Contents

[Development checklist 4](#_Toc433379209)

[Create placeholders for images during web page design. 4](#_Toc433379210)

[https://placehold.it/ 4](#_Toc433379211)

[Add vendor prefixes using a generation tool. 4](#_Toc433379212)

[What to prefix: 4](#_Toc433379213)

[Postcss/Autoprefixer 4](#_Toc433379214)

[Fonts and common fallback fonts 5](#_Toc433379215)

[For FEWD\_P1 5](#_Toc433379216)

[General info 5](#_Toc433379217)

[Core fonts for the web 5](#_Toc433379218)

[Compatible Typography 5](#_Toc433379219)

[CSS Web Safe Font Combinations 5](#_Toc433379220)

[The @Font-Face rule and useful web font tricks 5](#_Toc433379221)

[How to use any font you like with CSS3 5](#_Toc433379222)

[Google fonts 5](#_Toc433379223)

[Typedia – encyclopedia of typefaces 5](#_Toc433379224)

[Five main groupings of fonts 5](#_Toc433379225)

[Humanist Sans: 5](#_Toc433379226)

[Old Style: aka “Venetian.” 6](#_Toc433379227)

[Transitional and Modern: 6](#_Toc433379228)

[Slab Serifs (aka “Egyptian”) 6](#_Toc433379229)

[The Principle of Decisive Contrast 6](#_Toc433379230)

[Best practices for combining typefaces: 6](#_Toc433379231)

[There are currently six web font formats, each with their own advantages and disadvantages. •TTF (TrueType) – supported on older IOS and Android 7](#_Toc433379232)

[Web Font Loader 9](#_Toc433379233)

[How to construct a good font stack 9](#_Toc433379234)

[Font Compression 9](#_Toc433379235)

[Compress data in file 9](#_Toc433379236)

[Compress font while being transmitted 9](#_Toc433379237)

[Inlineing fonts considered bad 10](#_Toc433379238)

[How to find out what fonts are on a given platform. 10](#_Toc433379239)

[Git Setup 11](#_Toc433379240)

[PowerShell environment 11](#_Toc433379241)

[Change prompt 11](#_Toc433379242)

[Run scheduled tasks 11](#_Toc433379243)

[Setup and clone Git repository from github: 11](#_Toc433379244)

[Git Common tasks 18](#_Toc433379245)

[Resolving merge conflicts 18](#_Toc433379246)

[Resolving remote missing 18](#_Toc433379247)

[Git Auto Commit 20](#_Toc433379248)

[Using MS tasks: 20](#_Toc433379249)

[Git Hooks 21](#_Toc433379250)

[Testing environment variables that Git passes to hooks 21](#_Toc433379251)

[Development Environment 22](#_Toc433379252)

[File and update structure 22](#_Toc433379253)

[Instructor Notes 23](#_Toc433379254)

[Experimental Work 23](#_Toc433379255)

[CSS variables 23](#_Toc433379256)

[Why you need to use CSS variables 23](#_Toc433379257)

[CSS custom properties for cascading variables 23](#_Toc433379258)

[CSS Calc() 23](#_Toc433379259)

[CSS3 Gems: the calc() function 23](#_Toc433379260)

[Submission notes for FEWD/P0: 24](#_Toc433379261)

[Set up GitHub, including SSL certificate https://help.github.com/articles/generating-ssh-keys/ 24](#_Toc433379262)

[Development to production path steps: 27](#_Toc433379263)

[Working with template repository: 27](#_Toc433379264)

[Setup the development environment 27](#_Toc433379265)

[Update the GitHub repository 27](#_Toc433379266)

[Working with development repository created from template: 27](#_Toc433379267)

[Publishing development repository: 27](#_Toc433379268)

[Sample creation and publishing of FEWD P0 28](#_Toc433379269)

[FEWD\_P1 Notes 28](#_Toc433379270)

[Fonts 30](#_Toc433379271)

[Performance 31](#_Toc433379272)

[SpeedIndex 31](#_Toc433379273)

[PageSpeed Insights 31](#_Toc433379274)

# Development checklist

## 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>

## Typedia – encyclopedia of typefaces

<http://typedia.com/learn/only/typeface-classifications/>

## Five main groupings of fonts

(from What Font Should I Use?”: Five Principles for Choosing and Using Typefaces, Smashing Magazine)

Geometric:

combination of Geometric, Realist and Grotesk. Based on strict geometric forms. Designed to be as simple as possible. Helvetica, Univers, Futura, Avant Garde, Akzidenz Grotesk, Franklin Gothic, Gotham.

