Using Arrow Functions - ECMAScript 6 Tutorial

The ECMAScript 6 arrow function syntax is a shorthand for the ECMAScript 5 function syntax. It supports both block and expression bodies. The value of `this` inside the function is not altered: it is the same as the value of `this` outside the function. No more `var self = this` to keep track of the current scope.

In this unit, you add a new function to calculate the mortgage amortization. You also modify the existing functions to use the new ECMAScript 6 arrow function syntax.

Steps

1. Open `js/main.js`. Right after the `calculateMonthlyPayment` function, add a `calculateAmortization` function defined as follows:

```
let calculateAmortization = (principal, years, rate) => {
    let {monthlyRate, monthlyPayment} = calculateMonthlyPayment(principal, years,
rate);
    let balance = principal;
    let amortization = [];
    for (let y=0; y<years; y++) {
        let interestY = 0;  //Interest payment for year y
        let principalY = 0; //Principal payment for year y
        for (let m=0; m<12; m++) {
            let interestM = balance * monthlyRate;       //Interest payment for
month m
            let principalM = monthlyPayment - interestM; //Principal payment for
month m
            interestY = interestY + interestM;
            principalY = principalY + principalM;
            balance = balance - principalM;
        }
        amortization.push({principalY, interestY, balance});
    }
    return {monthlyPayment, monthlyRate, amortization};
};
```

2. Modify the `calculateMonthlyPayment` function signature as follows:

```
let calculateMonthlyPayment = (principal, years, rate) => {
```

3. Modify the signature of the **calcBtn** click event handler as follows:

```
document.getElementById('calcBtn').addEventListener('click', () => {
```

4. In the **calcBtn** click event handler, invoke `calculateAmortization` function instead of `calculateMonthlyPayment`:

   ```
   let {monthlyPayment, monthlyRate, amortization} = calculateAmortization(principal,
   years, rate);
   ```

5. As the last line of the **calcBtn** click event handler, log amortization data to the console (you'll display the amortization table in the application in the next unit):

   ```
   amortization.forEach(month => console.log(month));
   ```

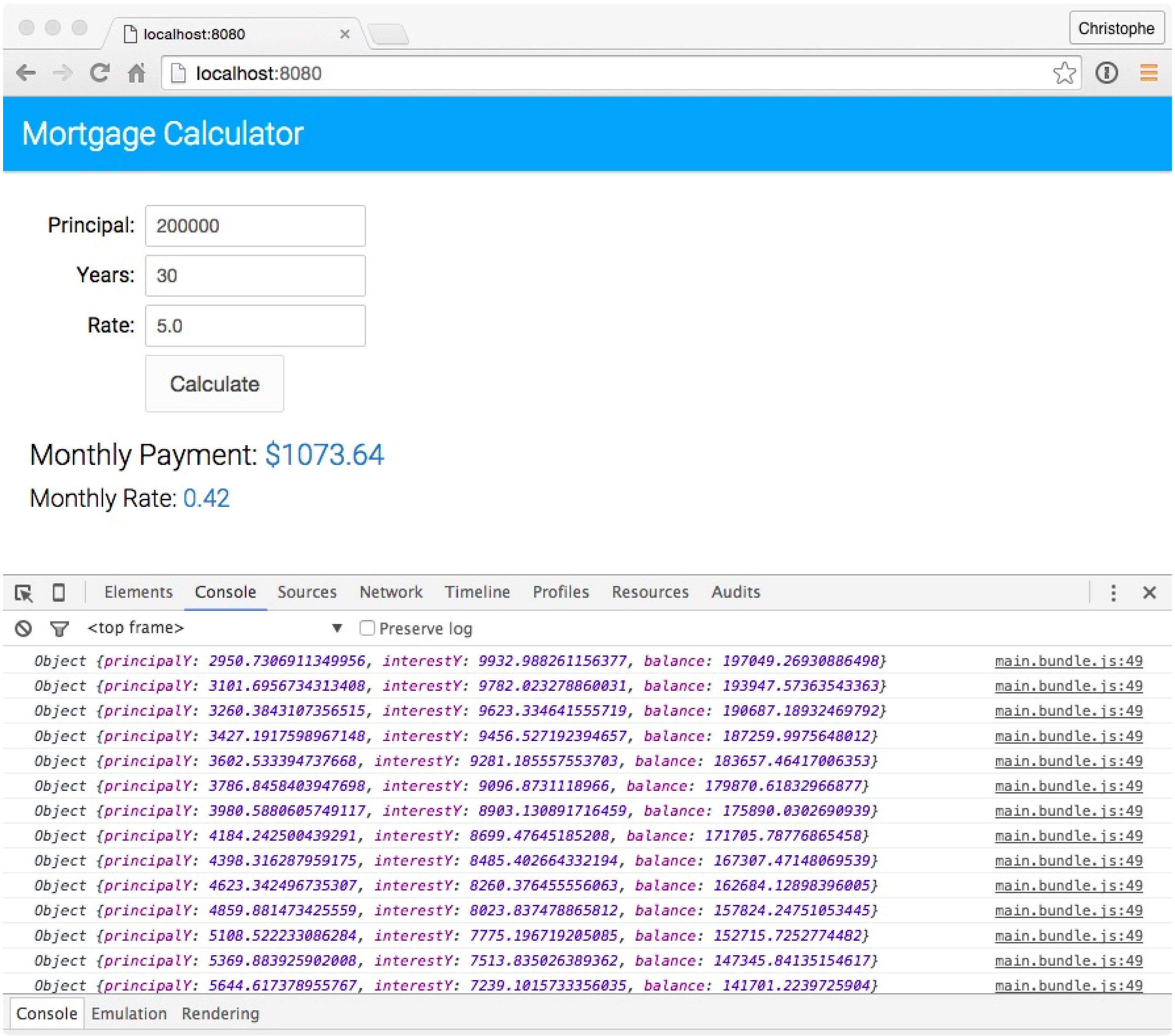
   This is an example of an expression body.
   The complete implementation of the button click handler looks like this:

