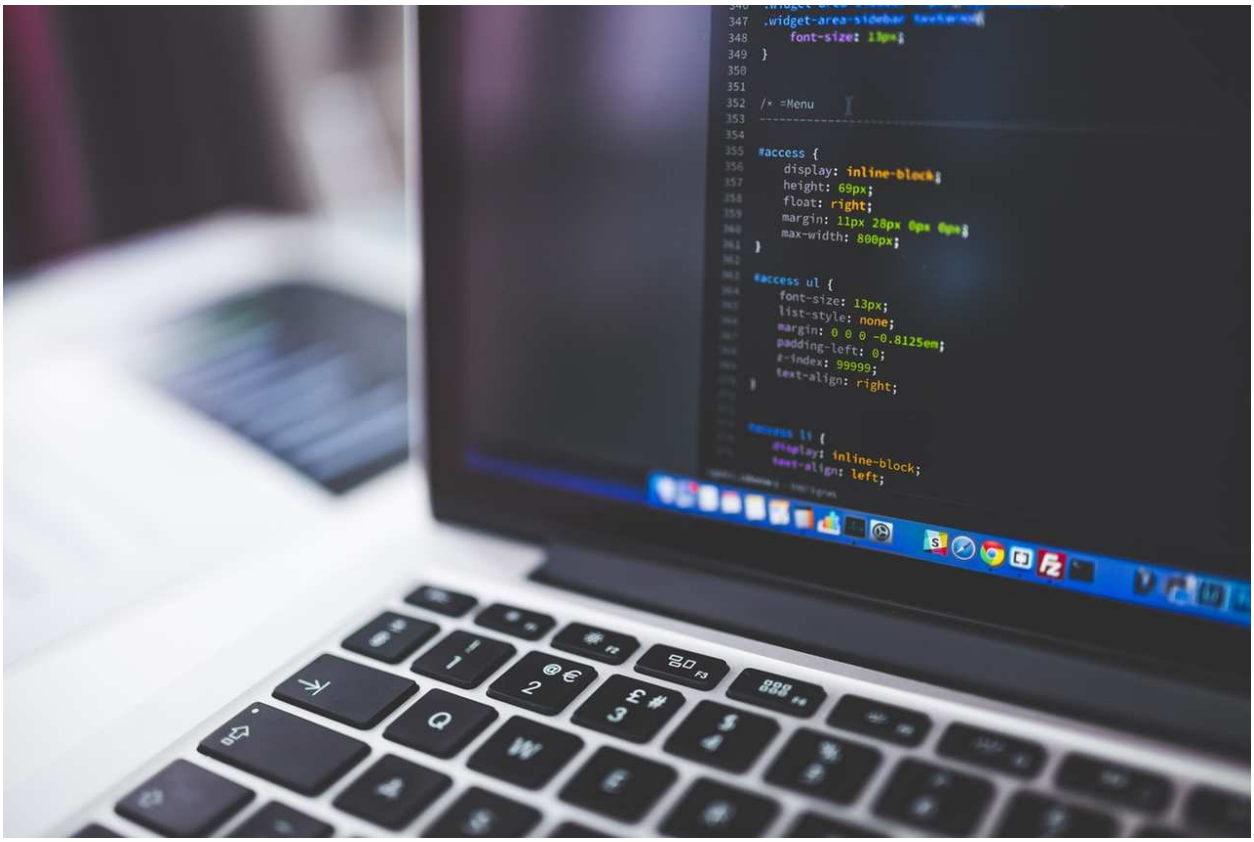
UX Developer Start Guide



Contents

Introduction

This is a getting started guide that will give you a brief overview of the corporate processes that govern how UX Development is organized and managed.

If you have any questions not covered here please email your supervisor.

