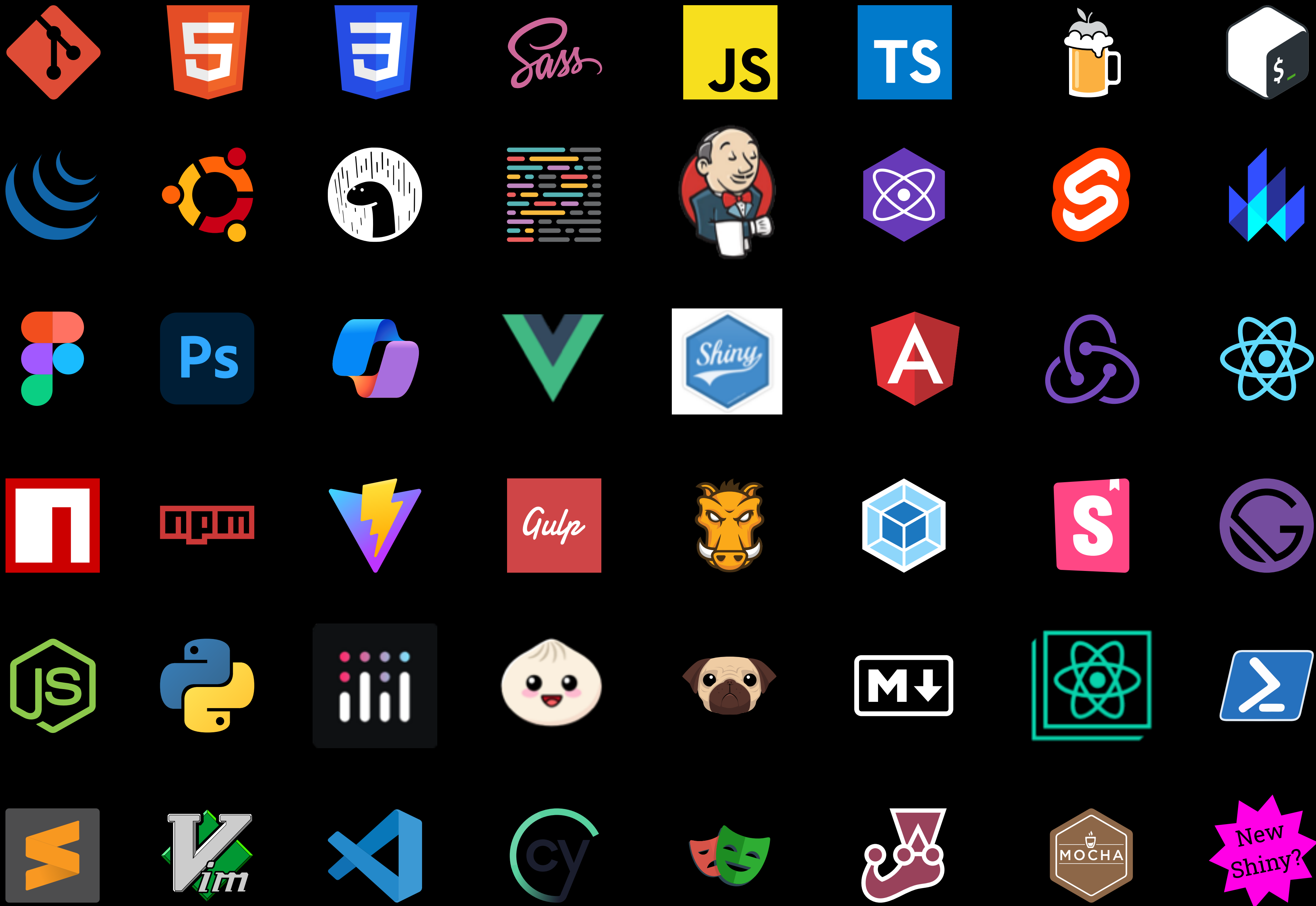
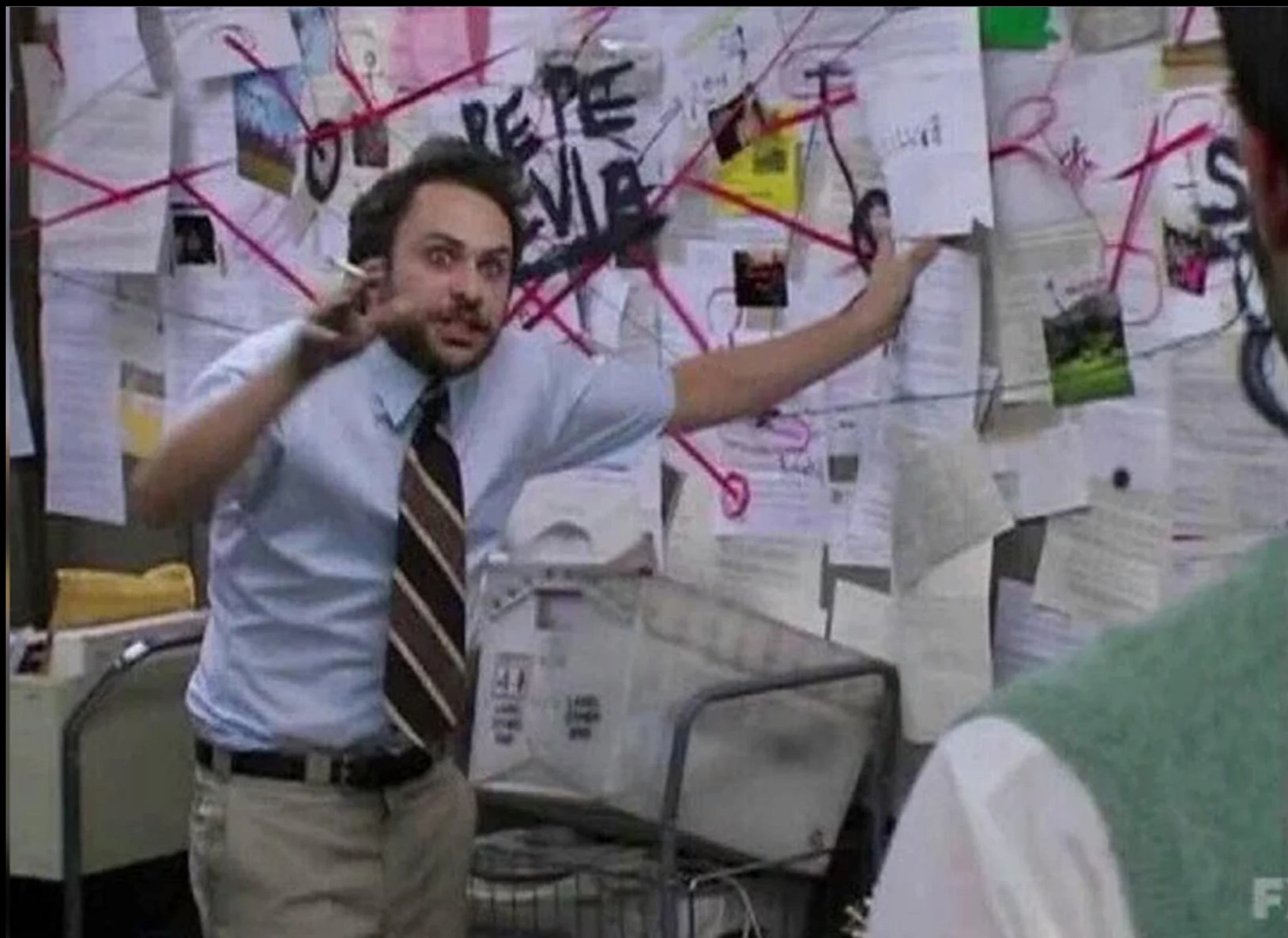


Less, but better.

“Good design is as little as possible.
Less, but better.
Simple as possible but not simpler.”

Dieter Rams

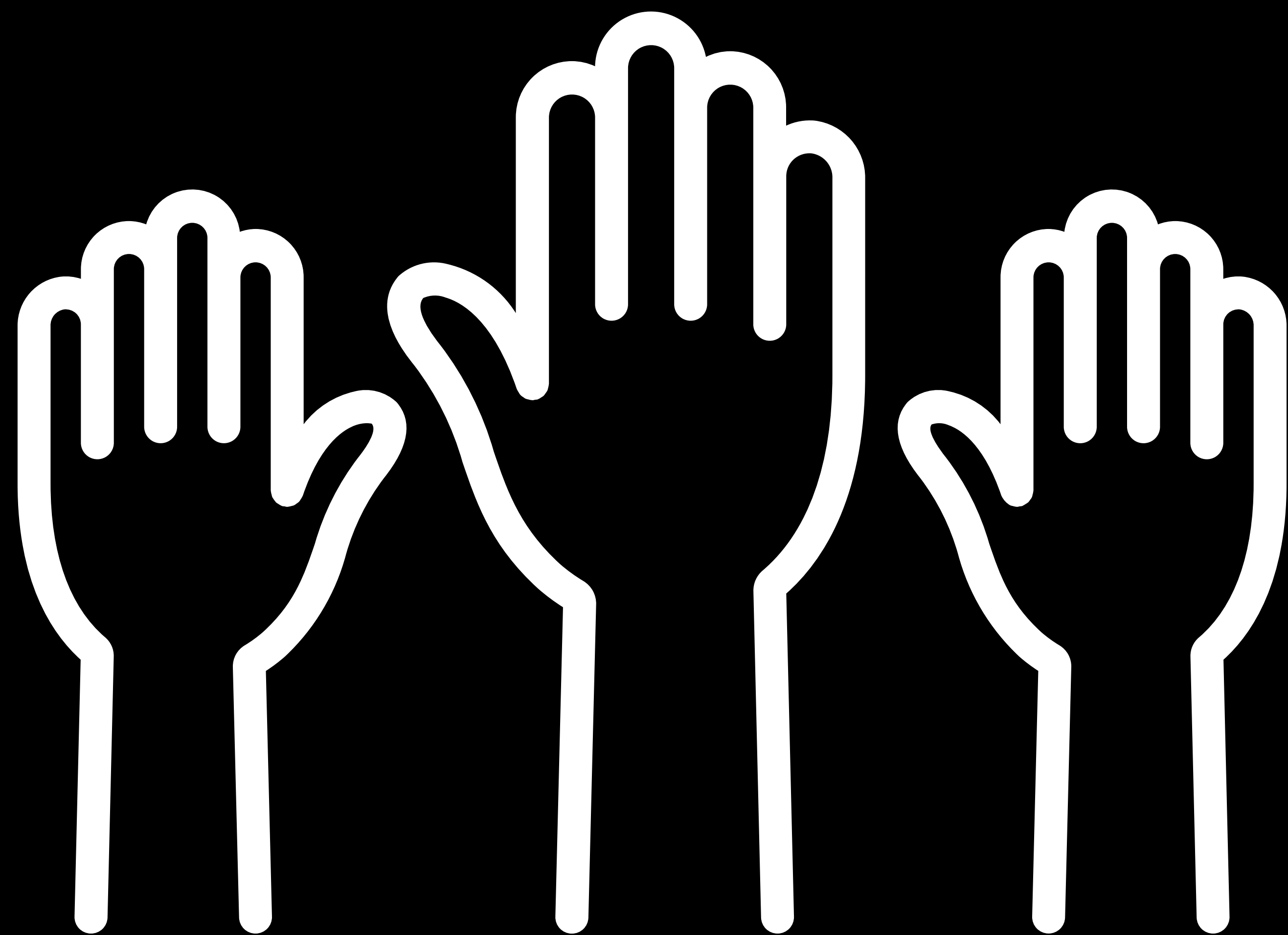




The complexity is in our builds,
not in the platform.

Zooming out

How long should our projects last?





