in and out before your app loads in a new browser tab. Hopefully, it looks like this:

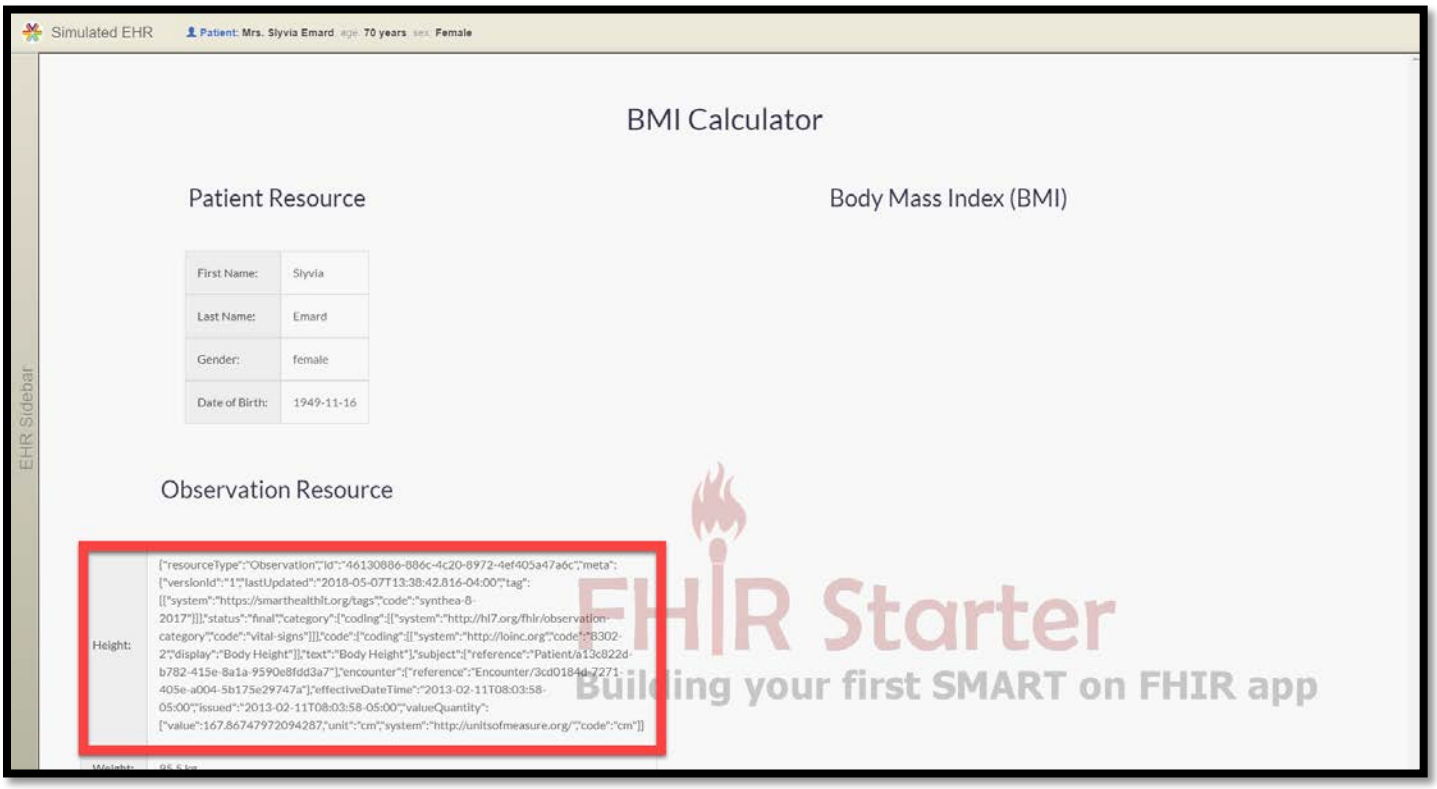


i. Troubleshooting common errors

1. *Failed to Call FHIR Service*
    a. Solution: Make sure you've selected R2 (DSTU 2) as the **FHIR Version**
2. *Endless three-dot animation*
    a. This likely represents an error somewhere in your code, most likely the *example-smart-app.js* file. Double check your code with this tutorial's instructions.
3. Nothing changed in your app
    a. Make sure you've closed both the http://127.0.0.1:5500/bmi-calculator/launch.html page & the page that launched when you clicked Launch App before launching a new version of your app.
    b. Make sure cookies are disabled [see above].

