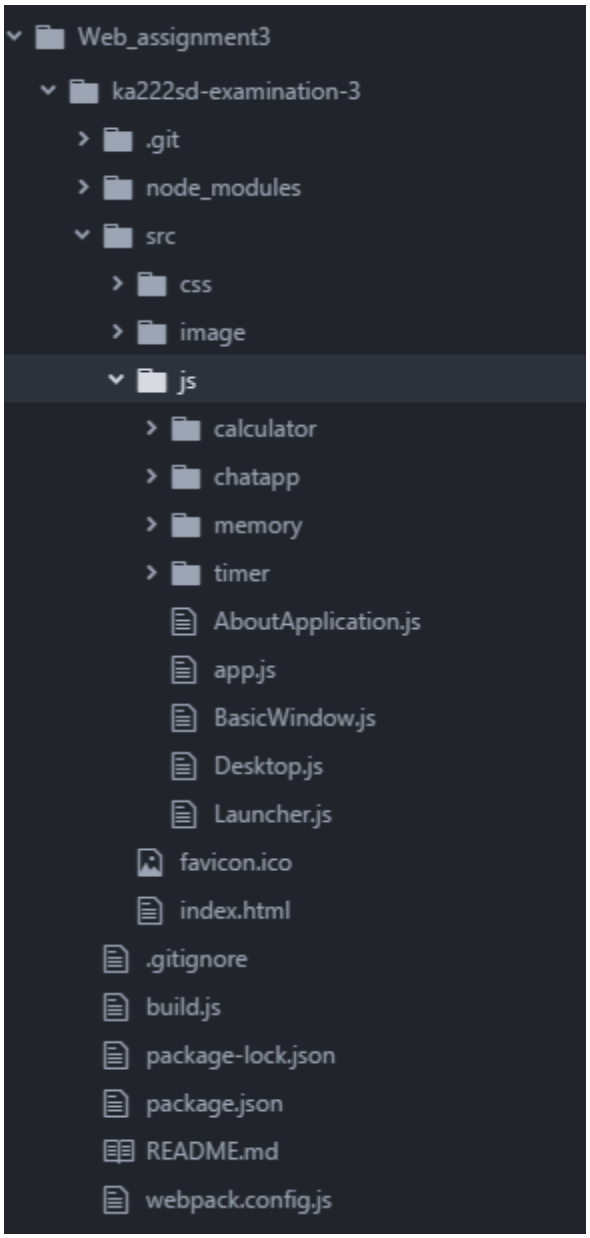
Documentation about Assignment 3: Personal Web Desktop SPA

Architecture of the application

I have used an object oriented approach using object prototypes in JavaScript for building my desktop web application. Firstly I have organized the files and divided them into folders .



App.js

The starting script of the application is app.js that is located under the js folder. In the app.js firstly we require a Desktop module object and then creating a new object of Desktop that calls the constructor of Desktop and then I am calling the init function of Desktop.

Desktop.js

In the Desktop class we require a Launcher module object , that we need for calling the main methods in Launcher.js and also to give as the parameter the desktop object. In the Desktop class the constructor initializes the main elements that an application need such as serialNumber(id),zIndex(foucus),clickX,clickY,offsetX,offsetY,windows(array that stores all the opened applications) and also creates an object of Launcher , and gives as the parameter a Desktop object. The init functions calls the init function of the launcher class. The main job of Desktop is to handle the basic functions through adding event listeners such as mouse down , mouse up, mouse move , key down , also handles which window button is clicked whether it is minimize , maximized or closed , and clears or closes a window by calling destroy function from Launcher which inherited from the application created and the application inherited it form the BasicWindow.js module object , also handles when an application window is moved by modifying the css (using left and top) .

