Week 12: DevOps: Setting Up a Game Development Environment

Note: While you may use a functional programming pattern for your final project, you are required this week to set up a build system in a new Cloud9 workspace and try using it by building the example game in Chapter 5 of our textbook.

Welcome to DevOps

The phrase "DevOps" was coined at the 2008 Agile Toronto conference, by Andrew Shafer and Patrick Debois. They introduced the term in their talk on "Agile Infrastructure". Since then, the term has been steadily promoted and brought into more mainstream usage.

DevOps is focused on the set-up of software tools in a development environment that supports and facilitates a proper workflow for writing code, building a project from the code, testing the code, and packaging the code for distribution to others.

Setting Up Your Own Build System

In this first part of our class, we will use a Cloud9 workspace that already includes many of the software tools needed for our build system, including the Git version control management system and the Node.js JavaScript runtime.

These DevOps software tools – as well as the others I walk through installing in the Screencast (and the book tours in Chapter 4) - fit into one or more of the key stages of the software development and delivery process:

Code — Cloud9 editor to code development and review, source code management tools, code merging
Build — Grunt, Browserify and Babel to provide continuous integration of code
Test — continuous testing tools that provide feedback on business risks
Package — Bower and npm to manage the packages, repository, and pre-deployment staging
Release — change management, release approvals, release automation
Configure — infrastructure configuration and management, Infrastructure as Code tools
Monitor — applications performance monitoring, end–user experience

Our DevOps Tools Explained

By the time you complete the steps outlined in this week's Screencast and Chapter 4 of An Introduction to HTML5 Game Development with Phaser.js, you will have established a cloud-based virtual server (workspace in Cloud9's terminology) that contains a vast amount of powerful JavaScript-based code

designed to support your own coding and creation process for making games. **What are all of these software tools?**

Git	Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It can be used to keep track of changes in any set of files. For us, it will support the copying of packages we need from other developers.
Node.js	Node.js is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side. Node.js runs scripts server-side to perform all sorts of operations. For us, Node.js is the execution environment for all of our build scripts.
npm (Node Package Manager)	npm consists of a command line client and an online database of packages, called the npm registry. The registry is accessed via the client, and the available packages can be browsed and searched via the npm website. npm can manage packages that are required for a project, as well as globally-installed JavaScript tools. npm can install, in one command, all the dependencies of a project through the package.json file.
Bower	If npm is the tool to manage the software packages we need as developers to establish our build system on the Cloud9 server, bower is the package manager we use to manage any software library packages that will be required for distribution with our final game. This will include the Phaser.js library file itself.
Browserify	Browserify is an open-source JavaScript tool that allows developers to write Node.js-style modules that compile for use in the browser. For us, it allows us to write our game code modularly, in separate well-organized files, which get combined into a single production JavaScript file.
Babel	Babel is a code transpiler. Transpilation is a combination of "translation" and "compilation" because what Babel does is convert code written in different versions of ECMAScript to make that code compatible with older browsers.
Babelify	Babelify is a package that bridges between Browserify and Babel, allowing the Babel transpilation process to happen in conjunction with the combination of our source files,

	resulting in a single JavaScript file that has been tweaked to be compatible with ECMAScript 5 used in most browsers.
Grunt	Grunt is a JavaScript task runner, a tool used to automatically perform frequently used tasks such as minification, compilation, and testing. It uses a command-line interface to run custom tasks defined in a file (known as a Gruntfile). There are more than five thousand plugins available in the Grunt ecosystem. For us, Grunt will give us a real-time build system that produces testable production code and a Web server to serve the files.
Grunt Connect	A Grunt plugin that provides a simple Web server to deliver our project files to us in a browser.
Grunt Watch	A Grunt plugin that watches our source code folder(s), triggering other scripts to run when any changes are made. This enables instant builds to be produced each time a source code edit is made.
Main Bower Files	A Grunt plugin that copies any Bower-managed packages into our final build. This includes the Phaser.js library file for us.
Yeoman	A template-based project setup tool that leverages the other build tools already discussed to allow you to produce a completely new project setup – with directories and build tools are configured – just by entering a single command in the terminal.