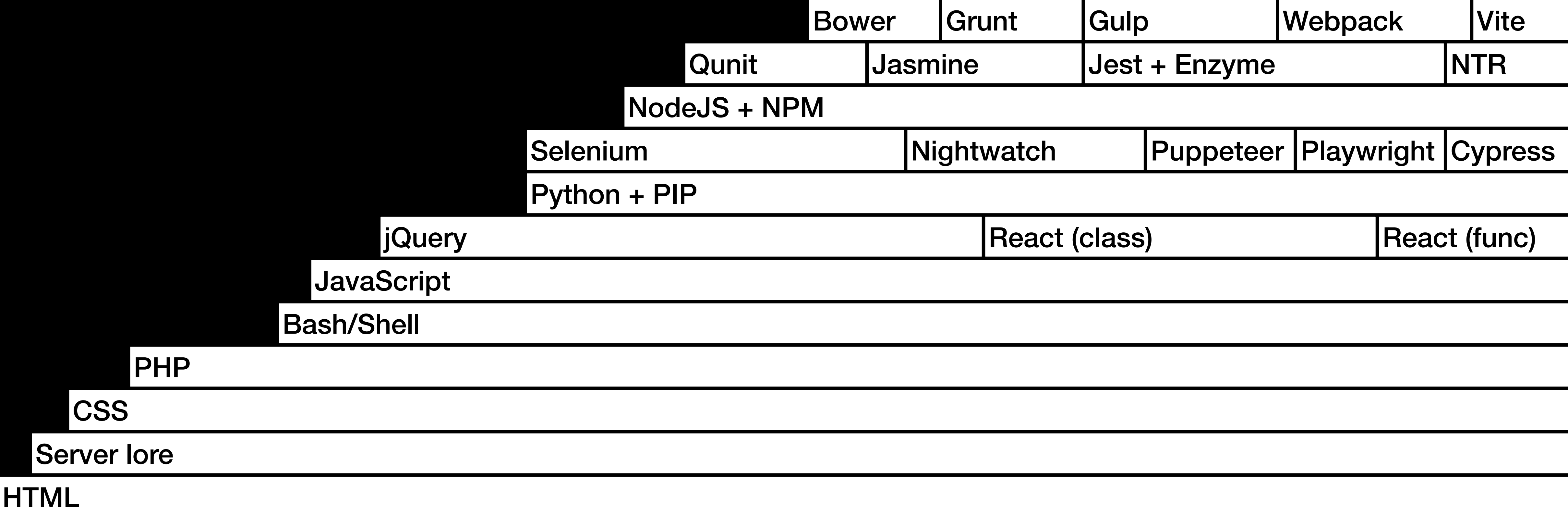
How long should our skills last?

