No Build

Making Javascript (and TypeScript) Fun Again

Fabian Paus

- 12+ years of C++ experience

  (backend services, robotics)
- ~3 years of web development
- Currently working on customer IAM

https://xkcd.com/303/

Building Websites has Become Complex

Type checking and compiling TypeScript to JavaScript

Bundling multiple files into a single file, e.g. webpack

Minification and tree shaking, e.g. esbuild

Building CSS, e.g. tailwind

Polyfills, …

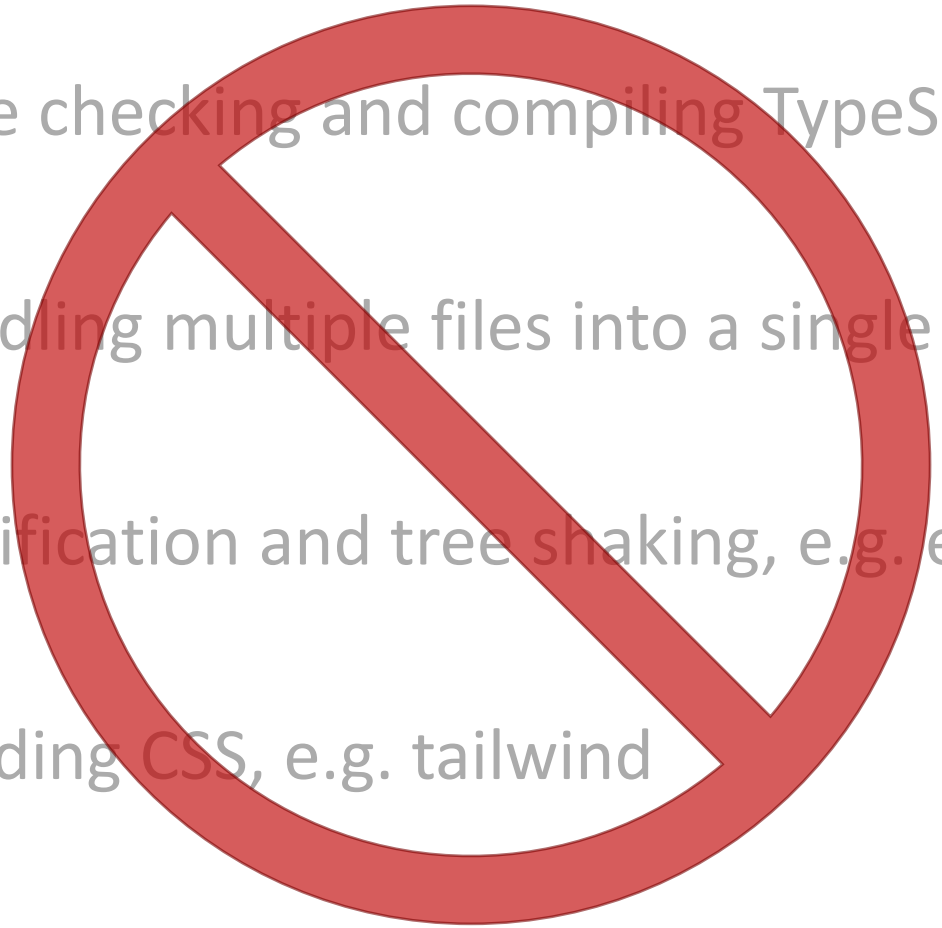
What is No Build?

