heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

01 - Introduction

Development environment

heig-vd

What are we going to do in this course ?

- Have fun 🎉
- Prepare our portfolio 💼
- Build production-ready web applications 🚀
- Use modern tools, libraries and techniques 🤓
- Use Javascript on the client and the server

How it feels to learn JavaScript in 2016

No JavaScript frameworks were created during the writing of this article.

https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f

heig-vd

Modern software development

heig-vd

How to Keep Up ?

Javascript community is constantly evolving so it can be hard to keep up. If you don't know how here is some suggestions:

- check sites like StateOfJS or BestOfJS
- Watch conferences and meetups recordings like JSConf
- Follow people on Twitter
- Subscribe to subreddits on Reddit
- See what's popular on Codepen
- **Discuss and debate with you friends and colleagues**

heig-vd

What about employers ?

After analyzing open *web developer* positions in the region from Glassdoor using a Node.js script, results are not surprising.

heig-vd

What about employers ?

After analyzing open *web developer* positions in the region from Glassdoor using a Node.js script, results are not surprising.

Most in demand languages
Javascript , Python, HTML/CSS Java, and C++, scala

Most in demand frameworks
Angular, React , Vue.js

Most in demand databases
SQL (MySQL, Postgres, SQL Server) and MongoDB

Others
Experience with Automated testing, Git, webpack, gulp, SASS, etc..

heig-vd

What about employers ?

After analyzing open *web developer* positions in the region from Glassdoor using a Node.js script, results are not surprising.

Most in demand languages
Javascript , Python, HTML/CSS Java, and C++, scala

Most in demand frameworks
Angular, React , Vue.js

Most in demand databases
SQL (MySQL, Postgres, SQL Server) and MongoDB

Others
Experience with Automated testing, Git, webpack, gulp, SASS, etc..

The data was collected and analyzed using tools like gdom, Cloud natural language API and string-similarity

heig-vd

Planning

1 week	HTML5 / CSS3 / JS	Labo
4 weeks	**Github Analytics**	Project
2 weeks	React / Graphql	Labo TE1
6 weeks	**Social App**	Project
3 weeks	Wrap up	TE2

heig-vd

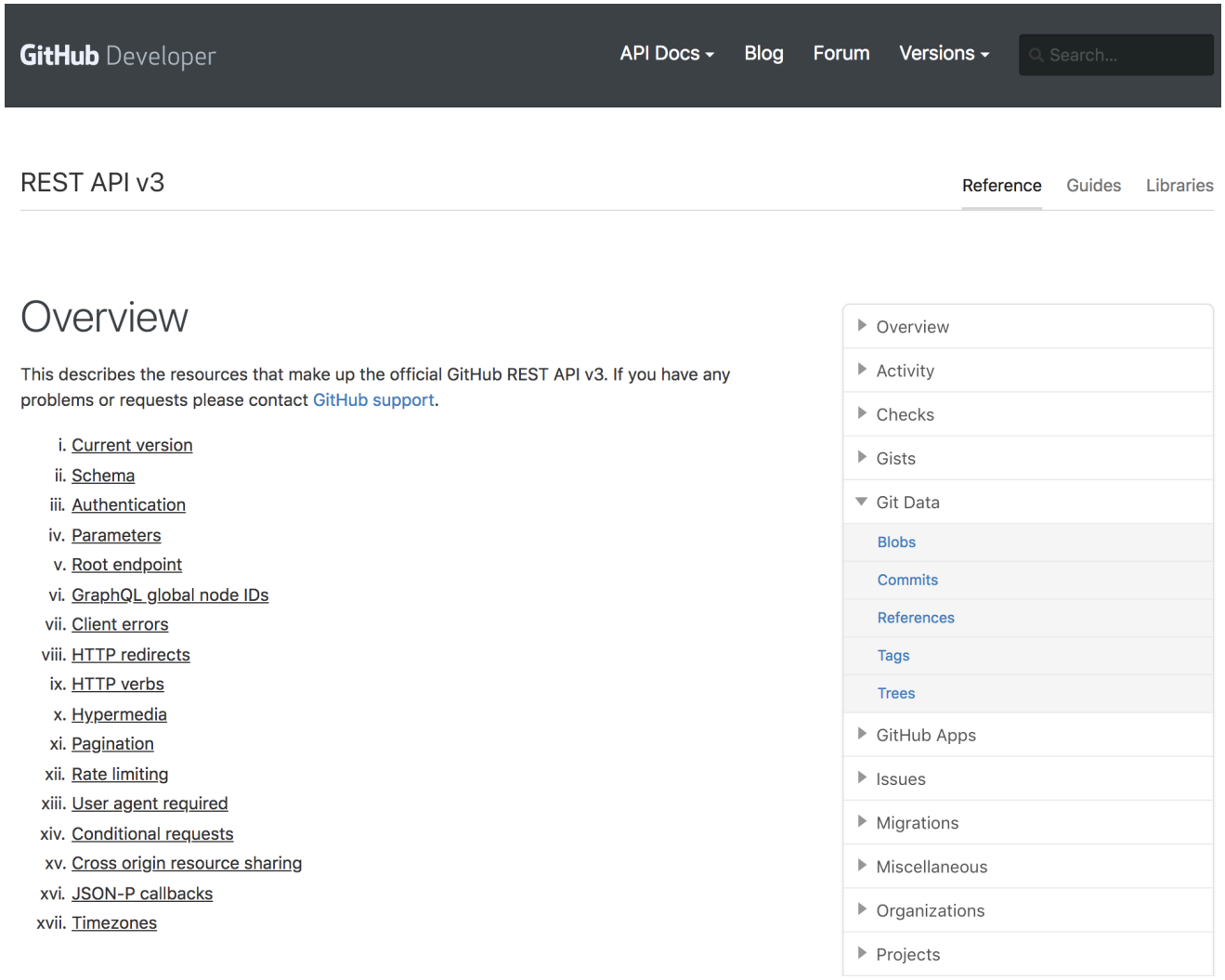
Github Analytics

For this first project, you'll be responsible of **collecting**, **analyzing** and **interpreting** data from Github. Your web application will provide **visualizations** to deliver insights about a topic of your choice.

