

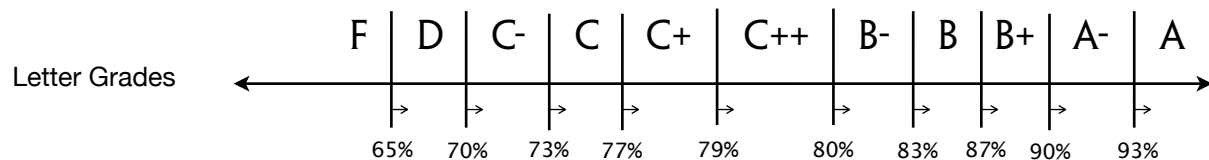
Web Programming I

ITS 210 • Fall 2009

-Background

When and Where	LT-135 Thursday evenings 6:30pm through 9:15pm	
Required Text	<i>Programming the World Wide Web, 5th edition</i> Robert Sebesta ISBN 978-0-13-607663-6	
Web Site	www.3nfconsulting.com/students-web1.aspx	
Instructor	Alan G. Labouseur LT 101 (office hours posted)	Alan.Labouseur@Marist.edu alan@3NFconsulting.com 845-575-3000 x2831 Marist phone 845-440-1102 alternate phone

-Grading



You can earn up to 1000 points over the course of the semester, broken down over the following areas:	Homework	15%	150 points: 3 at 50 points each
	Test 1	15%	150 points
	Project 1	20%	200 points
	Test 2	15%	150 points
	Project 2	20%	200 points
	Attendance	5%	50 points for consistently showing up
	Participation	5%	50 points for constructive participation
	Laziness Adjustment	3%	30 points for not being lazy
	Whining Adjustment	2%	20 points for not whining

-Objectives and Assessment

Assessment methods include assignments, quizzes, exams, discussions, presentations, and projects.	<ol style="list-style-type: none">1. Develop the skills to design cross-browser applications2. Be able to construct browser-based client applications using XHTML, Javascript, CSS, and the DOM event model3. Reinforce the core concepts of Object-oriented programming.4. Get used to and practice event-driven programming.5. Provide students an opportunity to develop software systems over the course of the semester, where they have to live with their past mistakes and shortcuts, or fix them. Either will teach a valuable lesson.6. Development is only half the battle. Debugging is a critical skill, and is stressed in this course.7. Students will get practice in finding some answers for themselves. Capable problem solvers never stop learning.
---	---

Web Programming I

ITS 210 • Fall 2009

-Proposed Schedule

#	Week	Ch	Topic	Required
1	3-Sep	1	<i>Theory:</i> Intro / The Internet in a Nutshell / Intro to DOM & HTML5 <i>Practice:</i> Our first Web App: TV and Movie data	<i>Presence</i>
2	10-Sep	1, 2	<i>Theory:</i> Clients and Servers / HTML5 (headers, linking, images, lists) <i>Practice:</i> DOM / TV and Movie library	<i>Practice</i>
3	17-Sep	2	<i>Theory:</i> More HTML5 (tables and forms) / more DOM <i>Practice:</i> New and improved TV and Movie library	Homework 1
4	24-Sep	3	<i>Theory:</i> Styles and CSS / more DOM <i>Practice:</i> Pretty-up the TV and Movie library	<i>Practice</i>
5	1-Oct	4	<i>Theory:</i> Introduction to JavaScript / Introduction to the Event model <i>Practice:</i> alert() / Calling JavaScript functions in response to events	Homework 2
6	8-Oct	4, 5	<i>Theory:</i> Event processing with JavaScript / more DOM / innerHTML <i>Practice:</i> innerHTML, Input validation and error messages	<i>Practice</i>
7	15-Oct	-	Test One / Practical Exam in class	<i>Expertise</i>
8	22-Oct	4	<i>Theory:</i> Review JavaScript so far / Math in Javascript <i>Practice:</i> Magic Square / getting and setting element values	Project One
9	29-Oct	4	<i>Theory:</i> Loops in JavaScript / Intro to JavaScript objects <i>Practice:</i> Computing aggregates with loops	<i>Practice</i>
10	5-Nov	4	<i>Theory:</i> JavaScript Objects / JavaScript Arrays <i>Practice:</i> error messages in arrays / TV program objects in arrays	Homework 3
11	12-Nov	-	<i>Practice:</i> Lab time for Project Two / Final Project work	<i>Practice</i>
12	19-Nov	-	<i>Theory:</i> Web 2.0, Ajax, JSON with guest speaker Jason Sliss <i>Practice:</i> Ajax, JSON	<i>Practice</i>
13	26-Nov	-	<i>No class meeting - Thanksgiving</i>	<i>Appetite</i>
14	3-Dec	-	<i>Theory:</i> OOP in JavaScript, JSON <i>Practice:</i> Complex classes in JavaScript, fun with JSON	<i>Expertise</i>
15	10-Dec	-	Test Two / Practical Exam in class	<i>Practice</i>
16	17-Dec	-	Final Project Presentations	Project Two

Web Programming I

ITS 210 • Fall 2009

-Policies

Tests

Tests cover material presented up to the class in which the test is administered. No makeup tests will be given. If you anticipate missing a test, make arrangements with me in advance to hand in the exam on or prior to its due date.