Type checking and compiling TypeScript to JavaScript

Bundling multiple files into a single file, e.g. webpack

Minification and tree shaking, e.g. esbuild

Building CSS, e.g. tailwind

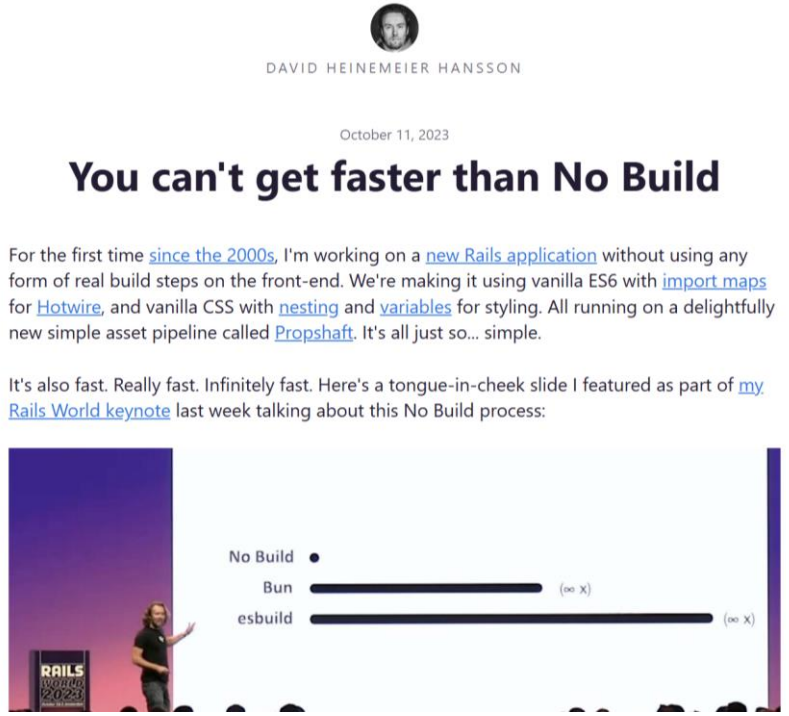


Polyfills, …

What is No Build?

A trend to build web applications **without a build system**

https://jvns.ca/blog/2023/02/16/writing-javascript-without-a-build-system/



https://world.hey.com/dhh/you-can-t-get-faster-than-no-build-7a44131c

https://mxb.dev/blog/buildless/

What is No Build? 

A trend to build web applications **without a build system**

Plan for today:

How to **type check** without compiling

Why **bundling** is not as important as you think

Why you might consider not to **minify**

TypeScript: Type Checking for Javascript

- Define types
- Annotate the code with types

➔ Catch errors at compile time

Javascript:

```
async function getWeather(
    latitude,
    longitude) {
```

TypeScript:

```
interface WeatherData {
  latitude: number;
  longitude: number;
  elevation: number;
  current: CurrentWeather;
  hourly: WeatherSeries;
}

async function getWeather(
  latitude: number,
  longitude: number
): Promise<WeatherData> {
```

TypeScript: Compilation

TypeScript:

```typescript
interface WeatherData {
  latitude: number;
  longitude: number;
  elevation: number;
  current: CurrentWeather;
  hourly: WeatherSeries;
}

async function getWeather(
  latitude: number,
  longitude: number
): Promise<WeatherData> {
```

tsc

Javascript:

```javascript
async function getWeather(
    latitude,
    longitude) {
```

execute

TypeScript Compiler (tsc)

- Checks for type errors
- Removes all type annotations
- Translates TypeScript specific constructs (enums, namespaces)

Javascript Runtime

- Executes Javascript code
- Provided by browsers, Node.js
- Examples: V8, SpiderMonkey

JSDoc: Annotate Types in Javascript

TypeScript:

```typescript
interface WeatherData {
  latitude: number;
  longitude: number;
  elevation: number;
  current: CurrentWeather;
  hourly: WeatherSeries;
}

async function getWeather(
  latitude: number,
  longitude: number
): Promise<WeatherData> {
```

Javascript with JSDOC:

```javascript
/**
 * Get the weather data for a location
 * via the Open-Meteo API.
 *
 * @param {number} latitude
 * @param {number} longitude
 * @returns {Promise<WeatherData>}
 */
async function getWeather(
  latitude,
  longitude) {
```

JSDoc

- API documentation generator for Javascript
- Plain Javascript with comments ➔ **no translation**
- tsc and IDEs can **type check** your code
- Almost as powerful as TypeScript (no enums, namespaces)

tsconfig.json

```jsonc
// Enable type checking for Javascript
"allowJs": true,
"checkJs": true,
// We do not need any output
"noEmit": true,
```