- Experiment with Javascript
- Work with charting libraries
- Fetch data from a REST API
- Process and save data to MongoDB
- **Innovate and develop your creativity**

Github Analytics

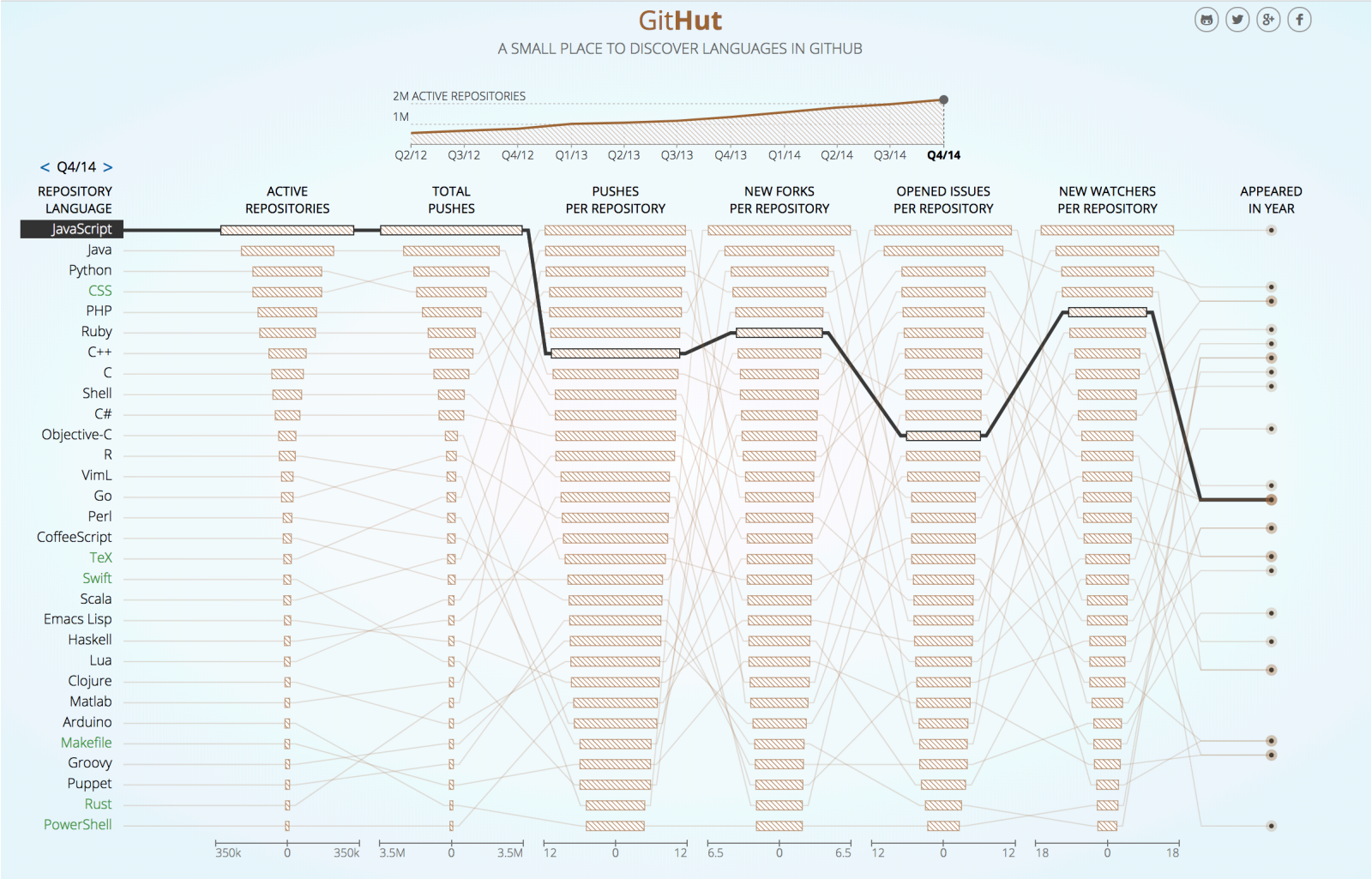
Github REST API



https://developer.github.com/v3/

Github Analytics

Example

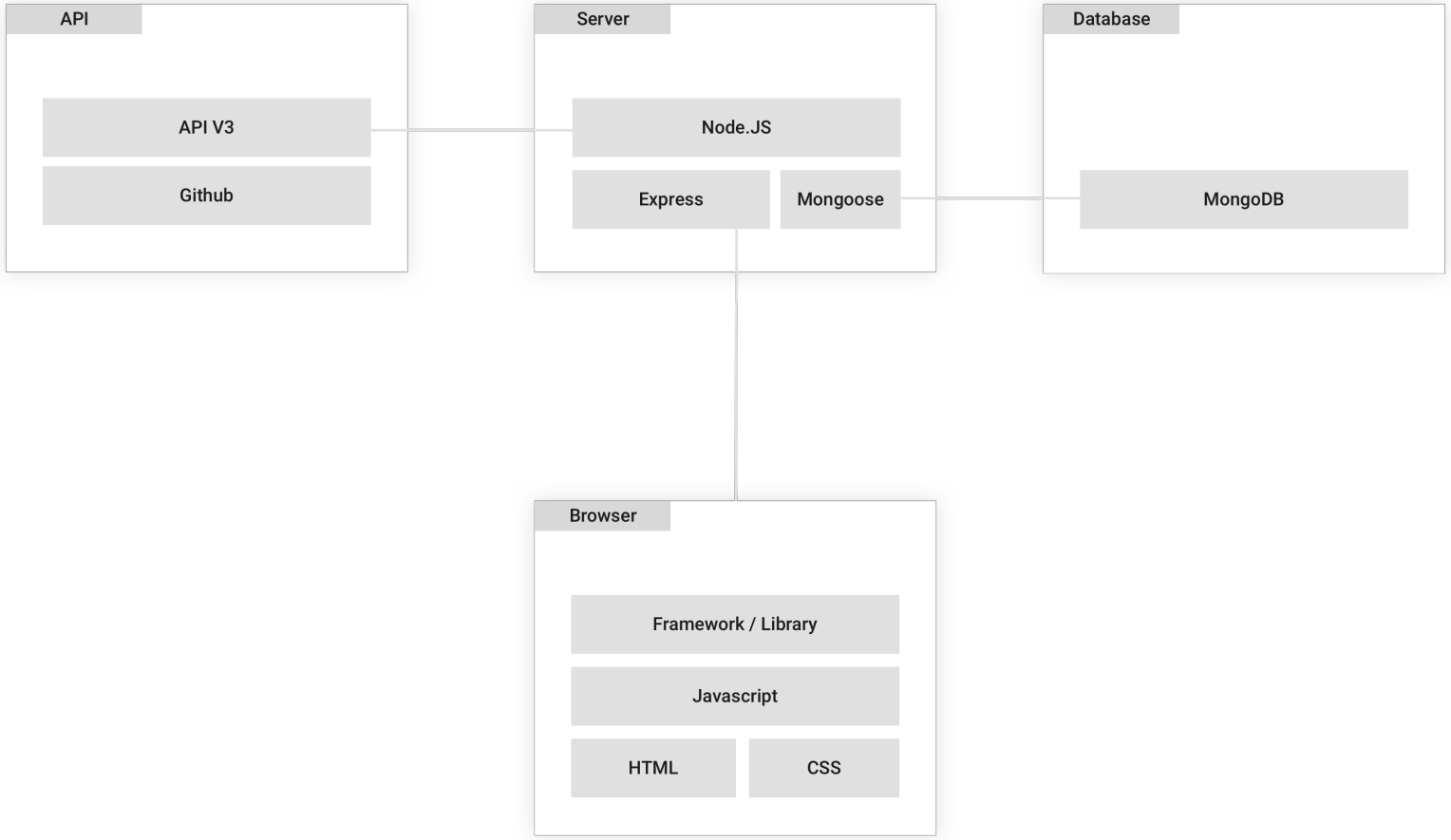


https://githut.info/

heig-vd

Github Analytics

Architecture overview



heig-vd

Big Project

- 2 courses, one project!
  - TWEB: UI, web technologies, REST API (or GraphQL if you insist), deployment
  - MAC: data access part
- Groups of 3
- Social network
- One type of resources (images, videos, books, etc.)
- Authentication, search, sharing, CRUD

heig-vd

Big Project

- Project deployed in real-life at the end
- Votes of people (newbies and developers) we'll choose for a bonus in the final note
- We like creativity and fun stuff, so don't be shy.
- Bonus if you do a special work on architecture, performances, security, ...

heig-vd

Introducing...

...James Nolan! (15.11.2018)

heig-vd

How do you see yourself in this course ?