Homework

All assignments must be handed in at the beginning of class on the day they are due. Since all homework assignments are outlined in this syllabus, arrange to submit homework on schedule, even when a class will be missed.

Late Submissions

No assignments will be accepted late because we will be going over the assignment in class on the day it's due. This is an important part of the learning process, and once we cover the assignment in class, you clearly cannot hand it in after that. (Inconceivable!)

Academic Honesty

As a part this class, I will uphold and **vigorously enforce** the general policies of this institution on academic honesty and plagiarism. All examinations, papers, projects, and homework assignments are subject to the usual standards of academic honesty as described in the Student Handbook and/or other related publications.

Furthermore, I expect my students to behave in a manner appropriate to Computer Science and Information Technology professionals. Professional ethics **demand** that you embrace traditional "thou shall not cheat" behavior, and also that you soundly reject additional forms of dishonesty and abuse which are uniquely possible working with computers.

Remember: Allowing someone to copy your work is every bit as dishonest as copying someone else's, and will be treated just as harshly.

Any violation -- actual or perceived (in my sole discretion) -- of this Academic Honesty policy will result in one or more of the following actions in addition to any other forms of recourse available as specified by the Student Handbook:

- You will be ejected from the course with a failing grade.
- A letter will be sent to your department chair, your Dean, and the president of the college.
- And more. (And worse!)

The bottom line is that I expect you to conduct yourself as a person of integrity. This means that **plagiarism in any form is completely unacceptable**. You are soon-to-be a computing professional, and I encourage you to consult the ACM professional code of ethics. See www.acm.org/about/code-of-ethics.

Web Programming I

ITS 210 • Fall 2009

-Policies

Appealing Grades

This semester I will implement an appeals process to handle any questions you might have about fairness related to the grading of your work. I will address each and every one of your concerns. To that end, and in order to be fair and efficient, I insist you to write a letter of appeal.

Rules for Submitting an Appeal

- Appeals must be in the form of a neatly written letter.
- Appeals must be on a separate paper and stapled to the work in question.
- Every appeal (if there is more than one) requires its own paragraph.
- Appeals are due the next class period after the work is returned to you.
- Appeals must be very specific.
- Appeals must be content-based, not personal or emotional.
- Insufficient time is not a basis for an appeal.
- You must communicate what action you would like taken, for instance give full credit, add points, etc.

This process empowers students, advances learning, and moves students toward academic maturity. As such, it benefits both the teacher and the student. Further, students are given a method to argue their points in an appropriate manner and explain their reasoning, while the teacher has an opportunity to learn whether or not he has understood students' reasoning.

Students with Disabilities

Any student requesting or wondering about accommodations based on a disability should see the fine folks at the Office of Special Services in Donnelley 226 and online at www.marist.edu/specialservices.

Web Programming I

ITS 210 • Fall 2009

-Program Evaluation

Guidelines for Grading Programs and Projects

All programs and applications must be free of syntax errors to receive any credit. Programs that ((compile or interpret) and execute) cleanly but contain logic errors will be graded based on the severity of the errors and how well your work demonstrates your approach to solving the problem.

When evaluating your programming assignments I will ask myself the following questions about your program:

- Is it correct? I.e., free from faults in specification, design, and implementation?
- Is it usable? About the interface...
 - ▶ Is it “clean” and well-organized? Would Mr. Monk be proud?
 - ▶ Does it obey the Laws of Least Astonishment?
 - ▶ Is is accurate?
 - ▶ Is it easy to use?
- Is it reliable and robust? I.e., can it perform its functions without breaking down, even given unexpected input/data or other circumstances?
- About the readability and maintainability of the source code...
 - ▶ Are the comments plentiful, clear, meaningful, and helpful?
 - ▶ Are the identifier names accurate, clear, and meaningful?
 - ▶ How is its object-oriented architecture?
 - ▶ Does it compile or interpret cleanly?
- About the Design...
 - ▶ Is the solution well-designed?
 - ▶ Is it re-usable?
 - ▶ Does it illustrate the points made and principles discussed in class?
 - ▶ Have any lazy shortcuts been taken?
 - ▶ Can the program be unit and system tested?
- About the results, are they accurate? I.e., are the qualitative outputs free of error?
 - ▶ Does it perform as assigned?
 - ▶ Is the output well-formatted and meaningful?

Remember, neatness and style count. If you hand in a program that works, but that does not adhere to reasonable style standards, is inadequately commented, or is poorly designed, you will be penalized. Good habits are important and I want you to develop some.