

tt()

Schedule

1. Show and tell: dataset & concept
2. Web architecture
3. Libraries, frameworks en bundlers
4. Boilerplate opzetten met Svelte en D3
5. All together!



Schedule

1. Show and tell: dataset & concept
2. Web architecture
3. Libraries, frameworks en bundlers
4. Boilerplate opzetten met Svelte en D3
5. All together!



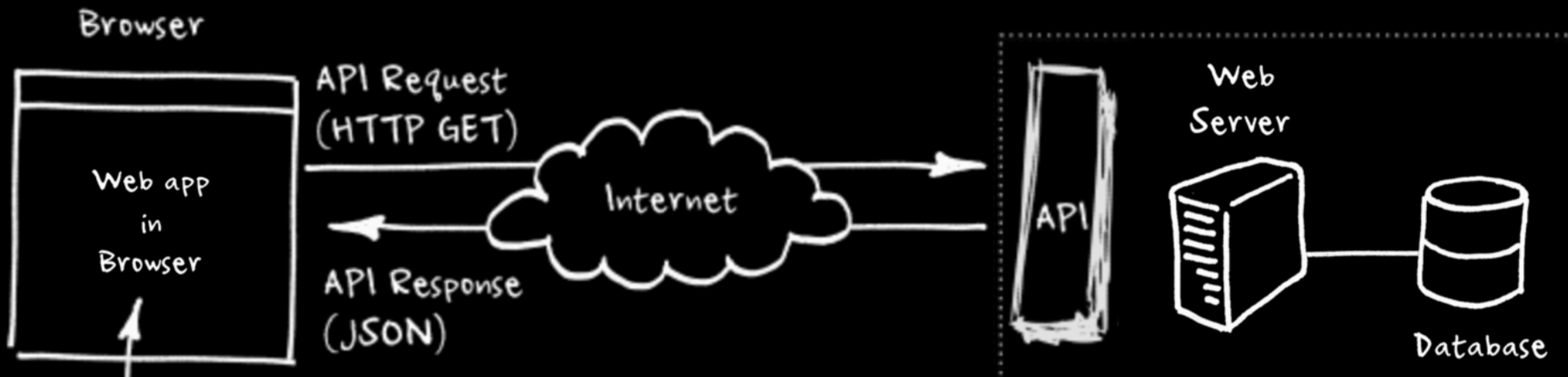


Schedule

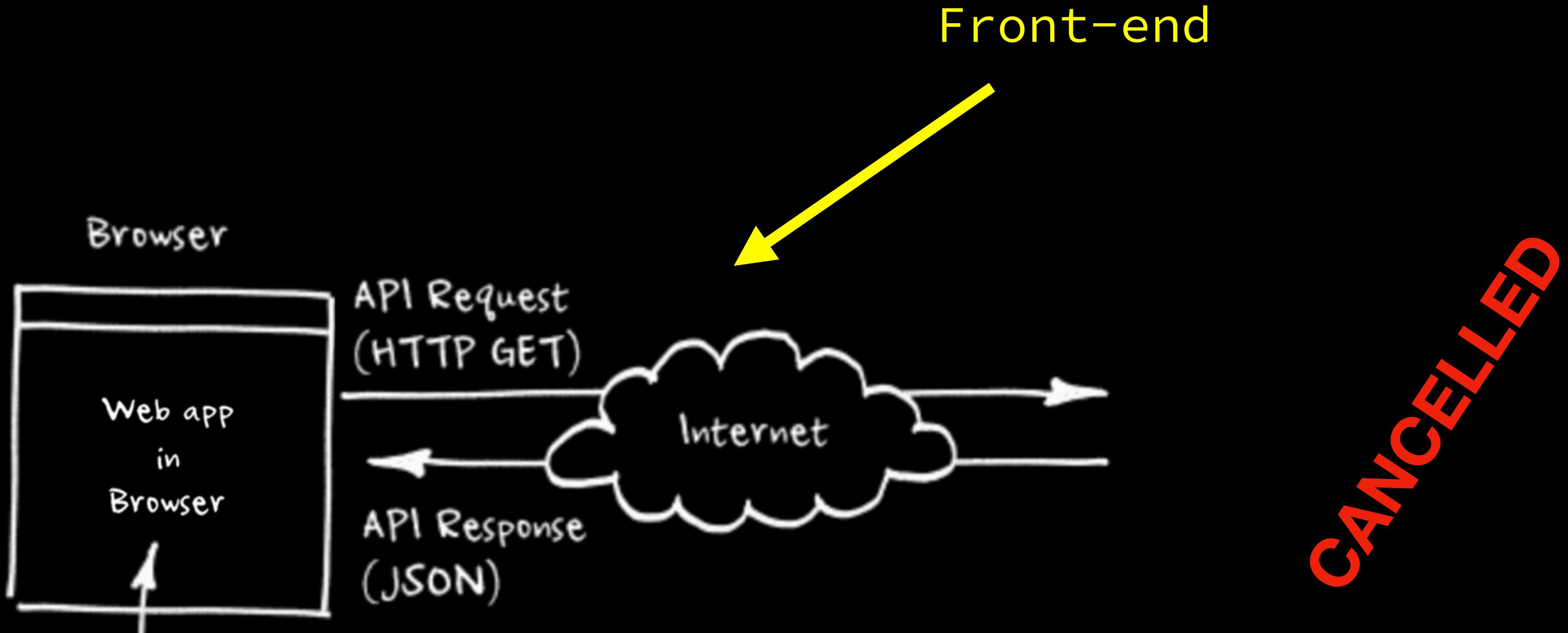
1. Show and tell: dataset & concept
- 2. Web architecture**
3. Libraries, frameworks en bundlers
4. Boilerplate opzetten met Svelte en D3
5. All together!



Web architecture



Web architecture



CANCELLED

Web architecture



Web architecture



```
<style type="text/css" src="styles/main.css"></style>
<style type="text/css" src="styles/partials/header.css"></style>
<style type="text/css" src="styles/partials/footer.css"></style>
<style type="text/css" src="styles/modules/main.css"></style>
<style type="text/css" src="styles/main.css"></style>
```

5(!) http requests?????

