Web Essentials Dojo

## Code:

Clone the project code from here:

<https://github.com/gartdan/WebEssentialsDojo.git>

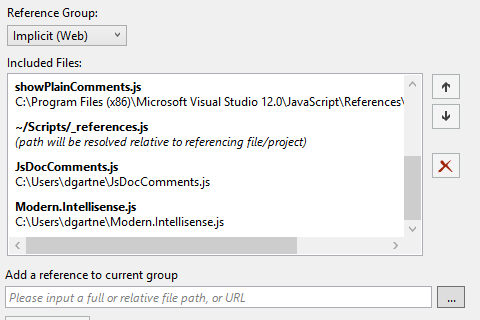
## Lab Guide

This lab guide can be found here:

<http://1drv.ms/Sett9S>

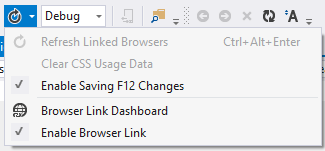
## Initial Setup

1. Install the latest version of VS Web Essentials @ <http://vswebessentials.com/>
2. In Visual Studio press Ctrl+Q to focus on quicklaunch, enter ‘java ref’ and hit enter
3. Ensure the following javascript Intellisense items are added (sometimes VS Web Essentials does not correctly add them). Manually add them from your c:\Users\[Username] directory If necessary



## BrowserLink Exercise

1. Open the **BrowserlinkDemo** Solution
2. Ensure BrowserLink is enabled



1. Click Browse-With, Select Google Chrome and Internet Explorer and click Browser
2. In Chrome, note the BrowserLink toolbar in the bottom right corner
   1. Hit the Ctrl key to show and hide this toolbar
3. Enable F12 auto-sync



1. Click Design, and move your mouse over the different HTML elements. Note how they are selected in Visual Studio
2. Click on the header text. Edit the text in the browser and these changes are reflected in Visual Studio
3. In Visual Studio, click the BrowserLink refresh icon to have all open browsers refresh with the latest changes
4. Right click on the large header, and click “inspect element.” Change the background color in the F12 dev tools, and it will be reflected in the style.css file
   1. (Note, common css libraries such as bootstrap are ignored by default. You can change the settings under Options > Web Essentials > Browser Link > CSS Usage files to ignore)

## LESS Exercise

1. Open Content\LessStyles.less
2. Copy the following code into the file:

@main-background: #ffffff;

@main-foreground: #000000;

@main-foreground-light: #dedfe7;

@header-color: #ff6a00;

@main-foreground-lightest: lighten(@main-foreground-light, 10%);

body {

    color: @main-foreground;

    background-color: @main-background;

}

.bordered{

    border-top: dotted 1px @main-foreground;

    border-bottom: solid 2px @main-foreground;

}

div.header

{

    color: @header-color;

}

#levNav.header

{

    color:@header-color;

}

th

{

    color:@header-color;

    background-color:@main-background;

    .bordered;

}

th.light

{

    color:@main-foreground-light;

}

th.lighter

{

    color:@main-foreground-lightest;

}

1. Enable the Preview Pane if it is not already enabled
   1. Tools > Options > Wen Essentials > LESS > Editor, Show Preview Pane
2. In the th.lighter selector, mouseover the @main-foreground-lightest, notice its value is shown
3. Right-click on @main-foreground-lighter, select Go To Definition
4. Select all of the variables at the top of the file, right click and select Web Essentials > Extract to file. This can be used to modularize your less files
5. Add a mixin to div.header by entering .bordered inside of the selector. Save the file and notice the changes in the preview pane.
6. Experiment with other changes in the file, load the Contact Page (Home/Contact) and see the changes immediately reflected.

## Javascript Exercises

### Bundling

1. Add a new javascript file to the **Scripts** folder called scripts.js
2. Enter the following javascript into the file

function sayHello(message) {

    alert(messge);

}

1. In VS, select bootstrap.js, jquery.1.10.2.js, jquery.validate.js, scripts.js, right-click and Web Essentials > Create Javascript Bundle File. Name the bundle MyBundle.js
2. Open MyBundle.js, scroll to the bottom and notice that the sayHello function is there
3. Close the MyBundle.js file
4. In scripts.js, make a change to the function
5. Save and rebuild your solution.

function sayHello(message, person) {

    alert(messge);

    alert(person);

}

1. Open MyBundle.js and scroll to the bottom. The function has been updated.

### Go to Definition & Intellisense

1. Open script.js in Visual Studio
2. Select jquery-1.10.2 in solution explorer and drag it to the first line of the file to add a reference
3. Type the following to view the Intellisense

$("#container").html("blah");

1. Right click on the html function, click ‘Go To Definition’
2. Right click on the html() function, click ‘Find All References’

### JSHint

1. Copy the following function into script.js and Save the file

function doCompare(item1, item2) {

    if (item1 == item2)

        return 'Equals';

    else

        return "Not Equals!"

