Contents

[Contents 1](#_Toc433035844)

[Development checklist 12](#_Toc433035845)

[Decide on design parameters: 12](#_Toc433035846)

[Which libraries may be added? Which devices will be supported, etc 12](#_Toc433035847)

[Create a wireframe model 12](#_Toc433035848)

[Create a content priority list 12](#_Toc433035849)

[Scroll behavior - http://blog.chartbeat.com/2013/08/12/scroll-behavior-across-the-web/ 12](#_Toc433035850)

[Check a reference spreadsheet for components and frameworks that might be useful 12](#_Toc433035851)

[https://docs.google.com/spreadsheet/ccc?key=0AiN0QfBTPpOCdDFjWlM0eU1ra21XanZkekxGbjA2WWc#gid=0 12](#_Toc433035852)

[Use http://caniuse.com/#search=column-gap to verify if chosen approach is valid across the target deployment platforms. 12](#_Toc433035853)

[Create placeholders for images during web page design. 12](#_Toc433035854)

[Add vendor prefixes using a generation tool. 12](#_Toc433035855)

[What to prefix: 12](#_Toc433035856)

[What to prefix tool/website: enter the CSS feature and see whether is needs prefixing 12](#_Toc433035857)

[http://shouldiprefix.com/ 12](#_Toc433035858)

[Postcss/Autoprefixer 12](#_Toc433035859)

[An interactive version is located at: http://autoprefixer.github.io/ 12](#_Toc433035860)

[Fonts and common fallback fonts 13](#_Toc433035861)

[For FEWD\_P1 13](#_Toc433035862)

[http://ufonts.com/download/gotham-thin/149718.html 13](#_Toc433035863)

[General info 13](#_Toc433035864)

[https://en.wikipedia.org/wiki/Web\_typography 13](#_Toc433035865)

[Core fonts for the web 13](#_Toc433035866)

[https://en.wikipedia.org/wiki/Core\_fonts\_for\_the\_Web 13](#_Toc433035867)

[Compatible Typography 13](#_Toc433035868)

[https://www.onedesigns.com/tutorials/font-families-for-cross-compatible-typography 13](#_Toc433035869)

[CSS Web Safe Font Combinations 13](#_Toc433035870)

[http://www.w3schools.com/cssref/css\_websafe\_fonts.asp 13](#_Toc433035871)

[The @Font-Face rule and useful web font tricks 13](#_Toc433035872)

[http://www.smashingmagazine.com/2011/03/the-font-face-rule-revisited-and-useful-tricks/ 13](#_Toc433035873)

[How to use any font you like with CSS3 13](#_Toc433035874)

[http://www.webdesignerdepot.com/2013/01/how-to-use-any-font-you-like-with-css3/ 13](#_Toc433035875)

[Google fonts 13](#_Toc433035876)

[https://www.google.com/fonts 13](#_Toc433035877)

[Git Setup 15](#_Toc433035878)

[PowerShell environment 15](#_Toc433035879)

[Change prompt 15](#_Toc433035880)

[http://stackoverflow.com/questions/18292063/what-does-the-prompt-master-1-0-0-mean-when-using-git-command-line 15](#_Toc433035881)

[Run scheduled tasks 15](#_Toc433035882)

[http://blogs.technet.com/b/heyscriptingguy/archive/2011/01/12/use-scheduled-tasks-to-run-powershell-commands-on-windows.aspx 15](#_Toc433035883)

[Setup and clone Git repository from github: 15](#_Toc433035884)

[1. Login to Github 15](#_Toc433035885)

[2. On home page click on “New repository” button 15](#_Toc433035886)

[3. Enter a name for the repository 15](#_Toc433035887)

[4. Check “Initialize this repository with a README” 15](#_Toc433035888)

[5. Click on “Create repository” button 15](#_Toc433035889)

[6. After repository is created, note the page that displays that shows how to create a client of the remote repository on the local computer. 15](#_Toc433035890)

[a. 15](#_Toc433035891)

[ 16](#_Toc433035892)

[7. If using “Quick setup” make sure to select the correct security protocol string, HTTPS or SSH. 16](#_Toc433035893)

[8. Switch to the client computer 16](#_Toc433035894)

[9. Start a git shell or command line shell that includes git path and environment variables. 16](#_Toc433035895)

[10. Change/create folders to one folder above where the remote repo will be created. 16](#_Toc433035896)

[11. Issue a clone command: 16](#_Toc433035897)

[a. For instance to clone a github hosted repository named UP0 under user Alanwea the following command will create a folder of the same name and then clone the remote repository to the local folder. 16](#_Toc433035898)

[Git clone https://github.com/alanwea/UP0.git 16](#_Toc433035899)

[12. Change folders into the root of the new local repository 16](#_Toc433035900)

[13. Check that git was successfully installed: 16](#_Toc433035901)

[Git --version 16](#_Toc433035902)

[14. Initialize the new cloned repository by: 16](#_Toc433035903)

[a. Set a user name: 16](#_Toc433035904)

[git config –global user.name “Alan Weatherhead” 17](#_Toc433035905)

[b. Set user email: 17](#_Toc433035906)

[git config –global user.email alanwea@hotmail.com 17](#_Toc433035907)

[c. Set a default editor for typing in messages, like the –m message that is required on a commit. Note especially the use of “ and ‘ to enclose the path to notepad++ and the entire command with arguments for 32-bit Notepad++ on a 64-bit Windows environment.git http://stackoverflow.com/questions/1634161/how-do-i-use-notepad-or-other-with-msysgit) 17](#_Toc433035908)

[git config --global core.editor "’C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin" 17](#_Toc433035909)

[15. Set autocrlf to handle line endings correctly. For git on Windows: 17](#_Toc433035910)

[Git config –global core.autocrlf true 17](#_Toc433035911)

[16. Check that the config settings were correctly set: 17](#_Toc433035912)

[Git config –list 17](#_Toc433035913)

