Module 2: Review ES6-Tutorial; React Intro

Edgardo Molina, PhD | Head Instructor

CUNY Tech Prep 2017-2018

Outline

Review: ES6-Tutorial

React Intro

ES6 Tutorial

Building an interactive Mortgage calculator

In ES6 Tutorial we built a mortgage calculator using Vanilla JavaScript (ES6).

The page updates **dynamically (interactively)** as we change form values and click the “Calculate” button.

Mortgage calculator

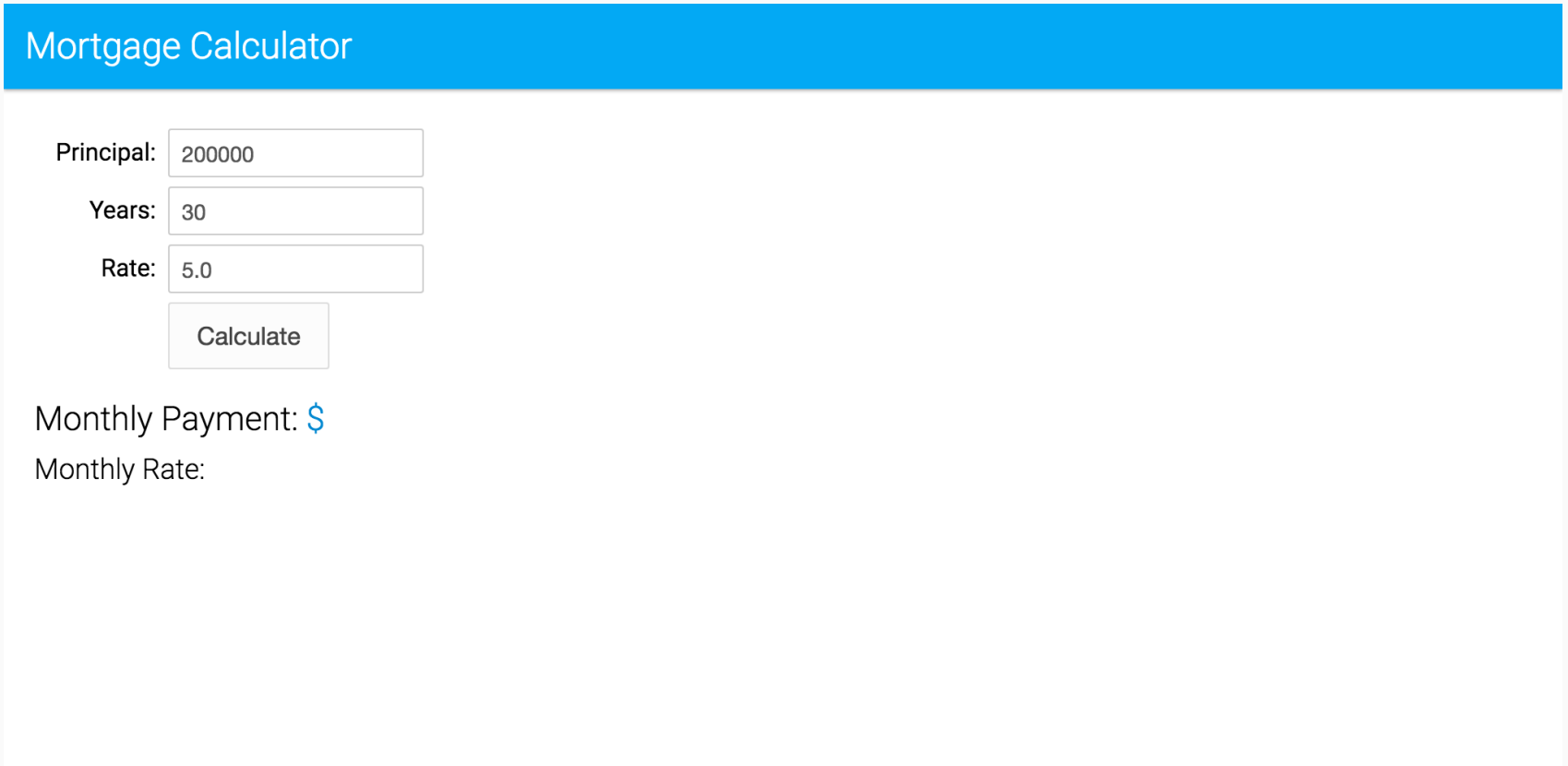

Figure 1: Mortgage Calc form

Mortgage calculator

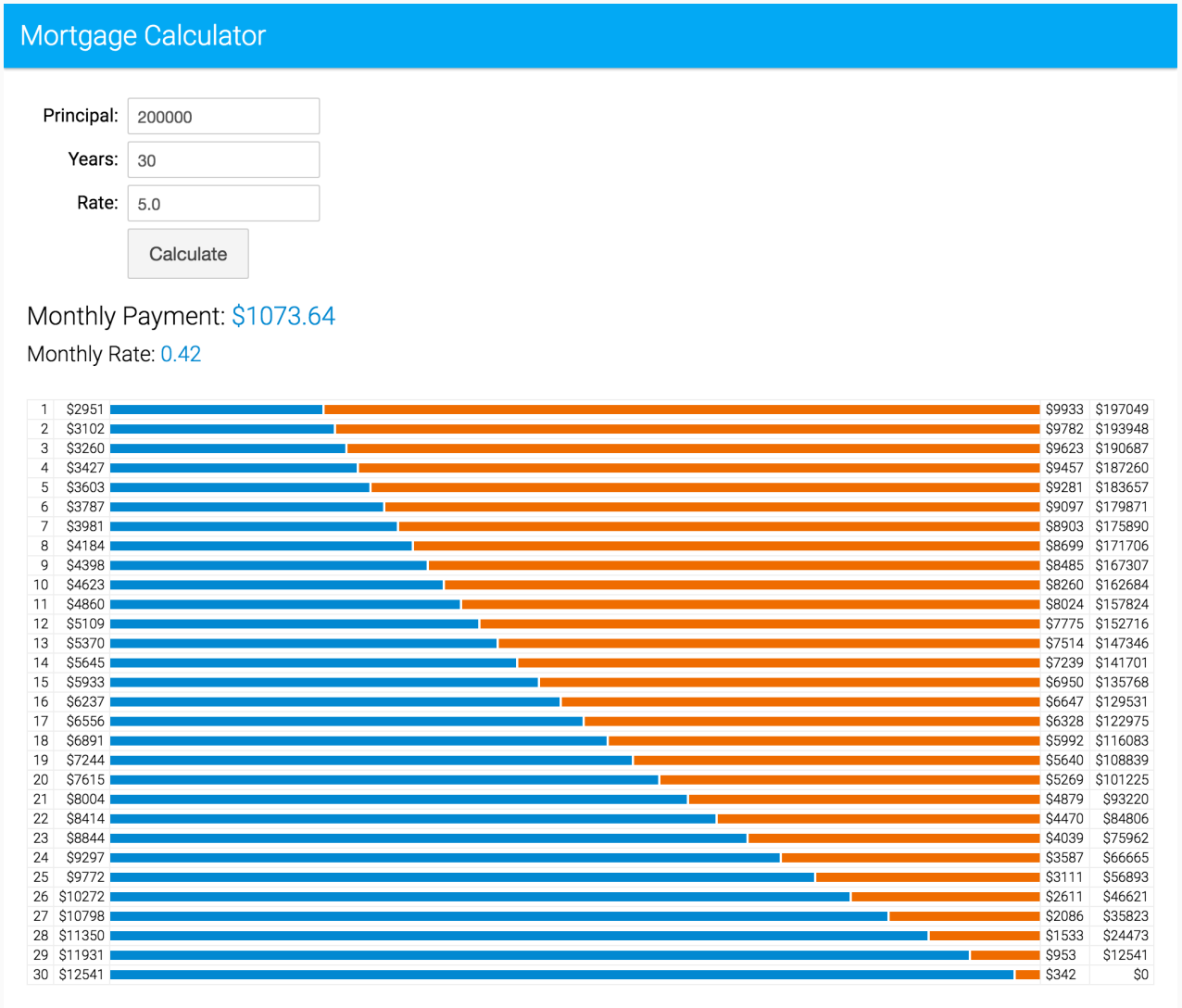

Figure 2: Mortgage Calc results

Mortgage calculator code

```
document.getElementById('calcBtn').addEventListener('click', () => {
    let principal = document.getElementById("principal").value;
    let years = document.getElementById("years").value;
    let rate = document.getElementById("rate").value;
    let mortgage = new Mortgage(principal, years, rate);