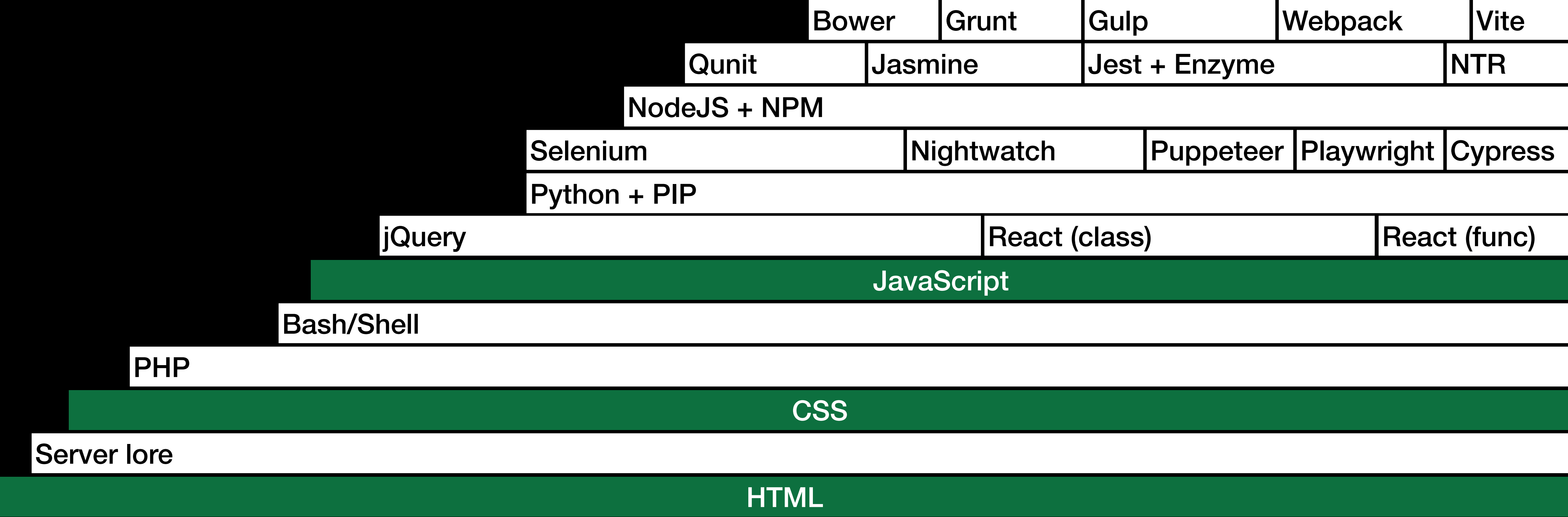
Vite - 1 year 

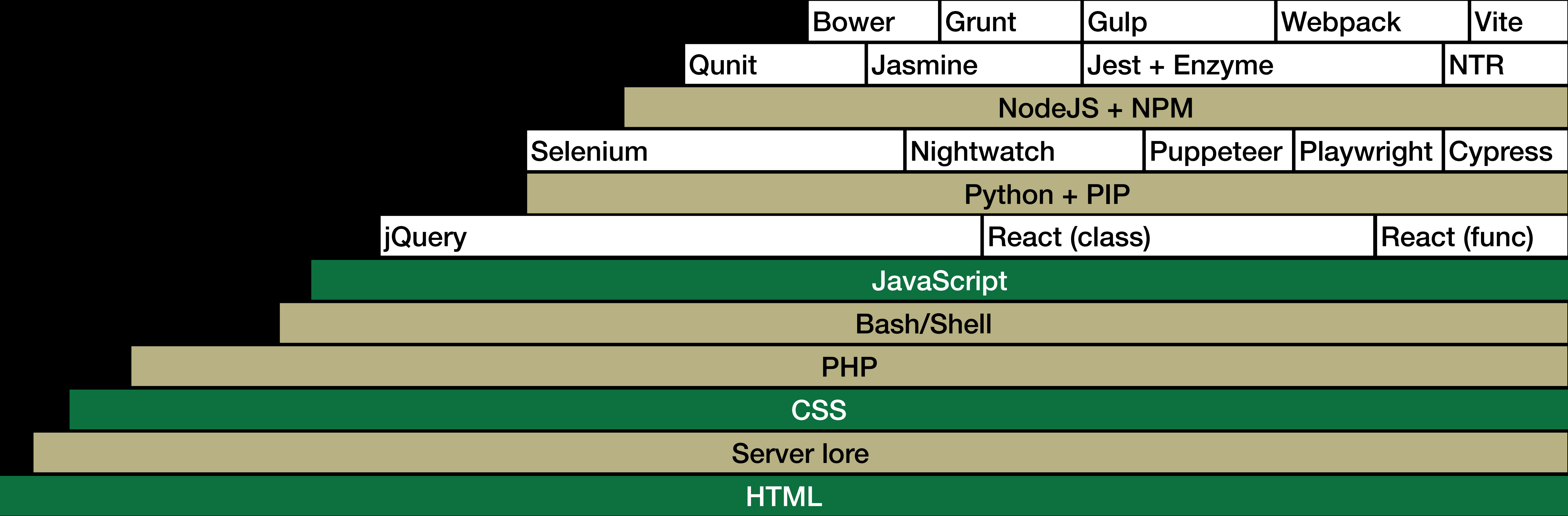
HTML - 28 years

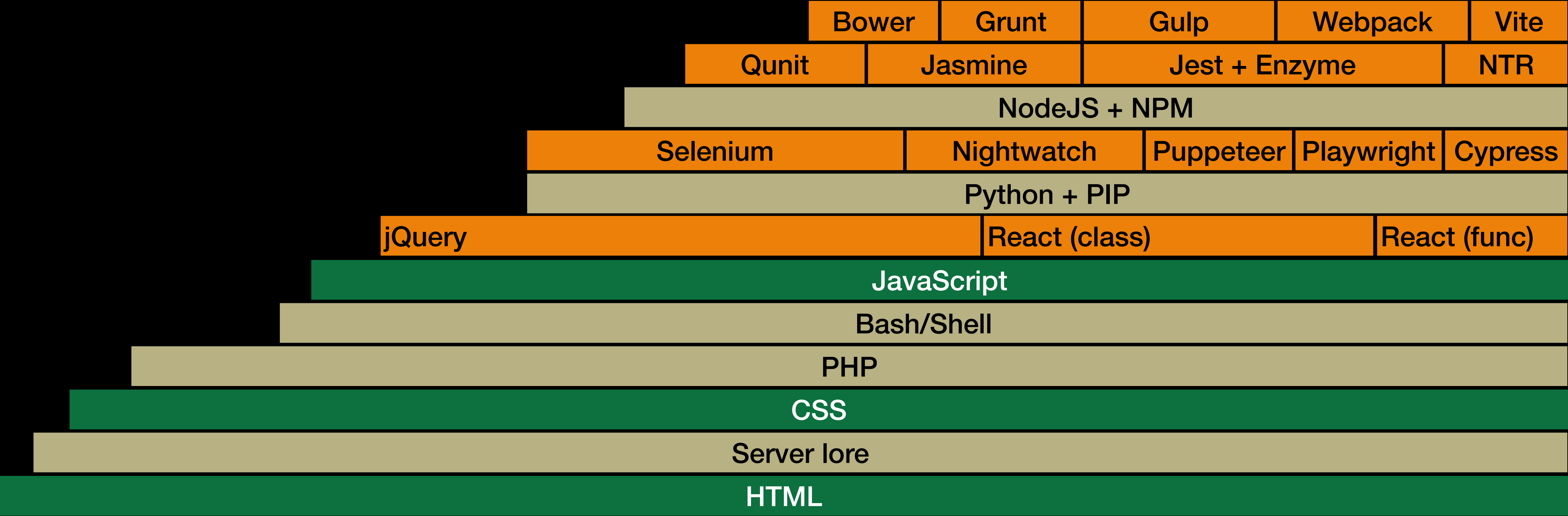
Vite

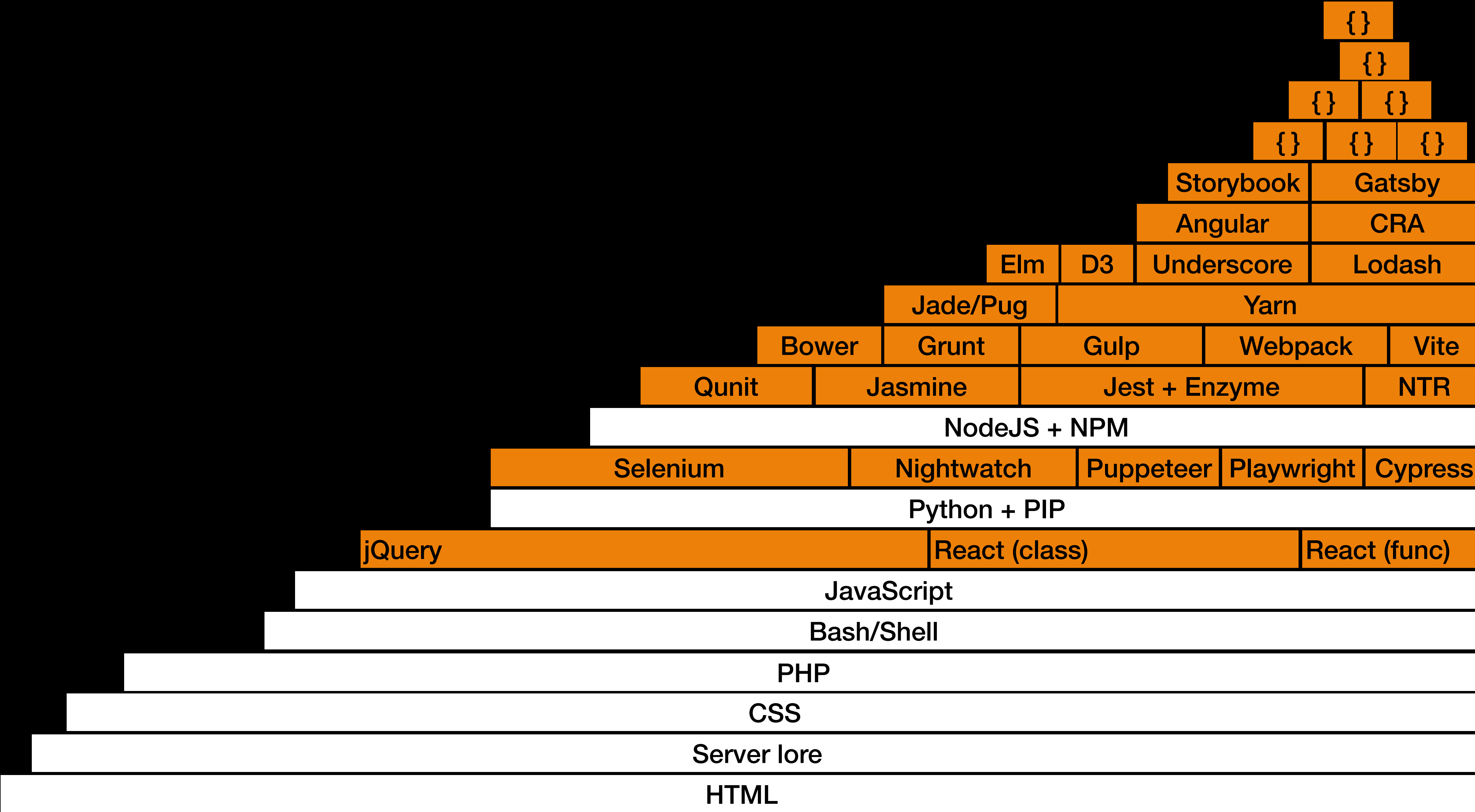
HTML











Fatigue Zone

{ }

{ }

{ }

{ }

{ }

{ }

{ }

Storybook

Gatsby

Angular

CRA

Elm

D3

Underscore

Lodash

Jade/Pug

Yarn

Bower

Grunt

Gulp

Webpack

Vite

Qunit

Jasmine

Jest + Enzyme

NTR

NodeJS + NPM

Selenium

Nightwatch

Puppeteer

Playwright

Cypress

Python + PIP

jQuery

React (class)

React (func)

JavaScript

Bash/Shell

PHP

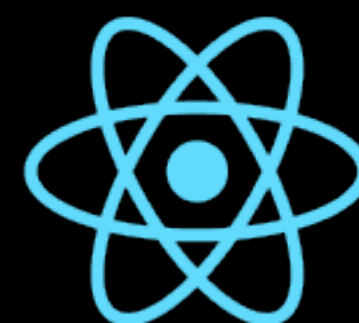
CSS

Server lore

HTML

Stable Zone

Volatility is expensive

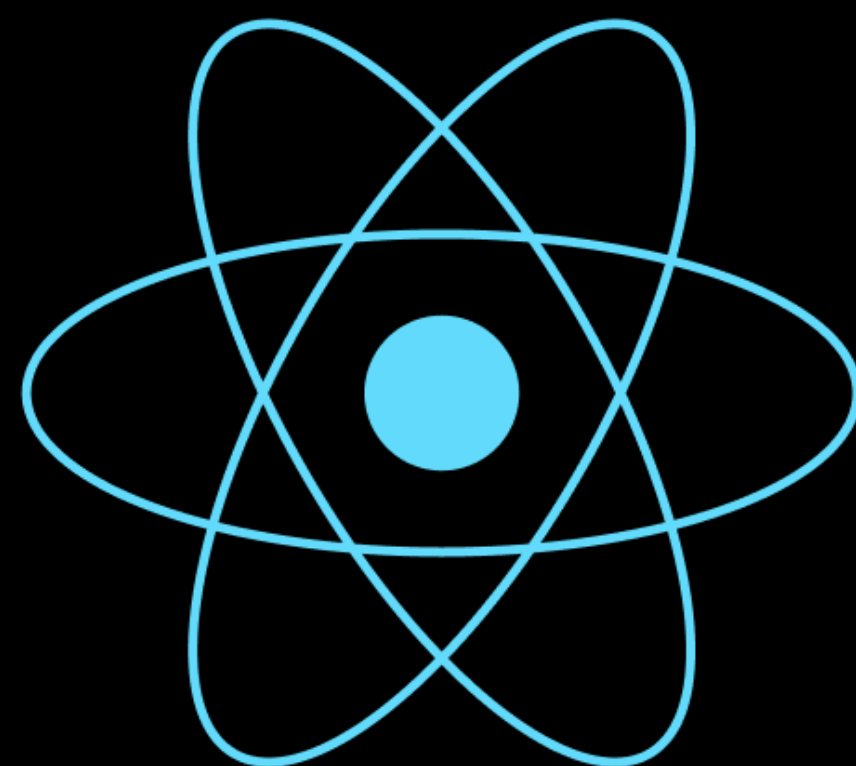


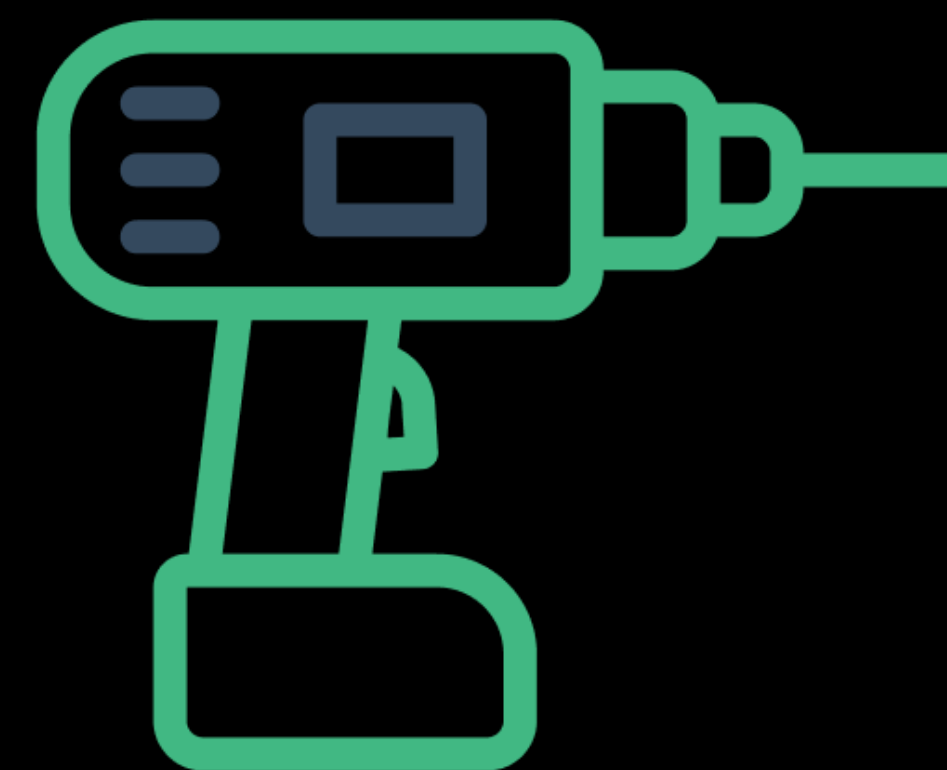
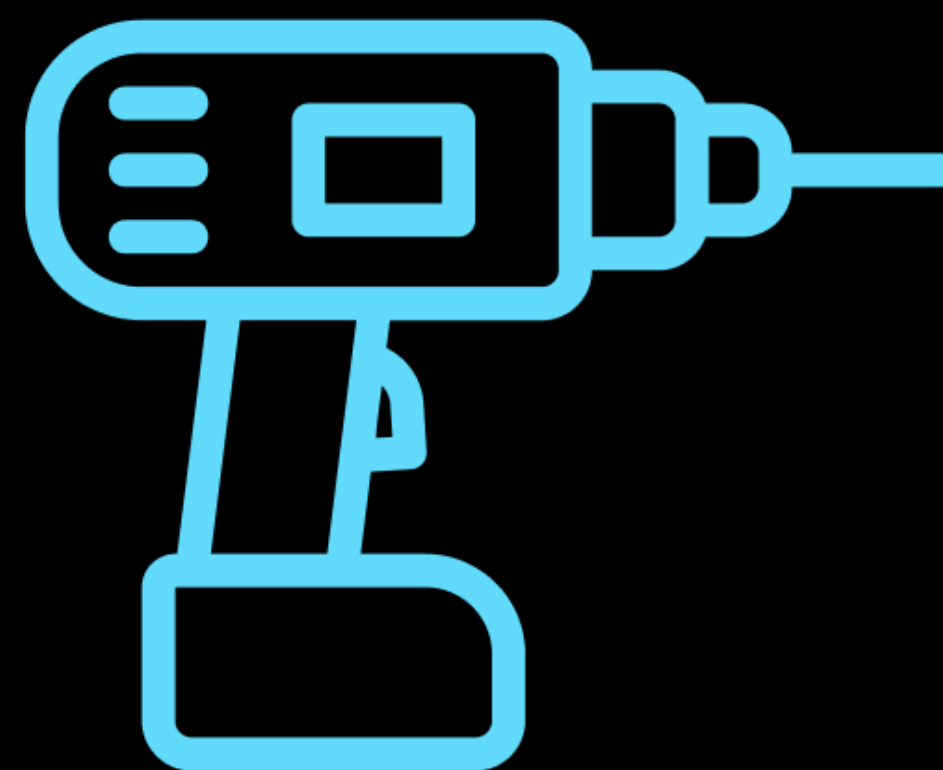
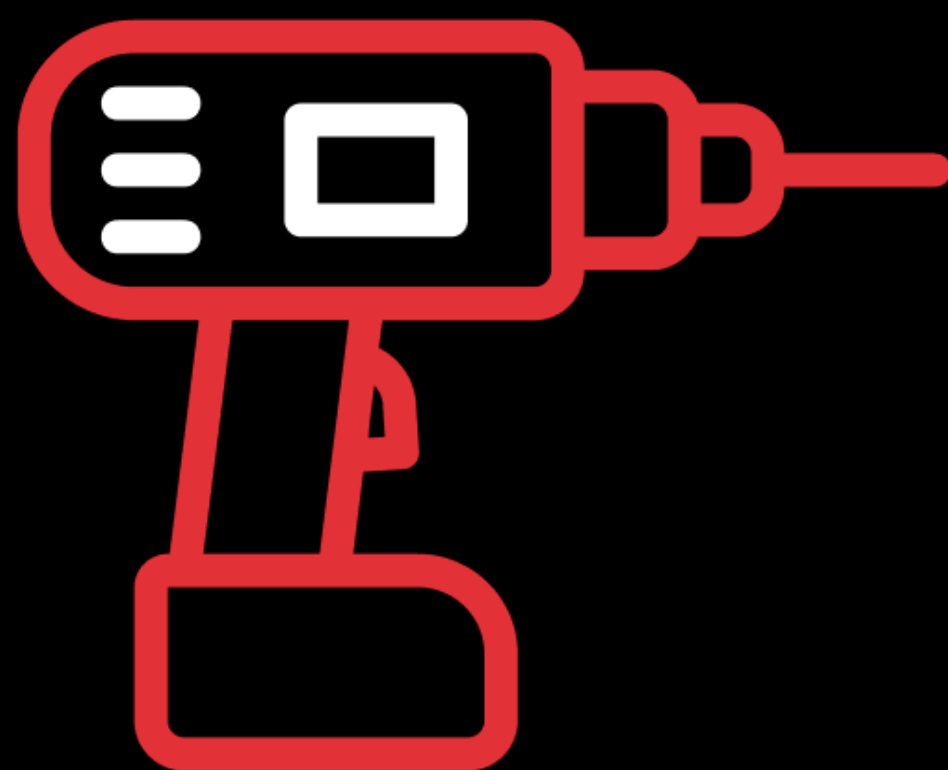
BABEL



