Dealing with Time Using `moment`

Time as We Know It

- Right now we interact with time either using the string dates we store in the database, or the Javascript `Date` object
- And the date strings we store in the databse aren't user friendly, and hard to read
- The `Date` object is incredibly unwieldly, and not fun to use
- What we want is an easy way of formatting, altering, and comparing dates

Dealing with Time - Database String

```
// app.js
app.get("/", function(req, res) {
    res.render("home", {
        userTime: req.user.get("createdAt"),
    });
});

<!-- views/home.ejs -->
<div class="user-time"><%- userTime %></div>

<!-- output -->
<div class="user-time">2017-07-07 05:16:10.125</div>
```

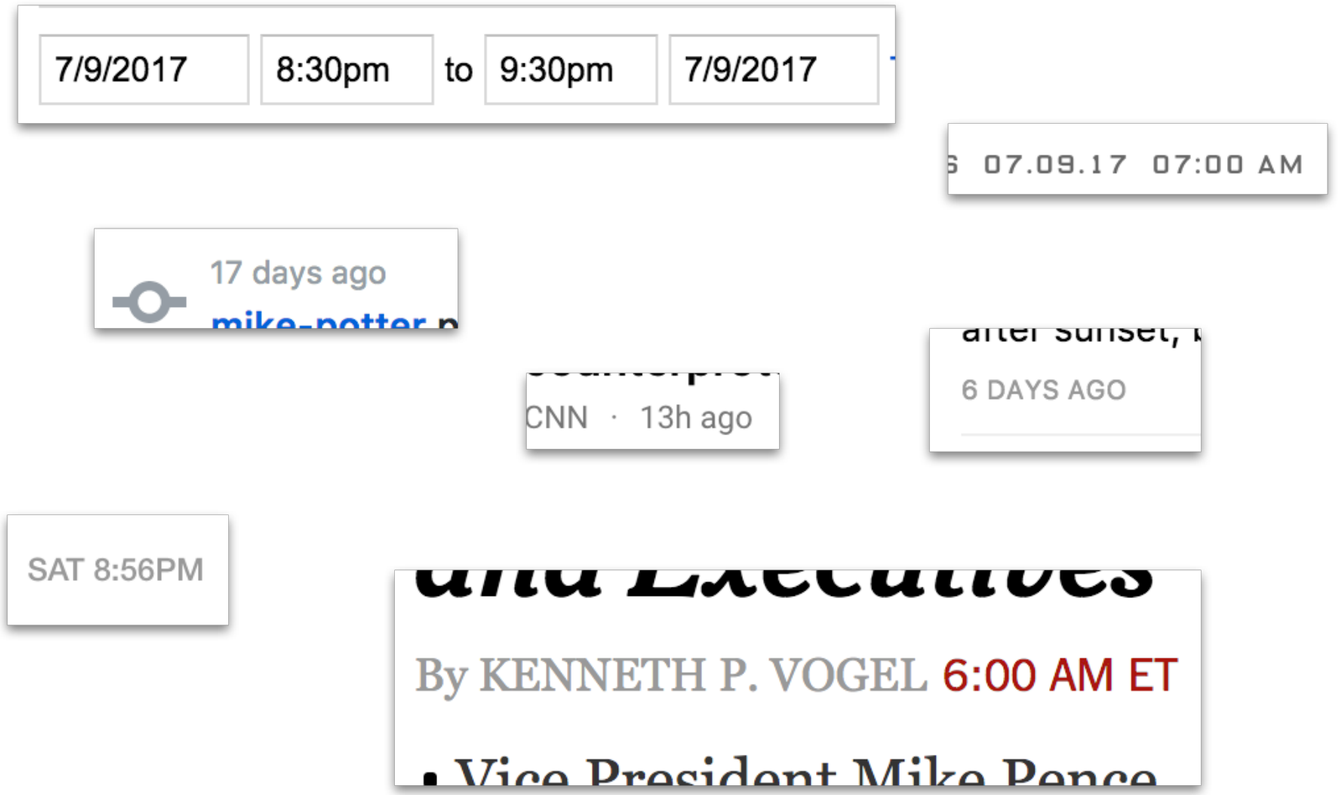
Time as We Know It - Date Object

- Rather than give you a real example of the Javascript Date object, I thought I'd show you a code sample
- This is a gnarly amount of code that only outputs *one format* for time
- Imagine having functions like this for every format you wanted to display time in

```
// util/formatDate.js, taken from
// https://stackoverflow.com/questions/25275696/javascript-format-date-time
function formatDate(date) {
    let hours = date.getHours();
    let minutes = date.getMinutes();
    const ampm = hours >= 12 ? 'pm' : 'am';
    hours = hours % 12;
    hours = hours ? hours : 12;
    minutes = minutes < 10 ? '0'+minutes : minutes;
    const strTime = hours + ':' + minutes + ampm;

    // All this, just to get "7/9/2017 6:30pm"
    return date.getMonth() + 1 + "/" + date.getDate() + "/" + date.getFullYear() + "  " + strTime;
}
```