Web architecture

```
<script src="d3.js"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/d3@7"></script>
```

Web architecture

- What if the CDN or externally loaded library **goes down?**
- What if you want to work in HTML/CSS/JS **components?**
- What if the library gets **updated with a new major version?**
- What if you want to use additional **(library) plugins?**

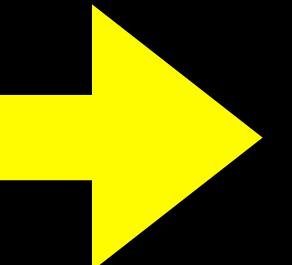
Web architecture

Client-side code

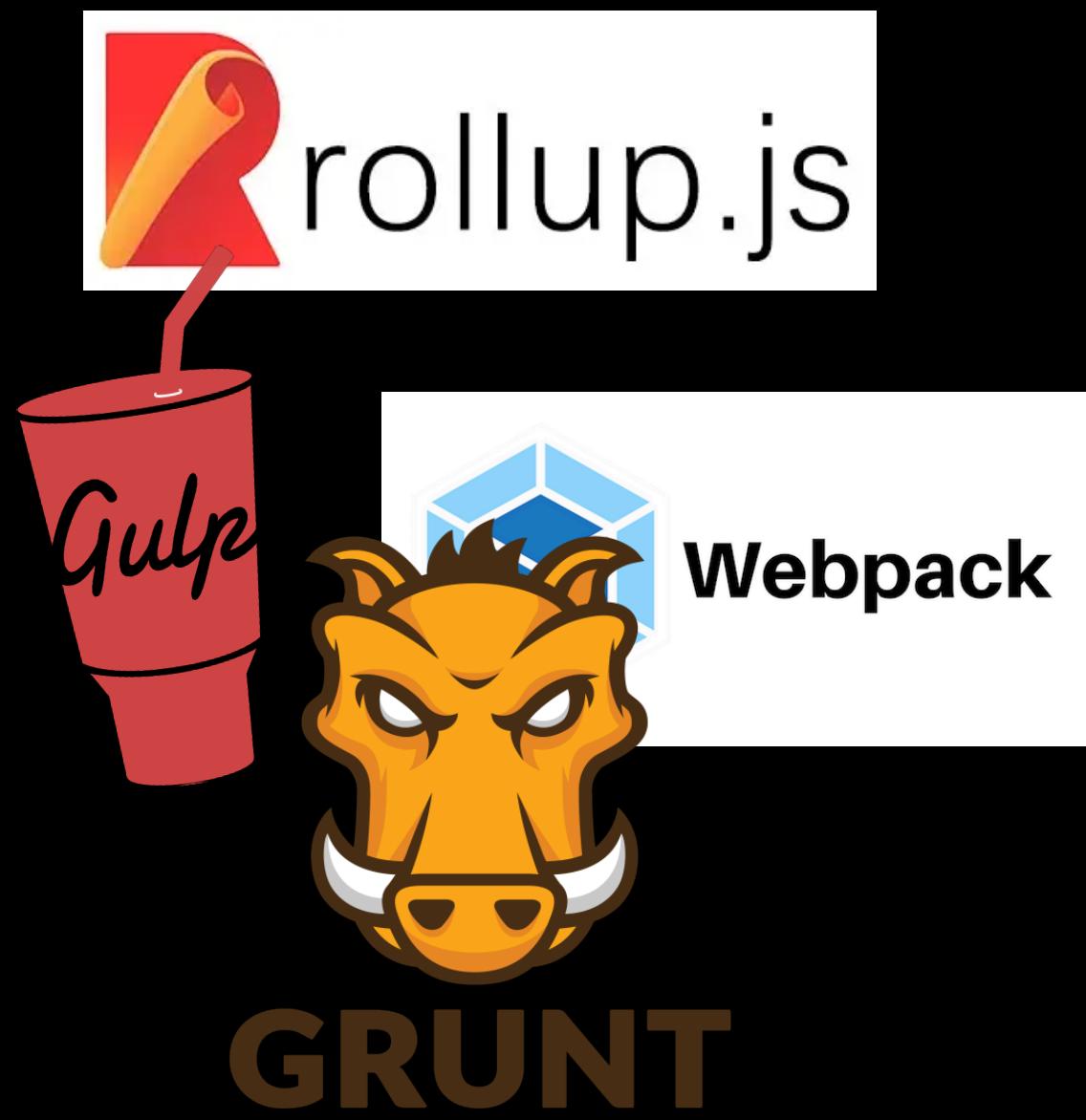


Web architecture

Client-side code



Build tools

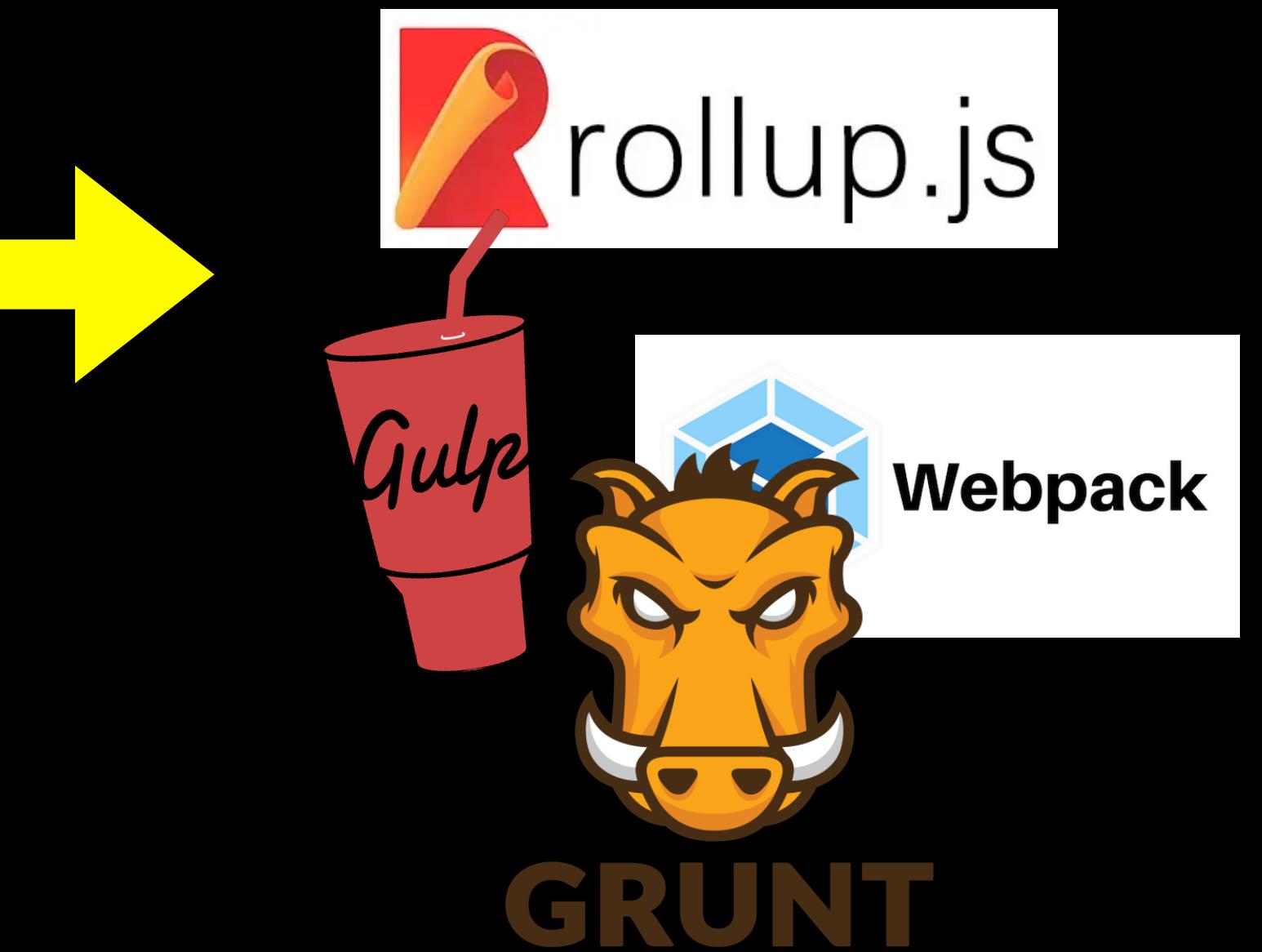


Web architecture

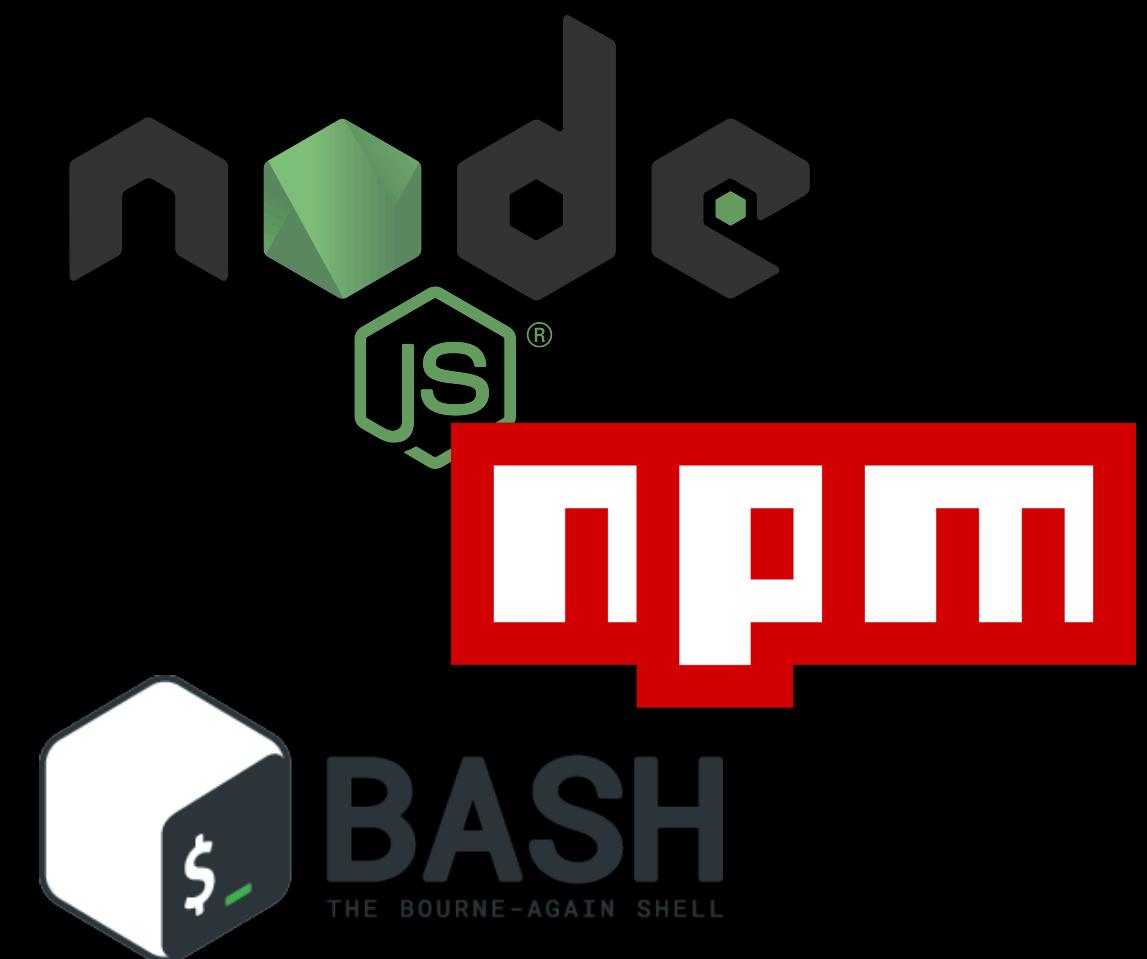
Client-side code



Build tools

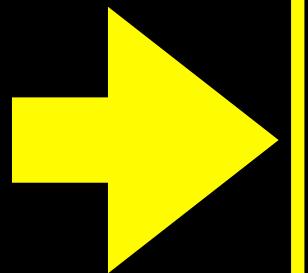


Scripting