aws



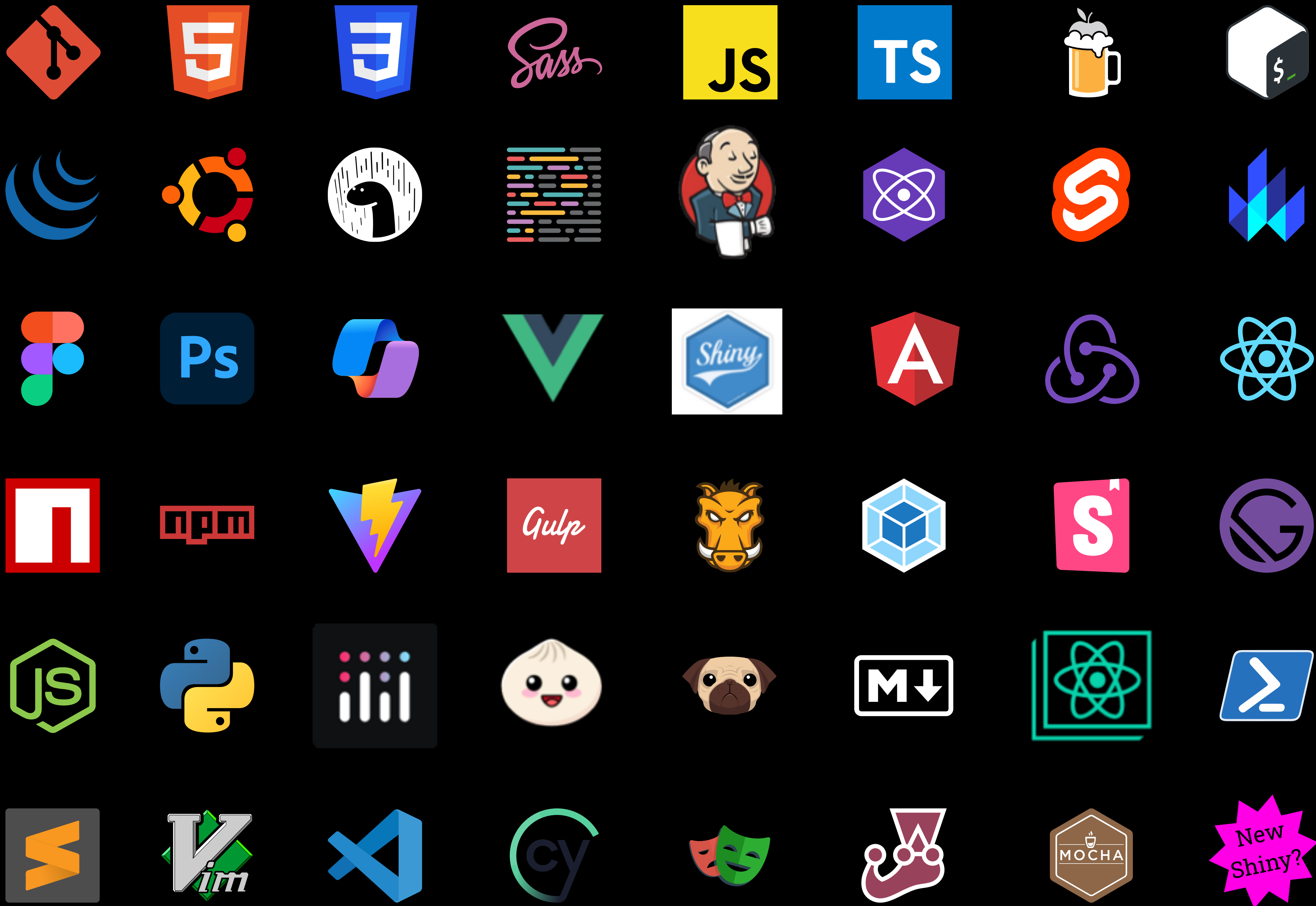






“Nobody wants a drill.
What they want is the hole.”

Old sales saying



By the pricking of my thumbs,
a new web era this way comes

Eras

Mid 1990s

HTML

Server
HTML

Mid 1990s

HTML

Server
HTML

Early 2000s

SSR

Server
HTML
CSS
JS

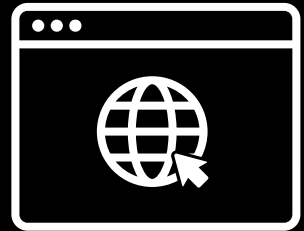
Mid 1990s	Early 2000s	Mid 2000s
HTML	SSR	DOM/Ajax
Server HTML	Server HTML CSS JS	Server HTML CSS jQuery E2E tests

Mid 1990s	Early 2000s	Mid 2000s	Mid 2010s
HTML	SSR	DOM/Ajax	SPA
Server HTML	Server HTML CSS JS	Server HTML CSS jQuery E2E tests	Cloud infrastructure CI/CD pipelines Package managers HTML CSS Typescript React React tests E2E tests

Eras all the way down

- Plugin content - *Applets, Flash, Silverlight*
- Mobile formats - *WAP, WML, iHTML, AMP*
- Content - *broadcast, mashup, syndication, walled garden, paywall, AI*

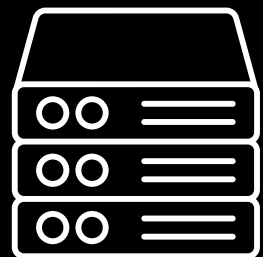
Our roles across eras



UI

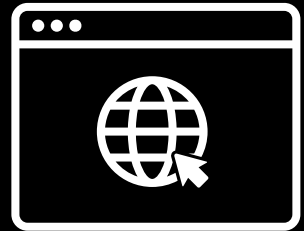
Frontend

HTML



Business Logic

Backend



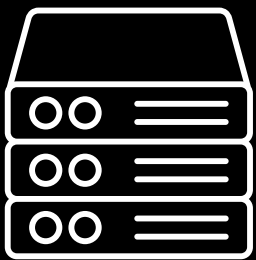
UI

UI

Frontend

HTML

SSR

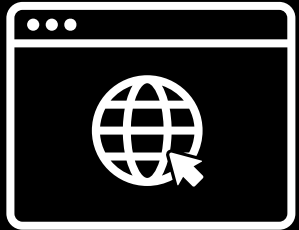


Business Logic

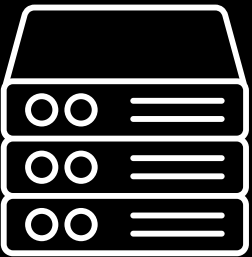
Templating (SSR)

Business Logic

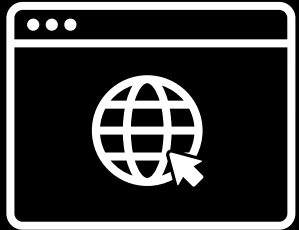
Frontend (in server-side languages)
Backend



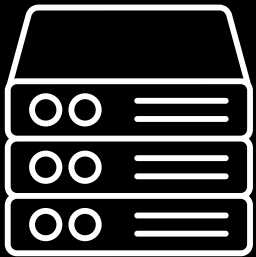
Frontend



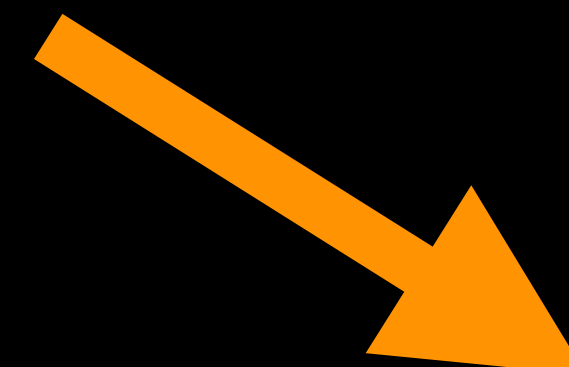
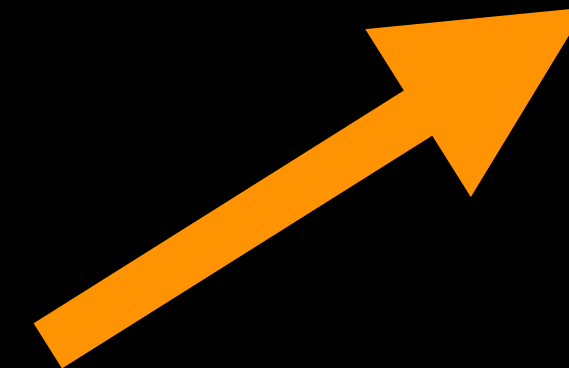
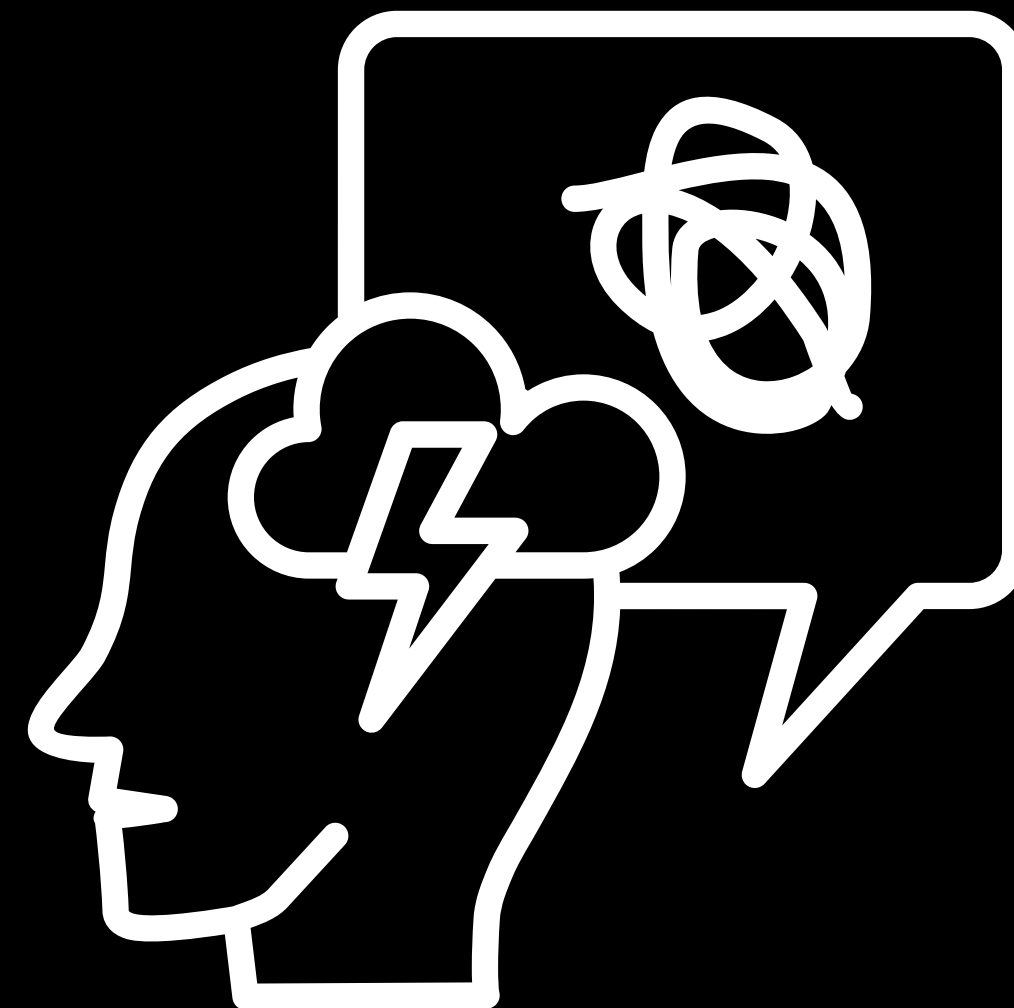
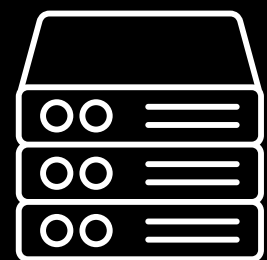
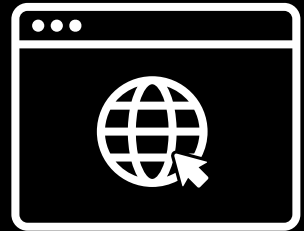
Frontend
Backend