BMI Calculator [Cont'd]

8) In your app, you should see the JSON object as a text string in the left column. It should look similar, minus the formatting, to the height resource here.



9) Now we're going to look at the function that extracted the value of this height resource from the JSON object. Go to line 125 within the *getQuantityValue* function. This function takes the JSON object and drills down to find the value within the name/value pair of the pair with the name "value." Sounds confusing. For example, if we wanted to return the type of resource, the function would return *ob.resourceType*.

```
118    // Get only numerical value of observations
119    function getQuantityValue(ob) {
120
121        if (typeof ob != 'undefined' &&
122            typeof ob.valueQuantity != 'undefined' &&
123            typeof ob.valueQuantity.value != 'undefined') {
124
125            return ob.valueQuantity.value;
126
127        } else {
128            return undefined;
129        }
130    }
```

10) Go back to line 69. Delete this line.

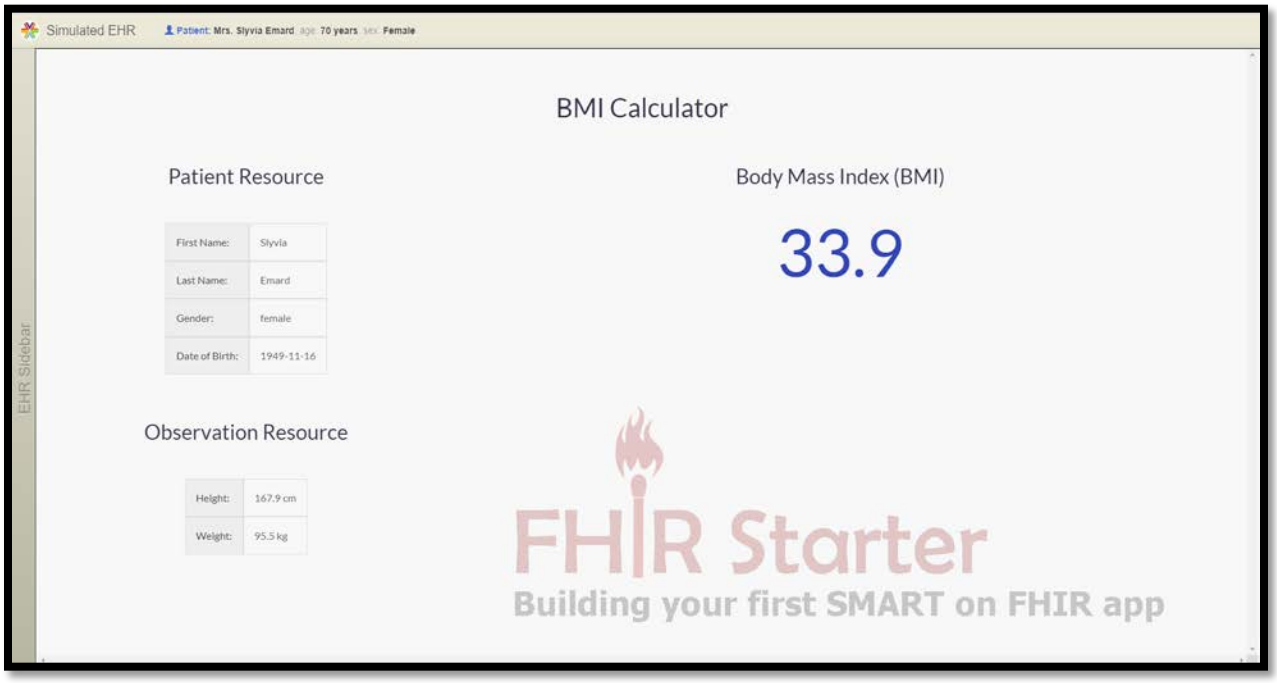
```
67          // Height
68          p.height = getQuantityValueAndUnit(height[0]);
69          p.height = JSON.stringify(height[0]) // Delete this line
70
```