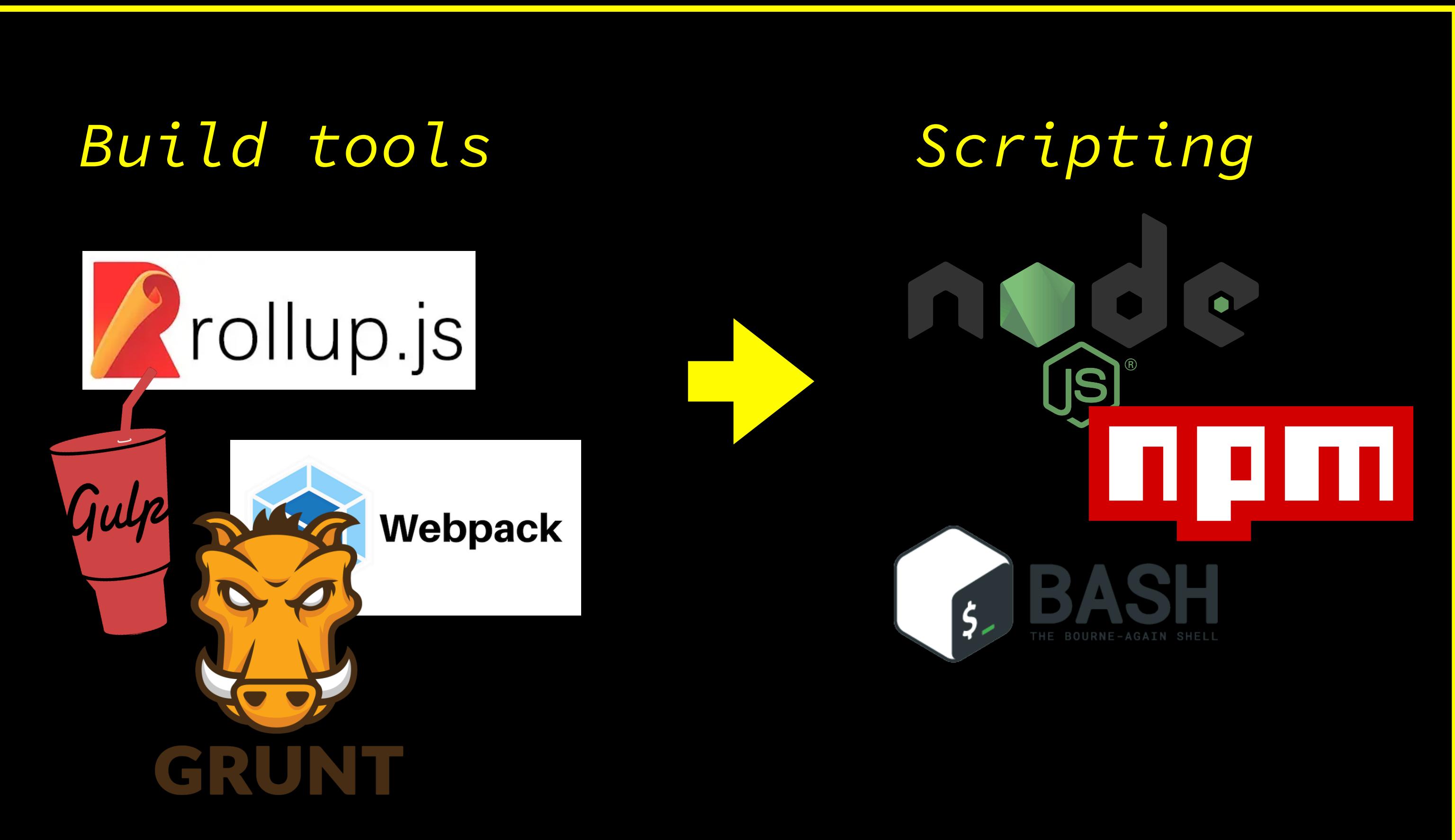


Web architecture

client-side code



Build tools



Scripting

Web architecture



*Om JavaScript op
je PC te runnen.*



*Om packages te
downloaden.*



*Om commando's
uit te voeren.*

Schedule

1. Show and tell: dataset & concept
2. Web architecture
- 3. Libraries, frameworks en bundlers**
4. Boilerplate opzetten met Svelte en D3
5. All together!



Tech stack

A **JavaScript tech stack** refers to the combination of technologies, tools, libraries, and frameworks that developers use to build web applications.

Tech stack

A **JavaScript tech stack** refers to the combination of technologies, tools, libraries, and frameworks that developers use to build web applications.