UI	UI	UI	UI
			Templating (JavaScript)
			Business Logic (JavaScript)
HTML	SSR	DOM/Ajax	SPA



	Templating (SSR)	Templating (SSR)	
Business Logic	Business Logic	Business Logic	Business Logic



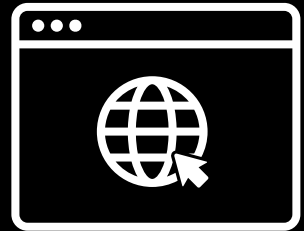
UI

Templating (JavaScript)

Business Logic (JavaScript)

SPA

Business Logic



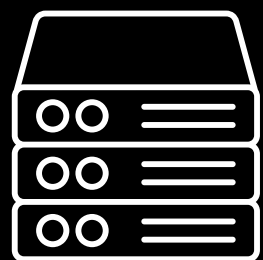
Frontend of the frontend
GenX of frontend
Backend of the frontend

UI

Templating (JavaScript)

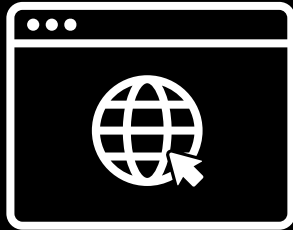
Business Logic (JavaScript)

SPA



Backend! Only backend! Nice pure backend!

Business Logic



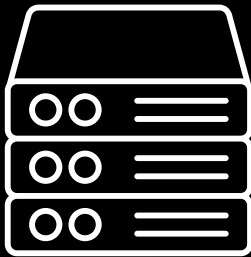
Full stack
Full stack
Full stack

UI

Templating (JavaScript)

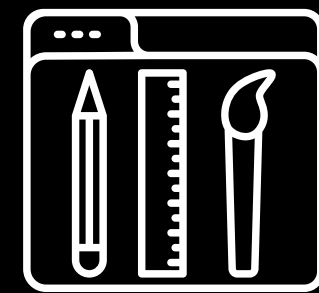
Business Logic (JavaScript)

SPA



Full stack

Business Logic



UX & Design

Design

Content

Tech Writer

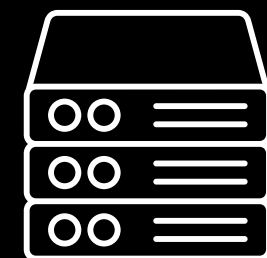


UI

Frontend

Business Logic in client

Backend



Templates on server

Frontend

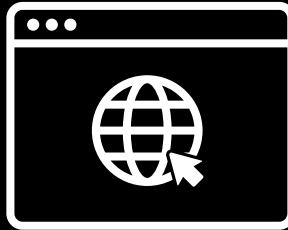
Business Logic on server

Backend



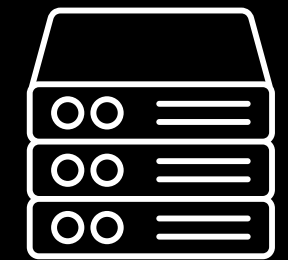
Data

Data



UI	UI	UI	UI	UI?
			Templating (JavaScript)	
			Business Logic (JavaScript)	

HTML	SSR	DOM/Ajax	SPA	Streamed SSR?
------	-----	----------	-----	---------------



	Templating (SSR)	Templating (SSR)		Templating (SSR)?
Business Logic	Business Logic	Business Logic	Business Logic	Business Logic?

Every era shapes the next

Frustrations drive innovation

- Repetition and low maintainability → templating, SSR, CSS
- Desire for rich designs → DOM scripting, improved media and fonts
- Dislike of C in CSS → CSS layers, scope, shadow DOM
- Desire to escape algorithms → fediverse, cozyweb

Frameworks shape the platform

- jQuery → vanilla JS features like queryselector, forEach and fetch
- SASS → vanilla CSS features like custom props
- CSS-in-JS → vanilla CSS features like scope, layers, and shadow DOM

Frameworks shape the platform

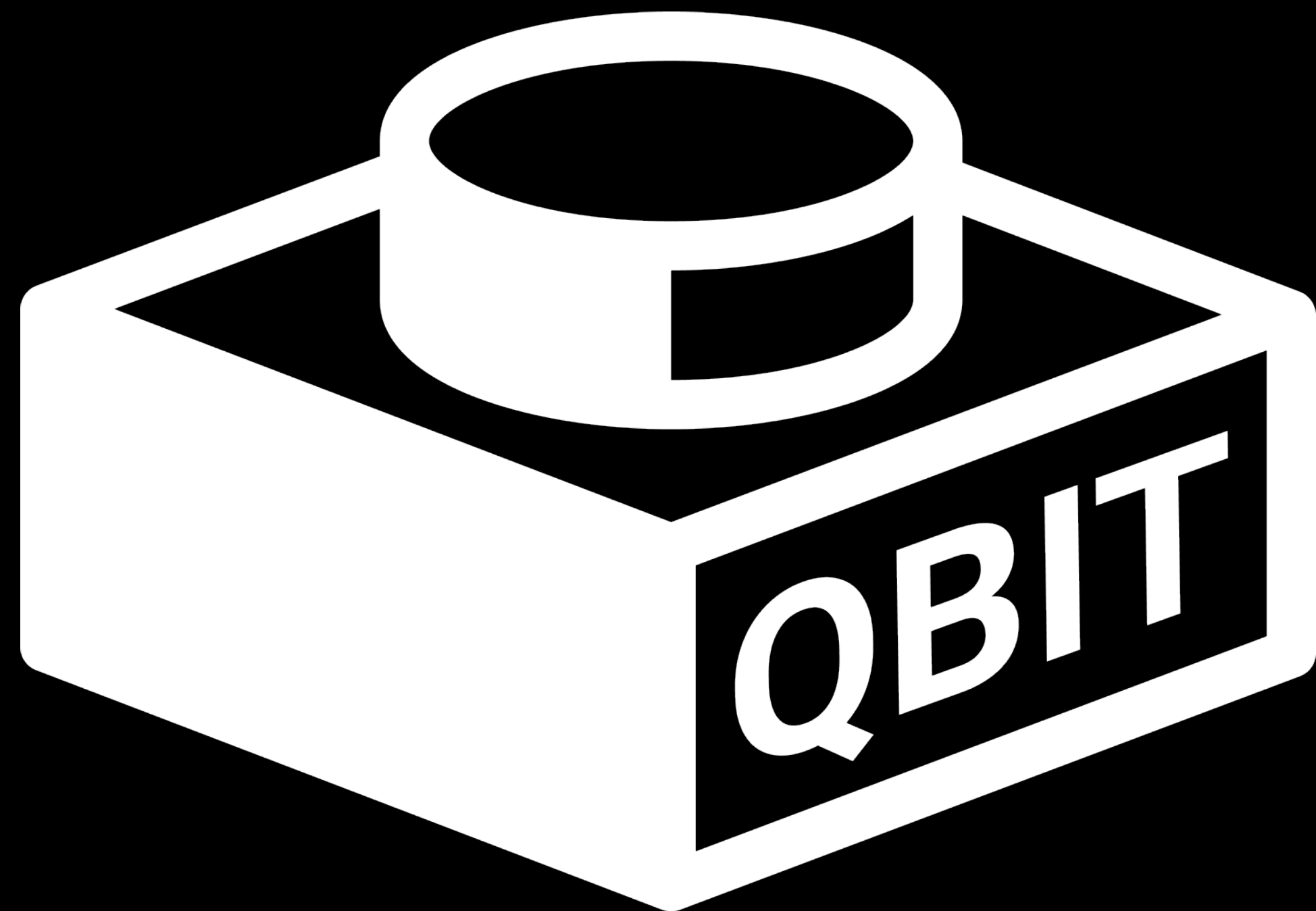
- JavaScript → jQuery → JavaScript (ES6)
- CSS → SCSS → CSS
- Coffeescript → Typescript → ...?

**You only get the benefit if you use
the new features of the platform.**

Beware of the
familiarity
comfort
zone



My own dream of a simpler web



Quantium's UI library



This picture is a lie.