### Humanist Sans:

sans = derived from handwriting. Less detail, less consistency, thicker and thinner stroke weights. Gill Sans, Frutiger, Myriad, Optima, Verdana

### Old Style: aka “Venetian.”

Oldest typefaces refined over centuries from calligraphic forms. Little contrast between thick and thin, curved letters tilt to left. Jenson, Bembo, Palatino, and — especially — Garamond,

### Transitional and Modern:

geometric, sharp, virtuosic. Contrasting thick and thin strokes. *Examples of transitional typefaces:* Times New Roman, Baskerville.

*Examples of Modern serifs:* Bodoni, Didot.

### Slab Serifs (aka “Egyptian”)

Strokes like sans faces. Solid, rectangular shoes stuck on the end. Specific, but contradictory associations. Clarendon, Rockwell, Courier, Lubalin Graph, Archer

## The Principle of Decisive Contrast

When adding a second typeface: keep it exactly the same or change it a lot.

Avoid two different faces from within one of the five categories.

Two typefaces work well together if they have one thing in common but are otherwise greatly different. Common aspect can be from same period of time, visual, same designer, etc.

Display typefaces:

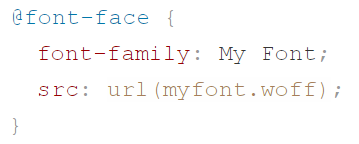
Use in small doses.

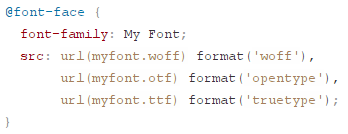
lots of personality; comic sans to candy-cane and bunny, …

## Best practices for combining typefaces:

<http://www.smashingmagazine.com/2010/11/best-practices-of-combining-typefaces/>

* Combine a Sans Serif with a Serif
* Avoid Similar Classifications
* Assign Distinct Roles
* Contrast Font Weights
* Create a Variety of Typographic Colors
* Don’t Mix Moods
* Contrast Distinct with Neutral
* Avoid Combinations That are Too Disparate
* Keep It Simple — Try Just Two Typefaces
* Use Different Point Sizes

Web fonts performance (from Smashing book 5: web fonts performance);

The src property will accept multiple comma-separated URLs to font files. Each URL can optionally be followed by a font format hint that browsers use to select the format they support (or prefer).

This @font-face rule uses three different font formats. Selected by browsers based on support and the order in which they occur.

Pick the formats that are most suitable for the browsers you support and give you the best compression rates.

## There are currently six web font formats, each with their own advantages and disadvantages. •TTF (TrueType) – supported on older IOS and Android

•OTF (OpenType) – supported on older iOS and Android

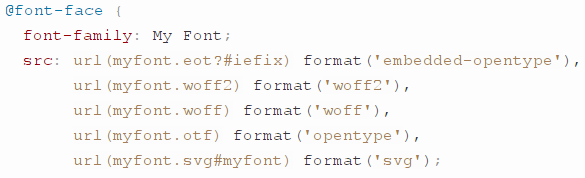
•EOT (Embedded OpenType) – IE 6,7,8 must use this

•WOFF (Web Open Font Format) – wider support, no support on older Android and iOS.

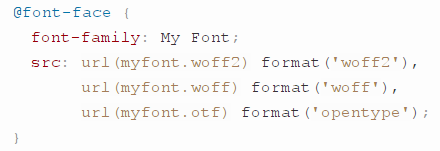
•WOFF2 (Web Open Font Format 2) – best compression rates, only supported in most recent browsers.