    document.getElementById("monthlyPayment").innerHTML = mortgage.monthlyPayment.toFixed(2);
    document.getElementById("monthlyRate").innerHTML = (rate/12).toFixed(2);
    let html = "";
    mortgage.amortization.forEach((year, index) => html += `
      <tr>
        <td>${index + 1}</td>
        <td class="currency">${Math.round(year.principalY)}</td>
        <td class="stretch">
          <div class="flex">
            <div class="bar principal"
              style="flex:${year.principalY};-webkit-flex:${year.principalY}">
            </div>
            <div class="bar interest"
              style="flex:${year.interestY};-webkit-flex:${year.interestY}">
            </div>
          </div>
        </td>
        <td class="currency left">${Math.round(year.interestY)}</td>
        <td class="currency">${Math.round(year.balance)}</td>
      </tr>
      `);
    document.getElementById("amortization").innerHTML = html;
});
```

Mortgage Calculator code

This uses the browser DOM API.

This function is clear but it is *large*. Is this testable? What would you test?

React Intro

What is React?

React is a *declarative* and *component-based* front-end library for building interactive applications.

Initially built by Facebook and Instagram, now developed and used by a larger community.

Solves a few problems:

- Makes rendering highly dynamic and interactive apps fast!
- Makes developing interactive apps better/easier... (subjective)

React fundamentals

Instead of thinking of and developing HTML/CSS/JS for your entire app at once, we want to break it down into simpler (and possibly *reusable*) components.

React introduces JSX syntax to create/interact with HTML more naturally than using the DOM api directly

> *JSX looks like HTML*
>
> *but, JSX != HTML*

Given the following mockup...

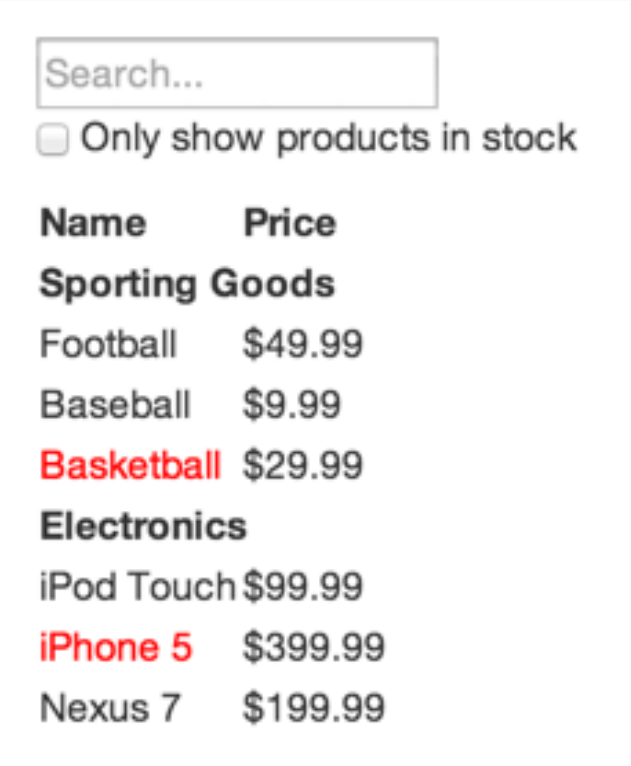

Figure 3: The app mockup

We want to break it into simpler components...