11) Look at what is now line 74 or 75. You should be able to see how BMI is calculated in JavaScript. Delete the two '//' to un-comment this line.
    a. Formula: weight (kg) / [height (m)]^2

```
74          // Calculate BMI
75          // p.bmi = (getQuantityValue(weight[0]) / (Math.pow((getQuantityValue(height[0]) / 100), 2))).toFixed(1);
76
```

ANSWER

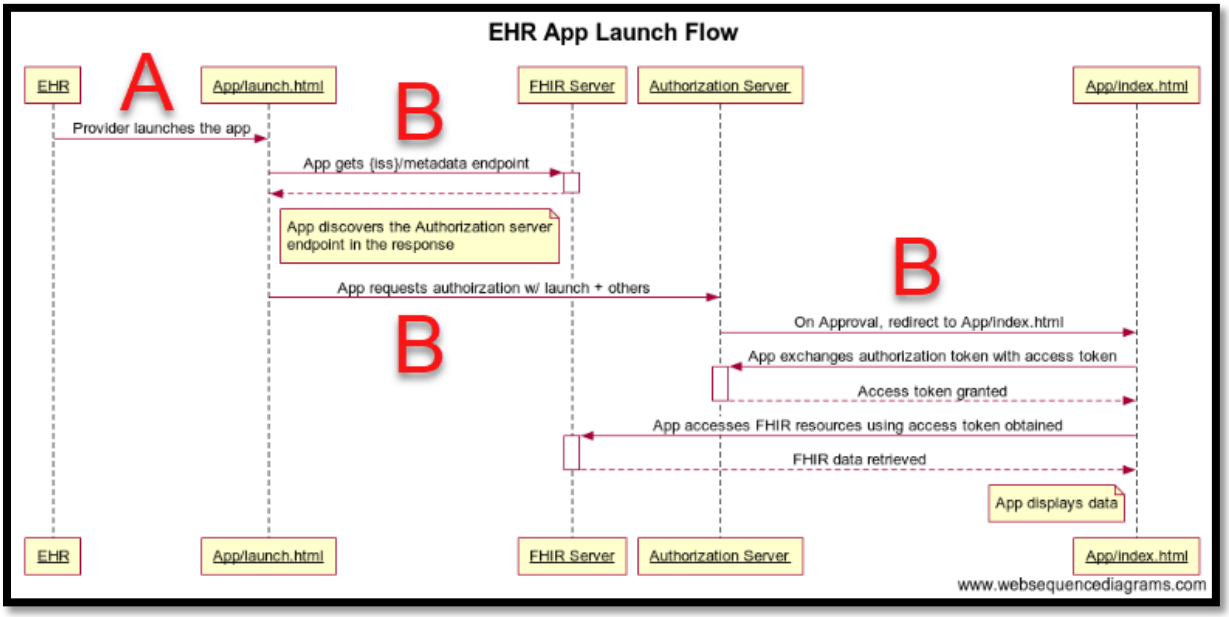
12) Launch App! again using the above steps. Congratulations! You've just created your first fully functioning SMART on FHIR application!



Full example-smart-app.js file

SMART authorization workflow

1) Now that you've created and launched a fully functioning SMART on FHIR app, you've indirectly experienced the SMART authorization workflow.