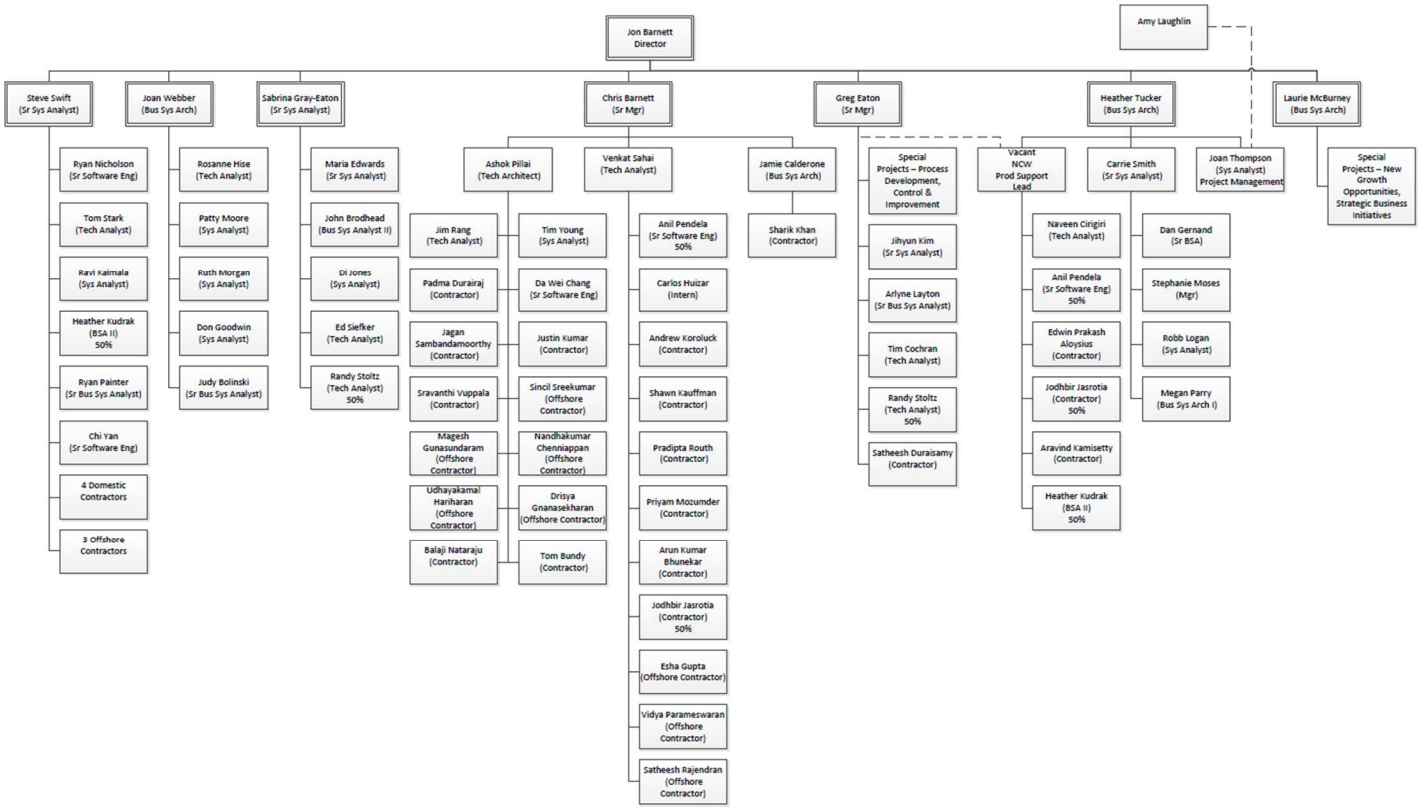
Internal Terminology

Below is a list of terms you'll come across while working here.

- NCW – Navient Customer Website – The website you will be mainly working on.
- Quality Center or QC – Internal ticketing system.
- Defect – A Quality Center ticket detailing a change that needs to be made.
- Compliance – Follow the rules laid out by business / department of education.
- BSL – Business Service Layer
- GBA - Global Business Analyst
- Prototype – A PowerPoint showing look and feel of a page or section.
- Use Case – A Word Document that details the content for a page or section.
- Migration – A process of combining all packages inside SCM and moving them to the test server.
- TSD – Technical Systems Design
- LRPD – Limited Route Project Deliverable
- MRPD - Micro Route Project Deliverable – Document for making a small change in production.

Organization Chart

This is an overview of the structure of the company. You can find the updated list on Navigator.



Security

Customer and Business security informs all decision-making. This is why small changes require approval, testing, and detailed documentation. When in doubt reach out and ask your supervisor. Volitions of security can result in termination.