[17. Move, copy or create files in the local repository root folder, then add them to the git tracking list. In this case the entire contents of the files in the local git repository will be added: 17](#_Toc433035914)

[18. 17](#_Toc433035915)

[19. If using SSH for authentication: Generate client side SSH key using ssh-keygen in the Git Shell. (Instructions adapted from https://help.github.com/articles/generating-ssh-keys/ ) 18](#_Toc433035916)

[20. Start the ssh agent: 18](#_Toc433035917)

[21. Add SSH key to the ssh-agent: 18](#_Toc433035918)

[22. Original instructions use “clip < ~/.ssh/id\_rsa.pub” to copy the generated SSH public key to the clipboard. This doesn’t work under PowerShell. Instead I used Notepad++ to open the SSH public file and copied it to the clipboard. 18](#_Toc433035919)

[23. Setting up GitHub side of SSH and security 18](#_Toc433035920)

[a. Login to GitHub account 18](#_Toc433035921)

[b. Select “Profile” then “Settings” from upper right corner of home page. 18](#_Toc433035922)

[c. Select SSH Keys on left menu 18](#_Toc433035923)

[d. Click Add SSH key 18](#_Toc433035924)

[e. Enter title, i.e. computer name 18](#_Toc433035925)

[f. Paste key from clipboard into the Key field 19](#_Toc433035926)

[g. Click Add key 19](#_Toc433035927)

[h. Enter GitHub password 19](#_Toc433035928)

[i. Test the connection 19](#_Toc433035929)

[i. 19](#_Toc433035930)

[24. Show current remotes with “git remote –v”. If SSH will look something like this: 19](#_Toc433035931)

[25. If HTTPS will look like this: 19](#_Toc433035932)

[19](#_Toc433035933)

[26. Push local repository to the remote: 19](#_Toc433035934)

[27. Get detailed info about a remote: 19](#_Toc433035935)

[a. Git manual says “git remote show” returns even more info, but that wasn’t the case when I did it, it returned the same info. Looks like there is an error in the manual. 19](#_Toc433035936)

[28. Rename a remote – STUBBED 19](#_Toc433035937)

[29. Removing a remote – STUBBED 20](#_Toc433035938)

[Git Common tasks 21](#_Toc433035939)

[Resolving merge conflicts 21](#_Toc433035940)

[https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/ 21](#_Toc433035941)

[Resolving remote missing 21](#_Toc433035942)

[“template” repository existed on Github. 21](#_Toc433035943)

[“template” repository existed locally, but repository was deleted and then recreated with a git init. 21](#_Toc433035944)

[Local template repository has changes that are not in the remote repository and the “git remote” returns nothing, “origin” does not exist. 21](#_Toc433035945)

[git fetch https://git@github.com:alanwea/template 21](#_Toc433035946)

[Works, but says there are no common commits, gives this at the end: “\* branch HEAD -> FETCH\_HEAD” 21](#_Toc433035947)

[git push 21](#_Toc433035948)

[result: “fatal: No configured push destination” 21](#_Toc433035949)

[ 21](#_Toc433035950)

[The remotes now exist. 21](#_Toc433035951)

[ 21](#_Toc433035952)

[This pulls the master branch from the remote repository into the local repository. Looks like the Jquery stuff was not in the local repository. Now the local repository is up to date, but the remote repository does not have the changes made in the local repository. 21](#_Toc433035953)

[ 22](#_Toc433035954)

[The local changes are pushed to the remote and everything should be good to go. 22](#_Toc433035955)

[Git Auto Commit 23](#_Toc433035956)

[Using MS tasks: 23](#_Toc433035957)

[Run Task Scheduler from Control Panel/Administrative. Create a new task to run at whatever start time and interval. Create an action to run a script in the git shell of the repository to be auto committed. In example, the argument is to create a debugging log, this is optional in production. Start in is the folder at the root of the target repository (that is a folder with a .git folder). 23](#_Toc433035958)

[23](#_Toc433035959)

[Create a batch file in the root repository folder, in this case it is named Autogit.bat and contains: 23](#_Toc433035960)

[ 23](#_Toc433035961)

[Add Autogit.log to .gitignore file? 23](#_Toc433035962)

[Add other commands as needed. 23](#_Toc433035963)

[Might add one to change the interval of the job itself. 23](#_Toc433035964)

[Might convert the .BAT to a full-fledged powershell script 23](#_Toc433035965)

[Add parametrization to bat file 23](#_Toc433035966)

[Git Hooks 24](#_Toc433035967)

[Testing environment variables that Git passes to hooks 24](#_Toc433035968)

[https://www.digitalocean.com/community/tutorials/how-to-use-git-hooks-to-automate-development-and-deployment-tasks 24](#_Toc433035969)

[Development Environment 25](#_Toc433035970)

[File and update structure 25](#_Toc433035971)

[25](#_Toc433035972)

[25](#_Toc433035973)

[HTML and CSS 26](#_Toc433035974)

[Instructor Notes 26](#_Toc433035975)

[Very nice article on CSS selectors. 26](#_Toc433035976)

[Documentation 26](#_Toc433035977)

[Browsers use default stylesheets to determine how to display HTML elements. You can view the default style rules for h1 and other elements for the following browsers: 26](#_Toc433035978)

[ WebKit (Chrome and Safari) 26](#_Toc433035979)

[ Firefox 26](#_Toc433035980)

[ Internet Explorer 26](#_Toc433035981)

[Experimental Work 26](#_Toc433035982)

[CSS variables 26](#_Toc433035983)

[Why you need to use CSS variables 26](#_Toc433035984)

[http://www.creativebloq.com/netmag/why-you-need-use-css-variables-91412904 26](#_Toc433035985)

[CSS custom properties for cascading variables 26](#_Toc433035986)

[https://drafts.csswg.org/css-variables/ 26](#_Toc433035987)

[CSS Calc() 26](#_Toc433035988)

[http://www.w3.org/TR/css3-values/ 26](#_Toc433035989)

[CSS3 Gems: the calc() function 26](#_Toc433035990)