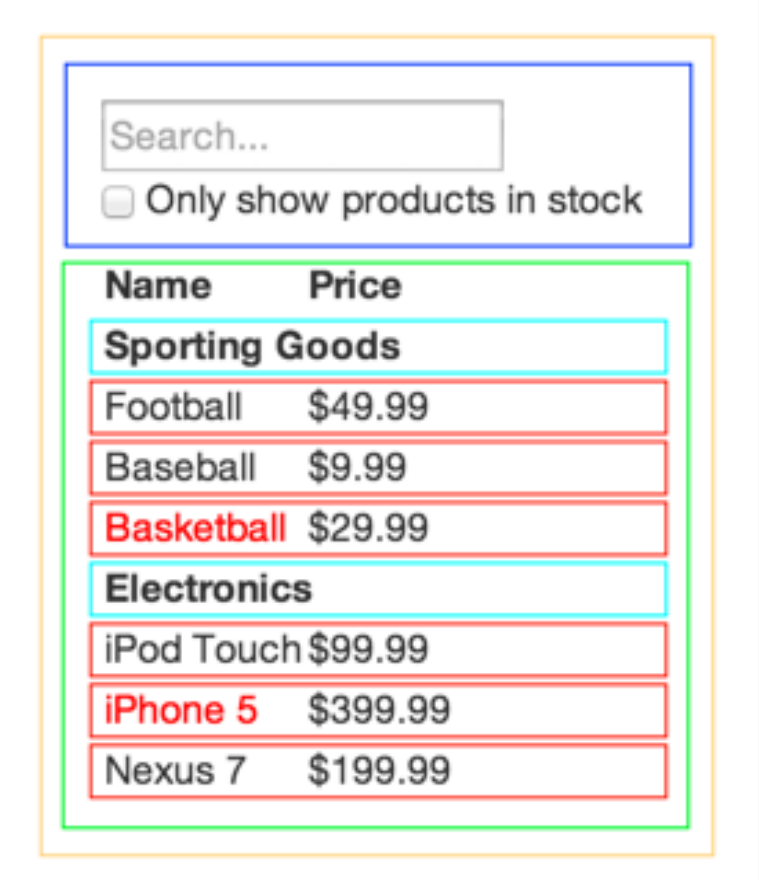

Figure 4: The app components

Thinking in React

A complex app like product search is broken down into smaller components.

Those Components can then be further divided

Thinking in React

READ THIS ARTICLE LATER:

https://facebook.github.io/react/docs/thinking-in-react.html

React, learning by example:

`http://buildwithreact.com/tutorial`

React Topics

- Immutability
- JSX
- Elements/Components
- Props/State

Immutability

Immutable variables or objects CANNOT change

- Think of constants / READ-only
- They take on values when created

Mutable variables or objects CAN change

- You can READ and WRITE to these

Q: Why is immutability a good thing?

JSX

Looks like HTML/XML

- It is actually a syntax extension to JavaScript

JSX compiles into JavaScript objects

These objects ensure the output HTML is safe and fast

JSX

```
const element = (
  <h1 className="greeting">
    Hello, world!
  </h1>
);

// compiles into:

const element = React.createElement(
  'h1',
  {className: 'greeting'},
  'Hello, world!'
);
```

JSX

Because this is JavaScript and not HTML, attribute names are slightly different, and use camelCase representation to indicate this is JS.

Examples:

- `className=` (*React*) vs `class=` (*HTML*)
- `onClick=` (*React*) vs `onclick=` (*HTML*)
- *JSX*: `<div className={someVar}>{anotherVar}<div/>`

React Elements

React elements are Immutable

Elements are the building blocks for React Applications

They are best created with JSX instead of

```
React.createElement({...});
```

Q: How do we update elements?

React Components

React components are reusable code composed of elements and other components

Components manage the lifecycle of the UI

- Mounting, updating, unmounting of the component

Components can track changes with state variables

Props/State

All components and elements have props

Props are READ only (immutable) properties set when the object was created

Components have state

State has READ/WRITE ability. Each component can update its state fields

Props

A React component should use `props` to store information that can be changed, but can only be changed by a different component.
A parent can send whatever prop values it likes to a child, but the child cannot modify its own props.

`state` allows a component to maintain some changing values, while props are the mechanism to propagate those values to children components.

State

A React component should use `state` to store information that the component itself can change.

But don't mutate state. Make a copy of state and use `setState()` on that new variable.

Don't change `this.state` directly. Instead use `this.setState(...)`, which also automatically calls render.

Resources:

Links

- Simple React Tutorial
- Immutability in React
- Intro to JSX
- JSX in depth
- TicTacToe Tutorial