Decision Chain

Higher Executives and other departments decide, based on business needs, how a web page or project will look and behave. This is called “Business-Driven” development. UX Developers are at the end of this decision chain.

Because of this intensive process some projects will be slightly behind schedule by the time they are handed-off to the UX team.

While working on projects UX Developers can offer suggestions to their direct supervisor on best practices and ways to solve specific page problems. Anything more complex must be ran up the chain, discussed, and approved before it can be implemented. Depending on the complexity of a suggestion, it can take a long time before a decision is reached.

Keep in mind that project timelines and budgets are already in place, which the UX Developers may or may not be aware of.

Scope

UX Developers have a narrow scope of service. UX Developers fix “look and feel” errors in pages for upcoming releases, as well as fixing “look and feel” defects on test and production servers.

The look and feel is always provided through Prototypes. Usually these are page mockups create in PowerPoint. These are not 100% accurate as PowerPoint is not a good tool for Mockups. Research is currently being done to test cross-platform tools that non-developers can use to mockup pages. The majority of mockups come from Business Analysts. UX must sometimes consult the BA’s who created the Mockups to determine if a change is accurate and/or to clarify the change request.

UX sometimes updates static text (content) on pages. The Use Case documents should be cross-checked with the person who wrote it to verify if the content presented is correct.

In the future UX Developers will create static templates for pages. These will be sent to the offshore team to add the dynamic content and connect the templates to the backend code.

Tools

UX Developers will use an array of tools to complete assigned tasks and track their work.

Here’s a link to the UX Developer Style Guide. This will take you directly to the resource page where you’ll find links to all the tools you’ll work with on a daily basis.

Below is a list of the tools you’ll use:

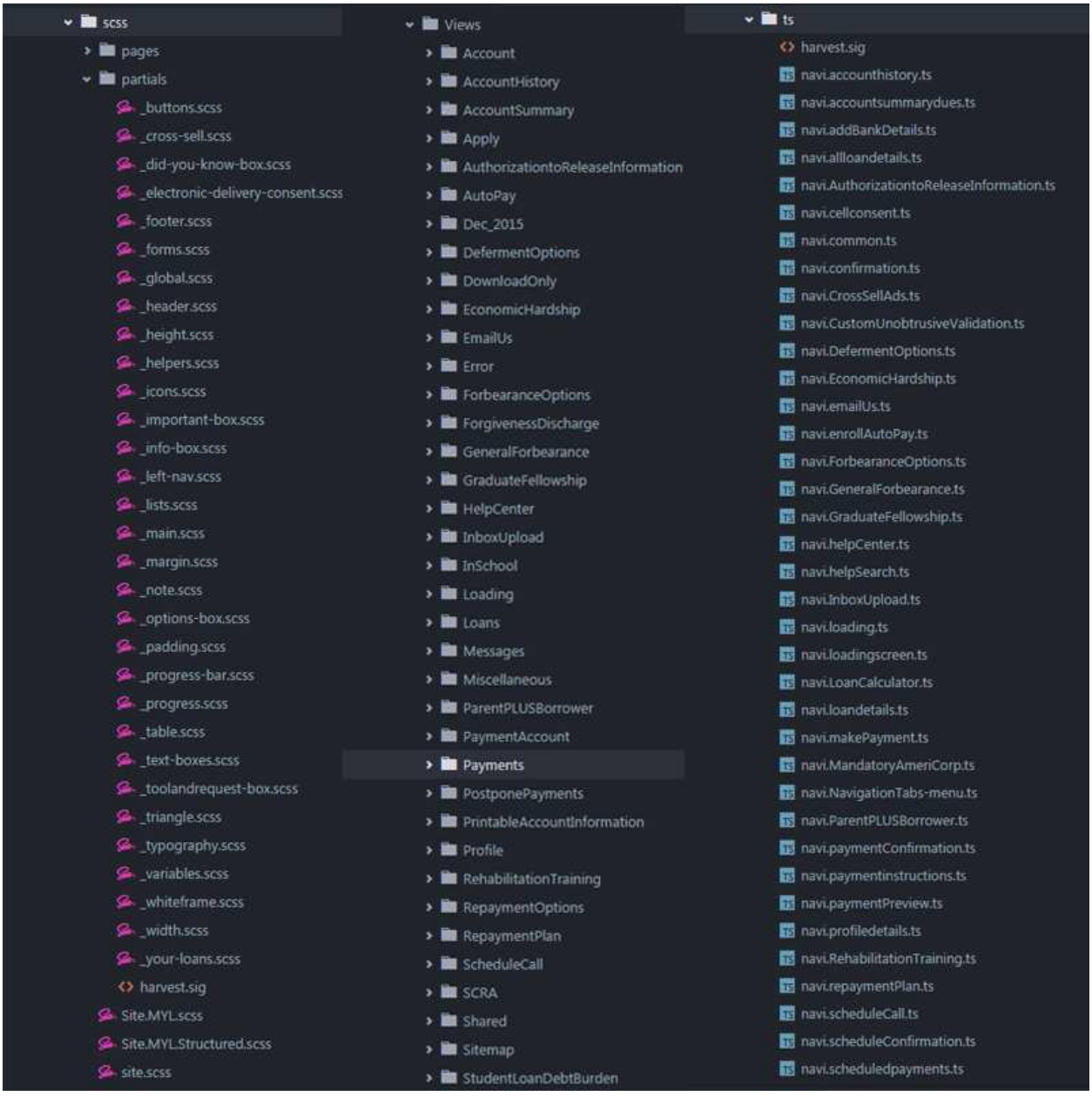
- Outlook / Office 365
- Word
- PowerPoint
- Skype
- SCM
- QC
- Visual Studio
- Trello
- Atom (or another code editor)
- Browsers: Chrome, Firefox, IE 11, and Safari*.
- Cmdr to run Node and Gulp