[http://www.sitepoint.com/css3-calc-function/ 26](#_Toc433035991)

[Submission notes for FEWD/P0: 27](#_Toc433035992)

[Set up GitHub, including SSL certificate https://help.github.com/articles/generating-ssh-keys/ 27](#_Toc433035993)

[http://stackoverflow.com/questions/13800289/configure-git-clients-like-github-for-windows-to-not-ask-for-authentication/18607931#18607931 27](#_Toc433035994)

[http://stackoverflow.com/questions/2916845/different-default-remote-tracking-branch-for-git-pull-and-git-push 27](#_Toc433035995)

[http://corlewsolutions.com/articles/article-11-guide-to-setting-up-ssh-on-windows-7 27](#_Toc433035996)

[Installed Git in the folder I’m using for P0 (I’m using command line Git for now, although I know that TortoiseGit, etc. exist). Configured user name, email and default editor (Notepadd++). Setup .gitignore to exclude files that don’t need to be tracked. Added existing P0 files to repository. Committed them and then pushed to GitHub using SSL. 27](#_Toc433035997)

[I’m hosting on my own website using Microsoft Expression Web (for historical reasons), so I added Git to the folder holding the website production files. I haven’t done a complete end to end test yet, but the idea is to do development on my local machine and pull/Push to GitHub. When my website needs to be updated, I’ll create a release branch on GitHub and pull changes from GitHub to my production website folders, then publish to a test area on my website and when satisfied, publish to the live site. Any test area code changes I’ll push/merge back to the GitHub release and main branches. 27](#_Toc433035998)

[Completed the Udacity HTML/CSS classes up to the first couple of minutes of Bootstrap, I know we will be looking at that later, so stopped there. 27](#_Toc433035999)

[Finally found a picture for the P0 assignment that I’m happy with (had to ask someone to take it). 27](#_Toc433036000)

[Looked online for suggested folder structures for development. Lots of controversy as you might expect. For P0, I’m going to start using a hybrid structure based on the many suggestions that I found to get in the habit of starting this way, even though I know that P0 doesn’t require it. 27](#_Toc433036001)

[I’ve had some past experience with JavaScript, but it’s been a while so I reviewed articles on: 27](#_Toc433036002)

[Lazy loading JavaScript (aka deferred loading) – not needed right now, but will be eventually for fast load of rich content pages. 27](#_Toc433036003)

[Internationalization of Web pages ( http://www.smashingmagazine.com/2014/06/css-driven-internationalization-in-javascript/ , http://www.w3.org/TR/its20/ ) – also not needed, but a consideration later if website will be seen by non-English speakers. 27](#_Toc433036004)

[Ease of use – that is how to present to web site viewers with disabilities? Keystroke handling, text to speech, voice response. 27](#_Toc433036005)

[JavaScript event handling – this is going to be needed eventually too, but I was also interested in how handling events only for navigable items on pages, could be combined with lazy loading to speed up web page loading. Secondary, was looking at events generated by touch screens, I’ve only every handled click events before. Also, if it made sense to funnel all page events through a table driven event handler, maybe as a state machine for responses (?). (http://www.smashingmagazine.com/2013/11/an-introduction-to-dom-events/ ) 27](#_Toc433036006)

[Looked at concepts of responsive and adaptive design – responsive was briefly covered in the HTML/CSS classes. 28](#_Toc433036007)

[Looked at what alternatives there are to Javascript ( I know, this isn’t needed for P0 either). I have some experience with VBScript, but not the other alternatives: ActionScript, Dart, Typescript and Python (though have it installed). 28](#_Toc433036008)

[( https://en.wikipedia.org/wiki/Client-side\_scripting#List\_of\_Client-Side\_Scripting\_languages ) 28](#_Toc433036009)

[Install JQuery 28](#_Toc433036010)

[https://jquery.com/download/ 28](#_Toc433036011)

[A more in-depth look at CSS selectors (https://css-tricks.com/how-css-selectors-work/ ) 28](#_Toc433036012)

[Haven’t looked at testing frameworks/techniques yet. 28](#_Toc433036013)

[Finally, I was ready to begin the P0 assignment: 28](#_Toc433036014)

[Development to production path steps: 29](#_Toc433036015)

[Working with template repository: 29](#_Toc433036016)

[Setup the development environment 29](#_Toc433036017)

[1. Move to folder where template repository will be cloned. 29](#_Toc433036018)

[2. Install git in the folder (if not already there). See git init and setup information located elsewhere in this document. 29](#_Toc433036019)

[3. Clone the repository with: 29](#_Toc433036020)

[git clone remote https://git@github.com/alanwea/template . 29](#_Toc433036021)

[Note period at the end 29](#_Toc433036022)

[4. Change to the template folder and start developing. 29](#_Toc433036023)

[Update the GitHub repository 29](#_Toc433036024)

[1. In the local template development folder: 29](#_Toc433036025)

[2. git push 29](#_Toc433036026)

[Working with development repository created from template: 29](#_Toc433036027)

[1. Move to folder where development repository will be. 29](#_Toc433036028)

[2. Install git in the folder (if not already there). See git init and setup information located elsewhere in this document. 29](#_Toc433036029)

[3. Fork the repository with: 29](#_Toc433036030)

[git clone remote https://git@github.com/alanwea/template . 29](#_Toc433036031)

[Note period at the end 29](#_Toc433036032)

[4. On GitHub, create a repository for the new project and make it master and origin 29](#_Toc433036033)

[5. Change to the development folder and start developing. 29](#_Toc433036034)

[Publishing development repository: 29](#_Toc433036035)

[If 29](#_Toc433036036)

[Sample creation and publishing of FEWD P0 30](#_Toc433036037)

[CD E:\udacity\fewd 30](#_Toc433036038)

[FEWD\_P1 Notes 30](#_Toc433036039)

[1. Login to GitHub, create a new repository for FEWD\_P1 30](#_Toc433036040)

[a. Clone to local repository with “Git Shell” 30](#_Toc433036041)