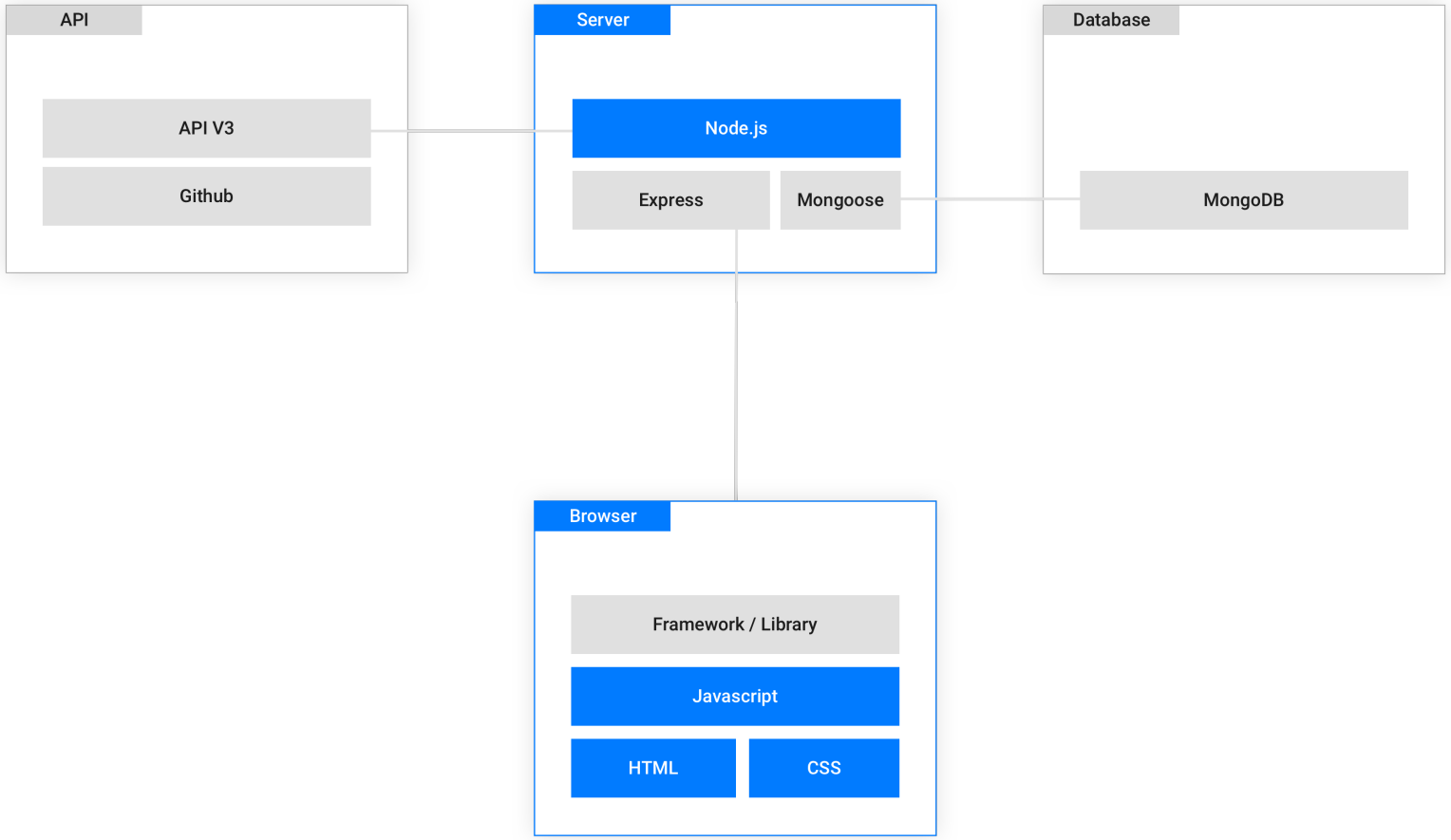
heig-vd

Daily menu

- Setup our development environment.
- Introduce a first set of tools that will make us better and happier
- Study the anatomy of a dynamic web page.

heig-vd

Daily menu



heig-vd

What tools should I use to be a productive developer ?

> I could use a simple text editor to write HTML and Javascript.
>
> I could debug my code with `console.log('argh')`.
>
> I could assemble my web application manually, repeating same tasks over and over again.
>
> I would **lose time** and deliver **lower quality** software.

Solution

- Invest time to make most of your IDE.
- Use a debugger on the client and on the server side.
- Use a build pipeline and take advantage of various automated tasks.

In practice

- Microsoft Visual Studio Code
- Debug client-side code
- Node.js & npm (nvm)
- Debug server-side code
- Let’s start without a build pipeline

Development environment

For this course, you'll need to install the following tools.

- A decent browser with developer tools - Google Chrome is recommended.
- A code editor - we recommend Microsoft Visual Studio Code

Install Node.js

Node.js is an open-source and cross-platform runtime environment used for development of server-side web applications

There is two different ways to install Node.js

1. Go to the download page, get the installer for your system and launch it. Be sure to install the version labeled LTS

2. (**Recommended**) Install Node.js via Node Version Manager (nvm) which is a command line tool that let you install several node versions at the same time and switch between them as you wish.

Install Node.js via nvm

To install or update nvm, you can use the install script using cURL. Copy and past the following command into your terminal:

```
curl –o– https://raw.githubusercontent.com/creationix/nvm/v0.33.11/
```

When the installation has completed familiarize yourself with common commands:

- `nvm current` - Display currently activated version of Node.js
- `nvm list` - List installed versions
- `nvm install <version>` - Install a specific version number
- `nvm use <version>` - Switch to another version

> Note: nvm does not support Windows but you can use nvm-windows as an alternative which has a similar interface.

Install live-server

Along with the node command you also have access to a command called npm. npm stands for Node Package Manager and gives you access to an enormous collection of modules created by the community one of them is live-server.

Live server is a little development server written in Node with hot reload capability. It'll save you time by reloading your page after changes to your files.

Open your **terminal** and type the following command:

```
npm install -g live-server
```

Test live-server

You can easily test your server by creating a directory with an `index.html` file.

```
$ mkdir html-playground
$ cd html-playground
$ touch index.html
```

Then write a simple and valid HTML document. A Valid HTML document contains at least a `DOCTYPE` declaration, `html`, `head` and `body` tags.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML playground</title>
  </head>
  <body>
    <h1>Welcome TWEB 2018</h1>
  </body>
</html>
```

> Note that the structure of your HTML document must be valid for live-server to be able to automatically reload your page.

Test live-server

Now we are good to go. Run the following command from the directory you just created:

```
$ live-server
```

Your default browser should open an page at `http://127.0.0.1:8080/`. Try making changes to the `index.html` file and hit save. You should instantly see those changes reflected in your browser.

> Your page looks ugly ? Don't worry we'll soon make it a bit better.

heig-vd

Debug client-side code

Now we'll see how a set of development tools called Chrome DevTools can increase our productivity.

Chrome DevTools can do a lot - code-coverage and memory analysis, benchmarking, mobile device simulation and the list goes on and on.

For now we'll see the most commonly used features

- Editing any web page on the fly
- Debugging network activities
- Debugging Javascript

You'll discover other tools as you need them when working on your projects.

Setup a project structure

Open the `html-playground` project you just created with Visual Studio Code

> Pro tip: Open a project with Visual Studio Code from your terminal by running `code .` from your project directory.

And setup a basic project structure by adding a script and a stylesheet.

```
 html-playground
 ├── index.html
+├── css
+│   └── styles.css
+└── js
+    └── app.js
```

heig-vd

Write a simple web page

Write a simple page that loads your stylesheet and your script - then launch live-server and open your http://localhost:8080 with Google Chrome.