•SVG fonts (Scalable Vector Graphics) – do not use

SVG fonts do not support any of the OpenType features required to properly display text, such as kerning, ligatures and advanced positioning. Browser support for SVG fonts is limited; it’s been removed from Chrome, and have never been supported on IE or Firefox. The combined lack of OpenType features and limited browser support means there really is no reason to use SVG fonts either now or in the future3.



Bulletproof @font-face, to trick older browsers into ignoring rest of src after “eot” font, which is all that they support.

Covers IE9 and all modern browsers, both desktop and mobile.

Index to supported web types

Up-to-date data on support for type and typographic features on the web.   
Search or choose from the features below to get started.

!!! Browser’s font-matching algorithm will return fonts that match only on family name but not on weight, style, or variant. Consider this scenario: the browser has found a match for family name but not the required variation (bold, bold italic, etc.). Instead of rendering the text in a fallback font the browser will use the variation it already has and attempt to generate the required variation. This is the cause of so-called faux bold, faux italic and faux small caps, also officially known as “font synthesis” Avoid faux synthesis by always including web fonts for all styles on your site. If site uses thin, thin italic, regular, for instance, must have separate @font-face rules for each. Faux styles can occur for local fonts, but rarely: most installed versions include bold and italic variations by default.

See “Setting Weights And Styles With The @font-face Declaration6” <http://www.smashingmagazine.com/2013/02/setting-weights-and-styles-at-font-face-declaration/>

The State of Web Type <http://www.stateofwebtype.com/>

Browser steps in finding a font:

* If font family matches @font-face rule, but font is not already cached, then download font and apply it.
* If font family matches @font-face rule, but font is already cached, then immediately apply the font.
* If font family matches a local font, then apply it immediately.

Drawbacks: if CSS web font selector does not match a DOM element, then font not downloaded. Browsers only know what characters are in font, after they have been downloaded. Can use “unicode-range” property in @font-face rules to avoid this, but not supported by all browsers.

Unicode-range currently has some support limitations. Tells browsers what characters are in a font.

Performance:

Some browsers show placeholder font while downloading actual font, called “flash of unstyled test” (FOUT), but most browsers just hide the text until font is downloaded, then display it. Called “flash of invisible text” (FOIT). Especially visible when web and local fonts mixed, local will render first, then web later.

For FOIT, Mozilla/Google research says average 3 second download for web font, so recent chrome, opera and firefox have 3 second timeout before failing over to next font.

NOT YET IMPLEMENTED: “font-rendering” property to standardize behavior of FOUT and FOIT.

## Web Font Loader

Loader is a collaboration between Typekit and Google, and it gives you the ability to load custom fonts, as well as fonts from several web font services. It also provides font load events so you can use custom styles or scripts when a font loads (or fails to load). Web Font Loader gives you added control when using linked fonts via [@font-face](https://github.com/font-face). <https://github.com/typekit/webfontloader>

## How to construct a good font stack

a font with a similar x-height (the height of the lowercase characters) and em-width (the width of characters)

giving font stacks for body text and headlines a different structure.

“Ideal” = desired font, rest are fallback. “Fit” = close in style and metrics to the “ideal” font. “Common” = common on most platforms and still roughly matches design. “Generic” = one of the common font families, ie. Serif, san-serif, monospace,…

For headline fonts, favor style over metrics.

Always test stacks with web font disabled to test fallback and user experience

Consider variations in weight, style, etc.

Theoretical way to avoid visual discontinuity is to use metric-compatible web and local fonts. Helvetica and Arial are famous example, but metric-compatible fonts are rare so of limited use.