Launcher.js

Firstly the Launcher.js requires all modules of the applications that I have created (MemmoryApp,ChatApp, AboutApp,CalculatorApp,timerApp) . The constructor of the Launcher initialize the variables that are used for the clock and also saves the parameter(object of desktop) in a variable(this.desktop).The **init** function that it is called from the Desktop.js adds a click event listener to the launcher (document.querySelector(".launcher")) part of the application (it is found in the index.html) and also calls the updateClock() function every second. When an element of the launcher div it is clicked it is called the **launcherClick** function.

launcherClick calls the **getClickedLauncherElement**(event.target) that return the div that is clicked and then we are saving the value of it in the value variable by using getAttribute (if we click calculator application it will return calculator) , in the icon variable we are saving the icon of the application that got clicked and also the title of the application into the title variable , handles also the close and open clicks in the launcher . The **laucherClick** function checks also if the click is in running windows than we use **swichToWindow** function , if not we start other

application by calling the **startApplication** function and adding also three parameters (value , icon , title).

startApplication firstly initialize two variables marginX and marginY by multiplying by 10 marginX with the offsetX(that we initialized with 1 in Desktop.js) and marginY with the offsetY(that we initialized with 1 in Desktop.js) that it is need for coordinating the application when it is opened. After that we create an object with name appOptions in which we add all the properties that we need for an application window (id,x(marginX) ,y(marginY) ,tabIndex (serialNumber) ,zIndex(focus) , icon , title , maximable(boolean), keyActivated(boolean)) some of the properties are unique for every application such as id , zIndex and tabIndex . After that we call the **createApplication** function by adding two parameters type(application name) and appOptions. The **createApplication** function checks the type of the application(name) and then initialize the new application by creating an object and then we call the initialize or print function of the new application(depends from the application) ,Example :

```
switch (type) {                    //If we click chat application
        case "chat":
         {
                //set option to be able to maximize window
                appOptions.maximizable = true;
                newApp = new ChatApplication(appOptions);//create an object of chat app
                 newApp.init();//calling the init function of the chat that starts the chat app

                break;
        }
}
```

After that we are return the object that we created and store it in newApp and then in the **startApplication** function we add listener to the window- buttons , save the object to window array ,add to the running-app list , increase the serialNumber , offsetX , offsetY, set the focus of new app and also we call the function **checkBounds**(newApp) to check it if the app-window is out of bounds and get it into bounds . So every application has its own elements and also its own unique id.

BasicWindow.js

BasicWindow.js is inherited by all the applications and handles the functionality and sets up the window for each particular application created in the Desktop by the Launcher.
BasicWindow.js is inherited by all the applications because every application is contained within a window with the same functionalities such as: destroy a window , print a window , minimize , maximize and to clear the content of the window application.

The constructor of the BasicWindow class gets app options as a parameter , sets up the id, x and y (left and top offset), window title , icon and adds the option of maximizable which depends on the application if we decide to make it available to maximize or not when we create the javascript file of the new application.

Example of a new Application

Firstly we should create a list in the launcher div that it is found in the index.hml and we should give it a value , name and an icon .(Example timer application)

```
<li value="timer">
<i class="material-icons md-30 launch-icon">av_timer</i>
          <div class="tooltip-container">
              <div class="tooltip-line">
                  <span class="tooltip-title">Timer</span>
              </div>
          </div>
      </li>
```

In index.html also we should create an template with an unique id and inside it the elements that we will need for the application (Example timer app) This template will be used to create every instance of the timer application.

```
<template id = "template-timer-application">
  <div id = "timer">
    <h1>00:00:00</h1>
    <button id="start" class="b" value = "start">Start</button>
    <button id="stop" class="b" value = "stop">Stop</button>
    <button id="clear"class="b" value="clear">Clear</button>
  </div>
</template>
```

After that, we should create a TimerApplication file i.e. a class that is used to setup the application. The TimerApplication class inherits the BasicWindow class so that we will be able to minimize, maximize , print and destroy the instance of the app. In the TimerApplication we instanciate the Timer, in our case **timerMain**.

timerMain is the class that holds all the methods and properties of a timer. So each timer application that is started will be an instance of this class. I have created a constructor of timerMain class that initialize the variables that I need it and also calls the function **addEvents**. **AddEvents** add the click event in this application so when a button is clicked calls the **click** function .The **click** function check if a button was clicked and calls the **buttonClicked** function with adding in parameter the button that was clicked. The **buttonClicked** function gets the value

of the button that was clicked and checks if the values is start , stop and clear. If it is start calls the **start** method that calls the **add** function which increase the seconds and writes it in the application and calls the **timer** function, the **timer** function calls the **add** function again every second until there is not clicked stop or clear button , if the stop button is clicked clears the Timeout or if clear button is clicked resets the seconds and also resets the text content of h1 (00:00:00).