[i. git clone git@github.com:alanwea/FEWD\_P1 30](#_Toc433036042)

[1. This uses SSH syntax, allowed because of a previously and separately created SSH key downloaded to the development machine from GitHub. 30](#_Toc433036043)

[2. Config user.name, user.email and editor not required, since they were set in FEWD\_P0. 30](#_Toc433036044)

[2. Copy template folders and code developed in the FEWD\_P0 project to FEWD\_P1 repository (it should be possible to pull from the template repository on GitHub to create the files in FEWD\_P1 and then break the push connection, but will leave that for later, right now just copying is easier). 30](#_Toc433036045)

[a. Will use Visual Studio 2015 Community edition to develop FEWD\_P1, so create a shortcut to Visual Studio in the FEWD\_P1 folder (for future convenience). 30](#_Toc433036046)

[3. Using text editor (I’m using Notepad++), open .gitignore and review patterns. 30](#_Toc433036047)

[a. The Visual Studio shortcut doesn’t need to be tracked so add it as a pattern. Also add “jquery” and “jquery-ui-1.11.4” since these are externally retrieved libraries that we shouldn’t changing anyway. 30](#_Toc433036048)

[4. Switch into FEWD\_P1: 30](#_Toc433036049)

[a. “Git status” to verify that .gitignore is working 30](#_Toc433036050)

[b. “Git add \*” to stage previously untracked items. 30](#_Toc433036051)

[c. “Git status” to make sure all is green. 30](#_Toc433036052)

[d. “Git commit” to commit to local repository 30](#_Toc433036053)

[e. “Git push” to the GitHub remote repository 30](#_Toc433036054)

[f. Check on Github to make sure the push was successful. 30](#_Toc433036055)

[1. Configure Visual Studio 2015 Community edition in preparation to develop FEWD\_P1. 30](#_Toc433036056)

[a. Under Tools menu, select “Extensions and Updates” and the expand “Updates” and “Product Updates” If there is an update to Visual Studio, apply it. 30](#_Toc433036057)

[b. Expand “Online” then “Visual Studio Gallery” then “Tools” then “Coding” and select “GitHub Extension for Visual Studio,” download and install it. Restart Visual Studio (VS). 30](#_Toc433036058)

[c. Select menu item “Team” and then “Manage Connections.” Look to right panel for connection and GitHub should be there. 30](#_Toc433036059)

[d. Under the GitHub connection, select “Local Git Repositories” and “Add” (since , in this case, the local FEWD\_P1 repository has already been created outside of VS). 31](#_Toc433036060)

[i. Navigate to the FEWD\_P1 repository and press return 31](#_Toc433036061)

[e. Create VS project for FEWD\_P1: 31](#_Toc433036062)

[i. File | Open | Web Site 31](#_Toc433036063)

[1. Navigate to FEWD\_P1 31](#_Toc433036064)

[ii. On right panel, switch to Solution Explorer view, the FEWD\_P1 files should all be shown there. 31](#_Toc433036065)

[f. Start developing 31](#_Toc433036066)

[i. Open index.html in editing window, press F5. 31](#_Toc433036067)

[1. Debugging Not Enabled might popup, tell it to add a new Web.config file. Add it to .gitignore. 31](#_Toc433036068)

[1. Start the FEWD\_P1 project: 31](#_Toc433036069)

[a. With index.html in the main editing window: 31](#_Toc433036070)

[i. Find and edit the TODO’s, as required 31](#_Toc433036071)

[1. Uncomment the development CSS style so we can see boxes. 31](#_Toc433036072)

[2. Press F5 and check the Web Page is blank. 31](#_Toc433036073)

[b. Boxify the PDF mockup – by hand or otherwise. 31](#_Toc433036074)

[c. Determine the characteristics of each visual item> 31](#_Toc433036075)

[i. Udacity logo and divider line are not identified as graphics by Adobe. But Select All shows that Udacity logo is boxed inside of orange circle box and divider line is also boxed 31](#_Toc433036076)

[ii. BOXed: For “Jane Doette” Adobe Acrobat (since we were provided with a PDF) Tools | Content Editing | Edit Text and Images, idefntifies the font as Gotham HTF with a font size of 42.91 in black. 31](#_Toc433036077)

[iii. Boxed: Front-End ninja is also Gotham HTF with font size of 15.26 in black 31](#_Toc433036078)

[iv. Boxed: main image of HTML with “meta name” displayed 31](#_Toc433036079)

[v. Boxed: Featured work is also Gotham HTF with font size of 26.71 31](#_Toc433036080)

[vi. Each boxed: Appify, sunflower and bokeh are all Gotham HTF with size 27.47 31](#_Toc433036081)

[vii. Each boxed: Links at bottom at all Gotham HTF at size 10.98 31](#_Toc433036082)

[d. 31](#_Toc433036083)

[e. Check color index versus what designer provided, using a color sensing tool (I’m using 31](#_Toc433036084)

[Fonts 31](#_Toc433036085)

[Adobe Acrobat says that Gotham is needed 31](#_Toc433036086)

[http://ufonts.com/download/gotham-thin/149718.html 31](#_Toc433036087)

[Flexbox 32](#_Toc433036088)

[Performance 32](#_Toc433036089)

[https://developers.google.com/web/updates/2013/10/Flexbox-layout-isn-t-slow?hl=en 32](#_Toc433036090)

[**When using flexbox, always author for the new stuff**: the IE10 tweener syntax and the new updated flexbox that’s in Chrome 21+, Safari 7+, Firefox 22+, Opera (& Opera Mobile) 12.1+, IE 11+, and Blackberry 10+. In many cases you can do a fallback to the legacy flexbox to pick up some older mobile browsers. 32](#_Toc433036091)

[What’s more important is optimizing what matters. Always use the timeline to identify your bottlenecks before spending time optimizing one sort of operation. 32](#_Toc433036092)

[https://jakearchibald.com/2013/progressive-enhancement-still-important/ 32](#_Toc433036093)