Node.js: Strip Types

TypeScript:

```typescript
interface WeatherData {
  latitude: number;
  longitude: number;
  elevation: number;
  current: CurrentWeather;
  hourly: WeatherSeries;
}

async function getWeather(
  latitude: number,
  longitude: number
): Promise<WeatherData> {
```

--experimental-strip-types

- Let's you execute TypeScript directly
- Replaces annotations with whitespace
- No type checking
- **Default** starting with Node.js 23.6

```
# With Node.js < 23.6.0
node --experimental-strip-types example.ts

# With Node.js >= 23.6.0
node example.ts
```

Node.js: Transform TypeScript

TypeScript:

```typescript
interface WeatherData {
  latitude: number;
  longitude: number;
  elevation: number;
  current: CurrentWeather;
  hourly: WeatherSeries;
}

async function getWeather(
  latitude: number,
  longitude: number
): Promise<WeatherData> {
```

--experimental-**transform**-types

- Let's you execute TypeScript directly
- Transforms TypeScript features (enums, namespaces)
- Requires source maps for debugging

```
# Transform TypeScript features before execution
node --experimental-transform-types example.ts
```

Demo: Type Checking without Build Step

- Simple script to get the current temperature
  - Query the Open-Meteo API
  - Parse the results (JSON)
  - Output the current temperature


```
PS C:\Users\pausf\Documents\2025\no-build-intro\jsdoc> node .\index.mjs
Current temperature in Waldkirch: 13.9 °C
Written weather history to out/weather.csv
```

- TypeScript with and without build step
- Javascript with JSDoc

What is No Build? 

A trend to build web applications **without a build system**

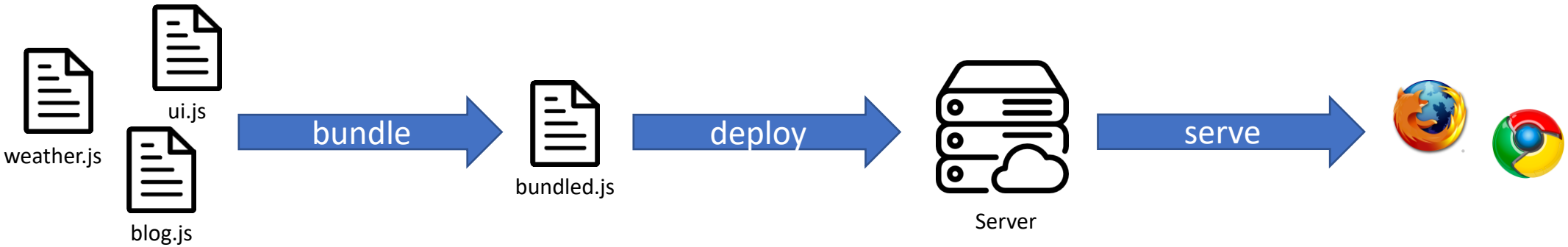
Plan for today:

How to **type check** without compiling

Why **bundling** is not as important as you think

Why you might consider not to **minify**

What is Bundling?



- Bundle all your Javacript files into a single file
- This bundle is deployed to your server
- Browsers only need to **download one Javascript file**

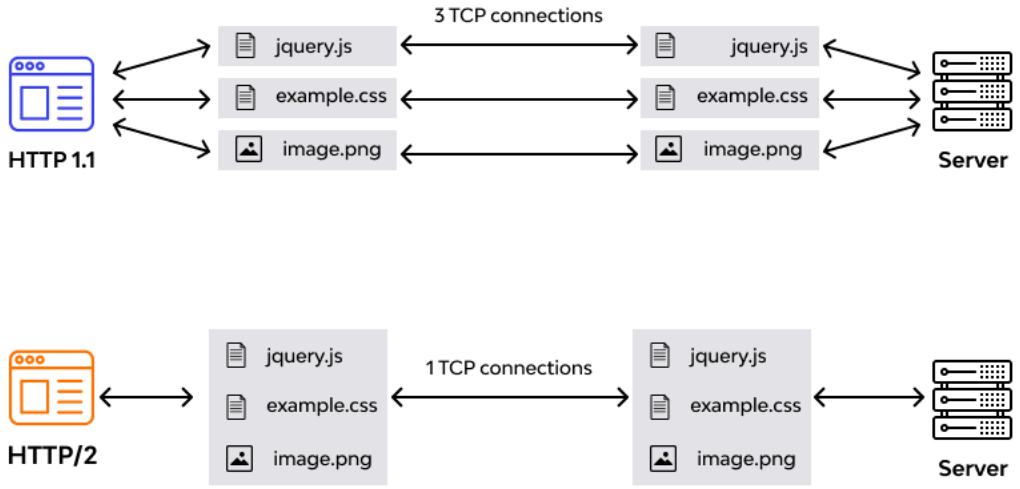
➔**Reduce round** trips between browser and server