1. *Library*

2. *Framework*

3. *Bundler*

Libraries, frameworks & bundlers



Library



Framework



Bundler

Libraries, frameworks & bundlers



Library



Framework



Bundler

Library (D3)

A **JavaScript library** is a collection of pre-written JavaScript code that provides specific, reusable functions and utilities to help developers accomplish common tasks more easily. Instead of writing complex code from scratch, you can leverage a library to perform repeated tasks.

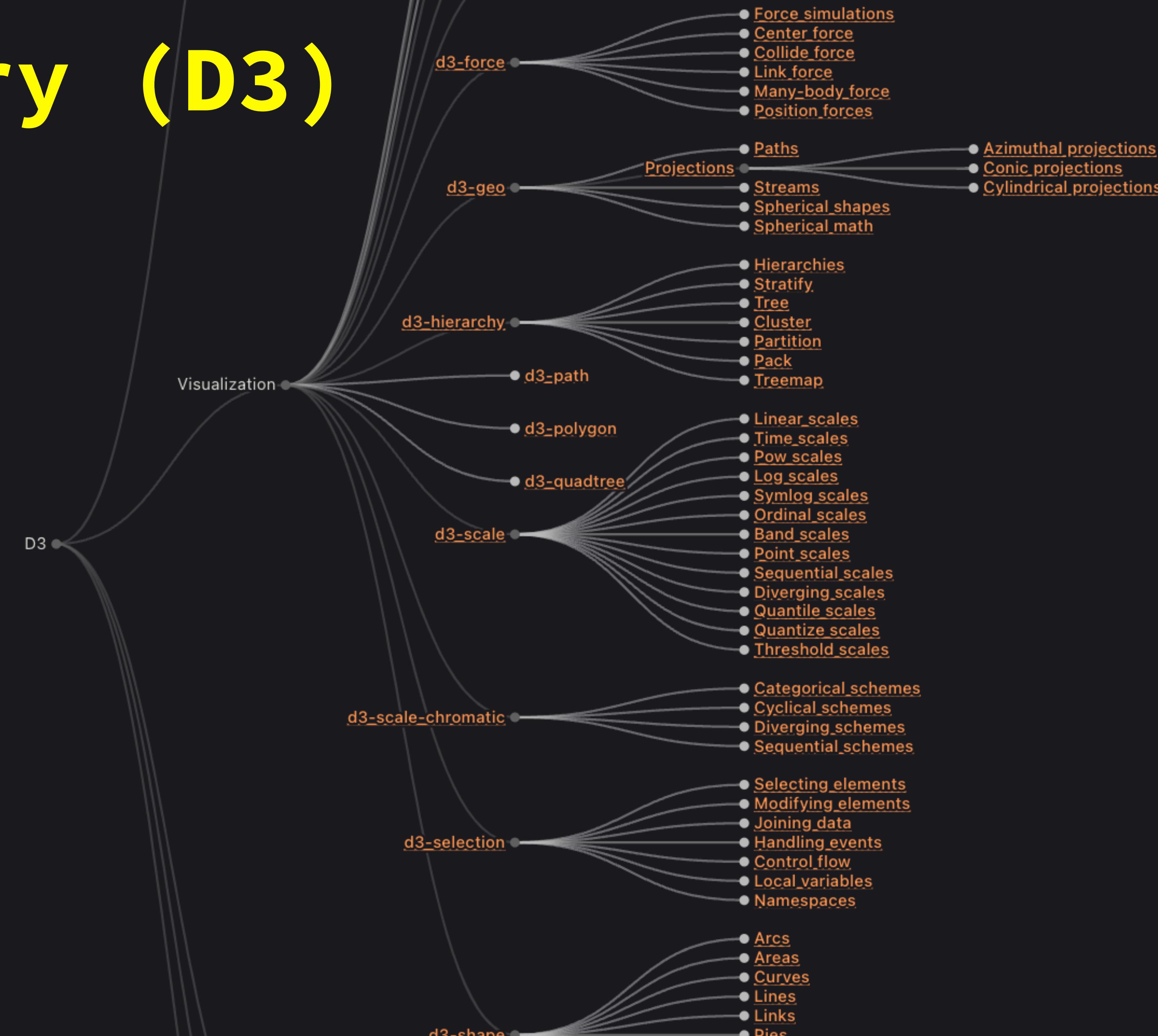
Library (D3)

D3 (or **D3.js**) is a free, open-source JavaScript library for visualizing data. Its low-level approach built on web standards offers unparalleled flexibility in authoring dynamic, data-driven graphics.

