1. **COURSE ID:** CIS 114  
**TITLE:** JavaScript/Ajax Programming  
**Units:** 4.0 units  
**Hours/Semester:** 48.0-54.0 Lecture hours; 48.0-54.0 Lab hours; and 96.0-108.0 Homework hours  
**Method of Grading:** Grade Option (Letter Grade or P/NP)  
**Recommended Preparation:**  
Eligibility for ENGL 838 or ENGL 848  
Eligibility for ESL 400, Completion of CIS 111 or CIS 254.

2. **COURSE DESIGNATION:**  
Degree Credit  
**Transfer credit:** CSU; UC

3. **COURSE DESCRIPTIONS:**  
**Catalog Description:**  
Study of the JavaScript programming language. Provides an overview of HTML5 and CSS, client-side programming, variables, arrays, functions, closures, event handlers, objects, form validation, cookies, and the DOM. Introduces Ajax (Asynchronous JavaScript and XML) technologies, design patterns, server-side programming, RSS, JSON, open-source libraries, and advanced topics such as MVC frameworks, security, performance, and Web Services. Intended for students with previous programming experience.

4. **STUDENT LEARNING OUTCOME(S) (SLO'S):**  
Upon successful completion of this course, a student will meet the following outcomes:  
1. Develop interactive Web applications that integrate HTML5 with JavaScript using event handlers.  
2. Explain object-based programming and the Document Object Model (DOM).  
3. Develop interactive Web applications that integrate client- and server-side programming using JavaScript and a server-side language.  
4. Employ Ajax technologies to fetch XML, RSS, or JSON data asynchronously from the server.  
5. Explain JavaScript design patterns and illustrate how they are used to create various applications.  
6. Create an advanced project using various libraries and frameworks, with attention to security and performance.

5. **SPECIFIC INSTRUCTIONAL OBJECTIVES:**  
Upon successful completion of this course, a student will be able to:  
1. Develop interactive Web applications that integrate HTML5 with JavaScript using event handlers.  
2. Explain object-based programming and the Document Object Model (DOM).  
3. Develop interactive Web applications that integrate client- and server-side programming using JavaScript and a server-side language.  
4. Employ Ajax technologies to fetch XML, RSS, or JSON data asynchronously from the server.  
5. Explain JavaScript design patterns and illustrate how they are used to create various Ajax applications.  
6. Create an advanced project using various libraries and frameworks, with attention to security and performance.

6. **COURSE CONTENT:**  
**Lecture Content:**  
1. Introduction  
   A. History of JavaScript  
   B. HTML5  
   C. CSS  
   D. Editors/IDEs  
   E. Debuggers  
2. JavaScript Basics  
   A. Data and Operations  
   B. Conditional Statements and Control Structures  
   C. Functions  
   a. Methods  
   b. Closures
c. Recursion
d. Scope
e. Event Handlers

D. Arrays
   a. Array Object
   b. Array Methods
   c. Associative Arrays
   d. Multidimensional Arrays

3. Objects
   A. Native Object Types
   B. Prototype Inheritance
   C. Custom Object Types
   D. Browser Objects
   E. Document Object Model (DOM)
      a. DOM Scripting
      b. DOM Nodes
      c. HTML Nodes

4. Input/Output
   A. Validating Form Input
      a. Regular Expressions
      b. Exception Handling
   B. Reading/Writing Cookies
   C. Browser Detection
   D. Keystroke Detection

5. Introduction to Ajax
   A. Ajax Technologies
      a. XMLHttpRequest
   B. History of Ajax
   C. Real-World Examples

6. Ajax Design Patterns
   A. Observer Pattern
   B. Singleton Pattern
   C. Model-View-Controller

7. Ajax and the Server
   A. Ajax Libraries
   B. Server-Side Designs
   C. Exchanging Data
   D. Writing to the Server
      a. Using HTML Forms
      b. Using XHR Objects
      c. Managing User Updates

8. Integrating the Client and Server
   A. Client-Side Code
   B. Server-Side Code
   C. Browser Issues

9. Building Stand-Alone Ajax Applications
   A. Rich User Interface
   B. Loading XML and RSS Feeds
   C. Web Services
   D. Open-Source Libraries and Frameworks
   E. JSON

10. Ajax Security and Performance
    A. JavaScript and Browser Security
    B. Communicating with Remote Services
    C. Protecting Confidential Data
    D. Restricting Access to Web Data

Lab Content:
1. Programming project using a loop, a switch statement, and event handlers.

2. Programming project employing functions and arrays for data storage.
3. Programming project with object creation and manipulation.
4. Programming project with form submission, string and image processing.
5. Task management Web application with form submission.
6. Simple auction Web application with form submission.
7. Programming project with popup windows and cookies.
8. Programming project with select menus, checkboxes and/or radio buttons.
9. Programming project using regular expressions for input validation.
10. Programming project with a server-side PHP script to fetch JSON data.
12. Simple stock ticker application using a Web service for data retrieval.
13. Programming project using exception handling for error management.
14. Programming project using Ajax libraries such as jQuery and YUI.
15. Programming project using prototypes and prototypical inheritance.
16. Programming project using a public Web service (such as iTunes or weather reports).
17. Programming project using templating libraries such as Mustache, Underscore, or Handlebars for data management.
18. Data-driven programming project using MVC/MVP frameworks such as AngularJS or Backbone.
19. Programming project using Web APIs such as Google Maps, YUI, language translators, or similar Web services.
20. Final project using Ajax libraries, frameworks, Web services, with attention to security and performance.

7. REPRESENTATIVE METHODS OF INSTRUCTION:
   Typical methods of instruction may include:
   A. Lecture
   B. Lab
   C. Discussion
   D. Other (Specify): Teacher will model problem-solving techniques. Class will solve a problem together, each person contributing a potential "next step". Teacher will create and manage an Internet conference for discussion of course topics. Students will work in small groups to solve programming assignments.

8. REPRESENTATIVE ASSIGNMENTS
   Representative assignments in this course may include, but are not limited to the following:
   **Writing Assignments:**
   Weekly programming assignments
   **Reading Assignments:**
   A. Reading assignments accompanied by self-test questions and running coding samples
   **Other Outside Assignments:**
   A. Studying posted lecture notes and relevant handouts. Computer programming assignments. The reading assignment frames the concepts covered and provides the basic knowledge necessary to do the self-test questions and understand the sample code. The lecture notes and handouts provide a more in-depth look at topics and distills the information down to what the faculty thinks is most important. The computer programming assignments provide hands on practice of the concepts covered in the readings.
9. REPRESENTATIVE METHODS OF EVALUATION
Representative methods of evaluation may include:

A. Class Work
B. Exams/Tests
C. Homework
D. Lab Activities
E. Projects
F. Quizzes
G. Written examination
H. Bi-weekly quizzes (short answer–from textbook material) to provide feedback to students and teacher; Assessment of student contributions during class discussion and project time; Individual programming assignment; Midterm and Final exams (short answer (textbook material)), general problem solving (similar to in-class work), short program segments (similar to programming assignments); Assessment of group participation on course projects, including peer-assessment of participation and contribution to the group effort.

10. REPRESENTATIVE TEXT(S):
Possible textbooks include:


**Origination Date:** September 2016
**Curriculum Committee Approval Date:** October 2016
**Effective Term:** Fall 2017
**Course Originator:** Melissa Green