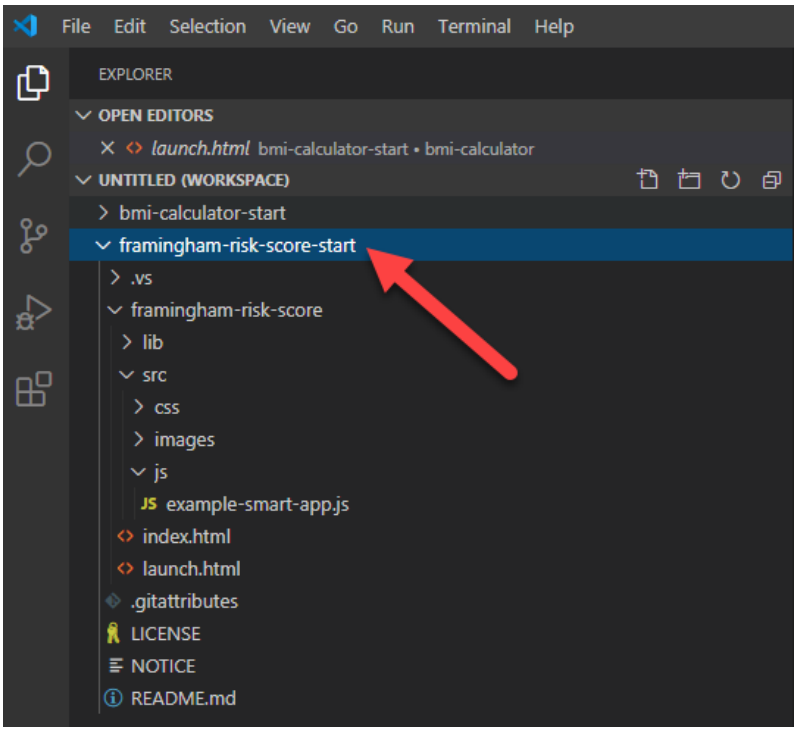
2) The http://launch.smarthealthit.org/ site is taking care of step A above while lines lines 33-36 in the *launch.html* file are taking care of the steps labeled *B* above.

```
33          FHIR.oauth2.authorize({
34              'client_id': 'YOUR-SMART-HEALTH-IT-CLIENT-ID-HERE',
35              'scope': 'patient/*.read launch online_access openid profile'
36          });
```

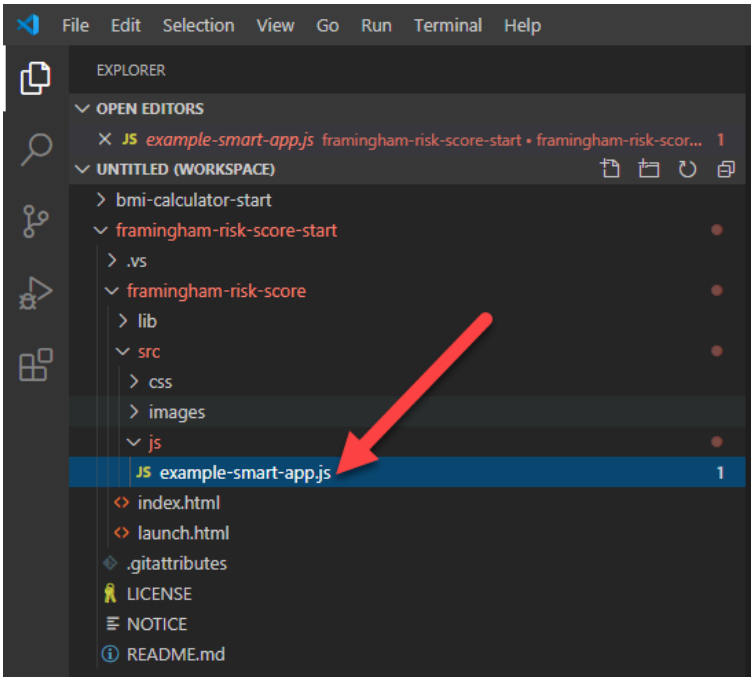
3) As you can see, the *FHIR.oath2.authorize* function from the FHIR Client JavaScript library makes the authorization step easy. Since the http://launch.smarthealthit.org/ site does not validate the *client_id*, we leave this blank. In a SMART sandbox for Cerner, Epic, or another vendor, this likely would not be the case. Under *scope*, the following arguments do the following,
    a. *patient/*.read*: permission to read all the resources for the current patient
    b. *launch*: permission to obtain launch context; required when launching within EHR
    c. *online_access*: request a refresh_token to obtain a new access token to replace an expired one and will be usable for as long as the user remains online; required when launching within EHR
    d. *openid*, *profile*: permission to obtain information about user; required when launching within EHR