Examples of Time Output



Moment To The Rescue

- Both of these methods are pretty bad, so fortunately there's a module called `moment` that's great for dealing with time
- Moment can take in almost any standard date format, and output almost any display format
- It uses a friendly template syntax for defining how to output dates
- It also provides a ton of utilities for altering and comparing times

```
# In your terminal
npm install --save moment

// In javascript
const moment = require("moment");
```

Moment - Inputs

```
const moment = require("moment");
let time;

// Current time
time = moment();

// Date object
time = moment(Date.now());

// Database-style time
time = moment("2017-07-07 05:16:10.125");

// Unix epoch time
time = moment(31499429958686);

// Custom formats too!
time = moment("03/09/1997", "MM/DD/YYYY");
```

Moment - Outputs

```
const moment = require("moment");
const time = moment("1977-08-05 15:25:50.228");

// "Sun, 3PM"
time.format("ddd, hA");

// "Aug. 5, 1977"
time.format("MMM. d, YYYY");

// "Sunday, August 5th 1977, 3:25:50 pm"
time.format("dddd, MMMM Do YYYY, h:mm:ss a");

// "1977-08-05T03:25:50-05:00" (Not very useful)
time.format();
```

- Given a template string, output the moment time as a string
- See more about template strings at the end of the slides

Moment - Relative

```
const moment = require("moment");
const oneHour = 3600000;
const oneDay = oneHour * 24;
const lastWeek = Date.now() - oneDay * 7;

// "in an hour"
moment(Date.now() - oneHour).to();

// "in 2 hours"
moment(Date.now() - oneHour).to(Date.now() + oneHour);

// "a day ago"
moment(Date.now() - oneHour * 24).to();

// "20 minutes"
moment(Date.now() - oneHour / 3).to(true);

// "Last Saturday at 11:00am"
moment(Date.now() - (oneHour * 24 * 6)).calendar();
```

- We can do relative times using `.to(time, removeSuffix)`, which is great for knowing how long until something, or how long ago
- We can also use `.calendar(time, formats)` for more precise relative time

Moment - Operations

```
const moment = require("moment");
const time = moment();
const twoWeeks = 3600000 * 24 * 14;

// "in one week"
time.add(7, "days").to();

// "a week ago"
time.subtract(twoWeeks).to();

// "20 years ago"
time.set("year", 1997).to();

// "Jan 1st 1997"
time.startOf("year").format("MMM d YYYY");

// "January 31 '97"
time.endOf("month").format("MMMM D 'YY");
```

- We can `add()` and `subtract()` time from moment objects, or directly `set()` one or many of its values
- We can also jump to `startOf()` or `endOf()` of units of time
- All of these "mutate" the object, changing its time value forever, so be make a new `moment()` instance if you want to keep the old time!

Moment - Queries

```
const moment = require("moment");

// true
moment().isBefore(Date.now() + 100);

// false
moment(Date.now() - 1).isAfter();

// true
moment("2017-06-01").isBetween("2017-01-01", "2017-12-31");
```

- In addition to displaying times, moment also makes doing logic with time easy
- `isBefore()` and `isAfter()` both determine if the moment object is before or after
- `isBetween()` takes two dates, and determines if the moment object is between them

Using Moment

- Moment does all of what we just saw, and a ton more
- It can be great both on the server logic side, and on the template display side
- Feel free to send a `moment` object to your templates, and let them determine how time should be formatted
  - But don't `require` moment in your templates, send the object instead!
- Be sure to check out the moment documentation to see more of what you can do with it

Additional Reading

- moment guides - Easier to read guides on the basics
- moment docs - Parsing - List of all formats moment can parse
- moment docs - Display - List of all moment methods for display
- moment docs - Queries - More about conditionals and logic using time

Challenge - Codepen Moment Outputs

- Go to the codepen linked below
- See if you can fill in all of the correct formats!
- You should **only edit the JS**, not the HTML or CSS
- Every challenge should use moment, not just copying the desired output

https://codepen.io/wbobeirne/pen/KqxGPp/left?editors=0010