WebApp Checklist

Setup

GitHub

- The project should be in a separate repository named `webapp`
- Live branch should be `gh-pages` (to allow viewing on GitHub Pages)

Files

- `index.html` - The app
- `css/main.css` - App styles
- `js/main.js` - App logic
- `js/plugins.js` - Any helper functions (e.g. personal library, shims)

Style Guide

- Adhere to the given Style Guide
  https://drive.google.com/file/d/0B6ZIsommzsgVMU1mLW16Z1RURk0
- OR some community driven JSCS style (like jQuery's, Crockford's, etc.)

Source Files

- https://drive.google.com/file/d/0B_q5xsDIvfrhbXMyOGx6dFYyRlU
- Copy the contents of the `Source` folder into your project root

Checklist

Stage 1

- Download H5BP as the base for your project http://h5bp.com
- Copy the HTML content from the index.html (from Source Files) of the WebApp project into the index.html that came with H5BP
- `fonts` folder can be ignored for now
- Implement styling and fluid layout based on the screenshots and demo video (in the Source Files as well)

Stage 2

- Copy the AJAX plugin into `js/plugins.js`
  - Reference - AJAX Plugin v2
    http://codepen.io/netcraft/pen/Kvyte?editors=001
- Implement WebApp notifications
  - On page load, make a GET request to `data/config.json`
  - If the file exists and has a `notification` key in it, display the content in the notification area, otherwise, notification area should stay hidden
- Add functions to the library that implement cross browser event handling (support IE8+) (`addEvent`, `removeEvent`)

    - Usage
        - `addEvent(elem, type, handler)`
        - `removeEvent(elem, type, handler)`
    - Reference
        - See Events https://docs.google.com/document/d/1P44wmU7epwEvcxOJ8gsSheD7Gs7ZCDneRqKkyTkh4BY
        - Cross Browser Event Handling in JavaScript http://www.anujgakhar.com/2013/05/22/cross-browser-event-handling-in-javascript/
        - A cross-browser implementation of addEventListener/AttachEvent https://gist.github.com/eduardocereto/955642
        - A cross-browser way of assigning event handlers http://javascript.info/tutorial/introduction-browser-events#a-cross-browser-way-of-assigning-event-handlers
- Implement tabs navigation behavior with JS
    - Use the `addEvent` plugin you wrote
    - Each tab click should change the URL `hash` value (without jumping to an `id` of the same value)
        - http://www.w3schools.com/jsref/prop_loc_hash.asp
        - https://developer.mozilla.org/en-US/docs/Web/API/WindowEventHandlers.onhashchange
        - http://stackoverflow.com/questions/3870057/how-can-i-update-window-location-hash-without-jumping-the-document
    - Changing the `hash` value in the URL should trigger the relevant tab
    - Support tabs navigation using the keyboard
    - Make sure there's a `focus` state similar to `hover` when navigating with the keyboard
- Extra (optional)
    - Support dropdowns navigation with the keyboard

Stage 3

- Update `plugins.js` with JS Library v3 http://codepen.io/netcraft/pen/rJtbd?editors=001
- Add HTML content for each tab
    - Quick Reports and My Team Folders tabs
        - A form that allows saving a website as a setting (name, URL)
        - Should have a wheel icon (top right), clicking will toggle the form visibility and focus the first input
        - Should have Save button and Cancel link
            - Cancel - Hide the form
            - Save
                - Validate the form

            - If Name or URL entered, the other value must be filled as well.
            - If anything is invalid, add a red border around it, focus on the first invalid input
        - If valid, hide the form
        - Add an iFrame with the URL of the last report settings
        - Add a dropdown (top left) with the report names
            - Choosing a report should update the iFrame
        - Add an arrow icon (top right) next to the wheel
            - clicking will open the iFrame URL in a new tab
    - The whole form should be accessible using the keyboard
        - Tab press through the inputs, buttons and icons
        - Enter inside any input should act as Save
        - Esc inside any input should act as Cancel
    - Removing reports
        - Deleting entries in the form should remove the report from the dropdown
        - If all entries are deleted, revert to the initial state
    - Page refresh
        - For now, save reports data in a JS object, this will be lost after page refresh. Later we'll do it with cookies or localStorage
- My Folders and Public Reports tabs
    - An iFrame with a predefined URL (choose whatever you like)
    - An arrow icon (top right), clicking should open the iFrame URL in a new tab

Stage 4

- Add input placeholders for all forms, and the search box (support IE8)
- Validate URL with RegEx (support `http` and `https`)
    - Add protocol if omitted (when Saved), e.g. `netcraft.co.il` will become `http://netcraft.co.il`
- Make the Search box (top right) search a specific report
    - Searching a report name (or part of it) should find it under "Quick Reports" or "My Team Folders" and make it the visible report
    - If not found, add a message in the notification area "The searched report _____ was not found."
    - If multiple reports with the same name exist, choose the first one
- Continue IE8 support (layout + behavior)

Stage 5

- localStorage
    - Save the reports settings using localStorage

        - Use only 1 key for the whole data
    - On page load, restore the saved reports data into the forms and dropdowns
    - Also save the last selected tab