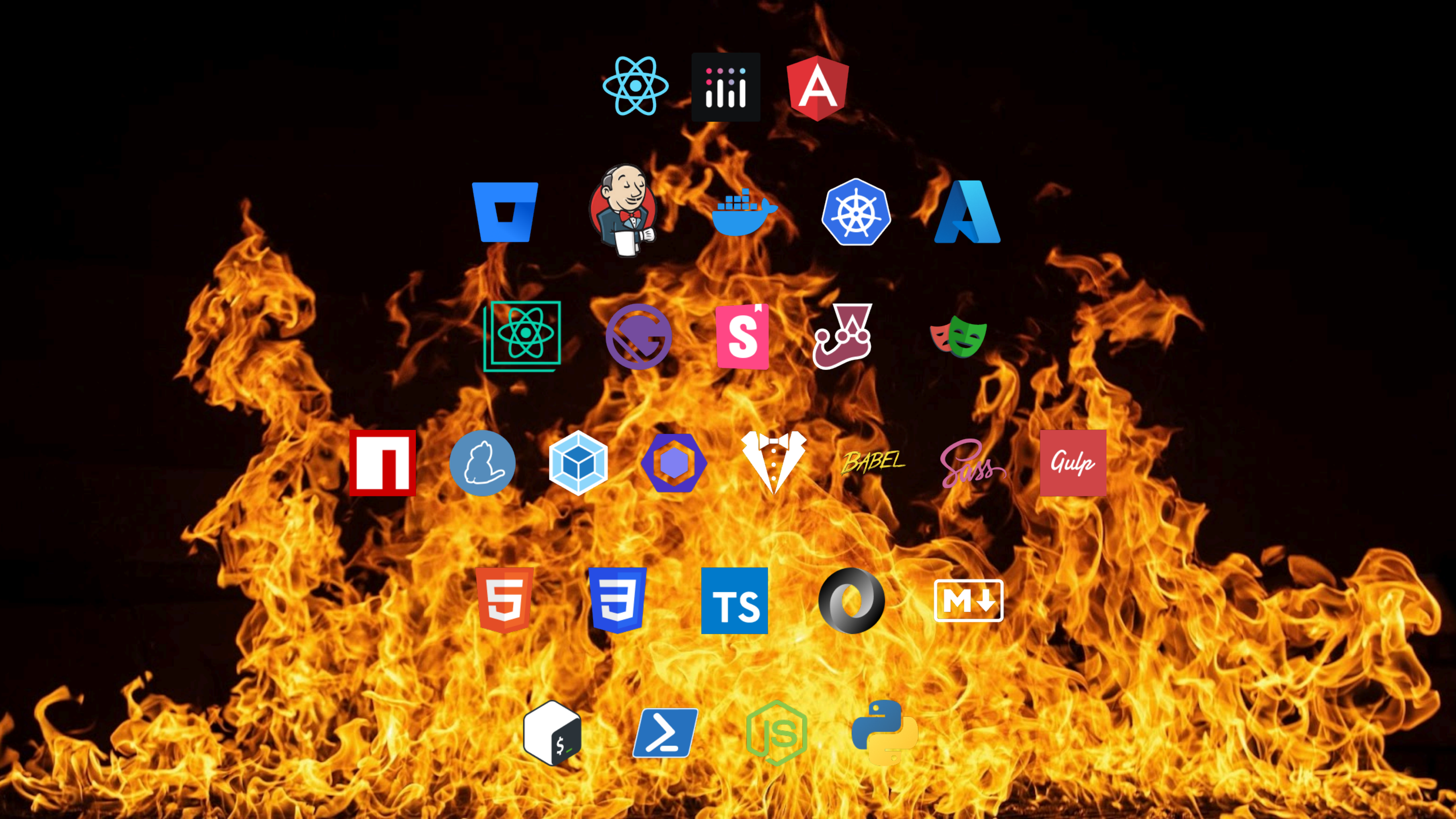




BABEL

Sass





BABEL

SASS



Industry-endorsed tooling
only lasted five years.

Never fear, people had
suggestions!



Bun!

Deno!

Lerna!

JSR!

Turborepo!

**React Test
Library!**

PNPM!

Nx!

**Go back to
Storybook!**





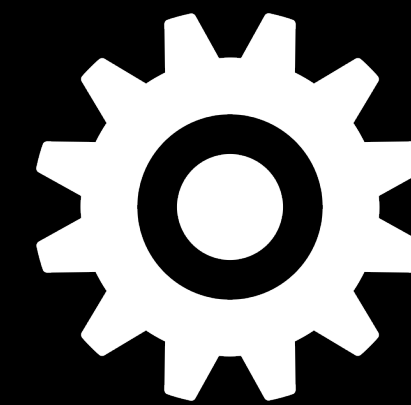
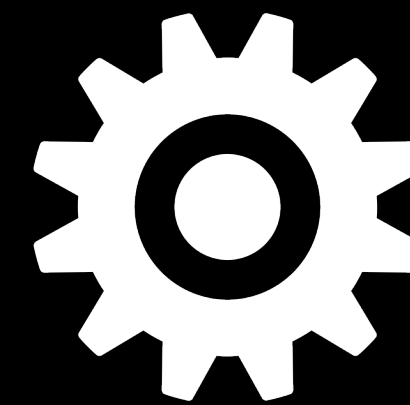
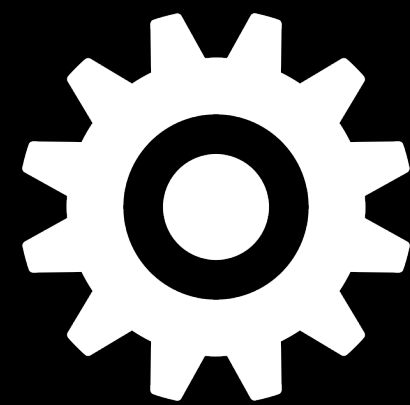
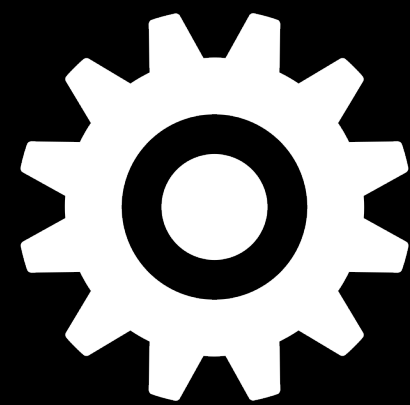
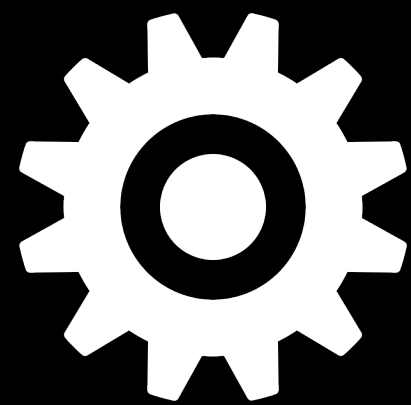
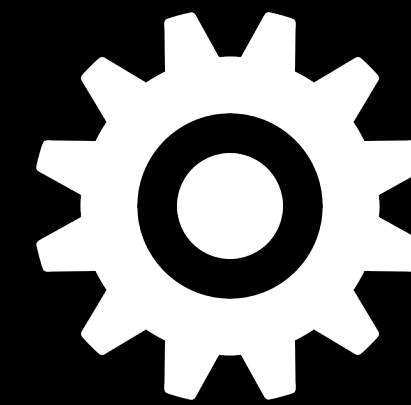
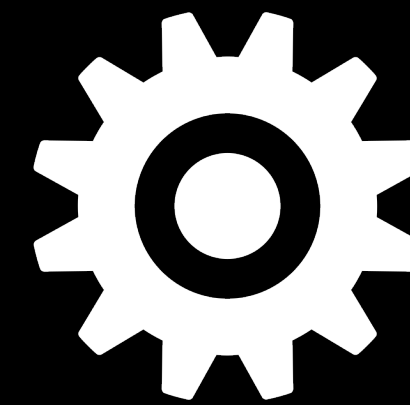
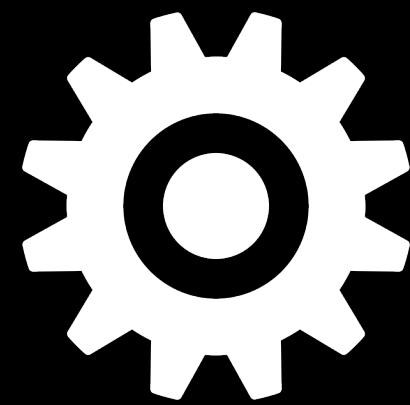
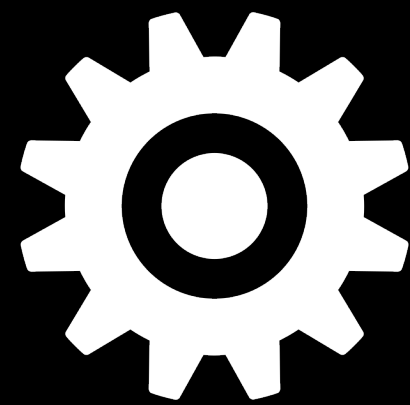
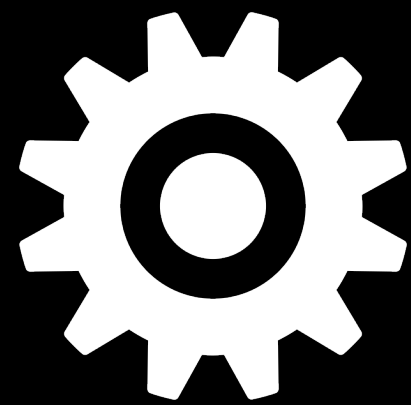
“The definition of insanity is doing the same thing over and over again and expecting a different result.”

Einstein did not actually say this

Reduce risk, add simplicity

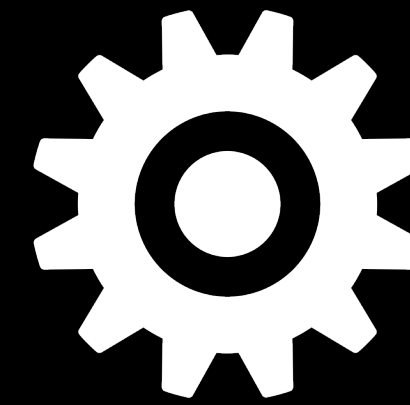
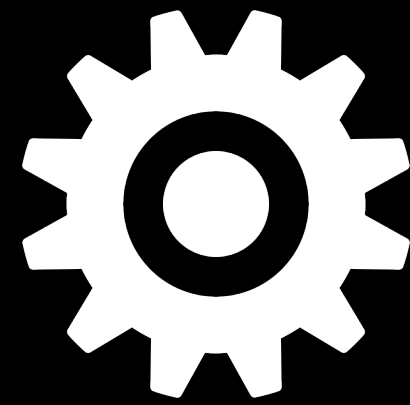
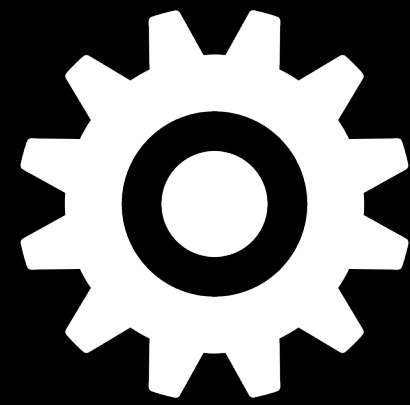
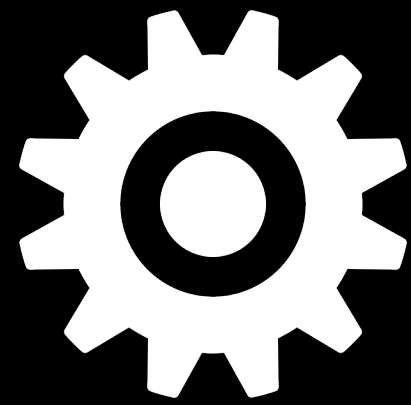
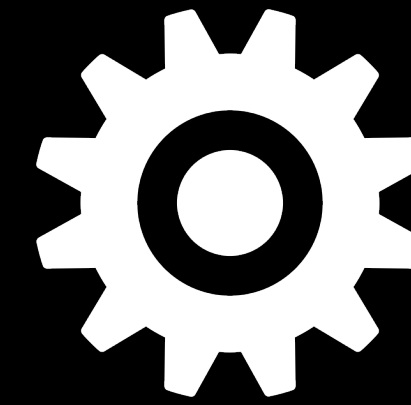
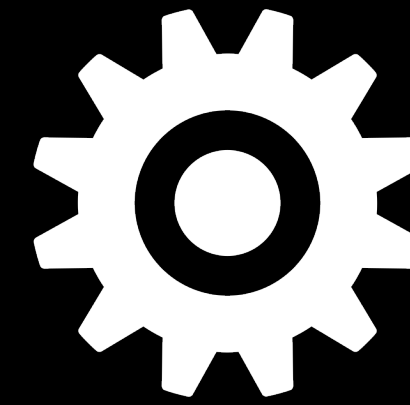
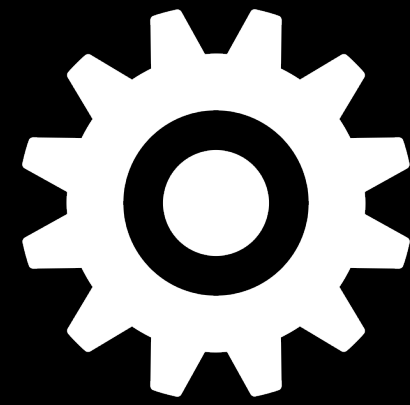
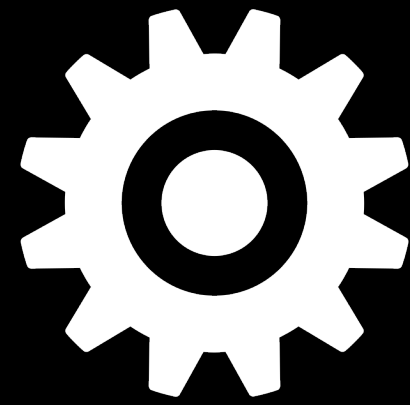
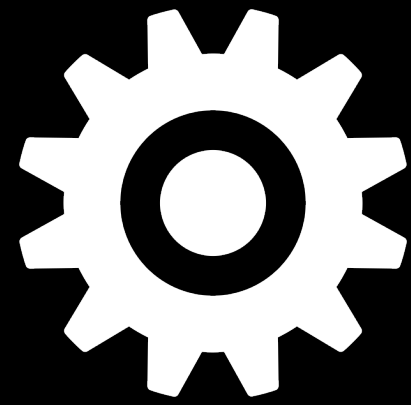