```html
<!DOCTYPE html>
<html>
  <head>
     <meta charset="UTF-8">
     <title>HTML playground</title>
     <link href="/css/style.css" rel="stylesheet">
  </head>
  <body>
     <h1>Welcome TWEB 2018</h1>
     <script src="/js/app.js"></script>
  </body>
</html>
```

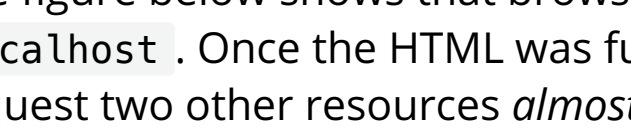
```css
/* css/style.css */
h1 {
  color: blue;
  text-align: center;
}
```

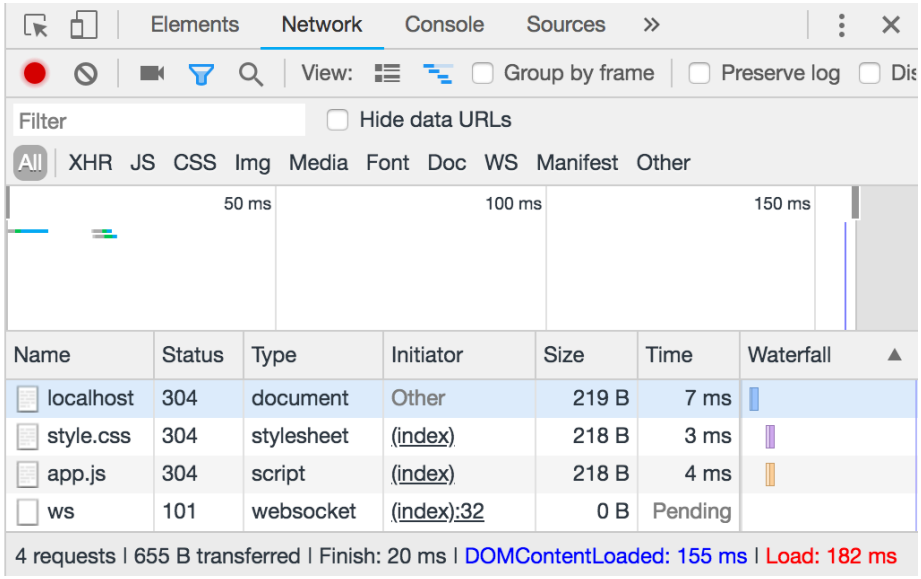
```javascript
/* js/app.js */
var msg = 'hello console';
console.log(msg);
```

Chrome DevTools - Network

Once the page is open, launch Chrome DevTools with a `right-click` on the page > and choose `Inspect`. In the `Network` tab you'll see all requested resources with a lot of useful information.

The figure below shows that browser first requested our HTML file - listed as `localhost`. Once the HTML was fully downloaded and parsed, the browser request two other resources *almost* at the same time - `style.css` then `app.js`.

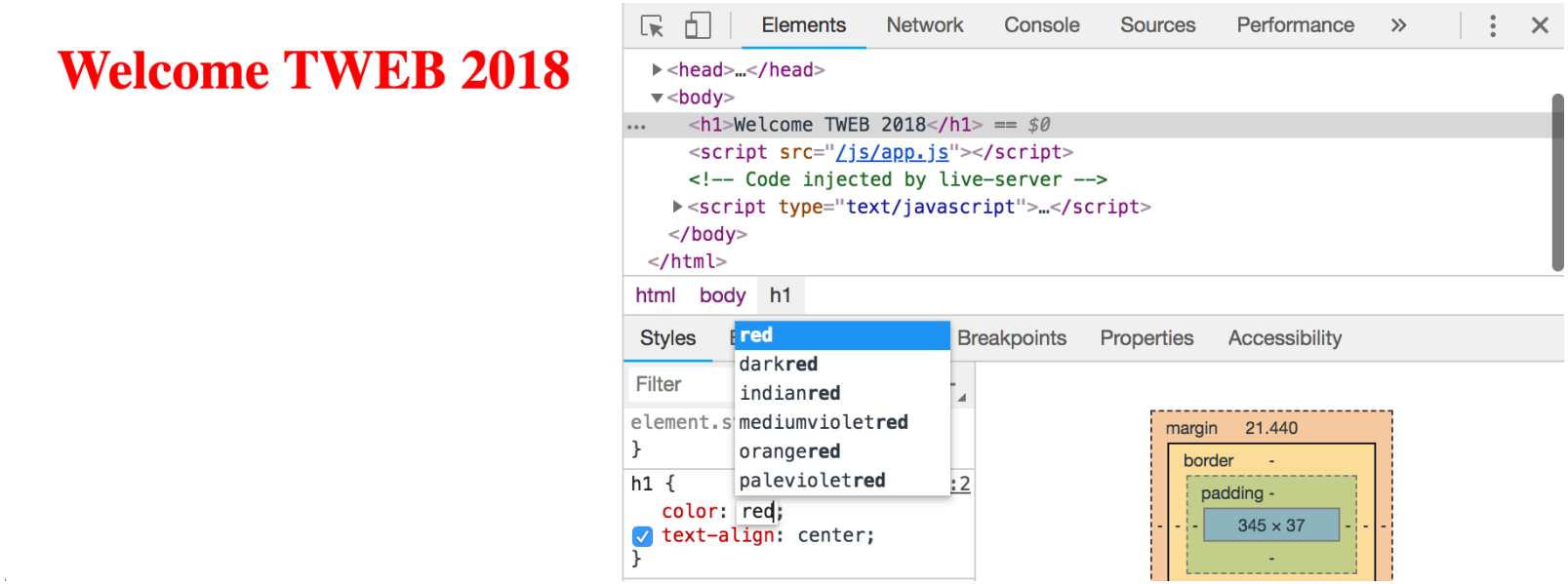




Chrome DevTools - Elements

`Elements` tab is where you can debug your HTML page and CSS. It allows you to see which CSS rules is being applied to any element.

You can also edit your HTML or CSS rules. This is really great for prototyping and debugging.

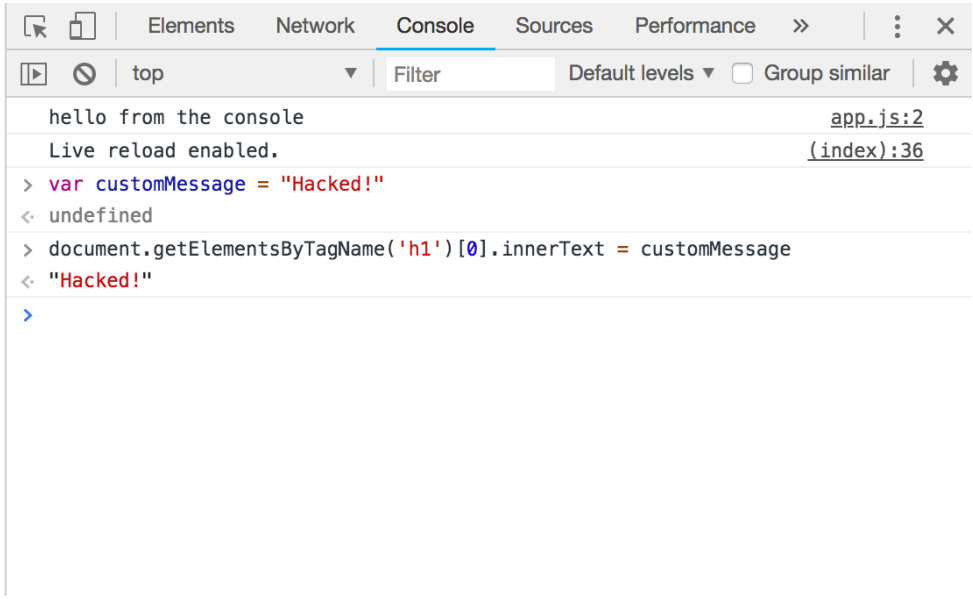


Chrome DevTools - Console

