FRONT-END BUILD TOOLS

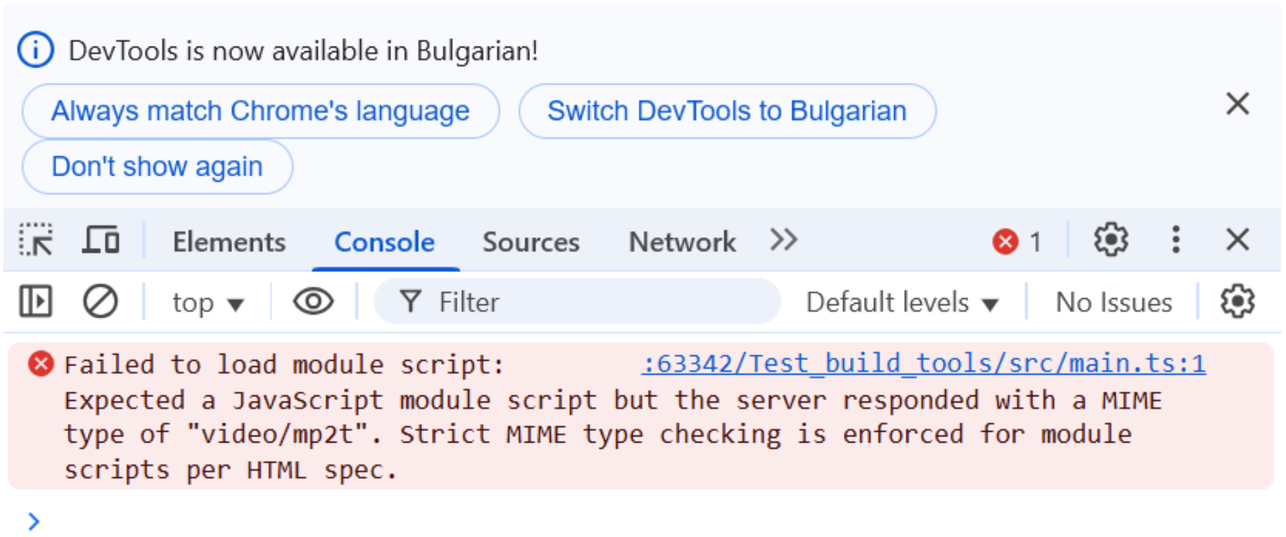
Какво представляват?

- **Build tools(или инструментите за предварителна обработка на сорс кода)** представляват софтуерни инструменти, които автоматизират процеси, свързани с подготовката и оптимизацията на изходния код, така че той да бъде готов за изпълнение в браузърна среда.

Основни действия:

- **Компилация** – преобразуване на код, написан на езици като TypeScript или Sass, в стандартен JavaScript или CSS, разбираем за браузърите
- **Bundling (обединяване)** – комбиниране на множество файлове в един или няколко, с цел по-малък брой заявки към сървъра и по-бързо зареждане на приложението
- **Minification (минификация)** – премахване на ненужни символи (интервали, нови редове, коментари) с цел намаляване на размера на файловете
- **Tree shaking** – премахване на неизползван код от крайния пакет
- **Live reloading и наблюдение на файлове** – автоматично презареждане на приложението при промени в кода, което ускорява процеса на разработка

