A 2 page (or so) overview of the project and the work toward completing the project. What were the interesting aspects of the project? Was there anything from class that you feel was better illuminated by the work on the project? What type of obstacles did you have to overcome and what did you do to overcome them? If you (or someone else) were to continue working on this project, what would you tackle next? What would a "final" version of your project look like if it were completely finished?

We originally chose our project topic to be investigating Flash. We both had an interest in learning how to write Flash animations, and thought it would be a fun project topic. Our project would be an investigation of Actionscript, the programming language of Flash. We were intrigued to find out that Actionscript was based on the same specification as Javascript, ECMAScript. Actionscript was a commercial endeavor by Macromedia, now owned by Adobe. Although based on ECMAScript, Actionscript is not compatible with it. One problem with Actionscript is it was not made to work in a variety of host environments, like ECMAScript. We were able to install a 30-day trial of the Macromedia flash mx designer. We found it very difficult to negotiate. The actionscript code was embedded in binary in a flash-format. The Actionscript specification [http://livedocs.adobe.com/specs/actionscript/3/wwhelp/wwimpl/js/html/wwhelp.htm](http://livedocs.adobe.com/specs/actionscript/3/wwhelp/wwimpl/js/html/wwhelp.htm) was not a very standards-driven one. It was more of a programming manual than a formal specification, so we were using the ECMAScript language specification as our main resource.

We decided that progress would be much speedier on our project, with the same formal analysis of semantics, if instead we just switched our language to Javascript, which is a direct implementation of Ecmascript. A great advantage of Javascript is that we already had the only tool needed to execute Javascript code, a web browser.

One of the challenges of the project was finding resources on Javascript that were authoritative. A great advantage of Javascript is that it is so widely deployed, and there are plenty of useful documents available online to help you. On the otherhand, there are many misconceptions about Javascript. This is possibly because of the wide variety of concrete syntax provided to make Javascript as programmer-friendly as possible. If you’re a Java programmer, you can write your Javascript program to look like Java code. If you normally use a functional programming language, you can write your code to look like that language. A big task of the project was sifting through technical
documents to find what concrete syntax matched to what abstract syntax. The most useful resource from class for this project were the paper and lectures on featherweight Java. JavaScript is a very full-featured multi-paradigm programming language. It is function, object-oriented, and imperative. Its development was at first industry-driven, then user-driven. We knew we were interested in analyzing how inheritance could work in a non-class-based language.

The biggest obstacle to overcome was how to type an Object. Javascript has only one non-primitive type, Object. Function, a built-in object, seems to fall in a very special class of this and is not very formally explained in the Ecmascript specification. In Javascript, objects of type Object are constructed with constructor objects of type Function. We were able to get around the typing problem of Object by checking for well-formedness rather than trying to type. We followed the model set by FeatherweightJava very closely. Object expressions were only used for field selection, and field selection always evaluated to a value.

If we were to go further with the project, we would endeavor to come up with typing rules for objects. The final version of our project would be a more complete analysis of Javascript to the level of Featherweight Java. Objects would be allowed to have properties of any type (rather than just number). We would also work on a better way to formalize the marriage of constructors as functions and objects, so that constructors do not have to be treated as a very special case of the Object type.