When something doesn't work as expected with your site, your first reaction should be checking the `Console` and see if any error was reported.

In addition to viewing logging and error messages, you can also use the Console to evaluate arbitrary Javascript.

Hacked!

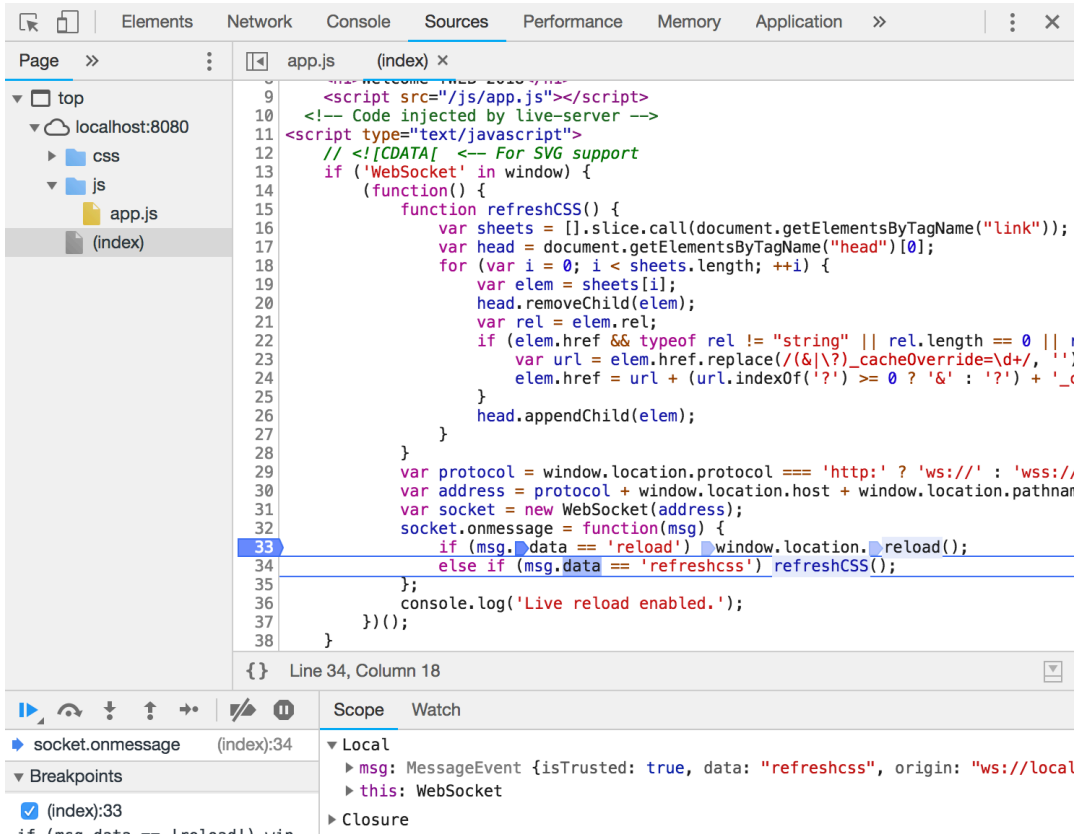


heig-vd

Debug client-side code

Chrome DevTools - Debugger

The `Sources` tab is where you can debug your code using breakpoints. A breakpoint lets you pause your code in the middle of its execution, and examine all values at that moment in time.



Try debugging the code injected by `live-server` at the end of your `index.html` file to understand how it automatically reloads your page

References

- Client-side debugging (Webcast)
  https://youtu.be/qAVGS85SmSg
- Viewing And Changing CSS
  https://developers.google.com/web/tools/chrome-devtools/css/
- Debugging JavaScript in Chrome DevTools
  https://developers.google.com/web/tools/chrome-devtools/javascript/

heig-vd

Debug server-side code

Visual Studio Code has an integrated debugger that you can use to debug your server-side code.

Let's create a boring Node.js server that respond `Hello world` to all requests:

- Create a folder for your project. I call it `node-playground`.
- Run `npm init` from your project directory. This will create a `package.json` file.
- Create an `server.js` file. This is where you'll write your code.