Library (D3)

D3 is not a charting library in the traditional sense. It has no concept of “charts”.

Library (D3)



Library (D3)

- Flexibel: er zijn geen ‘chart’ types dus veel customizability
- Standaarden: wat er al op het web is DOM, SVG
- Dynamisch: goed voor interactieve viz want data joining

Libraries, frameworks & bundlers



Library



Framework

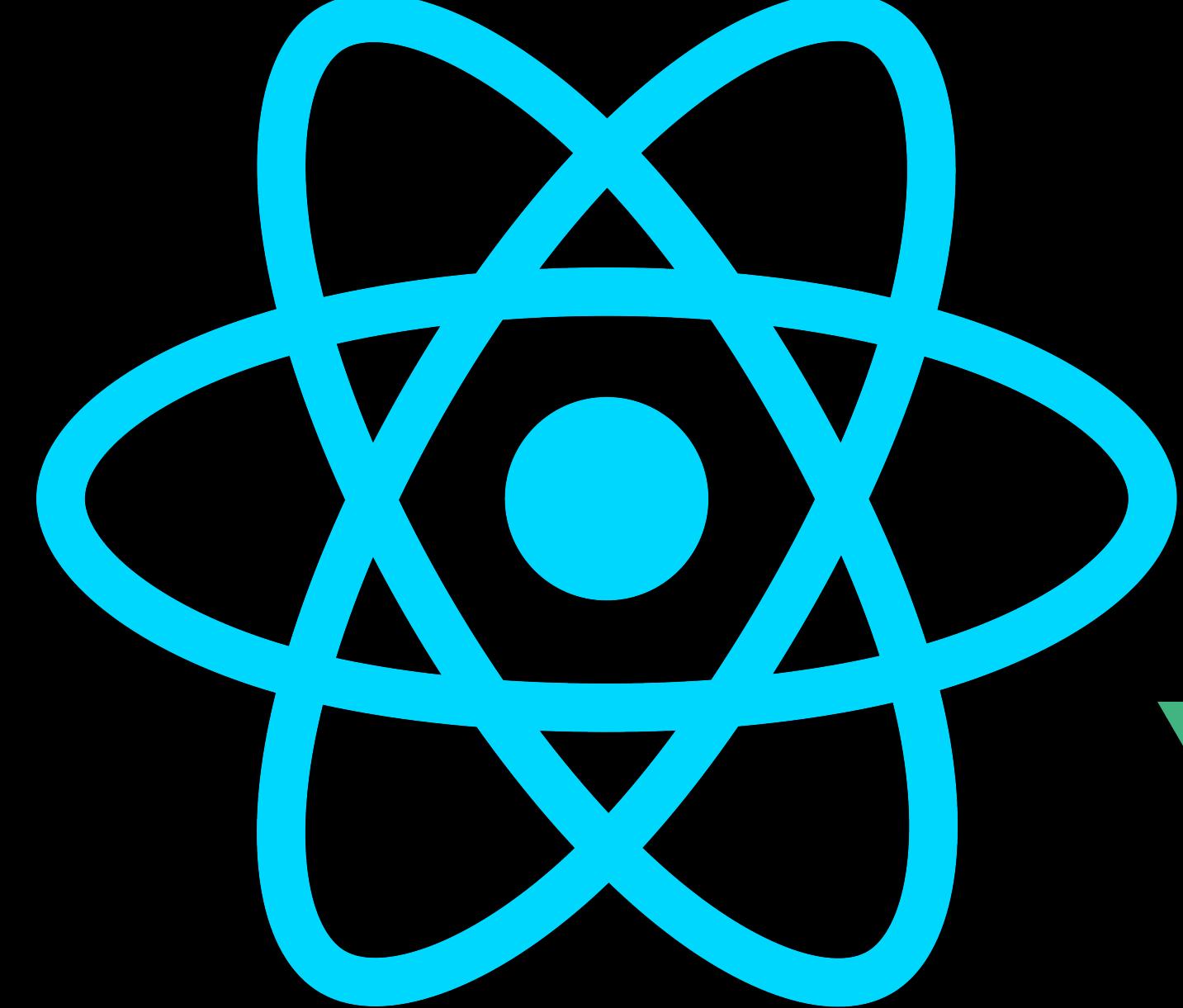


Bundler

Framework (**svelte**)

A **JavaScript framework** provides a structured environment and a set of tools for building web applications, offering predefined architecture and guidelines. Usually comes with built-in features like routing, state management, and data handling.

Framework (svelte)



Framework (svelte)

my linkedin profile

R, python, javascript, shiny, dplyr, purrr, ditto,
ggplot, d3, canvas, spark, sawk, pyspark, sparklyR,
lodash, lazy, bootstrap, jupyter, vulpix, git,
flask, numpy, pandas, feebas, scikit, pgm, bayes,
h2o.ai, sparkling-water, tensorflow, keras, onyx,
ekans, hadoop, scala, unity, metapod, gc, c#/c++,
krebase, neo4j, hadoop.

