Chapter 10

Building Your First News App

Summary: Allowing people to explore data, see the state of the world around them and put them in control of their own news consumption: Those are the ideas behind a news app, and you'll build one.

Skills you will learn:

1. Brainstorming the structures you'll need to build
2. Pulling together code concepts from previous tutorials to bring it together

This tutorial will assume that you have already completed the tutorials on HTML and CSS as well as the JavaScript tutorial. If you have not yet completed those tutorials, it is highly recommended that you go finish them before proceeding.

When building a news app, the goal is always to solve a problem. Whether it's a problem journalists have or a problem your readers have, writing code is another tool to solve that problem.

In this tutorial, we will build an app to help prospective home buyers understand their minimum down payment requirements since new mortgage lending rules came into effect. The app will allow our readers to enter a home price and see the minimum down payment required to purchase that home.

To start off, we'll need to create a blank HTML document and give it a title.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>


</body>
</html>
```

From there, we can start adding the elements to the page that we know we're going to need.

These will include an input field, so our readers can type in their purchase price, and a submit button so we can trigger the calculation.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>

    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">


</body>
</html>
```

Input elements are very versatile. The "type" attribute is what defines most of it. By defining the "type" as "text" it means we create a long text box for our readers to fill in. The "name" attribute is going to come in handy for referencing that field later on. The "ID" is how JavaScript will be able to identify it later and the "placeholder" is what will show up in the text-box to give our readers a sense of what they should enter, but will disappear to accept a real number once your reader starts typing.

The type “submit” is a button that appears after the input, and triggers a JavaScript function when it’s clicked. We’ve called this function “calculate” for now, but it could be called whatever we like, it’s just a name.

From there, we can add some descriptive text to give our readers a sense of how to interact with this tool.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">


</body>
</html>
```

A headline at the top, with a paragraph of text will go a long way to helping them. There’s also a “label” element here that will automatically align itself with the input field it belongs with. The “for” attribute in the label element aligns with the “name” attribute of the input field. If they’re the same, they’ll match up.

Now we need to inject some JavaScript to actually handle the calculation.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>

    </script>

</body>
</html>
```

We're actually going to use two <script> tags here, because we're going to import jQuery for this project.

jQuery is a library of JavaScript functions that allows you to use the expertise of others to make your code easier to write and more efficient. Instead of solving all the problems that other developers have already solved, like how to fade an element into the background, using jQuery means you can just import those solutions ready-made and incorporate them into your code. That's what the first <script> tag does with the long link after it. That link is pointing at another JavaScript file that will now be available to you to use.

The second <script> tag is where we're going to write our code.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>
        function calculate() {
            console.log('fired');
        }
    </script>

</body>
</html>
```

The first thing we're going to do in that segment is add a function called "calculate" that will be run when someone clicks the "submit" button. The "onclick" portion of the submit button and this function name must align with each other or nothing will work.

As written right now, this code will log the word “fired” to the console as soon as someone presses the “submit” button on the page. Give it a try now and see if it works.

Now to write the calculation code. The first thing we need to do is know the price of the house our reader is thinking about buying. So we need to grab the number entered into the input field, and save it to a variable.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>
        function calculate() {
            var price = $('input#dollars').val();

        }
    </script>

</body>
</html>
```

We use ‘var’ to establish a variable, and then we’ll fill in a jQuery function.

The $ indicates that we’re using jQuery. That’s the universal shortcut for “go into jQuery.” Then the portion in parentheses outlines what element on the page we want to act upon. So we have “input#dollars” which means an input element that has the ID of “dollars.” The pound sign is the universal short code in jQuery for an ID. Similarly, you can access classes with a period in place of the pound sign. Then we can use .val() at the end to mean “grab the value of this element.” The value of the element is what the user entered. So if they entered 500000 into the text-field, the value of the element will be 500000.

This means we now have a variable called “price” that contains the value of the house being purchase.

The next step is to calculate the cost of the down payment with a little bit of math. The new rules state that we must have a down payment of at least 20% of the value of the house. So to calculate the down payment, we multiply the cost of the house by 0.2.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>
        function calculate() {
            var price = $('input#dollars').val();
            var downpayment = price * 0.2;

        }
    </script>

</body>
</html>
```

Now we have a new variable called “downpayment” that contains our calculated number. Now we have to show it to our readers.

This will involve adding a new element to the page and entering some text into it.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>
        function calculate() {
            var price = $('input#dollars').val();
            var downpayment = price * 0.2;
            $('body').append('<p>Your minimum down payment on this house is: $' + downpayment + '</p>');
        }
    </script>

</body>
</html>
```

The $('body') means we're going to act on the body using jQuery. The .append() means we're going to append a new element to the end of the body element of our HTML file. The text inside the append() parentheses is what we want to append.