Framingham Risk Score

1) This second app is slightly more complicated than our first. Here, we'll be pulling in an additional resource type and utilizing another API. To start, open the *Framingham Risk Score* Visual Studio Code project that we created toward the beginning. The names of all the applicable files are the same as the *BMI Calculator* project. We'll be using this formula to calculate the risk.
    a. Note: When patients are missing smoking status and medication information, we treat missing as 'No.'



2) Open the *example-smart-app.js* file. It should look similar to the same file from the *BMI Calculator*. In lines 10-31, we created all the variables for the constants used in the Framingham formula. In lines 42-70, we again pull in all the observation resources that match a certain LOINC code, in this case, all the data points needed to determine a patient's Framingham risk score.



```
10          // Framingham Coefficients for Men
11          var age_men = 52.00961;
12          var tcl_men = 20.014077;
13          var hdl_men = -0.905964;
14          var sbp_men = 1.305784;
15          var bpTx_men = 0.241549;
16          var smk_men = 12.096316;
17          var ageTcl_men = -4.605038;
18          var ageSmk_men = -2.84367;
19          var age2_men = -2.93323;
20          var con_men = 172.300168;
21
22          // Framingham Coefficients for Women
23          var age_women = 31.764001;
24          var tcl_women = 22.465206;
25          var hdl_women = -1.187731;
26          var sbp_women = 2.552905;
27          var bpTx_women = 0.420251;
28          var smk_women = 13.07543;
29          var ageTcl_women = -5.060998;
30          var ageSmk_women = -2.996945;
31          var con_women = 146.5933061;
32
```

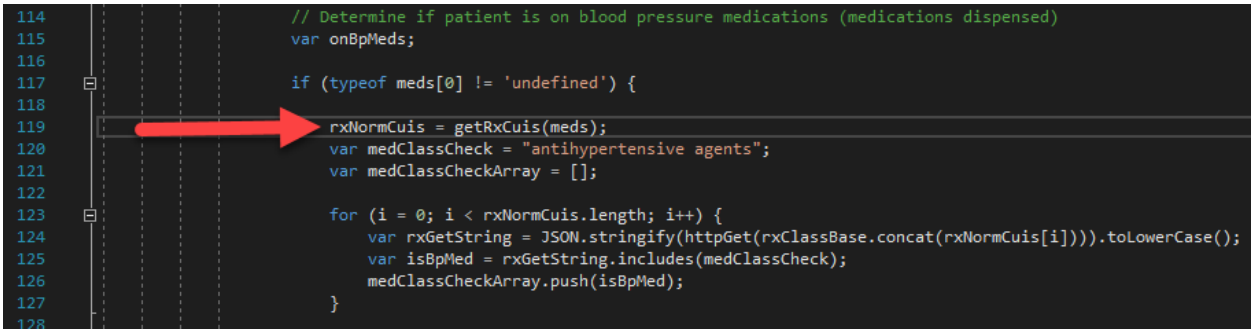
```
42          var obv = smart.patient.api.fetchAll({
43              type: 'Observation',
44              query: {
45                  code: {
46                      $or: [
47                          'http://loinc.org|8462-4', // DBP
48                          'http://loinc.org|8480-6', // SBP
49                          'http://loinc.org|2085-9', // HDL
50                          'http://loinc.org|2089-1', // LDL
51                          'http://loinc.org|13457-7', // LDL
52                          'http://loinc.org|18262-6', // LDL
53                          'http://loinc.org|2093-3', // Total Cholesterol
54                          'http://loinc.org|55284-4', // Blood pressure systolic & diastolic
55                          'http://loinc.org|30525-0', // Age
56                          'http://loinc.org|21611-9', // Age (estimated)
57                          'http://loinc.org|21612-7', // Age (reported)
58                          'http://loinc.org|29553-5', // Age (calculated)
59                          'http://loinc.org|72166-2', // Tobacco smoking status in social history
60                          'http://loinc.org|81229-7', // Tobacco smoking status - Tobacco Smoker
61                          'http://loinc.org|11366-2', // Tobacco use status
62                          'http://loinc.org|11367-0', // Tobacco use status
63                          'http://loinc.org|39240-7', // Tobacco use status
64                          'http://loinc.org|2571-8', // Triglycerides (mass/volume in Serum or plasma)
65                          'http://loinc.org|3043-7', // Triglycerides (mass/volume in Blood)
66                          'http://loinc.org|3049-4', // Triglycerides (mass/volume in serum or plasma) - Deprecated
67                      ]
68                  }
69              }
70          });
```

