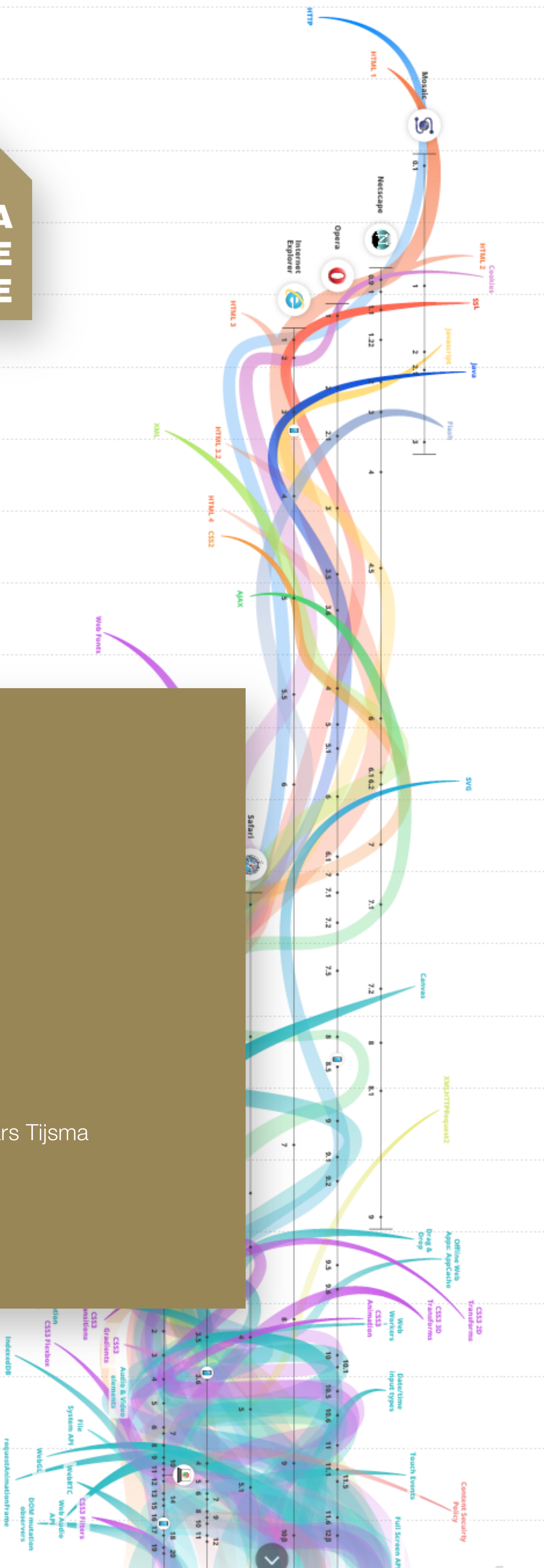
ICA

INFORMATICA
COMMUNICATIE
ACADEMIE



courses guide

develop a web application

server-side web development

client-side webdevelopment

....

year:	2021 - 2022
opleiding:	Informatics
authors:	Robert Holwerda, Sander Leer & Lars Tijsma
version:	1.0

....

HAN

image source: http://www.evolutionoftheweb.com/

table of contents

1. introduction

In the first half of the DWA semester you are going to follow two courses:

- **Client-side Web Development** (CWD). This course focuses on the code you write that runs in a browser.
- **Server-side Web Development** (SWD). This course is mainly about code that runs on a web server.

Actually, not all subjects that are important for Web development fall neatly into one of these categories. For example, we'll use JavaScript as the programming language for both courses, but the introduction to the language is part of SWD, even though CWD is just as suitable to host the JS intro. Websockets and Rest are other subjects that are important to both server-side and client-side development.

This Course Guide describes how both courses are organized, and what we expect from you, and what you can expect from us in return.
We wish you a fun and productive time.

For any remarks, suggestions or complaints about this guide in particular, please contact the maintainer of this document, Lars Tijsma (lars.tijsma@han.nl).

2. server-side web development

There are many programming platforms for building the server code for a website. In this course, we focus on the platform that has become very important and popular in the last few years: Node.js, a platform for using JavaScript on the server. In contrast to many other platforms, a Node.js web application does not need a separate web server to host it: your Node.js program will *contain its own* HTTP server, included as one of the built-in libraries of Node.js. We'll combine the basic HTTP libraries of Node.js with the Express framework which is used by most Node.js developers for structuring complex web applications.