*Note: Currently we do not have any corporate approved Apple products in the office available for testing.

Codebase

UX Developers mainly work within three folders, “SCSS”, “Views”, and rarely “TS”.

- SCSS holds the style sheets which are trans-piled to native CSS using Node and Gulp.
- Views is a collection of folders which hold ASP.Net web pages which use the razor syntax for dynamically including data and/or other page fragments. These pages sometimes contain C# code which runs the dynamic logic on the page.
- TS is for TypeScript which is Microsoft’s attempt to turn JavaScript into a type safe language like C#. Using the latest revision of JavaScript as the target output TypeScript is very much a pre-processor, like SCSS, for JavaScript. Again we use Node and Gulp to trans-pile TS to JS.



Workflow

UX Developers should understand the concept of DRY, “Don’t Repeat Yourself”. Here are some additional tips and tricks to help you be as efficient and fluid as possible within this environment.

Unless told otherwise your typical workflow will follow this pattern:

1. Sync and get latest code from SCM.
2. Check in Trello and/or QC for the current defects to work on.
3. Use Visual Studio, Atom, or the code editor of your choice to make the fix.
4. Screenshots of the fixed element.
5. Update the Trello / QC defect with screenshot, path, and file name. As well as a brief description of what you did to fix the issue.
6. Sync SCM to make sure you are not over-writing anyone else’s code. Then (if you already have a package) commit your changes to SCM.
7. In QC set your changes to either “Fixed” or “Pending Migration” depending on the situation.
8. If it was a production issue you will have to complete either a TSD or an MRPD. Your supervisor will let you know which one and what is required.

Tips:

When you first get the codebase from SCM you’ll need to use windows terminal, Git Bash, or Cmdr to navigate to the solution folder. Then run “npm install”. It will install all the required node modules to run the solution as well as prepare the gulp file for execution.

When you load a solution into Visual Studio. First you’ll want to clean the solution and then build it.

If you get build errors, there are a few things you can do to fix it.

First, check the Visual Studio error log to assess if it is an error due to missing code. You’ll need to check SCM for any files you might have missed.

If you get any other errors, it usually from NPM or Read-only folder permissions. These are easy to fix.

Try running “npm install” again. It should only take a few seconds.

Go to the folder which contains your codebase. Right click on the codebase folder and click “properties”.

Uncheck the “Read-only” box and click apply. Another dialog will pop-up asking if you want to apply the change to just the folder or all subfolders and files? It will default to the latter so just click OK and wait for it to finish. Then click OK on the Properties dialog.

Go back to Visual Studio and run the build again. It should succeed and allow you to run the solution.
If you’ve tried these steps and you are still getting build errors, you’ll need to reach out to your supervisor or a full-stack developer.