3) In lines 72-78, we're going to pull in an additional resource type to determine if the patient is taking any medications for blood pressure control. Go here to identify this "resourceType" and fill in the missing code on line 73.
    a. Note: Line 76 shows how you would search for individual medications.

```
72          var meds = smart.patient.api.fetchAll({
73              type: /* [Insert FHIR resource type here] */,
74              query: {
75                  status: "completed"
76                  //code: 'http://www.nlm.nih.gov/research/umls/rxnorm|153666' // "irbesartan 150 MG Oral Tablet [Avapro]"
77              }
78          });
```

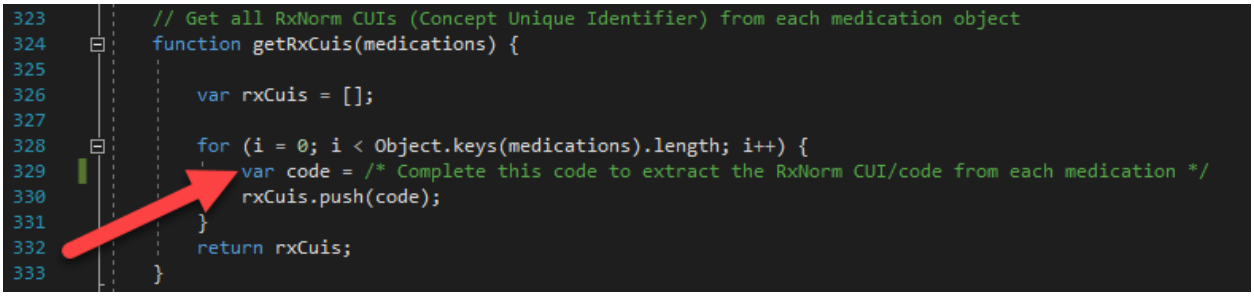
ANSWER

4) Find line 119. This line uses the function *getRxCuis* to extract the RxNorm concept unique identifier (CUI) from each medication. Now we're going to look at this function in detail.
    a. Note: Line 124 uses the RxClass API to help identify medications that could be used to control blood pressure. Take a look at this code and related functions in your own time.



```
114             // Determine if patient is on blood pressure medications (medications dispensed)
115             var onBpMeds;
116
117             if (typeof meds[0] != 'undefined') {
118
119                 rxNormCuis = getRxCuis(meds);
120                 var medClassCheck = "antihypertensive agents";
121                 var medClassCheckArray = [];
122
123                 for (i = 0; i < rxNormCuis.length; i++) {
124                     var rxGetString = JSON.stringify(httpGet(rxClassBase.concat(rxNormCuis[i]))).toLowerCase();
125                     var isBpMed = rxGetString.includes(medClassCheck);
126                     medClassCheckArray.push(isBpMed);
127                 }
```