By including other libraries, a Node.js app can be a server (or client) for many other protocols too, such as FTP, or (more interestingly) the new WebSocket protocol. WebSocket is, officially, an extension to the HTTP protocol that underlies the entire World Wide Web, but it is also a radical departure from how HTTP works: Instead of browser only sending request for pages or images, and the server waiting until a request comes in, with WebSockets, a server can send info to the browser without the browser having asked for it. This allows for completely new kinds of web applications, and these new kinds of web apps have been easier to build with Node.js than with many other mainstream web platforms.

You will also meet a new kind of database in SWD. Most databases in use today are relational databases: records in tables, linked by foreign keys and queried using SQL. But several new kinds of databases promise to be faster and easier to program with, and one important class of databases are *document-oriented* databases. In the Node.js world, these databases are used often, and MongoDB is the most important one, partly because it is very much a JavaScript database: the data structures in MongoDB are

JavaScript data structures, and the language to query and update the data is JavaScript.

Here is a week-by-week overview of the subjects in SWD:

1. Basic JavaScript
2. Asynchronous JavaScript
3. Express
4. Sessions & Rest
5. WebSockets
6. MongoDB & Mongoose

3. client-side web development

In computer science, a *client* is a piece of software that make use of services provided by a server. In web development, the client is almost always a browser (although these days, mobile phone apps are often also clients of HTTP servers). In CWD, we'll focus on building (real-time) single page web apps. These are web applications that seem like a single page to the browser (i.e. the page is never thrown away by fetching a new one from the server), but present a rich, changing and responsive user-interface to the user.

This is possible because browsers allow JavaScript code to change the page by manipulating it through something called the Document Object Model (DOM), and because JavaScript code can talk to servers using either HTTP and Ajax, or WebSockets, so these are all subject that web developers must be able to work with.

Browser-based JavaScript application can get very complex, and much effort is spent by the front-end development community to create JavaScript libraries and frameworks to make manipulating the browser page easier, and to create some structure to manage the growing complexity of modern web apps. In this course, we'll be looking at two of these libraries/frameworks: React.js (for manipulating the DOM) and Redux (for structuring your app).

Finally, we have to pay attention to two subjects that could just as well have been part of the SWD course: authentication (logging in) and Rest (an architecture for communicating data between client-side JavaScript and servers).

Here is a week-by-week overview of the subjects in CWD:

1. HTTP & DOM
2. Higher order functions and ES6 Classes
3. React
4. More React.js
5. Redux
6. More Redux

4. how do the courses work?

Both SWD and CWD use the "**flipped classroom**" style of education. This means in general that most sessions (meetings of students and teacher in a classroom) require the student to prepare by reading articles, watching videos and doing assignments. The session itself will be largely based on the work that the students have handed in, and focus on deepening your understanding of the subject at hand. This means two things that are important:

1. You will not be spending a lot of time listening to teachers lecturing from PowerPoint slides. The classes are much more **interactive** in nature.
2. Doing your **homework** *before* the session in class has to be *required*: If you don't prepare, the session has little to offer you. If, incidentally, personal circumstances made it difficult for you to prepare everything, your teacher will understand if you explain. If, however you fail to prepare your homework too often, you are likely to fail the course: you will not get a passing grade.

units and sessions

In DWA, we're using the term "***unit***" to refer to a week, because weeks run between Monday and Friday, but in DWA, our programme is sometimes scheduled to run between Wednesday and Tuesday, or some other phase-shift of a few days.
All units consist of three **sessions** and a **small-test**. There are seven units to each course. The final week of the course is reserved for doing your final assignment, and some workshops in preparation of the project phase of the semester.

Here is a typical timeline for a typical unit:

1. Your teacher **publishes the unit** online. Each unit describes its learning objectives, what homework you'll do to prepare each session, what'll happen during the session, and what to study for the small-test. Units are published in **Github.com**.
2. You **prepare for session 1**. You'll read articles, watch video's, formulate questions for your teacher (we call this **Q&A**), and/or do some assignments.
3. You **attend session 1.** Your teacher will spend time answering the **Q&A questions** that have been handed in and **discussing the assignments** you've made. Sessions may also contain **demo's** (your teacher demonstrating some programming techniques) and/or **workshops** (in-session time to work on programming assignments).

4. You prepare for session 2, similarly to session 1.
5. You attend session 2.

6. You **study for the small-test.** There is almost never any specific preparation for session 3, except studying for the test.
7. You **attend session 3.** In this session, you get the opportunity to ask the teacher questions about the subject of the test. Other activities may be guest-lectures, or demos of subjects that are interesting but not part of the test.
8. You take **the small-test**. Sometimes, the small-test happens during session 3, but, depending on the schedule, we may have to conduct it at another time.