This is, of course, written in HTML because we're actually appending code to our file. So it's a new paragraph element that we're adding to our file, and it has opening and closing <p> tags to make sure it's all coded properly.

Inside those <p> tags is a sentence to explain to our readers their result.

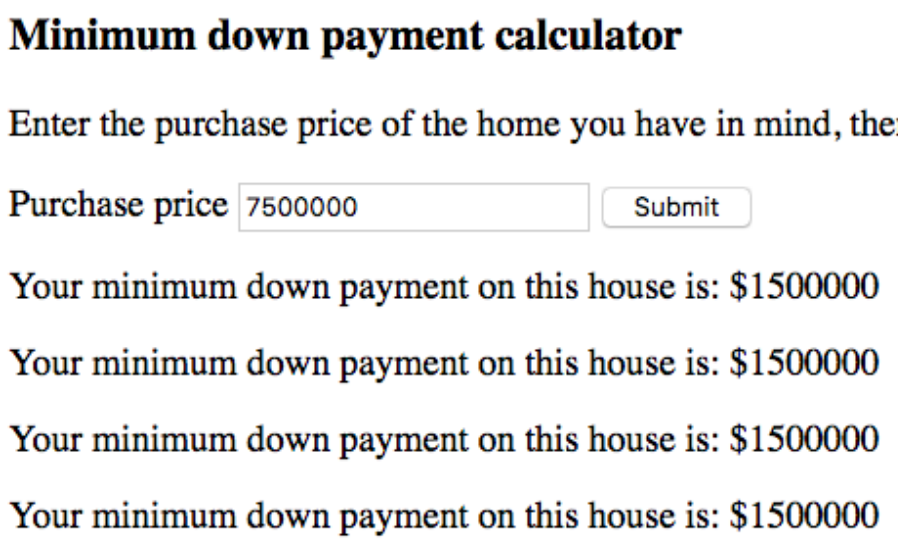
```
'<p>Your minimum down payment on this house is: $' +
downpayment + '</p>'
```

As in Python, you can concatenate text with variables. That means you can string together blocks of text and insert data from a variable into the middle of it.

In this case we're saying the down payment on the house is: then we have a dollar sign, because houses are purchased with dollars in this scenario, then we close off the text string by closing the quotation mark. Then a + sign means we're not done yet and we're going to add more text. But this bit of text is going to be our calculated number from our math operation one step earlier.

Once the downpayment number is in our text field we add another + sign and close the tag properly and then close the parentheses to finish the append() function.

Now save your file and give it a try.



But there's one problem. Hitting the "submit" button more than once delivers the same results over and over again. We need to accommodate for people using our tool more than once. So back to the drawing board to continue adjusting.

We need a way to delete the old tag when the submit button is pressed a second time. To do that, we need a way to identify it on the page. Let's give it a class of "results."

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>
        function calculate() {
            var price = $('input#dollars').val();
            var downpayment = price * 0.2;
            $('body').append('<p class="results">Your minimum down payment on this house is: $' + downpayment + '</p>');
        }
    </script>

</body>
</html>
```

This new <p> tag's class is going to be in the last line of JavaScript where we add it to our page.

So now, right before that operation, let's remove all <p> tags that have a class of results. By removing them all *before* we add a new one to the page, we keep everything nice and clean.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>First News App</title>
</head>
<body>
    <h3>Minimum down payment calculator</h3>
    <p>Enter the purchase price of the home you have in mind, then click "submit".</p>
    <label for="purchase-price">Purchase price</label>
    <input type="text" name="purchase-price" id="dollars" placeholder="500000">
    <input type="submit" value="Submit" onclick="calculate()">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js"></script>
    <script>
        function calculate() {
            var price = $('input#dollars').val();
            var downpayment = price * 0.2;
            $('p.results').remove();
            $('body').append('<p class="results">Your minimum down payment on this house is: $' + downpayment + '</p>');
        }
    </script>

</body>
</html>
```

The $ again means we're going to use a jQuery shortcut. This time we're selecting <p> tags that have a class of "results." That's what the period between p and results

means: It's short hand for "with a class of." Then the .remove() method does exactly what it says it does. It removes that element from the page.

Now that problem is solved and we have a nice clean page to work with every time we hit submit.

Congratulations, this was your first simple news app using HTML and JavaScript. You can go back to the lessons we learned with CSS to add some nicer styles to it and make it more pleasing for your users to explore, too.