Ten parts at 95% reliability



~40% chance of failure

$$0.95^{10} = 0.5987$$



More dependencies = more risk

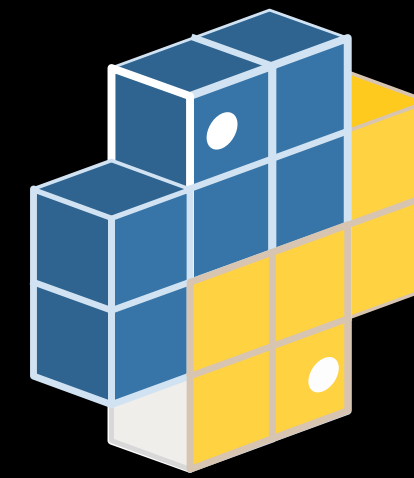
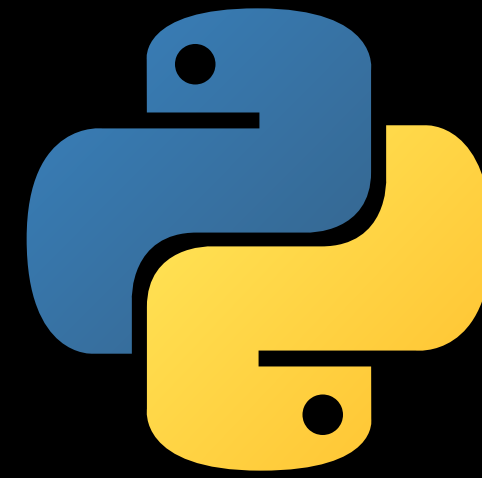
Dependency risks

- Bad license
- Missing entirely
- Package compatibility clash
- Runtime compatibility clash
- Breaking changes
- Deprecation or EOL
- Security compromised
- Abandonware

Not all dependencies are equal

- Why use an intermediary when you can just call the CLI?
- If you only use one function, why load a whole library?
- Why load a whole library for minor features?

Adding simplicity



The baseline is full of features!

- Shell, Node and Python scripting
- NPM and PIP for packages
- NPM Workspaces
- NodeJS Test Runner for unit tests
- Cypress for component, E2E and accessibility tests
- And of course the web platform itself - HTML, CSS, JS

Principles

- Use as little as possible
- Choose native where possible
- Make unproven technology as fungible as possible

Hard “no”

- Powershell
- Yarn, Webpack, Gulp, Babel
- Monorepo tools
- Intermediary libraries
- React-specific test frameworks
- Gatsby



Additions

- Vite for docs and React test fixtures
- SCSS for our design token integration
- STMUX to run things in parallel
- onchange for hot reloading
- Plus the actual executables like SASS, Stylelint, etc



/apps

/dashtest



/docs



/reacttest



/packages

/core

/dash



/react



/tests



Remaining complexity

- I *really* wish NPM run would build in a watcher
- Extracting TS definitions remains disappointingly difficult
- SCSS may still go, we barely need it
- Github Actions is intrusive, we may push more to shell scripts
- Building Dash is a bit niche and strange, but probably not of concern to many

So... do we like it?

Before

- 5610 dependencies (!!!)
- ~5-10 minute first-time setup
- ~1-2 minutes to start

After

- 1525 dependencies (-72%)
- ~2-3 minute first-time setup (-70%)
- ~25-30 seconds to start (-50%)

Less,

but better.

- 72% less dependencies
- 70% less build time
- 50% less start time

- All existing functionality maintained
- New components added
- Massively improved test coverage

Best of all...

Contributors no longer waste hours getting the project to run.

They ship work that makes a difference to customers.

Activity

Show: [All](#) [Comments](#) [History](#) [Work log](#) [Checklist history](#)



Jessica Gruen changed the Status 3 May 2024 at 12:25

[IN REVIEW](#) → [DONE](#)



Jessica Gruen updated the Resolution 3 May 2024 at 12:25

None → Done



Gemma Croad updated the Fix versions 3 May 2024 at 09:56

None → 5.1.0



Jessica Gruen changed the Status 3 May 2024 at 09:54

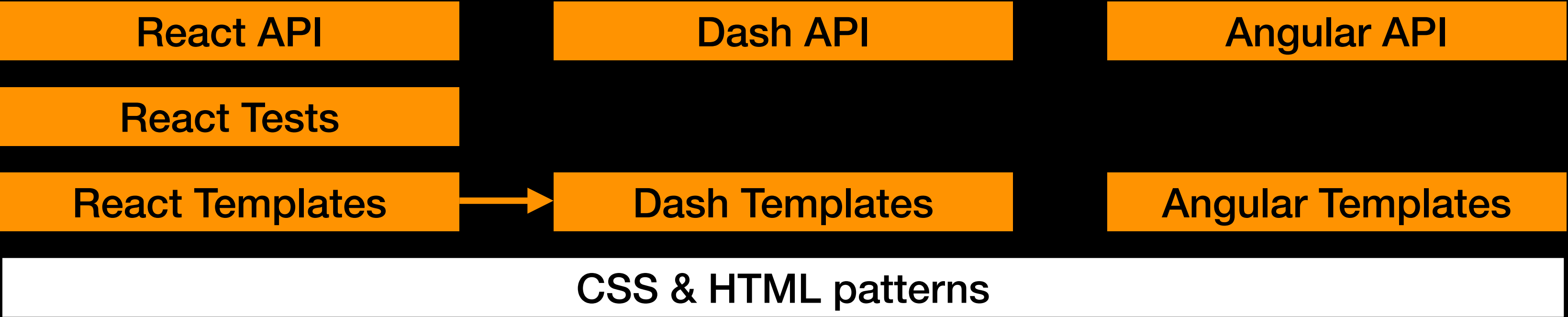
[IN PROGRESS](#) → [IN REVIEW](#)



Jessica Gruen changed the Status 2 May 2024 at 14:00

[SELECTED FOR DEVELOPMENT](#) → [IN PROGRESS](#)

What's next?



Vue!

HTMX!

Rshiny!

React API

Dash API

Angular API

React Tests

React Templates

Dash Templates

Angular Templates

CSS & HTML patterns

Svelte!

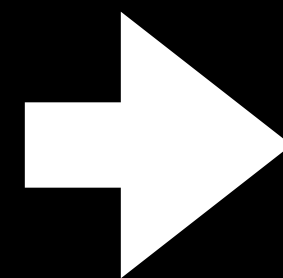
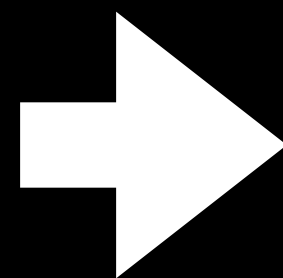
Streamlit!

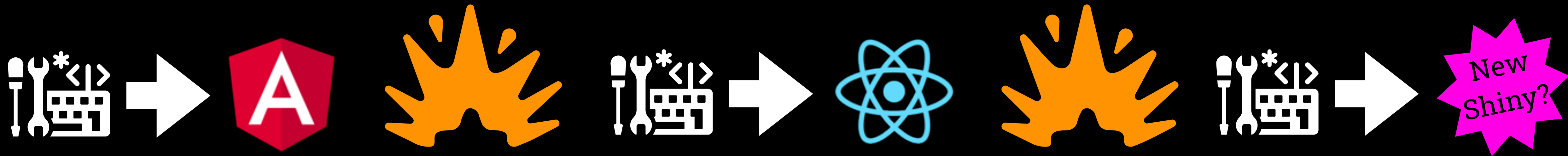
Web components?

**You'll never guess which new
and shiny thing I proposed!**



A business case for web components









Considerations for tech choices

- Strength of solution and its ecosystem
- Security and licensing
- Cost to buy
- Cost to migrate/roll out
- Cost to maintain
- Cost of training
- Impacts to hiring and onboarding
- ...and how long before you ask me to approve all of this all over again?

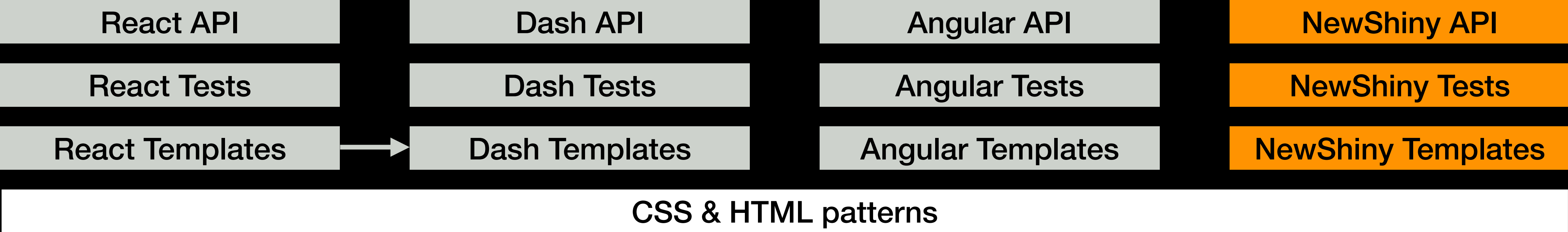
Web components

- Ecosystem - the web 
- License - free 
- Migration - incremental 
- Longevity - possibly indefinite 

You can keep your current framework.

You can support multiple frameworks.

This choice is the lowest risk you can offer to build and maintain a UI layer.