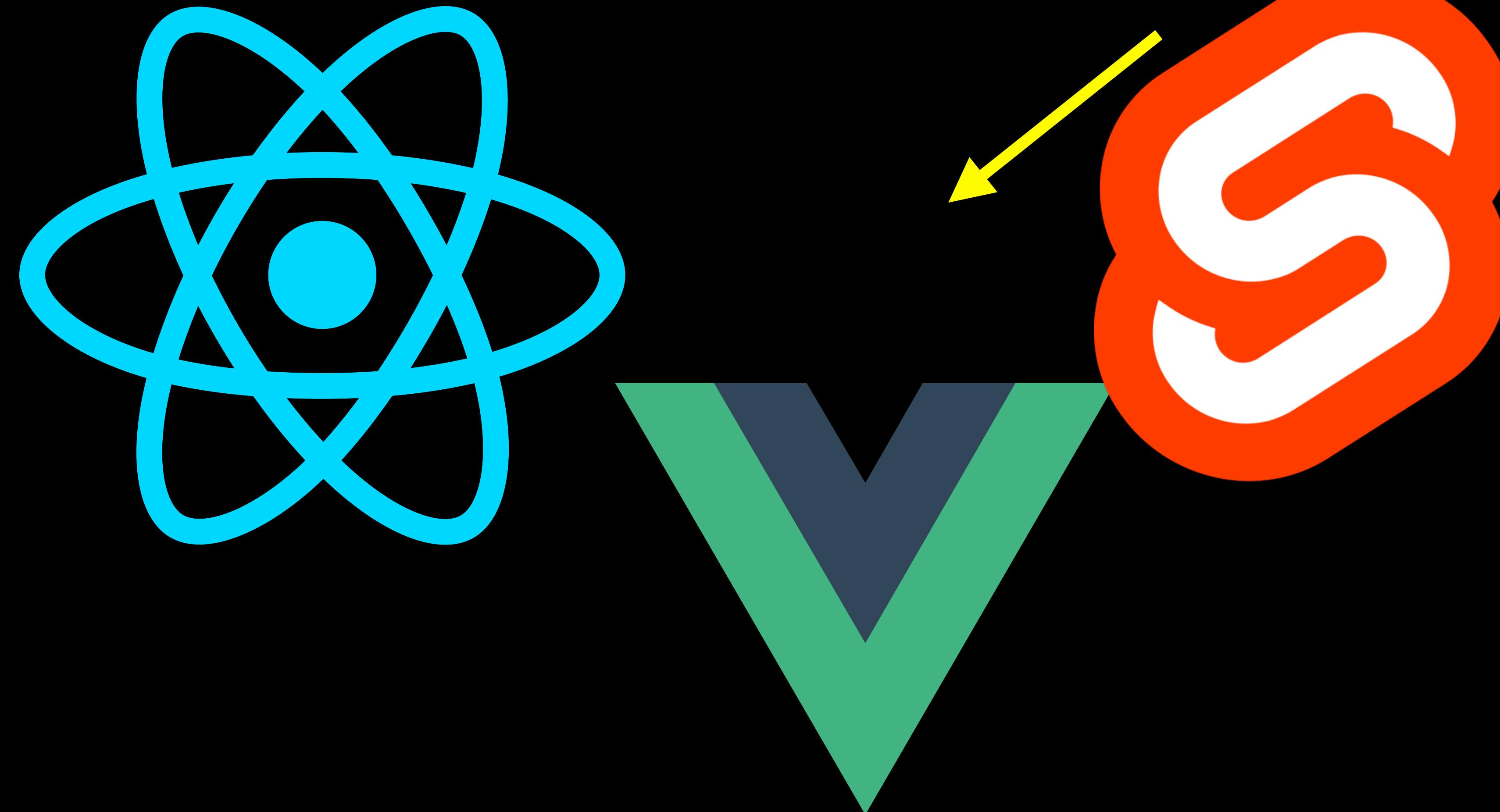
I typically ask recruiters to point out which of these are pokemon.

Vincent D. Warmerdam - @SchoutenDS - Runing.io - OnDataDriven 5



Framework (*svelte*)

All of these (at least in their current form)
will be obsolete in the next 3–5 years.



Framework (*svelte*)

- Architectuur en standaarden (opinionated)
- Component-gebaseerd, UI componenten, single file
- Reactieve data binding, dom updates
- Routing (SPA) tussen pagina's

Framework (*svelte*)

```
// Counter.js (React)

import { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}

export default Counter;
```

```
<!-- Counter.svelte (Svelte) -->

<script>
  let count = 0;
  function increment() {
    count += 1;
  }
</script>

<div>
  <p>Count: {count}</p>
  <button on:click={increment}>Increment</button>
</div>
```

Component, state management.

Framework (**svelte**)

It's better to learn **vanilla code**. **If you know JavaScript** as it enables you to switch between framework whenever the need requires it. This is also why we hammer on documentation skills, both writing and reading it.

Framework (*svelte*)



Use less or more
as you wish.

D3 in Svelte

As [with React](#), you can use Svelte exclusively for rendering if you like, and only use D3 modules that don't manipulate the DOM. Here is a line plot of an array of numbers that uses [d3-shape](#) and [d3-scale](#).

LinePlot.svelte

svelte

```
<script>
    import * as d3 from 'd3';

    export let data;
    export let width = 640;
    export let height = 400;
    export let marginTop = 20;
    export let marginRight = 20;
    export let marginBottom = 20;
    export let marginLeft = 20;

    $: x = d3.scaleLinear([0, data.length - 1], [marginLeft, width - marginRight])
    $: y = d3.scaleLinear(d3.extent(data), [height - marginBottom, marginTop]);
```

Libraries, frameworks & bundlers



Library



Framework



Bundler

Bundlers (vite)

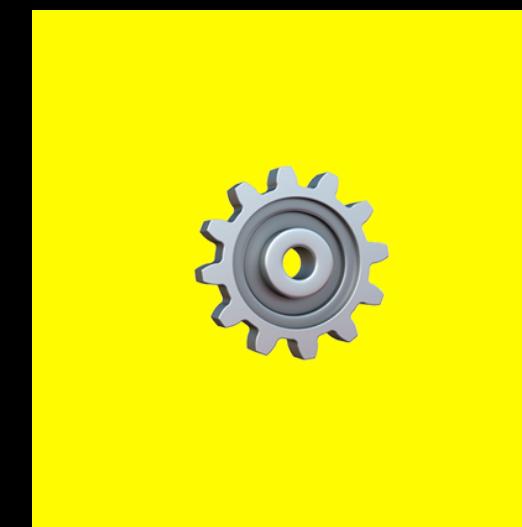
A **JavaScript bundler** is a tool that takes multiple files (and assets like CSS, images, etc.) and bundle them into a single (or few) output files for the browser. Mostly used for automating tasks and transforming code.

Bundlers (vite)