➔Resolve dependencies at bundle time (import, require)

Performance Improvements?

Bundling: Only one file needs to be downloaded
➔ One HTTP GET request, one network connection

HTTP 2.0: Only one TCP connection



Caching for bundled code:

- A change in a **single source** file invalidates the complete bundle
- Browser needs to download the complete bundle

Caching for non-bundled code:

- Only the **changed files** are invalidated
- Download size can be much smaller

Import Maps for Dependency Resolution

Resolving dependencies via **import maps**

- Use modern ESM syntax to declare dependencies (import)
- Define file locations in HTML

HTML

```
<script type="importmap">
  {
    "imports": {
      "square": "./modules/shapes/square.js",
      "circle": "https://example.com/shapes/circle.js"
    }
  }
</script>
```

JS

```
import { name as squareName, draw } from "square";
import { name as circleName } from "circle";
```

What is No Build? 

A trend to build web applications **without a build system**

Plan for today:

How to **type check** without compiling

Why **bundling** is not as important as you think

Why you might consider not to **minify**

What is Minification?

```
async function getWeather(
  latitude,
  longitude) {

  const url = new URL("https://api.open-meteo.com/
  url.searchParams.append("latitude", latitude.toF
  url.searchParams.append("longitude", longitude.t
  url.searchParams.append("current", "temperature_
  url.searchParams.append("hourly", "temperature_2

  const result = await fetch(url);
  if (!result.ok) {
    throw new Error(
      "Weather error: " + result.status + " " + re
    );
  }
```


```
"use strict";import c from"node:assert";import{wr
`;for(let e=0;e<r.time.length;++e){const a='"'+r.
i='"'+r.temperature_2m[e]+'"',m=a+","+i+``;t+=m}
await u(o,t)}const h=48.09585,d=7.96371,n=await l
console.log("Current temperature in Waldkirch:",n
```

Reduce code size by:

- Removing comments and whitespace
- Shorten variable and function names

➔ Does **not** affect how the browser **executes** the code

Why not to Minify?

```
"use strict";import c from"node:assert";import{wr
`;for(let e=0;e<r.time.length;++e){const a='"'+r.
i='"'+r.temperature_2m[e]+'"',m=a+","+i+``;t+=m}
await u(o,t)}const h=48.09585,d=7.96371,n=await l
console.log("Current temperature in Waldkirch:",n
```

It becomes **difficult to learn** and discover:

- New developers see an awesome websites
- They want to know how they work → View Source
- They see this **gibberish**

For our and the web's future, it's the morally right thing to do!

What is No Build? 

A trend to build web applications **without a build system**

Plan for today:

How to **type check** without compiling

Why **bundling** is not as important as you think

Why you might consider not to **minify**

Thanks for your Attention!

Example project on GitHub:

- https://github.com/fabian-paus/no-build-introduction

Blog posts:

- https://world.hey.com/dhh/you-can-t-get-faster-than-no-build-7a44131c
- https://jvns.ca/blog/2023/02/16/writing-javascript-without-a-build-system/
- https://mxb.dev/blog/buildless/

Videos:

- Interview with DHH: https://www.youtube.com/watch?v=mTa2d3OLXhg&t=3000s
- Ruby on Rails 2023: https://youtu.be/iqXjGiQ_D-A?t=1663