[Caution about using for dense layouts 32](#_Toc433036094)

[https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout/ 32](#_Toc433036095)

[growing problem on the web when it comes to content shifting around during loading. For large amounts of content, flexbox can cause this, whereas grid is less likely to, but more commonly content-shift is caused by JS modifying the DOM. 32](#_Toc433036096)

[Layout options 34](#_Toc433036097)

[Inline-block 34](#_Toc433036098)

# Development checklist

## Decide on design parameters:

Which libraries may be added? Which devices will be supported, etc

## Create a wireframe model

## Create a content priority list

Scroll behavior - <http://blog.chartbeat.com/2013/08/12/scroll-behavior-across-the-web/>

## Check a reference spreadsheet for components and frameworks that might be useful

<https://docs.google.com/spreadsheet/ccc?key=0AiN0QfBTPpOCdDFjWlM0eU1ra21XanZkekxGbjA2WWc#gid=0>

## Use <http://caniuse.com/#search=column-gap> to verify if chosen approach is valid across the target deployment platforms.

## Create placeholders for images during web page design.

<https://placehold.it/>

**How does it work? Just put your image size after our URL and you'll get a placeholder.**

**Like this:** [**http://placehold.it/350x150**](https://placehold.it/350x150)

**You can also use it in your code, like this:**

**<img src="http://placehold.it/350x150">**

## Add vendor prefixes using a generation tool.

### What to prefix:

What to prefix tool/website: enter the CSS feature and see whether is needs prefixing

<http://shouldiprefix.com/>

### Postcss/Autoprefixer

An interactive version is located at: <http://autoprefixer.github.io/>

Following is from Google developer documentation at <https://developers.google.com/web/tools/setup/workspace/setup-buildtools#dont-trip-up-with-vendor-prefixes>



## Fonts and common fallback fonts

### For FEWD\_P1

<http://ufonts.com/download/gotham-thin/149718.html>

### General info

<https://en.wikipedia.org/wiki/Web_typography>

### Core fonts for the web

<https://en.wikipedia.org/wiki/Core_fonts_for_the_Web>

### Compatible Typography

<https://www.onedesigns.com/tutorials/font-families-for-cross-compatible-typography>

### CSS Web Safe Font Combinations

<http://www.w3schools.com/cssref/css_websafe_fonts.asp>

### The @Font-Face rule and useful web font tricks

<http://www.smashingmagazine.com/2011/03/the-font-face-rule-revisited-and-useful-tricks/>

### How to use any font you like with CSS3

<http://www.webdesignerdepot.com/2013/01/how-to-use-any-font-you-like-with-css3/>

### Google fonts

<https://www.google.com/fonts>

# Git Setup

## PowerShell environment

### Change prompt

<http://stackoverflow.com/questions/18292063/what-does-the-prompt-master-1-0-0-mean-when-using-git-command-line>

### Run scheduled tasks

<http://blogs.technet.com/b/heyscriptingguy/archive/2011/01/12/use-scheduled-tasks-to-run-powershell-commands-on-windows.aspx>

## Setup and clone Git repository from github:

1. Login to Github
2. On home page click on “New repository” button
3. Enter a name for the repository
4. Check “Initialize this repository with a README”
5. Click on “Create repository” button
6. After repository is created, note the page that displays that shows how to create a client of the remote repository on the local computer.



1. If using “Quick setup” make sure to select the correct security protocol string, HTTPS or SSH.
2. Switch to the client computer
3. Start a git shell or command line shell that includes git path and environment variables.
4. Change/create folders to one folder above where the remote repo will be created.
5. Issue a clone command:
   1. For instance to clone a github hosted repository named UP0 under user Alanwea the following command will create a folder of the same name and then clone the remote repository to the local folder.

Git clone <https://github.com/alanwea/UP0.git>

1. Change folders into the root of the new local repository
2. Check that git was successfully installed:

Git --version

1. Initialize the new cloned repository by:
   1. Set a user name:

git config –global user.name “Alan Weatherhead”

* 1. Set user email:

git config –global user.email [alanwea@hotmail.com](mailto:alanwea@hotmail.com)

* 1. Set a default editor for typing in messages, like the –m message that is required on a commit. Note especially the use of “ and ‘ to enclose the path to notepad++ and the entire command with arguments for 32-bit Notepad++ on a 64-bit Windows environment.git <http://stackoverflow.com/questions/1634161/how-do-i-use-notepad-or-other-with-msysgit>)

git config --global core.editor "’C:/Program Files (x86)/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"

1. Set autocrlf to handle line endings correctly. For git on Windows:

Git config –global core.autocrlf true

1. Check that the config settings were correctly set:

Git config –list

1. Move, copy or create files in the local repository root folder, then add them to the git tracking list. In this case the entire contents of the files in the local git repository will be added:
3. If using SSH for authentication: Generate client side SSH key using ssh-keygen in the Git Shell. (Instructions adapted from <https://help.github.com/articles/generating-ssh-keys/> )

C:\Users\Alanwea\Documents\GitHub> ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/c/Users/Alanwea/.ssh/id\_rsa): C:\Users\Alanwea\Documents\GitHub>

C:\Users\Alanwea\Documents\GitHub> ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/c/Users/Alanwea/.ssh/id\_rsa):

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /c/Users/Alanwea/.ssh/id\_rsa.

Your public key has been saved in /c/Users/Alanwea/.ssh/id\_rsa.pub.