// Input

```
css/  
└── typography.scss  
└── layout.scss  
└── colors.scss
```

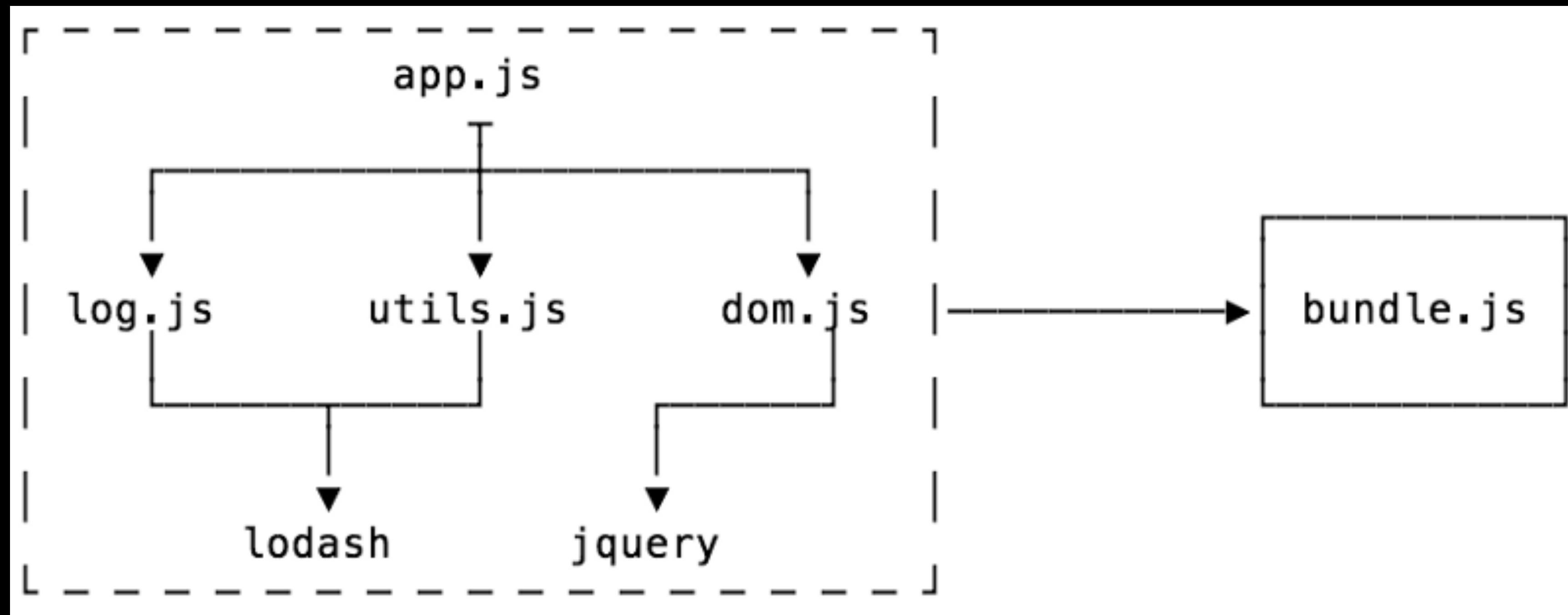
// Build



// Output

```
css/  
└── style.css
```

Bundlers (vite)



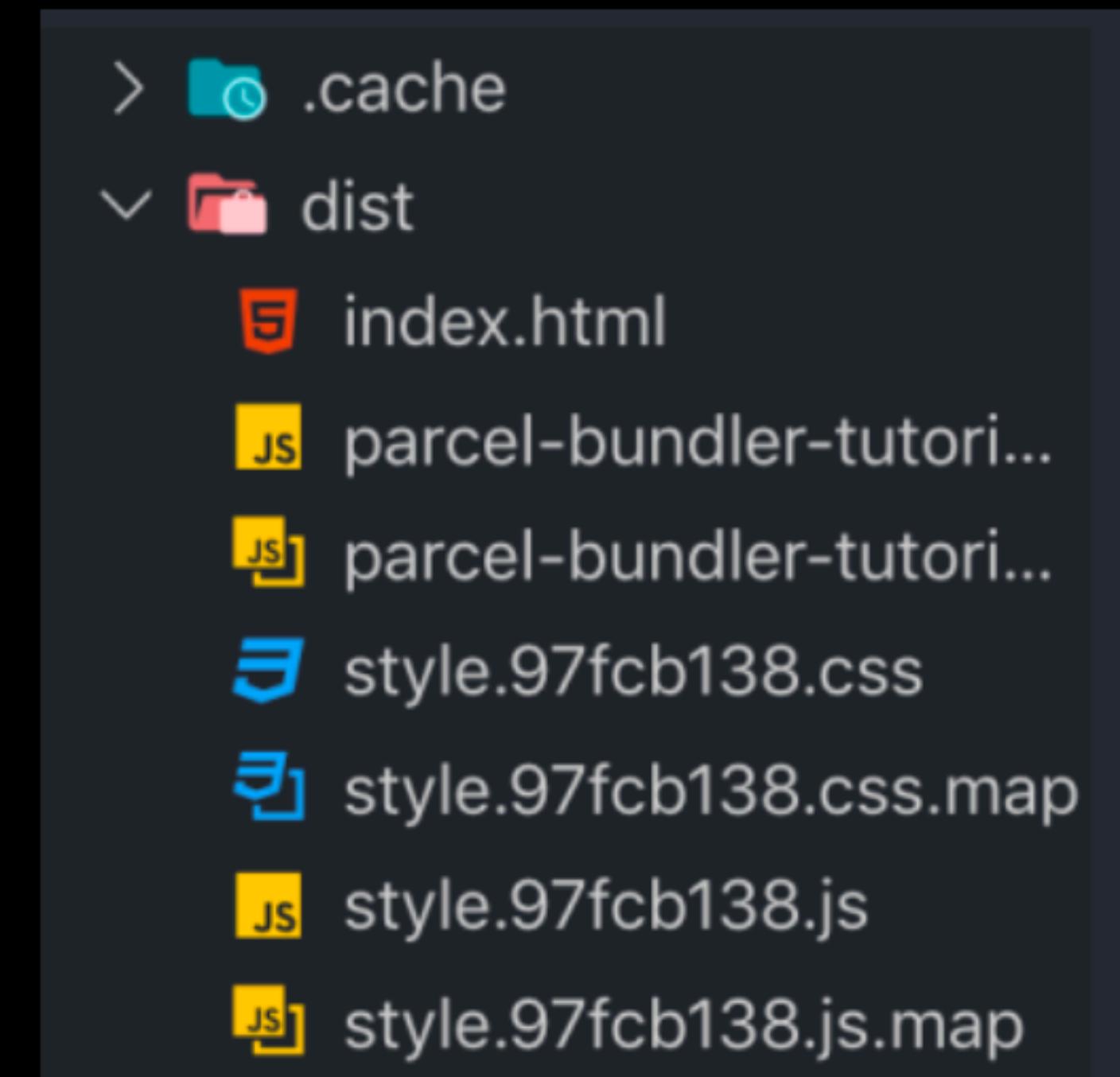
Bundlers (vite)

- Bundle HTML (components into complete markup)
- Bundle CSS (post/preprocess)
- Bundle (client-side) JavaScript components
- Use task runners to automate many more...