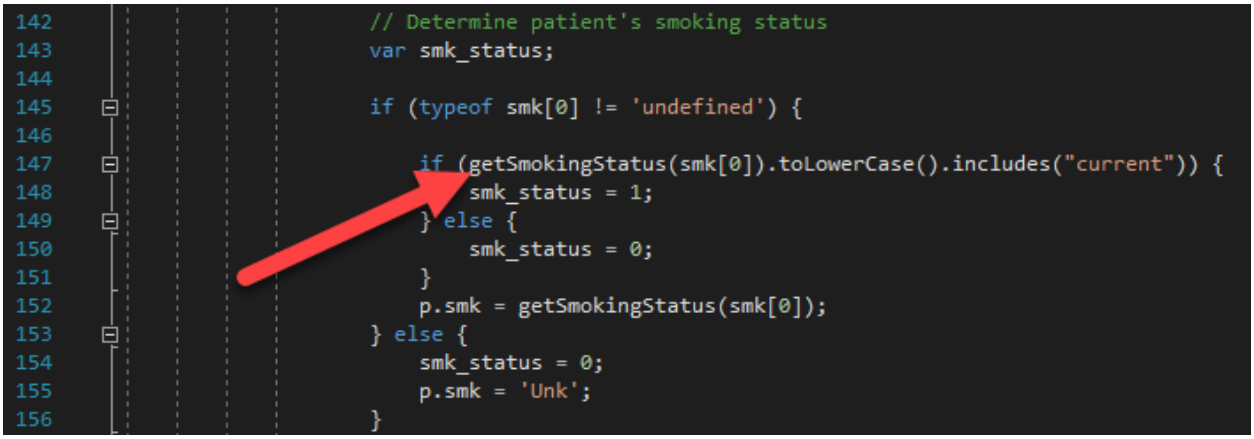
5) Find line 329. Complete this line. The goal of this function is to extract the RxNorm CUI (or code) for each medication, ie the value of "code" here.
    a. Hint 1: The start of the code is: *medications[i].*
    b. Hint 2: Objects inside of "coding" are in an array. Accessing the first item in this array would be written as *coding[0]*.



```
323 // Get all RxNorm CUIs (Concept Unique Identifier) from each medication object
324 function getRxCuis(medications) {
325
326     var rxCuis = [];
327
328     for (i = 0; i < Object.keys(medications).length; i++) {
329         var code = /* Complete this code to extract the RxNorm CUI/code from each medication */
330         rxCuis.push(code);
331     }
332     return rxCuis;
333 }
```

ANSWER

6) Find line 147 to begin our final task. This line uses the function *getSmokingStatus* to extract the smoking status as a text string from the relevant observation resources.