The key fingerprint is:

00:da:66:4b:67:d7:f5:3f:47:e5:1e:7e:20:e1:e2:41 Alanwea@ZOTZ

The key's randomart image is:

+--[ RSA 2048]----+

| . E o .|

| o . o o o ..|

| . = + . + o o.o|

| + + o . o ..=.|

| . S . o=|

| +|

| |

| |

| |

+-----------------+

C:\Users\Alanwea\Documents\GitHub>

1. Start the ssh agent:
2. Add SSH key to the ssh-agent:

C:\Users\Alanwea\Documents\GitHub> ssh-agent -s

SSH\_AUTH\_SOCK=/tmp/ssh-fmlHn5p5mzpx/agent.7544; export SSH\_AUTH\_SOCK;

SSH\_AGENT\_PID=8516; export SSH\_AGENT\_PID;

echo Agent pid 8516;

C:\Users\Alanwea\Documents\GitHub>

1. Original instructions use “clip < ~/.ssh/id\_rsa.pub” to copy the generated SSH public key to the clipboard. This doesn’t work under PowerShell. Instead I used Notepad++ to open the SSH public file and copied it to the clipboard.

C:\Users\Alanwea\Documents\GitHub> ssh-add ~/.ssh/id\_rsa

Identity added: /c/Users/Alanwea/.ssh/id\_rsa (/c/Users/Alanwea/.ssh/id\_rsa)

C:\Users\Alanwea\Documents\GitHub>

1. Setting up GitHub side of SSH and security
   1. Login to GitHub account
   2. Select “Profile” then “Settings” from upper right corner of home page.
   3. Select SSH Keys on left menu
   4. Click Add SSH key
   5. Enter title, i.e. computer name
   6. Paste key from clipboard into the Key field
   7. Click Add key
   8. Enter GitHub password

C:\Users\Alanwea\Documents\GitHub> ssh -T git@github.com

Warning: Permanently added 'github.com,192.30.252.131' (RSA) to the list of known hosts.

Hi alanwea! You've successfully authenticated, but GitHub does not provide shell access.

C:\Users\Alanwea\Documents\GitHub>

* 1. Test the connection

1. Show current remotes with “git remote –v”. If SSH will look something like this:

E:\udacity\frontendweb\up0 [master]> git remote -v

origin git@github.com:alanwea/UP0.git (fetch)

origin git@github.com:alanwea/UP0.git (push)

E:\udacity\frontendweb\up0 [master]>

1. If HTTPS will look like this:

E:\udacity\frontendweb\up0 [master]> git remote -v

origin https://github.com/alanwea/UP0.git (fetch)

origin https://github.com/alanwea/UP0.git (push)

E:\udacity\frontendweb\up0 [master]>

1. Push local repository to the remote:

E:\udacity\frontendweb\up0 [master]> git push --set-upstream origin master

Warning: Permanently added 'github.com,192.30.252.131' (RSA) to the list of known hosts.

Counting objects: 8, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (8/8), done.

Writing objects: 100% (8/8), 11.37 KiB | 0 bytes/s, done.

Total 8 (delta 0), reused 0 (delta 0)

To git@github.com:alanwea/UP0.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

E:\udacity\frontendweb\up0 [master]>

1. Get detailed info about a remote:
   1. Git manual says “git remote show” returns even more info, but that wasn’t the case when I did it, it returned the same info. Looks like there is an error in the manual.
2. Rename a remote – STUBBED
3. Removing a remote – STUBBED

E:\udacity\frontendweb\up0 [master]> git remote show origin

Warning: Permanently added 'github.com,192.30.252.130' (RSA) to the list of known hosts.

\* remote origin

Fetch URL: git@github.com:alanwea/UP0.git

Push URL: git@github.com:alanwea/UP0.git

HEAD branch: master

Remote branch:

master tracked

Local branch configured for 'git pull':

master merges with remote master

Local ref configured for 'git push':

master pushes to master (up to date)

E:\udacity\frontendweb\up0 [master]>

## Git Common tasks

## Resolving merge conflicts

<https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/>

### Resolving remote missing

“template” repository existed on Github.

“template” repository existed locally, but repository was deleted and then recreated with a git init.

Local template repository has changes that are not in the remote repository and the “git remote” returns nothing, “origin” does not exist.

git fetch <https://git@github.com:alanwea/template>

Works, but says there are no common commits, gives this at the end: “\* branch HEAD -> FETCH\_HEAD”

git push

result: “fatal: No configured push destination”



The remotes now exist.



This pulls the master branch from the remote repository into the local repository. Looks like the Jquery stuff was not in the local repository. Now the local repository is up to date, but the remote repository does not have the changes made in the local repository.



The local changes are pushed to the remote and everything should be good to go.

## Git Auto Commit

### Using MS tasks:

Run Task Scheduler from Control Panel/Administrative. Create a new task to run at whatever start time and interval. Create an action to run a script in the git shell of the repository to be auto committed. In example, the argument is to create a debugging log, this is optional in production. Start in is the folder at the root of the target repository (that is a folder with a .git folder).



Create a batch file in the root repository folder, in this case it is named Autogit.bat and contains:



Add Autogit.log to .gitignore file?

Add other commands as needed.

Might add one to change the interval of the job itself.

Might convert the .BAT to a full-fledged powershell script

Add parametrization to bat file

## Git Hooks

### Testing environment variables that Git passes to hooks

<https://www.digitalocean.com/community/tutorials/how-to-use-git-hooks-to-automate-development-and-deployment-tasks>

# Development Environment

## File and update structure



www.bvbites.com

Development

Production



Git clone <https://github.com/alanwea/UP0.git>

P0

E:\udacity\frontendweb

git push --set-upstream origin master

Microsoft Expression Web

C:\Users\alanwea\documents\my web sites\udacity

P0

HTML and CSS

## Instructor Notes

Very nice [article](http://css-tricks.com/how-css-selectors-work/) on CSS selectors.