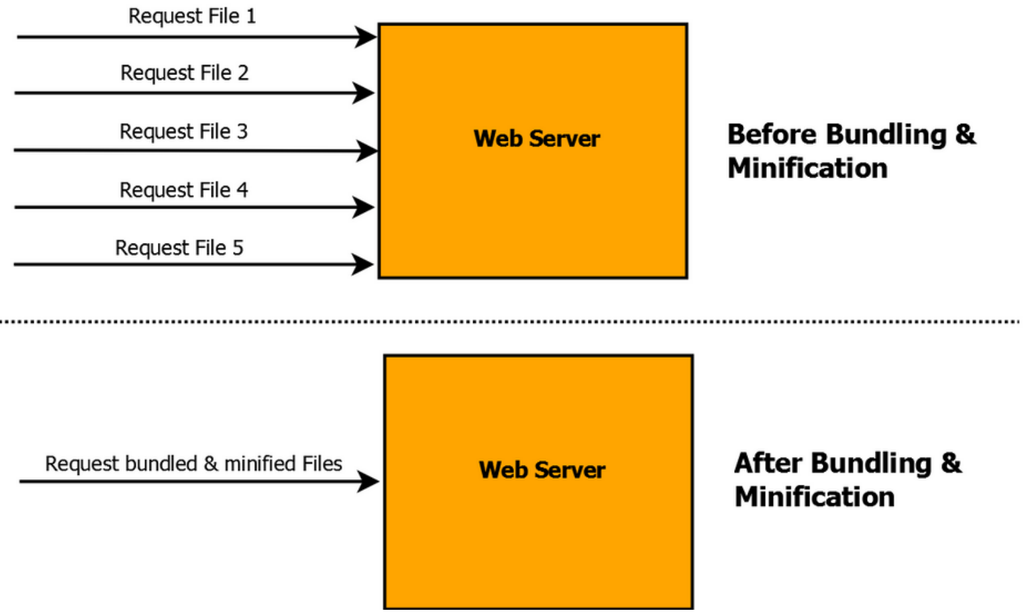
Защо са необходими?



Проблеми без build tools:

- Ръчно компилиране на TypeScript
- Много отделни файлове
- Бавно зареждане
- Грешки в браузъра с .ts файлове

Компилиране на TypeScript код до JavaScript



- Използване на инструмента tsc (официалният компилатор на TypeScript)
- Ръчно
- Import-ите трябва да са към .js файлове
- tsc не прави bundling, minification, tree shaking

- Използване на build tools
- Автоматично
- Позволява използване на import-и към .ts файлове
- Има оптимизации

WEBPACK

Защо използваме Webpack? 🤔

🚫 Без Webpack	✔ С Webpack
Зареждаме поотделно файлове в HTML чрез **<script>** и **<link>**	Един **<script src="bundle.js">** зарежда всичко
Не можем да импортваме **CSS** или изображения в **JS**	Можем да **имортваме CSS файл** в TypeScript
Трябва да следим **ръчно** реда на скриптове	Webpack **автоматично** подрежда зависимостите
Не можем да използваме **TypeScript** директно	Webpack компилира TS в JS с **ts-loader**
Трудна оптимизация за производство	Webpack **минифицира** и **премахва** ненужен код автоматично

Как работи Webpack?

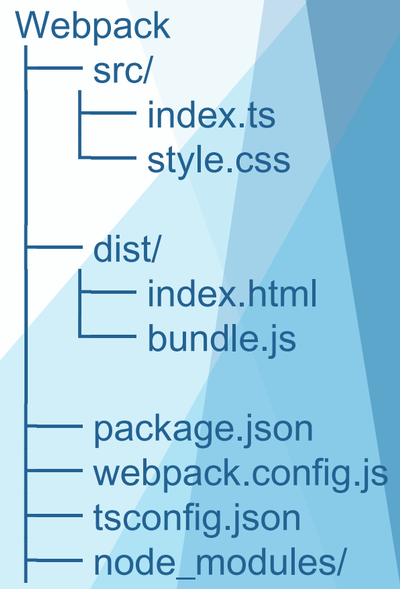
1. Старт от главен файл (entry)
2. Чете зависимостите (import-и)
3. През Loaders обработва типове файлове
4. Обединява всичко в bundle.js

Основни компоненти на Webpack

- Entry – откъде да започне
- Output – къде да запише резултата
- Loaders – как да обработи файлове
- Plugins – разширения за оптимизация
- Bundling – събиране в един файл