React API

Dash API

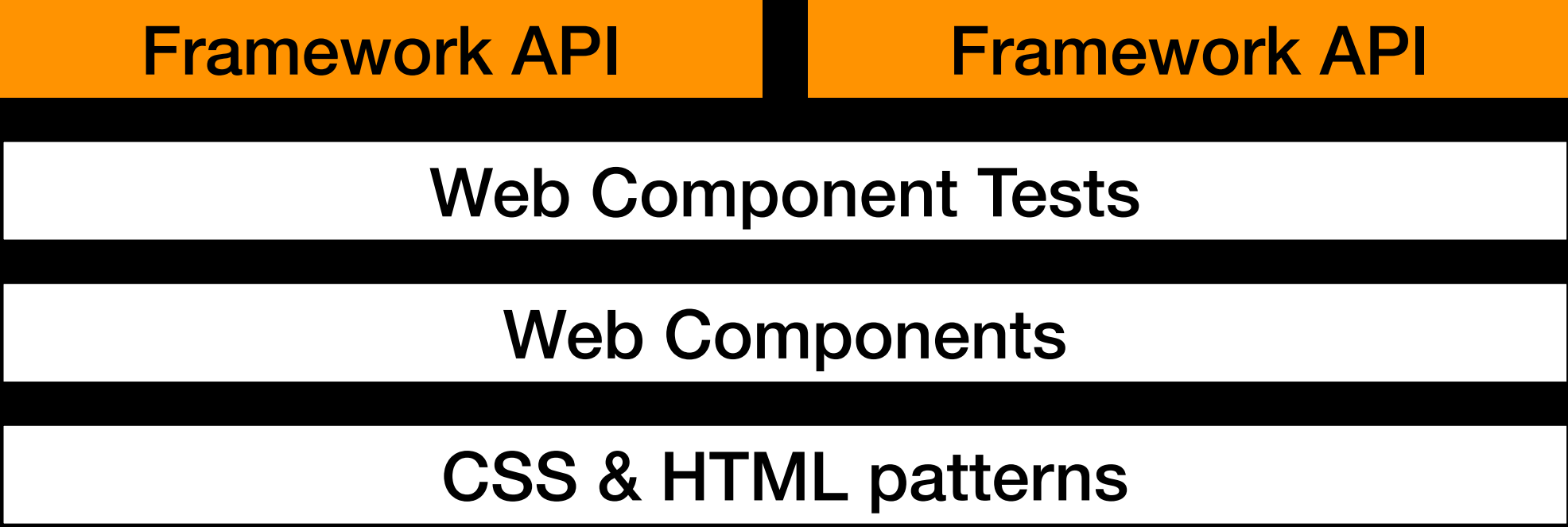
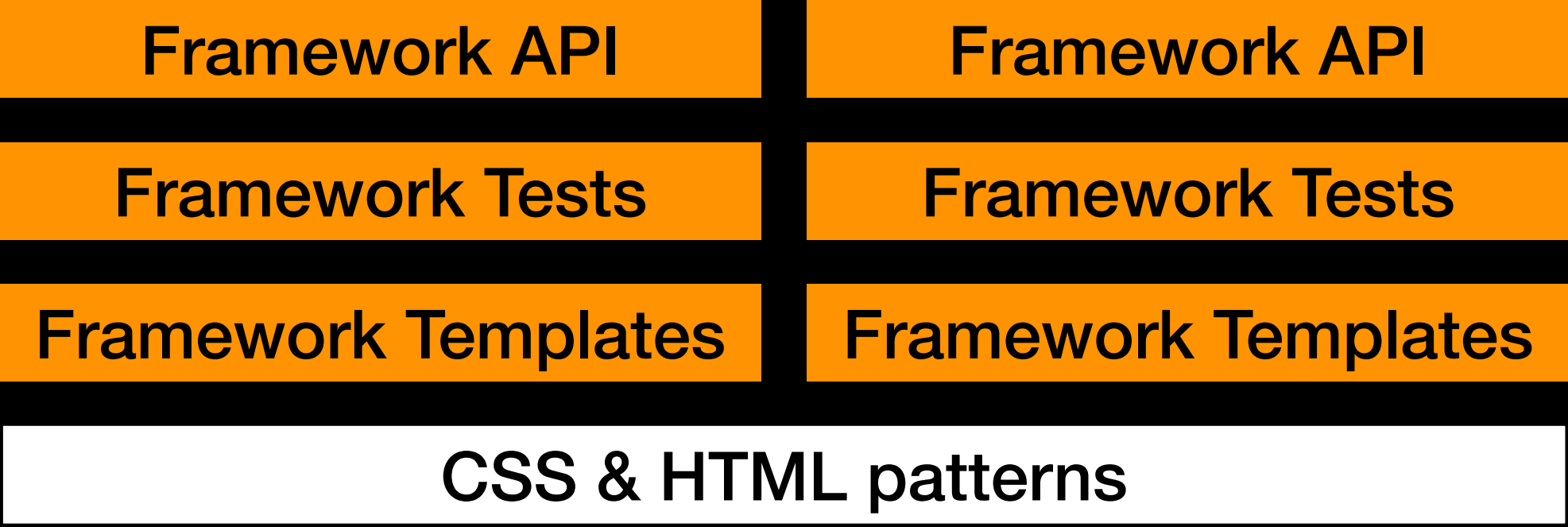
Angular API

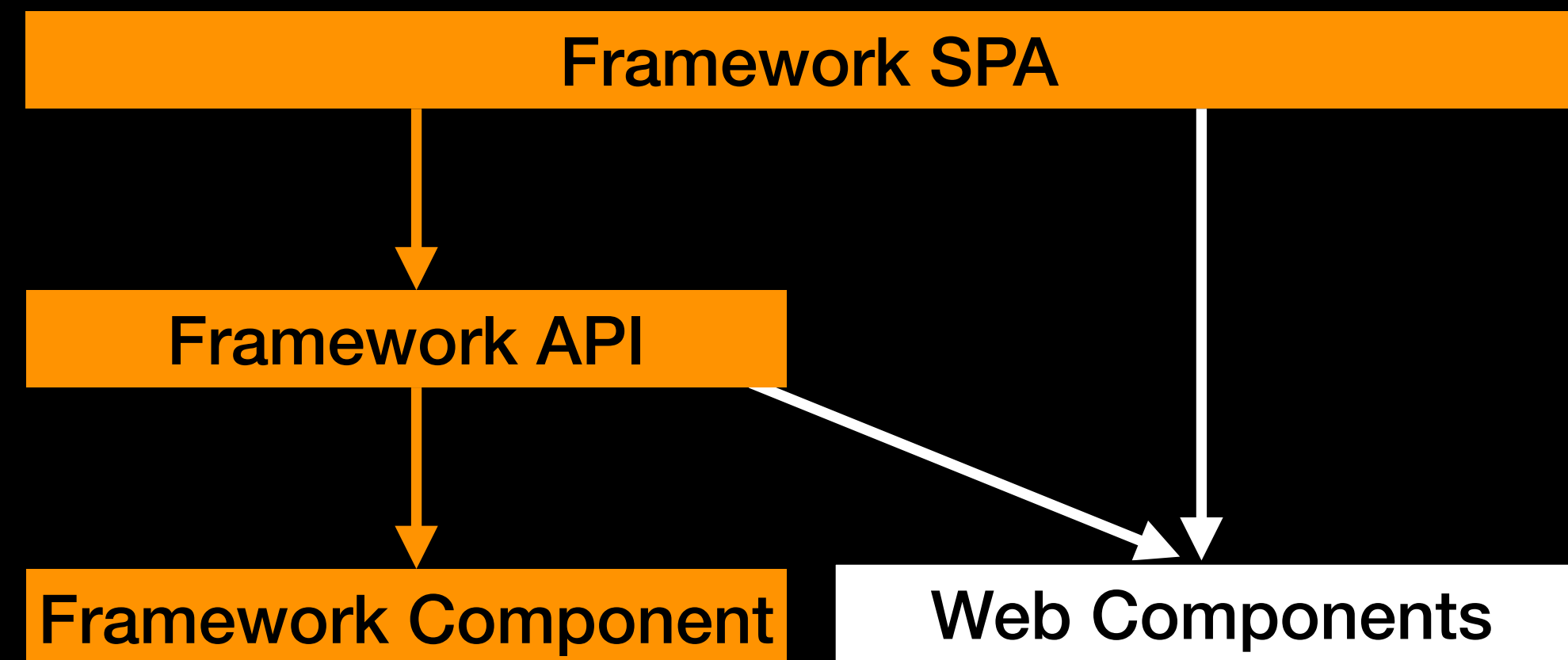
NewShiny API

Web Component tests

Web Components

CSS & HTML patterns





Mid 1990s

Early 2000s

Mid 2000s

Mid 2010s

Mid 2020s

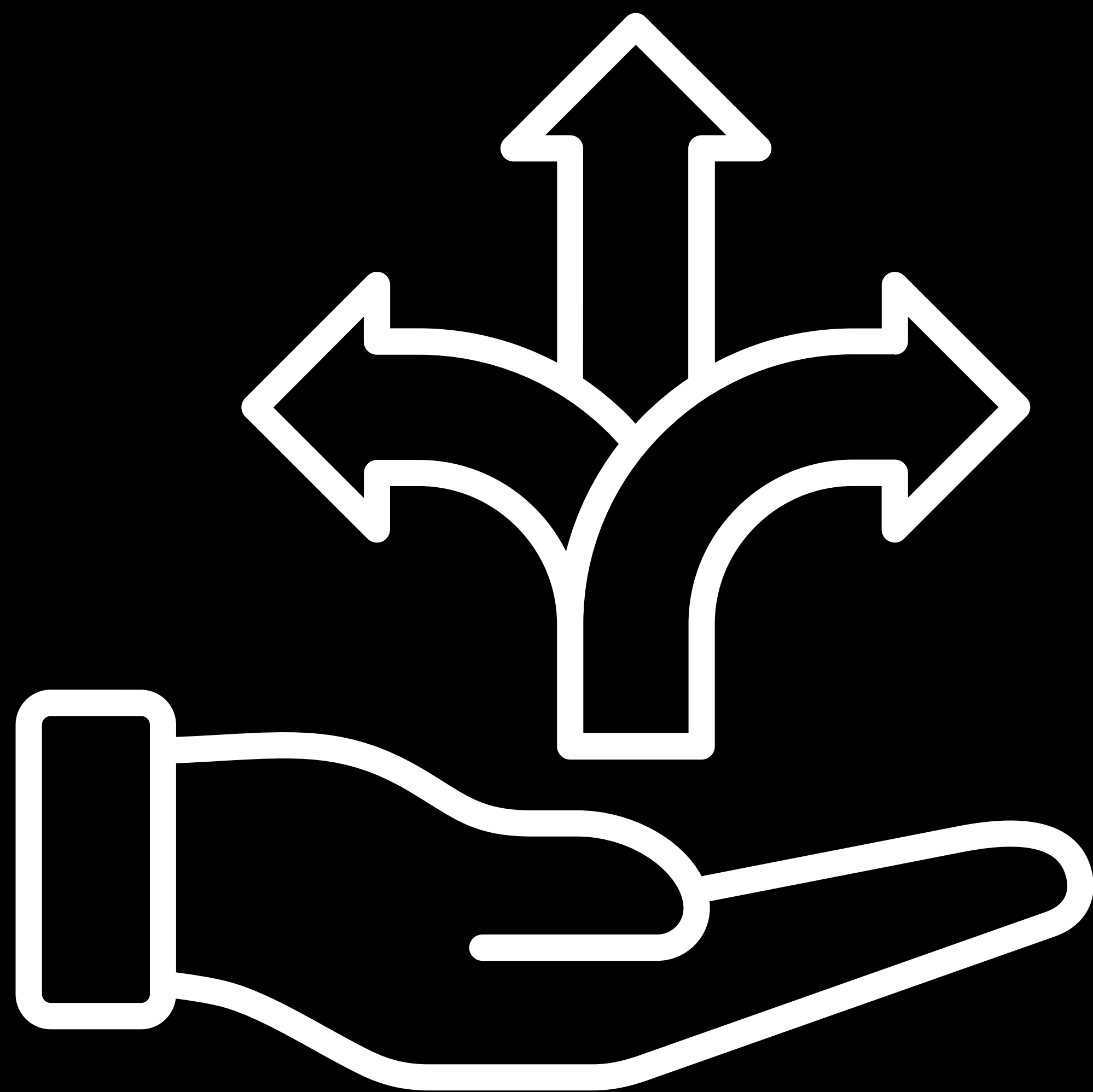
HTML

SSR

DOM/Ajax

SPA

Native web?



“Perfection is achieved,
not when there is nothing more to add,
but when there is nothing left to take away.”

Antoine de Saint-Exupéry

Less, but better.

Ben Buchanan / 200ok.blog / [@200ok@mastodon.social](https://mastodon.social/@200ok)