```
node-playground
├── package.json
└── server.js
```

Creating a simple server

No need for any external libraries to create an HTTP server with Node. In this example, we use Node's http module which is loaded on the first line.

- We use the `createServer` function to define a callback that should respond to any incoming request, also known as: `request handler`
- We define a port that the server should listen to for requests.

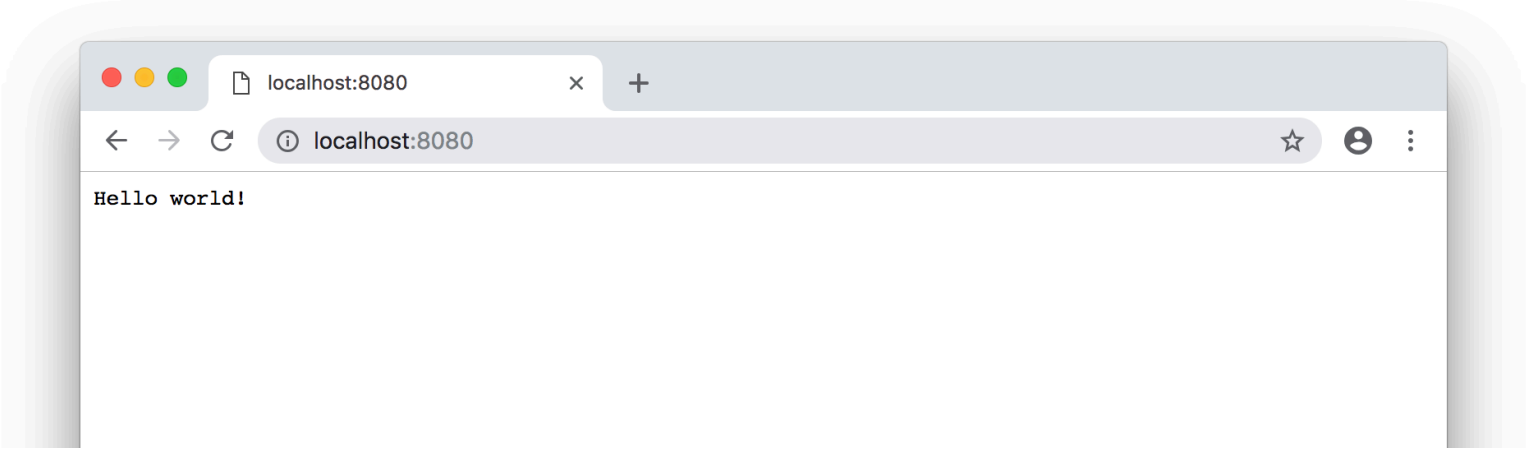
```
// server.js
const http = require('http');
const port = 8080;