Once we have the timer class ready, we can now add it to the Launcher **createApplication** method. We also give it an id (I named it 'timer') .(Example)

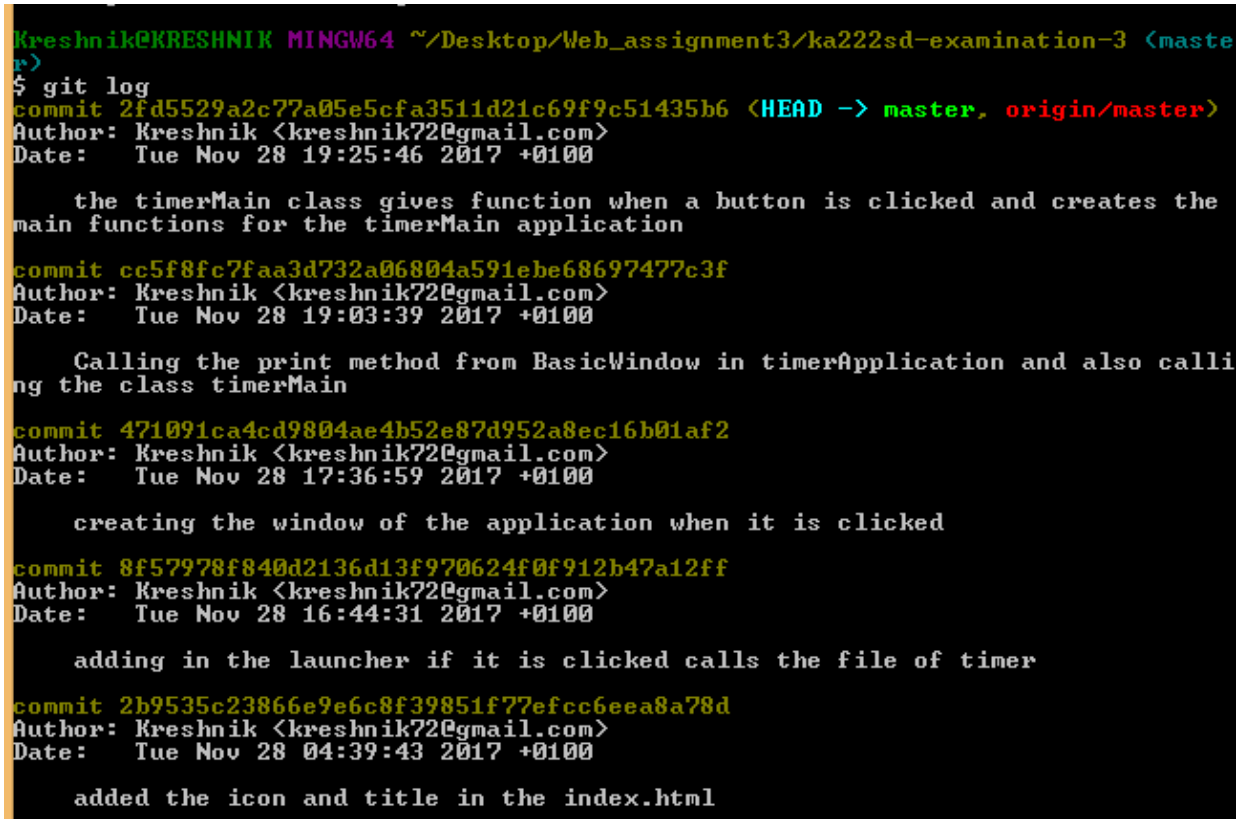
```
case "timer":
{

        newApp = new timerApplication(appOptions);
        newApp.init();

  break;
}
```

The launcher will be able to now detect when the 'timer' icon is clicked in the desktop in the launcher section. When the user clicks on the timer icon, the **launcherClick** method will be triggered with the parameter event passed on it so that we can detect which instance of the timer is clicked. (We can have many timers open at the same time, so it is important to pass in the context). **louncherClick** method will try to detect if we are starting a new timer instance or we reopen an existing one. If it's a request to make a new instance we call the **startApplication** function of the launcher class and we pass in type (the id 'timer'), icon (the icon of the application), and the title to be displayed in the window. Then, the **createApplication** method is called which checks which instance we want to make. This function will instanciate a new Timer Application and initialize it.

