What we left out

Notes

Web Application Development is a very broad subject. Here are some of the topics we left out, and where appropriate links for further study.

HTTP / REST / Database interaction / Web APIs

We barely started touching on HTTP, but the world of Web APIs and Web Services is very rich.

REST REpresentational State Transfer[1] is a fundamental approach to how we would represent resources and how to utilize the HTTP protocol.

CRUD Standing for Create-Read-Update-Delete[2], this suggests a design philosophy that allows our models to only perform 4 standard operations, that via REST translate to specific database actions

Databases There is a rich variety of databases available out there, as well as ways to access them. Some of the popular ones are MySQL[3], PostgreSQL[4], MongoDB[5], Redis[6].

Node Server There are ways to make very effective servers in Node. The Express Framework[7] is a great place to start.

Security

There are a number of security-related issues that we did not address at all. In truth, it is best that you follow a proper course on cryptography security, as there are many nuances that are easy to get wrong, with disastrous consequences.

Sessions There are different ways to maintain session information, via cookies and other technologies.

Password management How to safely transmit and store passwords is a complicated process. A dedicated course on security is recommended.

OATH There are ways to utilize authentification provided by other sites like Facebook, Google etc, and validate your users through those sites. These days this goes under the name of OATH 2.0[8].

---

[1]http://en.wikipedia.org/wiki/Representational_state_transfer
[2]http://en.wikipedia.org/wiki/Create,_read,_update_and_delete
[3]https://www.mysql.com/
[4]http://www.postgresql.org/
[5]https://www.mongodb.org/
[6]http://redis.io/
[7]http://expressjs.com/
[8]http://oauth.net/2/

HTTPS Network traffic sent via HTTP is fully readable by anyone monitoring the network. There are numerous ways to encrypt this information, HTTPS being one of the most popular ones.

Javascript parts

In our discussion of Javascript as a language, we left out a number of important parts.

WebWorkers WebWorkers[9] technology essentially allows a certain degree of threading. You can send a request over to a "parallel" worker, and continue your normal operation while waiting for the response, all the while never leaving the browser.

WebSockets WebSockets[10] allow a server and a client to establish a more direct and permanent connection than HTTP would allow, and to exchange efficiently many quick and small messages.

MVC Frameworks There are a number of frameworks that provide much of the basic model-controller functionality needed in Web Applications. A great starting point is to look at a basic TODO App[11] implemented in each of these frameworks.

Promises Programming for the web is a fundamentally asynchronous operation. An increasingly popular way to deal with the challenges arising from this asynchronicity is to employ a library providing promises[12]. Promises are essentially operations that will be resolved in the future, but that you can still pass around as if they were values and you can associate actions to be taken when the value becomes available. The return value of $.ajax is a step in that direction.

Testing

There are a number of issues related to testing, that we did not approach. And a lot of great resources out there.

Mocks/Spies Many libraries allow you to easily create mock objects or spy on certain functions. SinonJS[13] is but one such framework.

Browser Automation There are testing frameworks out there that allow you to automate the testing of web pages. Some worth learning about are PhantomJS[14] and BusterJS[15]. TravisCI[16] and Selenium[17] are also worth a look. And many more.

[9] https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers
[10] https://developer.mozilla.org/en-US/docs/WebSockets
[11] http://todomvc.com/
[12] https://promisesaplus.com/
[13] http://sinonjs.org/
[14] http://phantomjs.org/
[15] http://docs.busterjs.org/en/latest/
[16] https://travis-ci.org/
[17] http://www.seleniumhq.org/

Other Stuff

CoffeeScript[18] Compiles into Javascript, and is otherwise often easier and nicer to write in.

TypeScript[19] An extension of Javascript with static typing.

Sass[20] **and Less**[21] Both allow you to write CSS in a more modular way, more like programming.