[Documentation](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)

Browsers use default stylesheets to determine how to display HTML elements. You can view the default style rules for h1 and other elements for the following browsers:

* [WebKit (Chrome and Safari)](http://trac.webkit.org/browser/trunk/Source/WebCore/css/html.css)
* [Firefox](http://hg.mozilla.org/mozilla-central/file/tip/layout/style/html.css)
* [Internet Explorer](http://www.iecss.com/)

# Experimental Work

## CSS variables

### Why you need to use CSS variables

<http://www.creativebloq.com/netmag/why-you-need-use-css-variables-91412904>

### CSS custom properties for cascading variables

<https://drafts.csswg.org/css-variables/>

## CSS Calc()

<http://www.w3.org/TR/css3-values/>

### CSS3 Gems: the calc() function

<http://www.sitepoint.com/css3-calc-function/>

# Submission notes for FEWD/P0:

## Set up GitHub, including SSL certificate <https://help.github.com/articles/generating-ssh-keys/>

<http://stackoverflow.com/questions/13800289/configure-git-clients-like-github-for-windows-to-not-ask-for-authentication/18607931#18607931>

<http://stackoverflow.com/questions/2916845/different-default-remote-tracking-branch-for-git-pull-and-git-push>

<http://corlewsolutions.com/articles/article-11-guide-to-setting-up-ssh-on-windows-7>

Installed Git in the folder I’m using for P0 (I’m using command line Git for now, although I know that TortoiseGit, etc. exist). Configured user name, email and default editor (Notepadd++). Setup .gitignore to exclude files that don’t need to be tracked. Added existing P0 files to repository. Committed them and then pushed to GitHub using SSL.

I’m hosting on my own website using Microsoft Expression Web (for historical reasons), so I added Git to the folder holding the website production files. I haven’t done a complete end to end test yet, but the idea is to do development on my local machine and pull/Push to GitHub. When my website needs to be updated, I’ll create a release branch on GitHub and pull changes from GitHub to my production website folders, then publish to a test area on my website and when satisfied, publish to the live site. Any test area code changes I’ll push/merge back to the GitHub release and main branches.

Completed the Udacity HTML/CSS classes up to the first couple of minutes of Bootstrap, I know we will be looking at that later, so stopped there.

Finally found a picture for the P0 assignment that I’m happy with (had to ask someone to take it).

Looked online for suggested folder structures for development. Lots of controversy as you might expect. For P0, I’m going to start using a hybrid structure based on the many suggestions that I found to get in the habit of starting this way, even though I know that P0 doesn’t require it.

I’ve had some past experience with JavaScript, but it’s been a while so I reviewed articles on:

Lazy loading JavaScript (aka deferred loading) – not needed right now, but will be eventually for fast load of rich content pages.

Internationalization of Web pages ( <http://www.smashingmagazine.com/2014/06/css-driven-internationalization-in-javascript/> , <http://www.w3.org/TR/its20/> ) – also not needed, but a consideration later if website will be seen by non-English speakers.

Ease of use – that is how to present to web site viewers with disabilities? Keystroke handling, text to speech, voice response.

JavaScript event handling – this is going to be needed eventually too, but I was also interested in how handling events only for navigable items on pages, could be combined with lazy loading to speed up web page loading. Secondary, was looking at events generated by touch screens, I’ve only every handled click events before. Also, if it made sense to funnel all page events through a table driven event handler, maybe as a state machine for responses (?). (<http://www.smashingmagazine.com/2013/11/an-introduction-to-dom-events/> )

Looked at concepts of responsive and adaptive design – responsive was briefly covered in the HTML/CSS classes.

Looked at what alternatives there are to Javascript ( I know, this isn’t needed for P0 either). I have some experience with VBScript, but not the other alternatives: ActionScript, Dart, Typescript and Python (though have it installed).

( <https://en.wikipedia.org/wiki/Client-side_scripting#List_of_Client-Side_Scripting_languages> )

Install JQuery

<https://jquery.com/download/>

A more in-depth look at CSS selectors (<https://css-tricks.com/how-css-selectors-work/> )

Haven’t looked at testing frameworks/techniques yet.

Finally, I was ready to begin the P0 assignment:

# Development to production path steps:

## Working with template repository:

### Setup the development environment

1. Move to folder where template repository will be cloned.
2. Install git in the folder (if not already there). See git init and setup information located elsewhere in this document.
3. Clone the repository with:

git clone remote <https://git@github.com/alanwea/template> .

Note period at the end

1. Change to the template folder and start developing.

### Update the GitHub repository

1. In the local template development folder:
2. git push

## Working with development repository created from template:

1. Move to folder where development repository will be.
2. Install git in the folder (if not already there). See git init and setup information located elsewhere in this document.
3. Fork the repository with:

git clone remote <https://git@github.com/alanwea/template> .

Note period at the end

1. On GitHub, create a repository for the new project and make it master and origin
2. Change to the development folder and start developing.