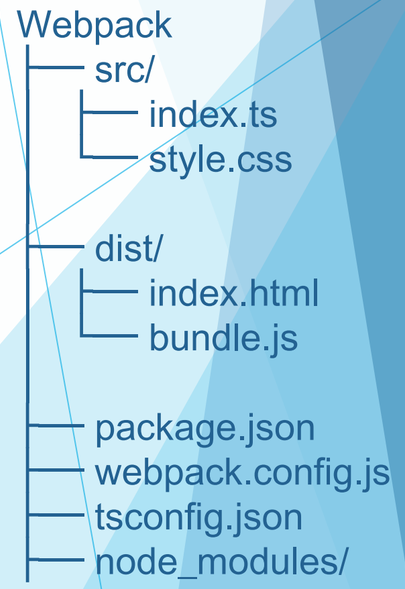
src/index.ts

```ts
import './style.css';

function component(): HTMLElement {
  const element = document.createElement('div');
  element.innerHTML = 'Здравей, Webpack!';
  element.className = 'greeting';
  return element;
}

document.body.appendChild(component());
```



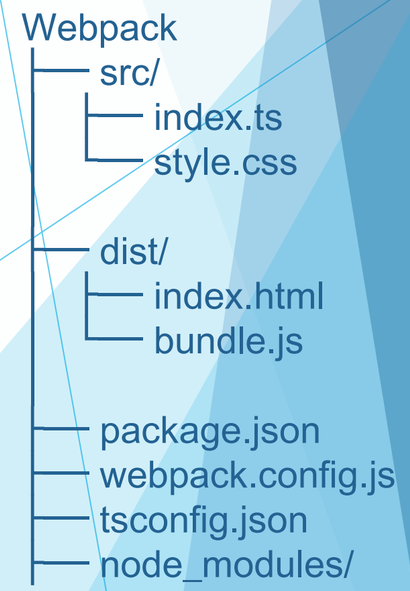
src/style.css

```css
.greeting {
    font-size: 24px;
    font-family: sans-serif;
    text-align: center;
    margin-top: 30px;
    color: purple;
}
```



dist/index.html

```html
<!DOCTYPE html>
<html lang="bg">
<head>
  <meta charset="UTF-8">
  <title>Webpack Демонстрация</title>
</head>
<body>
  <script src="bundle.js"></script>
</body>
</html>
```