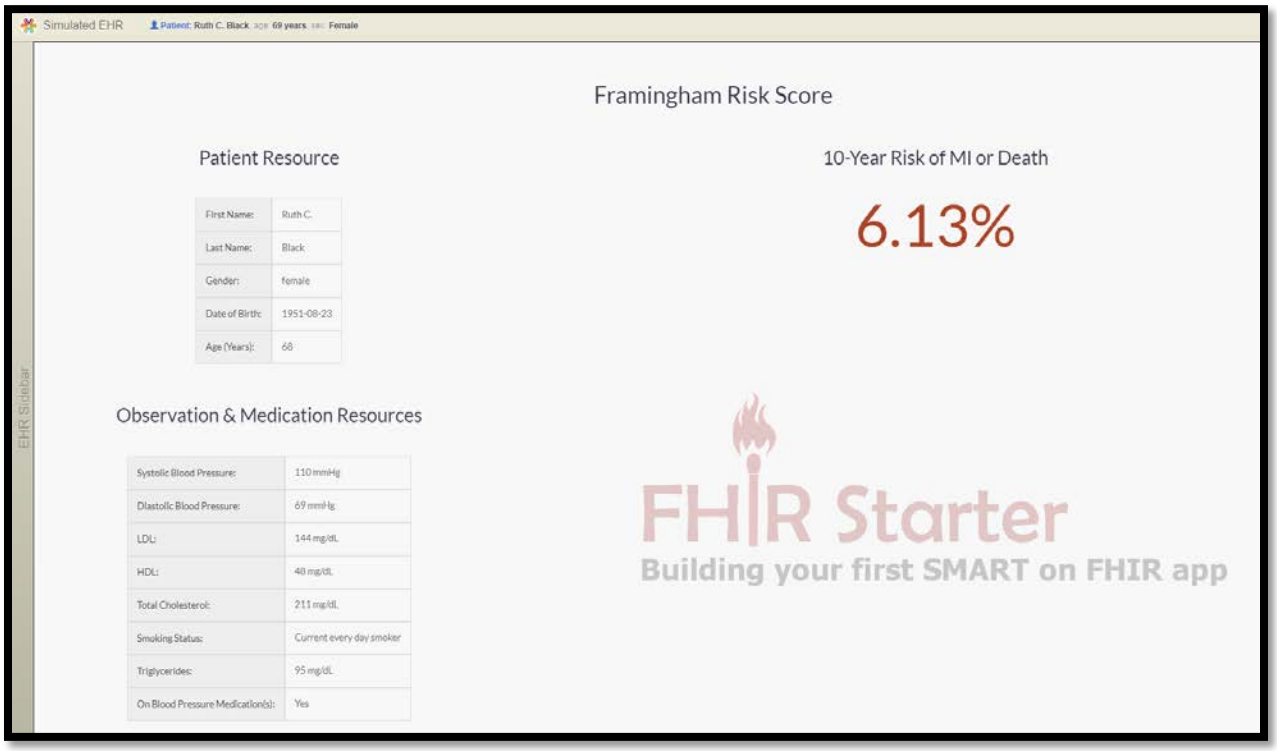
```
142             // Determine patient's smoking status
143             var smk_status;
144
145             if (typeof smk[0] != 'undefined') {
146
147                 if (getSmokingStatus(smk[0]).toLowerCase().includes("current")) {
148                     smk_status = 1;
149                 } else {
150                     smk_status = 0;
151                 }
152                 p.smk = getSmokingStatus(smk[0]);
153             } else {
154                 smk_status = 0;
155                 p.smk = 'Unk';
156             }
```

7) Find line 420. Complete this line. The goal of this function is to extract the value of "display" here.
    a. Hint 1: Objects inside of "coding" are in an array. Accessing the first item in this array would be written as *coding[0]*.

```
413     // Get smoking status
414     function getSmokingStatus(ob) {
415         if (typeof ob != 'undefined' &&
416             typeof ob.valueCodeableConcept != 'undefined' &&
417             typeof ob.valueCodeableConcept.coding != 'undefined' &&
418             typeof ob.valueCodeableConcept.coding[0].display != 'undefined') {
419
420             return /* Complete this code to extract the value of 'display' for  tobacco smoking status */
421
422         } else {
423             return undefined;
424         }
425     }
```

ANSWER

8) Launch App! again using the above steps. Congratulations! You've just created your ~~first~~ second fully functioning SMART on FHIR application!



Full example-smart-app.js file

Post workshop

1) Completed versions of each application are available below.
    a. BMI Calculator
    b. Framingham Risk Score
2) Please contact either Josh Herigon or Colby Uptegraft if you have any additional questions.
    a. Josh: Joshua.Herigon@childrens.harvard.edu
    b. Colby: colby.uptegraft@gmail.com