const server = http.createServer((req, res) => {
  res.writeHead(200, { 'Content-Type': 'text/plain' });
  res.end('Hello world!');
});

// Start the server
server.listen(port, () => {
  console.log(`Magic happens at http://localhost:${port}`);
});
```

Test your server

To run a node program, use the `node` command and pass in a filename to execute. Start your server by running the following command in your terminal.

```
$ node server.js
```

Then point you browser to http://localhost:8080 to see some magic.



Debugging your server

As with Chrome DevTools, Visual Studio Code lets you add breakpoints to our code and inspect the execution.

- You can debug **server-side** code - Launching your server in debug mode or attaching the debugger to a running process.
- You can also debug **client-side** code - Using the Debugger for Chrome extension.

> But for now, we'll just see how to debug our hello world server.

heig-vd

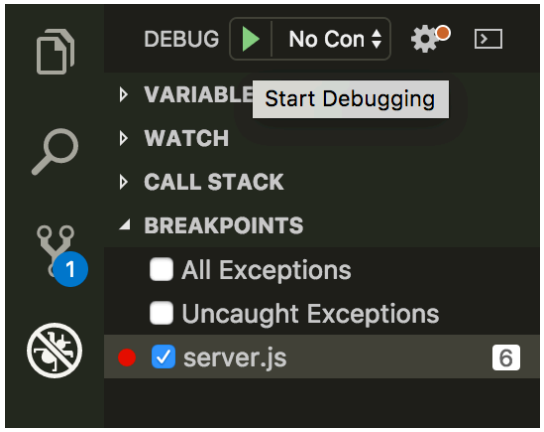
Debug server-side code

Debugging your server

To set a breakpoint in `server.js`, put the editor cursor on a line and press `F9` or click in the editor left gutter next to the line numbers. A red circle will appear in the gutter.

```
5   const server = http.createServer((req, res) => {
6     res.writeHead(200, { 'Content-Type': 'text/plain' });
7     res.end('Hello world!');
8   });
```

To start debugging, select the `Debug View` in the Activity Bar then click on the Debug tool bar green arrow or press `F5`.



References

- Webcast - Setup server-side debugging
  https://youtu.be/YMqIChybZCE
- Debugging in Visual Studio Code
  https://code.visualstudio.com/docs/editor/debugging

heig-vd

How do I know if I write *good* javascript?

> With Java, the **compiler** detects some of my mistakes. With Javascript,I often lose time because of typos and silly mistakes.

> Javascript is evolving a lot and I am not sure if my **coding style** needs an upgrade.

Solution

- Use coding guidelines
- Learn from others
- Enforce the rules in the IDE and in the build process

In practice

- Create your JS project with npm
- Install ESLint
- Configure ESLint
- Launch Visual Studio Code

ESLint

ESLint is a popular linting utility for Javascript that is frequently used to find **problematic patterns** or code that doesn't adhere to certain **style guidelines**

- It provides a level of **clarity** to your code
- It makes your code easier to **read** and **maintain**
- It can analyze your code and warn you of **potential errors**.

When configured and integrated in your code editor, ESLint will warn you as you type. See the following example in Visual Studio Code.

Airbnb JavaScript Style Guide

Eslint was designed in pluggable way to allow developers to create their own linting rules.

Airbnb maintains a very popular style guide which is a set of ESLint rules that you can use and extend.

The documentation explains the reason for their choices for each rule. Here is an example with the rules called: `no-undef` and `prefer-const`.

> Always use `const` or `let` to declare variables. Not doing so will result in global variables. We want to avoid polluting the global namespace. Captain Planet warned us of that.

