Extra Notes - Web Development - Time

- Dr Nick Hayward

Basic Timestamps

A basic example of using timestamps with JavaScript, and rendering to HTML.

Contents

intro

Time and timestamps in JS, and other development languages, refer to a specific date,

- 1st January 1970 00:00:00 (UTC)

This is commonly referenced as the **Unix Epic**.

So, a timestamp in milliseconds can be positive or negative relative to the starting point of zero. e.g.

- 0 = 1st January 1970 00:00:00 (AM)
- -1000 = 31st December 1969 11:59:59 (PM)
- 10000 = 1st January 1970 00:00:10 (AM)

and so on.

get timestamp with JS

We can get the current timestamp in milliseconds using JS's built in constructor and method, e.g.

```
var currentTime = new Date().getTime();
```

However, this return value will need to be formatted in various ways to present a meaningful time and data to a user.

MDN resources - Date object

One option for formatting the current value of a timestamp is to use available methods for the `Date`, as outlined in the MDN documentation,

- MDN Global Objects - Date - https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date

There are many available GETTER and SETTER methods to help us interpret, modify, and render a Date object.

date methods and output

We can call various methods on the `Date` object, e.g.

```
// get current date
var currentDate = new Date();
// get current time
var currentTime = currentDate.getTime();
// get current month
var currentMonth = currentDate.getMonth();
```

However, the return for `currentMonth` will simply be a number for the calendar month, starting at `0`.

We may write a custom function to convert this month number, e.g.

```
// format month number to concatenated month string
function monthFormatter(currentMonth) {
  const months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'June', 'July', 'Sep', 'Oct',
'Nov', 'Dec'];
  var month = months[currentMonth];
  return month;
}
```

Moment.js library

We can also use the *Moment* JS library to help format times for use in an app. *Moment* has now become the de-facto JS library for this type of utility. Further details are available at the following URL,

- Moment.js - https://momentjs.com/

We can install it using various tools, including NPM, Yarn, Bower &c.

```
npm i moment --save
```

or we can simply download a copy of the JS file for local reference in the app.

Moment usage examples

We can `require` Moment in our app, for Node apps, e.g.

```
var moment = require('moment');
```

A few usage examples include, e.g.

```
// get current date
var date = moment();

// check default date format
console.log(date.format());

// format date - month shorthand version - e.g. Jan
console.log(date.format('MMM'));

// format date - month & year shorthand version - e.g. Jan 2018
console.log(date.format('MMM Y'));

// format date - month & year shorthand version - e.g. Jan 18
console.log(date.format('MMM YY'));
```

Further details are available on the Moment.js website,

- Moment.js Docs - Display - https://momentjs.com/docs/#/displaying/

We can also calculate time differences, time used and so on with methods such as `add()` and `subtract()`. e.g.

```
// use add() method - add 10 years
date.add(10, 'year');
console.log(date.format('MMM Do, Y')); // e.g. Jan 10th, 2028

// use subtract() method = subtract 10 months
date.subtract(10, 'months');
console.log(date.format('MMM Do Y')); // e.g. Mar 10th 2027
```

We can also format and output times, both 12 and 24 hour clocks and use of AM and PM if required. e.g.

```
// output formatted time - e.g, 9:40 pm
console.log(date.format('h:mm a'));

// output formatted time - e.g 21:40
console.log(date.format('H:mm'));
```