## Publishing development repository:

If

# Sample creation and publishing of FEWD P0

CD E:\udacity\fewd

# FEWD\_P1 Notes

1. Login to GitHub, create a new repository for FEWD\_P1
   1. Clone to local repository with “Git Shell”
      1. git clone [git@github.com:alanwea/FEWD\_P1](mailto:git@github.com:alanwea/FEWD_P1)
         1. This uses SSH syntax, allowed because of a previously and separately created SSH key downloaded to the development machine from GitHub.
         2. Config user.name, user.email and editor not required, since they were set in FEWD\_P0.
2. Copy template folders and code developed in the FEWD\_P0 project to FEWD\_P1 repository (it should be possible to pull from the template repository on GitHub to create the files in FEWD\_P1 and then break the push connection, but will leave that for later, right now just copying is easier).
   1. Will use Visual Studio 2015 Community edition to develop FEWD\_P1, so create a shortcut to Visual Studio in the FEWD\_P1 folder (for future convenience).
3. Using text editor (I’m using Notepad++), open .gitignore and review patterns.
   1. The Visual Studio shortcut doesn’t need to be tracked so add it as a pattern. Also add “jquery” and “jquery-ui-1.11.4” since these are externally retrieved libraries that we shouldn’t changing anyway.
4. Switch into FEWD\_P1:
   1. “Git status” to verify that .gitignore is working
   2. “Git add \*” to stage previously untracked items.
   3. “Git status” to make sure all is green.
   4. “Git commit” to commit to local repository
   5. “Git push” to the GitHub remote repository
   6. Check on Github to make sure the push was successful.
5. Configure Visual Studio 2015 Community edition in preparation to develop FEWD\_P1.
   1. Under Tools menu, select “Extensions and Updates” and the expand “Updates” and “Product Updates” If there is an update to Visual Studio, apply it.
   2. Expand “Online” then “Visual Studio Gallery” then “Tools” then “Coding” and select “GitHub Extension for Visual Studio,” download and install it. Restart Visual Studio (VS).
   3. Select menu item “Team” and then “Manage Connections.” Look to right panel for connection and GitHub should be there.
   4. Under the GitHub connection, select “Local Git Repositories” and “Add” (since , in this case, the local FEWD\_P1 repository has already been created outside of VS).
      1. Navigate to the FEWD\_P1 repository and press return
   5. Create VS project for FEWD\_P1:
      1. File | Open | Web Site
         1. Navigate to FEWD\_P1
      2. On right panel, switch to Solution Explorer view, the FEWD\_P1 files should all be shown there.
   6. Start developing
      1. Open index.html in editing window, press F5.
         1. Debugging Not Enabled might popup, tell it to add a new Web.config file. Add it to .gitignore.
6. Start the FEWD\_P1 project:
   1. With index.html in the main editing window:
      1. Find and edit the TODO’s, as required
         1. Uncomment the development CSS style so we can see boxes.
         2. Press F5 and check the Web Page is blank.
   2. Boxify the PDF mockup – by hand or otherwise.
   3. Determine the characteristics of each visual item>
      1. Udacity logo and divider line are not identified as graphics by Adobe. But Select All shows that Udacity logo is boxed inside of orange circle box and divider line is also boxed
      2. BOXed: For “Jane Doette” Adobe Acrobat (since we were provided with a PDF) Tools | Content Editing | Edit Text and Images, idefntifies the font as Gotham HTF with a font size of 42.91 in black.
      3. Boxed: Front-End ninja is also Gotham HTF with font size of 15.26 in black
      4. Boxed: main image of HTML with “meta name” displayed
      5. Boxed: Featured work is also Gotham HTF with font size of 26.71
      6. Each boxed: Appify, sunflower and bokeh are all Gotham HTF with size 27.47
      7. Each boxed: Links at bottom at all Gotham HTF at size 10.98
   5. Check color index versus what designer provided, using a color sensing tool (I’m using

## Fonts

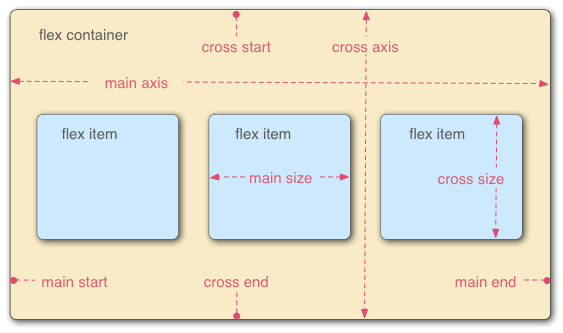
Adobe Acrobat says that Gotham is needed

http://ufonts.com/download/gotham-thin/149718.html

Flexbox

## Usage

<https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Using_CSS_flexible_boxes>



## Performance

<https://developers.google.com/web/updates/2013/10/Flexbox-layout-isn-t-slow?hl=en>

The new flexbox code has a lot fewer multi-pass layout codepaths. You can still hit multi-pass codepaths pretty easily though (e.g. flex-align: stretch is often 2-pass). In general, it should be much faster in the common case, but you can construct a case where it’s equally as slow.

That said, if you can get away with it, regular block layout (non-float), will usually be as fast or faster than new flexbox since it’s always single-pass. But new flexbox should be faster than using tables or writing custom JS-base layout code.

**When using flexbox, always author for the new stuff**: the IE10 tweener syntax and the new updated flexbox that’s in Chrome 21+, Safari 7+, Firefox 22+, Opera (& Opera Mobile) 12.1+, IE 11+, and Blackberry 10+. In many cases you can [do a fallback to the legacy flexbox](http://css-tricks.com/using-flexbox/) to pick up some older mobile browsers.

What’s more important is optimizing what matters. Always use the timeline to identify your bottlenecks before spending time optimizing one sort of operation.

<https://jakearchibald.com/2013/progressive-enhancement-still-important/>

Here's how most pages load:

1. HTML downloads
2. CSS downloads
3. CSS fetches additional assets
4. JS downloads
5. JS executes
6. JS fetches additional assets
7. JS updates DOM

## Caution about using for dense layouts

<https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout/>

growing problem on the web when it comes to content shifting around during loading. For large amounts of content, flexbox can cause this, whereas grid is less likely to, but more commonly content-shift is caused by JS modifying the DOM.

# Layout options

## Inline-block

<https://jakearchibald.com/2014/dont-use-flexbox-for-page-layout/>

See discussion thread at bottom

https://developers.google.com/web/fundamentals/performance/rendering/avoid-large-complex-layouts-and-layout-thrashing

Layout is where the browser figures out the geometric information for elements: their size and location in the page. Each element will have explicit or implicit sizing information based on the CSS that was used, the contents of the element, or a parent element. The process is called Layout in Chrome, Opera, Safari, and Internet Explorer. In Firefox it’s called Reflow, but effectively the process is the same

## Avoid layout wherever possible

When you change styles the browser checks to see if any of the changes require layout to be calculated, and for that render tree to be updated. Changes to “geometric properties”, such as widths, heights, left, or top all require layout.

**Layout is almost always scoped to the entire document**

<http://csstriggers.com/>

# CSS Triggers... A game of layout, paint, and composite.

## Use flexbox over older layout models

## Avoid forced synchronous layouts

## Avoid layout thrashing

# Debugging

## Chrome performance walkthrough

<https://jakearchibald.com/2013/solving-rendering-perf-puzzles/>

I opened up Chrome Devtools and made a [timeline recording](https://developers.google.com/chrome-developer-tools/docs/timeline)

Chrome has a lower-level timeline tool hidden in about:tracing. Let's use that to identify the cause.