Below is my git log history for the new application



Main steps for creating new application

- Firstly we should create a list in the launcher div that it is found in the index.hml and we should give it a value , name and an icon
- In index.html also we should create an template with an unique id and inside it the elements that we will need for the application
- We should create an javascript file to setup the application which should inherits the BasicWindow class so that we will be able to minimize , maximize , print and destroy the instance of the app .
- We should create another file to make the code more organized that will be write the main functions of the application that we want to create . This class is called from the previous javascript file and gives in the parameter the window-content(each application has its own window content and his own properties)

- Once we have finished with the javascript files for new application, we can now add it to the Launcher **createApplication** method.(Example)

```
case "timer":
{

        newApp = new timerApplication(appOptions);
        newApp.init();

  break;
}
```

What has been mainly written by me?

Both of (me and my colleague) decided that we would do a pair programming about the following parts "*Launcher* , *Desktop* , *BasicWindow* , *Chat* " and we decided that each of us will add a different application to this assignment. So I decided to add Calculator and Timer applications.

Calculator

This application I have created in same logic that I have explained above (how to add a new application).I have created to files for this application:
CalculatorApplication - inherits the BasicWindow class so that we will be able to minimize , maximize , print and destroy the instance of the app .Then I call the calculator class and pass to the parameter the window-content and also the part that will print the result when we make some calculation.
Calculator - I have a constructor that initialize the variables that I need (previousNumber , currentNumber , result) and also adding a click event in this application . I have a function click that checks if we click a button , if we click a button calls the calculatorClicked function to make the calculation and to show it.

Timer

This application I have explained above in the "*Example of a new Application" part*

What has been developed by both of us and what did we learn during this process?

Briefly what we improved in our knowledge during this assignment were:

- how to manipulate properly with DOM
- how to make it offline available even though currently it is not implemented in our application (maybe in the future we can add a service worker)
- local storage usage to store data locally within users browsers – mainly used in the Chat application for saving new Message (Chat.prototype.saveNewMessage and Chat.prototype.readStoredMessages)
- event handling such as mouse move, mouse down, clicked etc.
- Web Sockets how to connect, send, receive data from the server
- Using CSS Sprites (used in the Chat emoji)

Most of the code of the application has been developed together with my colleague, due to the fact that this project was so interesting, fun to code on and we wanted it to make as impeccable as possible. During this process we learned that one of the better ways for this SPA was to divide the code into different classes and modules to achieve better code organization and structure and the main goal was to ease the process of adding a new application in our SPA. We decided to use templates for the application printing and application windows. We learned that it will be better if we use the prototype property, so it will be easy to add new properties to an existing prototype but also it will be faster the object creation (ex the BasicWindow object instance or others) since we won't have to wait for other functions to be recreated each time a new object is being created. Module exports were used to export different functions and functionality that later on will be used in other applications. It helped us to : split our code into multiple files (which later in the future it will be easier for maintenance compared to a code written in only one file), reuse the code for multiple applications without copying the same code from one application to another, automatically load the code we need based on what the application needs rather than manually to specify what should be run. The main parts of the application as *BasicWindow*.js, Launcher,js, Desktop.js and the functionality of Chat Application were developed together. BasicWindow main idea for every particular window was to have different id (which later will be used for checking the windows properties in other functions too), mainly it will have its own options such as tabIndexing, title, icon, zIndex and will it be maximizable or not. *Launcher* main idea was to create a class that will contain applications that will be started or created on a click in the launcher div part in the body. As a functionality duty we had to make something that will handle the launcherClick, get the element that has been targeted by the click event and we wanted to check if it is a new application (than the function StartApplication will be called) or it was a click on the list of running apps for this one we built the Switch to Window function.

Desktop's main motivation for us was to make it to handle basic features such as handling mouse events and deals with the function to clearDesktop when needed and setting the focus on elements (in our case applications)

Chat's was the most fun part to work on it. Using websockets to connect to server we managed to try a simple communication between 2 of us in the beginning. Then we added functionality such as if we get an error from the server send the chat offline, how new message would be displayed and how the message will be send (.formSubmit). We added some other functions that will handle with the message being sent (checking, parsing, adding link or emoji). After a while we decided to add Settings on it (such as changing the username, server and the channel) and after each change the new Server connection will be initialized.