   ```
   document.getElementById('calcBtn').addEventListener('click', () => {
       let principal = document.getElementById("principal").value;
       let years = document.getElementById("years").value;
       let rate = document.getElementById("rate").value;
       let {monthlyPayment, monthlyRate, amortization} =
   calculateAmortization(principal, years, rate);
       document.getElementById("monthlyPayment").innerHTML = monthlyPayment.toFixed(2);
       document.getElementById("monthlyRate").innerHTML = (monthlyRate *
   100).toFixed(2);
       amortization.forEach(month => console.log(month));
   });
   ```

6. On the command line, type the following command to rebuild the application:

   ```
   npm run babel
   ```

7. Open a browser, access http://localhost:8080, and click the **Calculate** button. Open the developer console: you should see the amortization values in the console log.



Mortgage Calculator

Principal: 200000

Years: 30

Rate: 5.0

Monthly Payment: $1073.64

Monthly Rate: 0.42

```
Object {principalY: 2950.7306911349956, interestY: 9932.988261156377, balance: 197049.26930886498}    main.bundle.js:49
Object {principalY: 3101.6956734313408, interestY: 9782.023278860031, balance: 193947.57363543363}    main.bundle.js:49
Object {principalY: 3260.3843107356515, interestY: 9623.334641555719, balance: 190687.18932469792}    main.bundle.js:49
Object {principalY: 3427.1917598967148, interestY: 9456.527192394657, balance: 187259.9975648012}     main.bundle.js:49
Object {principalY: 3602.533394737668, interestY: 9281.185557553703, balance: 183657.46417006353}     main.bundle.js:49
Object {principalY: 3786.8458403947698, interestY: 9096.8731118966, balance: 179870.61832966877}      main.bundle.js:49
Object {principalY: 3980.5880605749117, interestY: 8903.130891716459, balance: 175890.0302690939}     main.bundle.js:49
Object {principalY: 4184.242500439291, interestY: 8699.47645185208, balance: 171705.78776865458}      main.bundle.js:49
Object {principalY: 4398.316287959175, interestY: 8485.402664332194, balance: 167307.47148069539}     main.bundle.js:49
Object {principalY: 4623.342496735307, interestY: 8260.376455556063, balance: 162684.12898396005}     main.bundle.js:49
Object {principalY: 4859.881473425559, interestY: 8023.837478865812, balance: 157824.24751053445}     main.bundle.js:49
Object {principalY: 5108.522233086284, interestY: 7775.196719205085, balance: 152715.7252774482}      main.bundle.js:49
Object {principalY: 5369.883925902008, interestY: 7513.835026389362, balance: 147345.84135154617}     main.bundle.js:49
Object {principalY: 5644.617378955767, interestY: 7239.1015733356035, balance: 141701.2239725904}     main.bundle.js:49
```