Client-side Verification of Credit Card Payment Information

CS453 Electronic Commerce
Homework #2 – JavaScript

Due: by 8pm Tuesday, July 21, 2009, via Collab
Collaboration rules: For summer term, you must work individually on this assignment. Ignore any comments in this document that refer to partners or teams of two persons. When submitting your work (which will be one HTML document), assure that your web page displays the names and email addresses of all authors.
What's allowed: If you have questions or problems with Javascript, you may ask your classmates. If asked, you are allowed to act as a "teacher" but not as a "subcontractor". In other words, explaining and showing to help someone learn is fine, but showing or giving code that directly solves the problem and is not fine.
Pledge: Your coding is your own. You may make use of any public repositories of useful information or code, but you must thoroughly understand any code you utilize and must be able to explain to the instructors any code that you submit.

A common e-commerce operation is to conclude an on-line sale by having the client (user) present credit card information (e.g., credit card number, expiration date, and credit card verification (CCV) number) for processing by the server. To reduce the load on the server and to reduce the communications bandwidth between the client and server, it is beneficial to use a client-side script to perform syntactic validation of the information provided. In other words, before the user can submit, the script checks to see that the required information is present and that the input conforms to whatever formatting rules are required by the specific credit card issuer. In this homework you will create a credit card payment page that asks the user to select (a) which credit card is being used, (b) its number, (c) its expiration date, and (d) its CCV. When the user clicks "submit" your JavaScript will apply the verification rules detailed below; if and only if the input adheres to all the verification rules will the form information be submitted.

Assume that items have already been selected for purchase, a shopping cart has already been created, all processing needed to arrive at a total price has been completed, and the user's order total is $123.45. This is the amount to be paid by credit card. Your store, Acme Electronics, now generates a page to collect the credit card information needed for payment processing. Use JavaScript within your web page to verify the information that the user submits according to the verification rules listed below. If and only if the user's input passes all the verification checks will your page submit the HTML form information to a server. But since at this moment we have no server-side code (that comes later using PHP or Perl), we'll substitute sending the credit card information to a special email address in place of actually invoking the payment processing server that does not yet exist.

While your web page design should be aesthetically pleasing, the details of that design are delegated to the artist within you.

Your web page must display:

(1) Your team members' names and emails—such as "Thomas B. Horton (tbh3f) and Brenda Lee Jones (blj9xa)". Display this information at the top of the page in a small and unobtrusive (but still readable) font.
(2) The name of your company, Acme Electronics, in a handsome and distinctive font (not Times Roman or Verdana).
(3) A picture or logo relevant to an electronics store.
(4) A text message indicating that the total price of the user's order is $123.45.
(5) Text asking the user to select whether the credit card is American Express, Visa, or MasterCard. Then provide three appropriately labeled radio buttons for these three choices.
(6) Text asking the user to input the credit card number, and then a text box in which the user can input the number.
(7) Text asking the user to input the month and year of the expiration date, and then a drop-down menu of twelve choices (January through December) for the month, and a drop-down menu of six choices (2009-2014) for the year.
(8) Text asking the user to input the CCV from the credit card, and then a text box in which the user can input the CCV.
(9) A Submit button and a Reset button.

Note on HTML and CSS: you must use CSS (either an internal or external style sheet) to make your page look nice, and to handle the requirements for font use listed above in (1) and (2). You should also use CSS to change some of the default formatting of HTML elements to make your page looking interesting but still nice.

After the user makes choices and fills in the requested information and clicks submit, your JavaScript runs the verification rules below on the input. Remember that the form data is conceptually transmitted to the server (in actuality, emailed to horton @cs.virginia.edu) if any only if all of the verification rules listed below are satisfied. If your JavaScript finds any deficiency in the user's input after the user clicks submit, issue an appropriate error (alert) message that accurately describes the nature of the error(s) and do not submit the form information for processing (i.e., do not email the form data to horton). If multiple errors are present, they may be detected and resolved all at once or one at a time (all at once is easier on the user, one at a time is easier on the programmer, so here is a decision point for the designer). Just be sure that no form data is submitted (emailed) unless it passes all the verification rules.

Verification rules:

(1) One of the three credit cards must been selected (so one radio button must be selected). If no selection is made, issue an appropriate error message.