}

1. View the messages in the error list

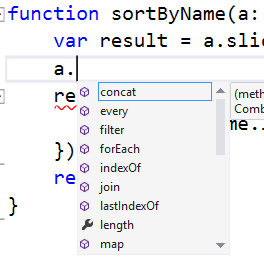
### Minification

1. Right-click on scripts.js, select Web Essentials > Minify

### Intellisense

1. Open \_references.js. This is the global Intellisense file

## TypeScript Exercise

1. Open The **TypeScriptDemo** Solution
2. Open **DemoStart\helpers.js** – we will learn TypeScript by adding type bindings to this source
3. Create a new TypeScript file in the DemoStart folder > **File > New > TypeScript > helpers.ts**
4. Copy the javascript in helpers.js into helpers.ts
5. Declare the **a** parameter in sortByName as an array by adding an **any[]** type annotation
   1. function sortByName(a: any[]) {
6. In the method body, type “a.” to see the Intellisense you now get.
7. 
8. Compile the project, notice the type bindings “evaporate” and helpers.js is unmodified. Typescript complies to javascript
9. Declare an interface and modify the sortByName parameter list to take arguments of type Entity (interfaces are duck-typed, you don’t have to explicitly implement):

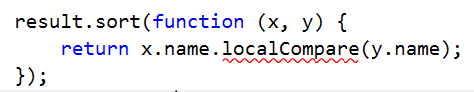
interface Entity {

    name: string;

}

function sortByName(a: Entity[])

1. Notice that we now detect an error that would have otherwise not been caught until runtime. There is no localCompare method on string.



* 1. Use your newly found Intellisense powers to correct the error (the method that actually should be called is **localeCompare**)

1. Enable the Web Essentials TypeScript Preview Pane
   1. Tools > Options > Web Essentials > TypeScript > Show Preview Pane: True
   2. Close and re-open the TypeScript file. When saving your TypeScript file, you can see the compiled output in a window next it.
2. Let’s add a method to the interface, and instantiate an object that implements the interface:

interface Entity {

    name: string;

    price: number;

    inStock?: boolean;

    getName(): string;

}

var e: Entity  = {

    name: 'Dan',

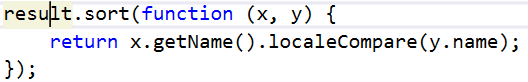
    price: 50.0,

    getName: function () {

        return this.name;

    }

};

1. Change the sortByName function to call getName. The type information flows through (as expected!)
   1. 
2. Declare a class type:

class Product {

    constructor(public name: string, public price: number) { }

    getName() {

        return this.name;

    }

}

1. Declare an array of Products

var products: Product[] = [

    new Product("Bananas", 3.50),

    new Product("Cabbage", 0.99),

    new Product("Peaches", 2.00),

    new Product("Artichoke", 2.00),

    new Product("Gala Apples", 3.50),

    new Product("Granny Smith Apples", 3.50)

];

1. Sort them:

var sorted = sortByName(products);

alert(sorted[0].getName());

1. Open demo.html and see the alert
2. You can also reference tons of existing JS libraries from TypeScript as well. Right-click on the project and select “Manage NuGet Packages”. Search online for “TypeScript”
3. Install the jquery bindings if not already installed
   1. Also, see <http://definitelytyped.org/> for the available js library bindings
4. Add a reference from jquery to the TypeScript file by dragging and dropping the jquery.d.ts file to the top of the helpers.ts file

/// <reference path="../scripts/typings/jquery/jquery.d.ts" />

1. Enjoy the jQuery Intellisense and give the demo some pizzazz!

Copy&Paste the following code after the alert.

function animate() {

    var block = $('#content');

    var r = Math.floor(Math.random() \* 256);

    var b = Math.floor(Math.random() \* 256);

    var g = Math.floor(Math.random() \* 256);

    block.animate({ backgroundColor: $.Color(block.css('backgroundColor')).rgba([r, g, b, 1.0]) }, 500, animate);

}

$("#content").html(sorted[0].getName());

$("#content").fadeOut(500).delay(500).fadeIn(750).fadeOut(1000).fadeIn(1000).animate({ top: '-=400' });

animate();