Simpliest way to improve cacheing is to use HTTP response headers with long cache expiration time. But, cache update (cache-invalidation) a problem: see <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/>

Good vale for HTTP Cache-Control header is set max-age value for CSS and font files is “604800” for (60 seconds x 60 minutes x 24 hours x 7 days)

*BETA Versions Chrome and Opera ONLY at time of writing: HTTP extension “stale-while-revalidate” browser uses cached resource even if stale, async retrieval of latest version to update the cache. Good value is 1209600 (60 seconds x 60 minutes x 24 hours x 14 days) keeps resources up to date while preventing redownloading content already cached.*

## Font Performance

Remove glyphs for languages and scripts that won’t be used. Called static subsets. Browsers treat subsets as two separate fonts- usually better to load minimal subset, then larger subset then two or more complementary subsets. Could be licensing restrictions on this. See Web Font Generator for a tool to do this: <http://www.fontsquirrel.com/tools/webfont-generator>

## Font Compression

### Compress data in file

### Compress font while being transmitted

WOFF2 and WOFF are already compressed. Opentype and truetype are not so should be compressed on web server. EOT supports compression but many tools won’t do it so compress it on the server.

Use Googles Zopfli compression algorithm, generates gzip. 2%-6% additional compression, even on WOFF.

## Inlineing fonts considered bad

See Why inlining everything is not the answer

<http://luiscameroon.blogspot.com/2011/12/why-inlining-everything-is-not-answer.html>

Also in the Performance section of this doc.

As a general rule, you should not inline fonts in CSS files unless you have a very specific use case, such as supporting only a single web font format or browser, storing fonts in localStorage, or low-bandwidth and high-latency connections.

## How to find out what fonts are on a given platform.

Compatibility tables for default local fonts, by OS. <http://fontfamily.io/>

Tinytype A compatibility table showing the available default system fonts across different mobile platforms <https://github.com/jordanmoore/tinytype>

## Better CSS Font Stacks article

<http://artequalswork.com/posts/better-css-font-stacks/>

## Dynamic subsetting, dynamic augmentation, see <http://artequalswork.com/posts/adventures-in-dynamic-subsetting/>

## Type Rendering Mix – adds classes based on browsers text rendering engine and antialiasing method.

## IE Swapping Method

## If your primary goal is to have your website’s content visible as soon as possible — regardless of whether it is using web fonts or a fallback font — you might consider asynchronously loading fonts and explicitly managing the cache.

## A much better approach is to load fonts asynchronously, and as soon as possible by using the browser’s cache. This is easily achieved using the native Font Load API (or the polyfill29).

## Sites that load lots of fonts

Flash faux text during download

“flash of faux text” article

<https://css-tricks.com/fout-foit-foft/>

<https://www.designernews.co/stories/42915-flash-of-faux-text--still-more-on-font-loading>

<http://www.zachleat.com/web/foft/>

Load complementary subsets of a single font

## Use Media queries – set default web font then use media queries to apply different font stack when the viewport matches desired dimensions. Used to apply a condensed version of typeface on smaller viewports, for instance.

Media queries let you selectively use custom font stacks. They should not, however, be used to enable or disable web fonts based on device pixel ratios or viewport dimensions. With ever increasing resolutions on modern handheld devices, your media queries might get executed even on devices that are often on unreliable and slow cellular connections. Likewise, mobile devices with small viewports may be on high-speed connections (such as Wi-Fi). Screen sizes or device pixel ratios don’t say much about whether or not the device’s internet connection is fast (which is the deciding factor for loading fonts or not). Use media queries as a way to load fonts based on your design and not simply to reduce page load.

# 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](https://en.wikipedia.org/wiki/Client-side_scripting" \l "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
   4. 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

# Performance

From <http://www.smashingmagazine.com/2015/08/understanding-critical-css/>

## **SpeedIndex**

**SpeedIndex measures the visual progress of the visible page loading** and computes an overall score for how quickly the content was painted. Try to [compare](http://www.webpagetest.org/video/) your SpeedIndex measurement before and after you have made changes by inlining your critical CSS

<https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics/speed-index>

## PageSpeed Insights

Make your web pages fast on all devices

<https://developers.google.com/speed/pagespeed/insights/>

If you use a content delivery network (CDN), it is also worth remembering that you should still **serve your non-critical CSS from the CDN**

**A counter statrement to putting css in the <head>**

<https://medium.com/@drublic/a-counter-statement-putting-the-css-in-the-head-f98103d09ce1#.9n66ka5dg>

Why inlining everything is not the answer:

<http://luiscameroon.blogspot.com/2011/12/why-inlining-everything-is-not-answer.html>