(2) If the credit card selected was American Express:
(2a) the credit card number (CCN) must contain only digits;
(2b) the CCN must be exactly 15 digits in length;
(2c) the CCN must not be all zeroes;
(2d) the credit card number, taken as a 15-digit integer, must be a multiple of 113.

Example: 139506171583899 is a multiple of 113, whereas 224604938284130 is not.

(3) If the credit card selected was Visa:
(3a) the CCN must contain only digits;
(3b) the CCN must be exactly 16 digits in length, but no larger than 9007000000000000;
(3c) the CCN must not be all zeroes;
(3d) the CCN, taken as a 16-digit integer, with the digit positions numbered from 1..16 from most- to least-significant, must have the sum of the four digits in positions 5..8 equal the sum of the four digits in positions 9..12.
Example: 2857463206813856 is valid because 4+6+3+2=15 and 0+6+8+1=15.

(4) If the credit card selected was MasterCard:
(4a) the CCN must contain only digits;
(4b) the CCN must be exactly 16 digits in length, but no larger than 9007000000000000;
(4c) the CCN must not be all zeroes;
(4d) the CCN, taken as a 16-digit integer, with the digit positions numbered from 1..16 from most- to least-significant, must have the digit is position 8 be even, the digit in position 12 be odd, and the digit in position 16 be even.
Example: 8473658671239584 is valid because 6 is even, 3 is odd, and 4 is even.

(5) If the user inputs an invalid CCN, the error message should be something generic like "Invalid credit card number. Please re-enter." Do not give more explicit error information because that would help a hacker.

(6) The expiration month (January—December) must be selected from the drop-down menu. If no month is selected, issue an appropriate error message.

(7) The expiration year (2009-2014) must be selected from the drop-down menu. If no year is selected, issue an appropriate error message.

(8) The selected combination of month and year must represent an expiration date that has not expired on the date the information is submitted (use JavaScript's date functions to gain access to the current date; *do not* hard code the date test using any particular date as a constant; this test must be accurate on whatever date it is run). If the card has expired, issue an appropriate error message.

(9) The CCV must be exactly three digits, but they can not be three identical digits. If the input is invalid, issue an appropriate error message. Examples: 123 and 997 and 003 are valid CCVs; 000 and 444 are invalid.

(10) At any point the user should be able to click the Reset button to clear his form.

(11) When the user's data passes all the verification rules, submit the form data to the server (which we will simulate by emailing it to tbh3f @virginia.edu). See the virtual labs and the JavaScript examples on the class-site for examples of how to do this.

In reality, once the payment information passes all these client-side tests and is submitted to the Acme Electronics server, the server repeats all these tests (and more), and in addition communicates with the credit card company to determine if a particular number is valid at that moment in time. We use client-side processing to reduce the time, cycles, and bandwidth required of the server.

Note: The only reason 16-digit numbers are not allowed to be larger than 9007000000000000 is because that is close to the maximum integer value (i.e whole number) that JavaScript can store.

Resources. Refer to our Virtual Labs on JavaScript, other tutorials and examples accessible from the class site. Also consider Google.com, sourceforge.net, and javascript.com.

Grading:

CRITERION	POINTS
All required elements are present on the web page (authors, firm name, picture/logo, explanatory text, price, instructions to the user and the textbox or drop-down menu associated with each input).	10
Quality of artistic design and use of CSS.	10
Enforces choice of one credit card; error message if no card is selected.	10
Enforces verification rules for AmEx CCN; error message if invalid.	10
Enforces verification rules for Visa CCN; error message if invalid.	10
Enforces verification rules for MasterCard CCN; error message if invalid.	10
Enforces selection of expiration month and year; error message if month or year not selected.	10
Validates expiration date by using JavaScript's supplied functions for determining today's date.	10
Enforces CCV validity rules; error message if invalid.	10
Submit payment information to the server (simulated by emailing to tbh3f) if and only if all information provided is valid.	10
TOTAL	100

Submission:

Name your file "HW2-lastname.html" for individuals or "HW2-lastname1-lastname2.html" for teams. If you use an external stylesheet, submit that too. Submit by the deadline listed at the start of this document at the link given on the class site.

Late penalty:

10% a day up to a maximum of two days late. Submissions later than two days won't be accepted. Contact me as soon as possible if any kind of problem arises.