```
// bad
superPower = new SuperPower();

// good
const superPower = new SuperPower();
```

Install ESLint

There is to ways to install ESLint.

- locally **(recommended)** - including ESLint as our project development dependency. Run the following command from your project directory:

  ```
  npm install --save-dev eslint
  ```

  This will create the `node_modules` directory in your current directory (if one doesn't exist yet) and will download the package to that directory.

- globally - By running this command in your terminal:

  ```
  npm install -g eslint
  ```

  As we did earlier when we installed live-server, the `-g` or `--global` argument will cause npm to install the package globally rather than locally

Configure ESLint

Once installed locally, you can run the `init` command which will help you setup a configuration file.

```
$ ./node_modules/.bin/eslint --init
```

Then answer the questions like bellow to configure ESLint with Airbnb's Javascript Style Guide:

```
? How would you like to configure ESLint? Use a popular style guide
? Which style guide do you want to follow? Airbnb
? Do you use React? No
? What format do you want your config file to be in? JavaScript
? Would you like to install them now with npm? Yes
```

Configure ESLint

After running `eslint --init` you'll have a `.eslintrc.js` file in your project directory. This file includes the line:

```
"extends": "airbnb-base"
```

Which turns on all rules listed in Airbnb Style Guide. You can use this file to customize your rules according to your team code style.

For example, if you fell tired of semicolons (which are required by default in `airbnb-base`) you can customize the `semi` rule to warn whenever they are used:

```
"extends": "airbnb-base"
"rules": {
    "semi": ["warn", "never"],
},
```

Or simply disable the rule with `"semi": "off"`. for more configuration options and details, see the configuration docs

Use ESLint

ESLint comes with a command line utility which you just used to run the `init` step. The following command **runs ESLint on every `.js` files** in the project root directory and prints out all errors and warnings:

```
$ ./node_modules/.bin/eslint **.js
```

This is equivalent to adding an npm script in your `package.json`

```
// package.json
"scripts": {
  "lint": "eslint **.js"
}
```

then run `npm run lint`.

npm will spawn a shell and run `eslint **.js`. The shell environment has your `./node_modules/.bin` folder added to the `PATH` which means any of the dependencies you have that install binaries will be runnable directly.

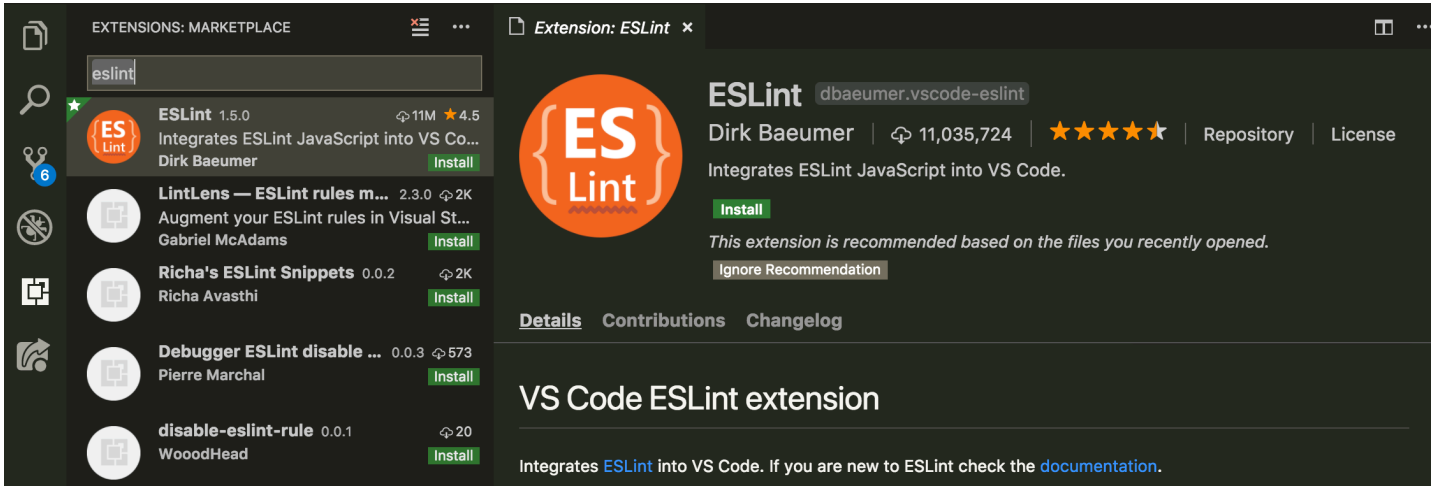
Use ESLint

In general you won't often use ESLint command line interface directly as it'll be redundant.

- The command line interface is generally integrated inside a **build pipeline** to prevent for example developers from deploying low quality code - many projects prevents `git commit` when the `lint` step fails.
- The command line interface can be used by code editors to **highlight errors** as you type or **automatically fix errors** for you. This is what we'll see next by integrating ESLint in Visual Studio Code.

ESLint in Visual Studio Code

Open Visual Studio Code and search for ESLint in the `Extensions` pan. Click on the `Install` button then restart VS Code.



ESLint extension

When the extension is installed - VSCode will look for the `eslint` binary in your project dependencies and the `.eslintrc.js` configuration file to start analyzing your code. And that's it 🎉!

References

- ESLint: configure and test in Visual Studio Code (Webcast)
  https://youtu.be/53IM2NYMH4Q
- Configuring ESLint
  https://eslint.org/docs/user-guide/configuring
- Airbnb JavaScript Style Guide
  https://github.com/airbnb/javascript