CSCE 120: Learning To Code

Manipulating Data I
Hacktivity 7.4

Introduction

Prior to engaging in this hacktivity, you should have completed all of the pre-class activities as outlined in this module. At the start of class, you will be randomly assigned a partner to work with on the entirety of this hacktivity as a *peer programming* activity. Your instructor will inform you of your partner for this class.

One of you will be the driver and the other will take on the navigator role. Recall that a driver is in charge of the keyboard and computer while the navigator is in charge of the handout and directing the activity. However, you are *both* responsible for contributing and discussing solutions. If you were a driver/navigator in the prior activity, switch roles for this activity.

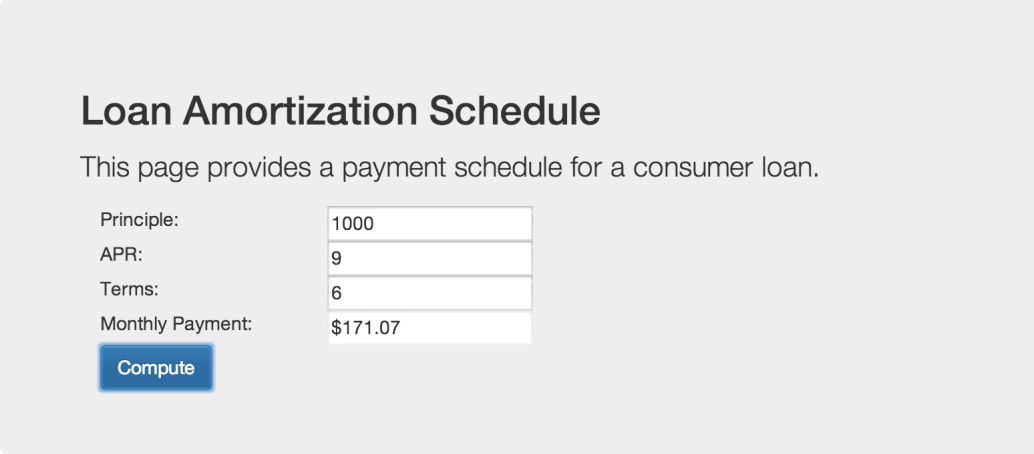
You will use the same project from the previous Hacktivity. If you need to, you can re-download it from GitHub using the URL, `https://github.com/cbourke/jQueryProject`.

Making Another App – Loan Amortization

Consumers take out loans for a certain *principle* p at a certain annual percentage rate (APR), r and pay the loan back monthly over a certain number of terms n. Each month, the consumer pays a monthly payment that can be computed with the following formula.

$$\text{monthly payment} = \frac{p \cdot i}{1 - (1+i)^{-n}}$$

where $i = \frac{r}{12}$ is the monthly interest rate. Note that this payment must be rounded to the nearest cent. Each month, part of the payment is applied to the interest (an amount equal to the monthly interest rate times the current balance) and the rest is applied to the principal. Your program will detail the month-by-month payment schedule including



Payment Number	Interest Payment	Principle Payment	New Balance
1	$7.50	$163.57	$836.43
2	$6.27	$164.80	$671.63
3	$5.04	$166.03	$505.60
4	$3.79	$167.28	$338.32
5	$2.54	$168.53	$169.79
6	$1.27	$169.80	$0.00

Figure 1: Loan Amortization Example

the amount of interest, the amount paid toward the balance, and the new balance after that month's payment.

As an example, suppose we have:

- Principle $p = 1000.00$
- APR: 9.00%
- Terms: 6 months

The monthly payment would be $171.07 with an amortization schedule as depicted in Figure 1.

We have provided you with starter code (see the files in the `LoanApp` folder). Complete the application by doing the following.

1. Write the `monthlyPayment()` function using the formula above. Be sure to use the provided `roundToCents()` function.
2. Write jQuery code to pull the form values for the principle, APR, and terms into the variables provided.

3. Validate the input values: the principle and terms should be positive and the APR should be in the range $[0, 100]$ and all three should represent numerical values (recall that you can use the `isNan()` to determine if the values are not numbers. Write a function or code to produce an error message using jQuery (produce a Bootstrap "alert" box using the `errorDiv` element).

4. Write code to convert the input values to numbers.

5. Compute the monthly payment and place this value into the "Monthly Payment" box. With this and all currency values, be sure to print the result to exactly two decimal places. Recall that you can use `value.toFixed(2)` to convert a number stored in the variable `value` to do this.

6. Code to compute each of the monthly values (month, interest payment, principle payment, and the new balance) has been provided to you. Write jQuery code to create a new HTML table row and columns using these values and add it to the table on each iteration of the loop.

7. Write similar code to handle the final monthly payment (the last payment must be handled differently as it may be slightly more or less than the monthly payment).

8. Finally, animate the new rows as a visual cue to the user using some jQuery effects (fade in or something similar, experiment with different effects).