Q&A questions

Almost every first and second session of a unit requires that you read some stuff, or watch one or more video's. Whenever you study some material, you will be asked to submit (usually) two **Q&A questions**, through some online tool (probably GitHub).
These questions must be unique: you can't submit questions that somebody else submitted before you. But you can ask a question that relates to someone else's question.

You teacher will spend part of the session answering the most important or interesting questions submitted by you and your classmates. We do this for three reasons:

- It gives you control over what will be discussed in class; the teacher talks about things that are relevant to the students.
- For some students, it encourages a more active style of reading or watching, because now you can evaluate each piece of information to see if you want to ask an interesting question about it.
- And yes, it allows your teacher some insight into who has prepared for the session and who hasn't.

It may seem difficult to come up with a question. Sometimes you feel like you've understood everything, so you have nothing that could use some clarification by the teacher. In such a case: think up a question that seeks more, deeper information. Or come up with a *discussion point*: something you did understand, but that you consider strange, stupid or surprising, and you'd like to discuss in class.
At other times, you may feel like you've understood so little that your question will seem silly to your classmates. Please let your teacher be the judge of that. If he thinks the rest of the class won't benefit from discussing the question, he'll seek you out and discuss it in private. But more often than not, he knows that other students are likely to have some misunderstanding about the same subject. In such a case, he will be grateful that you posed a question that allowed him to discuss the issue.

Q&A questions are **required**. You *have to* submit questions (even if you can't attend the session in person). You'll fail the course if you don't submit questions. See the chapter 5 "passing the courses" about this.

the small-tests

Every unit ends with a short written exam about the materials and assignments in that unit. These tests are small: you get 45 min to do them, but almost everyone will be done in 30 minutes, tops.

Because there are seven units for each course, you'll take seven small-test for each course. The combined grade for the small-tests counts as 50% of your final grade for the course. See chapter 5 "passing the courses".

the final assignment

The courses SWD and CWD share a large assignment at the end of the course period. In this assignment you, and a partner, will work on an interesting real-time single page web application for which you'll put almost everything you've learned to good use.

We'll hand out the assignment in the sixth week of the course. The final, eighth, week of the course is reserved primarily for working on the final assignment, although we are planning some workshops there in preparation of the project phase of the semester (see chapter “a note about the project”).

5. passing the courses

To pass either SWD or CWD, you need to get five things right:

- a passing grade for the small-tests of the course (≥5.5);
- a passing grade for the final assignment (≥5.5);
- almost all Q&A questions submitted on time;
- all assignments finished and pushed to your private DWA repository on GitHub;
- you must have attended (almost) all guest lectures.

small-tests

You'll take 6 small test for each course. Your grade will be the average of the 5 best grades. This gives you room to miss one small-test due to circumstances.
Except for tragic and unusual cases, you cannot get a re-take for one or more small-test.

final assignment

This is a single assignment that counts toward both your SWD grade and your CWD grade. You do this assignment in pairs. The assignment will be assessed in week 9, after the final week of the course.
If your final assignment fails, we may give separate grades for the SWD and the CWD parts, allowing you to pass at least one of the courses. It does not really matter though: you'll still fail the other course. You get to re-take an assessment for the final assignment the next time DWA is conducted (most likely in about 6 months).

q&a

Almost all sessions 1&2 of the seven units for each course require you to submit (usually two) questions.
All submitted questions must be unique, and must be **submitted *before* the relevant session**. If you miss three or more questions, you fail the course. You'll be allowed to fix this during the project phase by answering, in writing, all your classmate's questions for the Q&A's for which you handed in too few questions.

assignments

Besides the final assignment, we'll ask you to do quite a few programming exercises as part of the preparations for the course-sessions (and also *during* the sessions). Those **assignments are required**.

You will each receive a *personal private repository* on GitHub. For every assignment (or series of connected assignments), you create a directory in this repo. Handing in an assignment consists of three steps:

1. **Commit** your final version of the code to you repository on your local machine.
2. **Push** your repo to GitHub, making the code available to your teachers.
3. In GitHub, every commit gets its own URL. A final and important step is to **enter this commit-URL into a form** provided with the assignment. This gives us lists of who handed in which assignments, at what time. If you're not on one of those lists, you won't have handed in the assignment.

You hand the assignment in *before* the relevant session. If, due to circumstances, you are late with an assignment, your teacher may allow you to hand it in somewhat late. If, however, too many of your assignments are late, you will fail the course.