Bundlers (vite)

```
✨ Built in 444ms.

dist/parcel-bundler-tutorial.8bdf8f45.js      1.48 KB    241ms
dist/style.134bbd53.css.map                   779 B      5ms
dist/parcel-bundler-tutorial.8bdf8f45.js.map   756 B      2ms
dist/index.html                                639 B      5ms
dist/style.134bbd53.css                         337 B     13ms
```



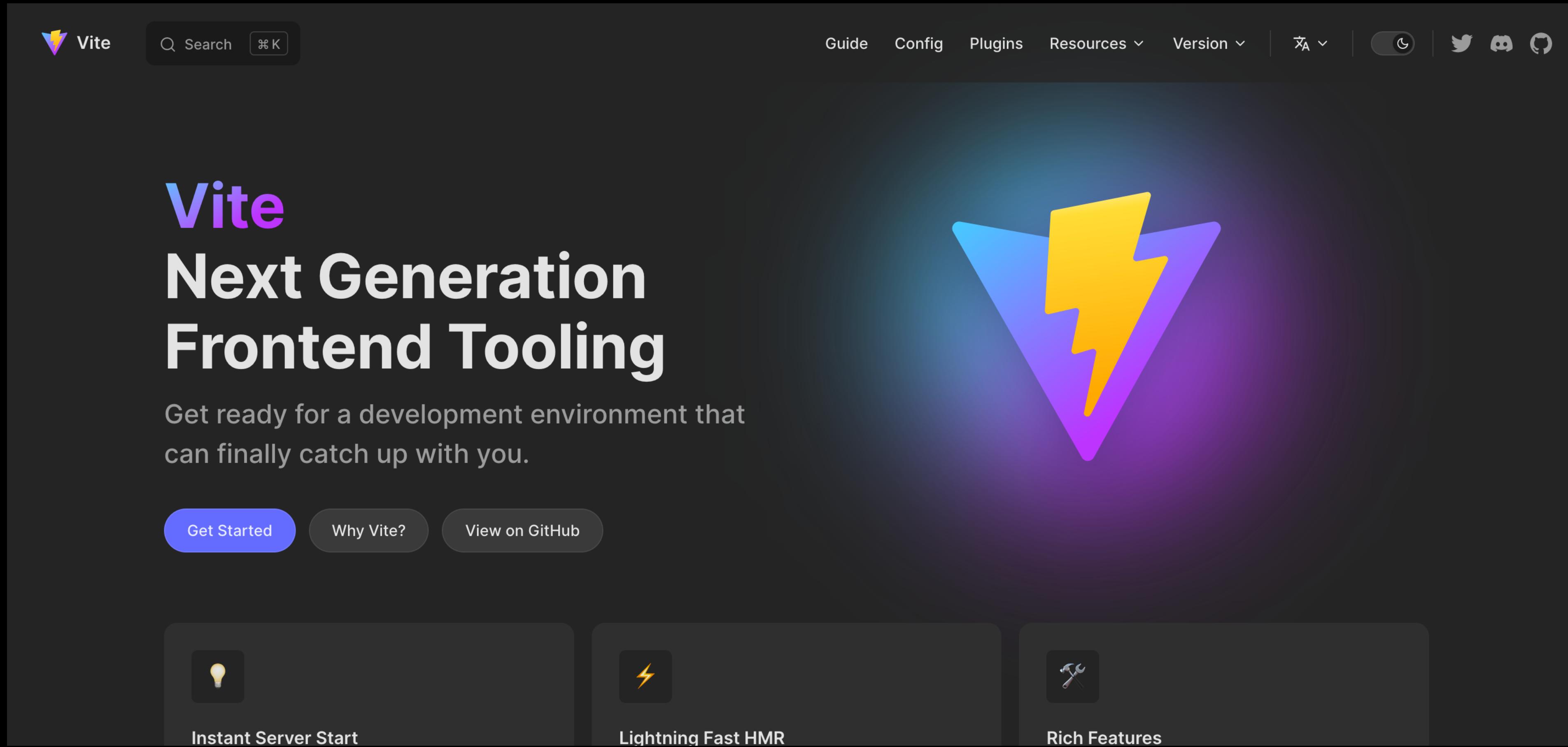
Bundlers (vite)

We used to have a lot of separate tools and libraries that were connected using task runners, Gulp, Grunt, Webpack

bla bla bla, etc...

But now:

Bundlers (vite)



The screenshot shows the official Vite website. At the top, there's a dark header with the Vite logo, a search bar, and navigation links for Guide, Config, Plugins, Resources, Version, and language selection. Below the header, the main title "Vite" is displayed in large purple and white letters, followed by the subtitle "Next Generation Frontend Tooling". A large, stylized yellow lightning bolt icon is centered on a blue-to-purple gradient background. Below the title, a subtext reads "Get ready for a development environment that can finally catch up with you." At the bottom of the main section, there are three buttons: "Get Started" (blue), "Why Vite?" (grey), and "View on GitHub" (grey). Below these buttons are three dark cards with icons and text: "Instant Server Start" (lightbulb icon), "Lightning Fast HMR" (lightning bolt icon), and "Rich Features" (wrench and hammer icon).

All-in-one toolkits (runtimes)

Libraries, frameworks & bundlers



Library:
*Om visualisaties
te maken.*



Framework:
*Om in componenten
te werken.*



Bundler:
*Om in componenten
te werken.*

Schedule

1. Show and tell: dataset & concept
2. Web architecture
3. Libraries, frameworks en bundlers
- 4. Boilerplate opzetten met Svelte en D3**
5. All together!



Boilerplate opzetten

The screenshot shows a dark-themed web page from the Svelte website. At the top, there's a navigation bar with the Svelte logo, a search bar, and links for Docs, Tutorial, Playground, and Blog. Below the navigation, the page title is "SVELTEKIT • GETTING STARTED". The main content area has a large heading "Project structure". To the left, there's a sidebar with sections for "Getting started" (Introduction, Creating a project, Project structure, Web standards), "Core concepts" (Routing, Loading data, Form actions, Page options, State management), and "Build and deploy" (Building your app, Adapters, Zero-config deployments, Node servers, Static site generation, Single-page apps, Cloudflare Pages, Cloudflare Workers, Netlify). On the right, there's a sidebar titled "On this page" with links to "Project structure", "Project files", and "Other files". The central content area contains a code block showing a typical SvelteKit project structure:

```
my-project/
├ src/
│ ├ lib/
│ │ ├ server/
│ │ │ └ [your server-only lib files]
│ │ └ [your lib files]
│ ├ params/
│ │ └ [your param matchers]
│ ├ routes/
│ │ └ [your routes]
│ ├ app.html
│ ├ error.html
│ ├ hooks.client.js
│ ├ hooks.server.js
│ └ service-worker.js
├ static/
│ └ [your static assets]
└ tests/
    └ [your tests]
├ package.json
└ svelte.config.js
└ tsconfig.json
```

Boilerplate opzetten

- Begin een nieuw project met SvelteKit
- Lees de documentatie, begrijp de structuur
- Installeer D3 binnen je project
- Kijk of D3 werkt door een example chart te maken

**Uncaught SyntaxError
Unexpected end of input**