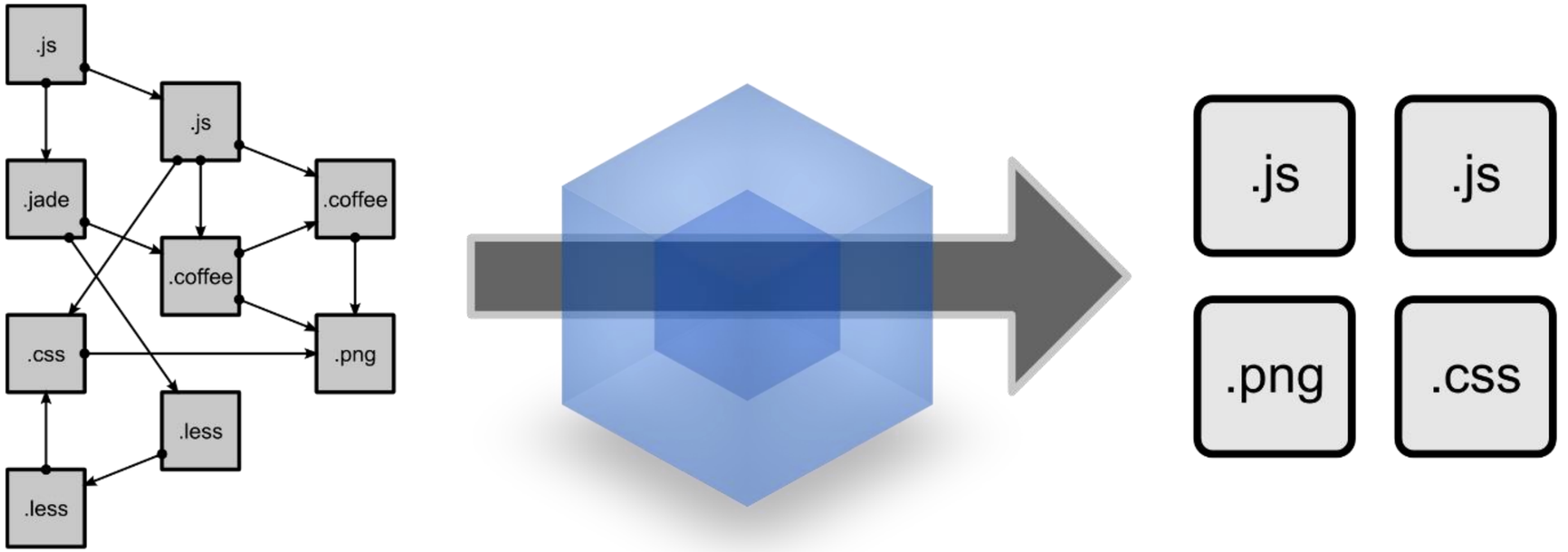
webpack.config.js

```
const path = require('path');

module.exports = {
  mode: 'development',
  entry: './src/index.ts',
  module: {
    rules: [
      {
        test: /\.ts$/,
        use: 'ts-loader',
        exclude: /node_modules/,
      },
      {
        test: /\.css$/i,
        use: ['style-loader', 'css-loader'],
      },
    ],
  },
  resolve: {
    extensions: ['.ts'],
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
};
```

tsconfig.json

```
{
    "compilerOptions": {
      "target": "ES6",
      "module": "ES6",
      "moduleResolution": "node",
      "strict": true,
      "esModuleInterop": true,
      "outDir": "./dist"
    },
    "include": ["src"]
  }
```

dist/bundle.js



SNOWPACK

Защо възниква Snowpack?

- bundler
- 2019 - ESM
- 2020 - Snowpack

Как работи Snowpack?

- **unbundled development подход**
- **Бързо, леко и ефективно разработване**

Snowpack в production среда

- snowpack build
  без bundler
- добавяне на плъгини
  @snowpack/plugin-webpack

WEBPACK

VS

SNOWPACK

Snowpack

- По-бърз
- По-малък
- Подходящ за малки проекти

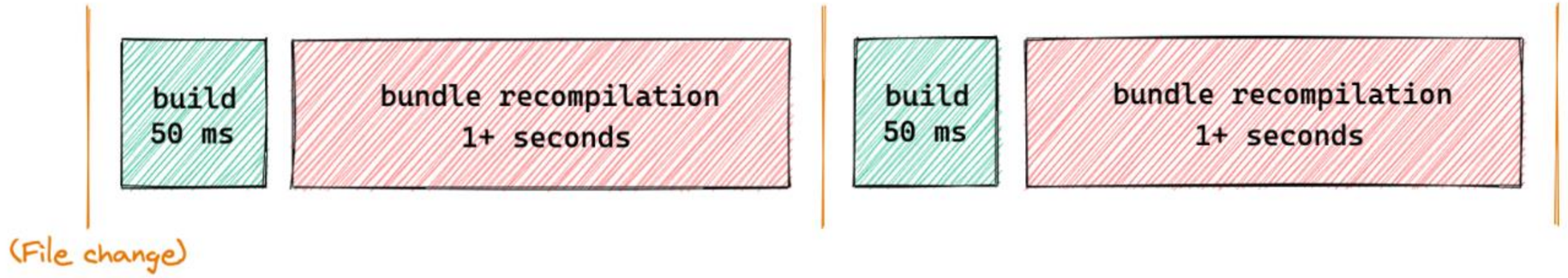
Като Tesla - стартира на мига, модерен

Webpack

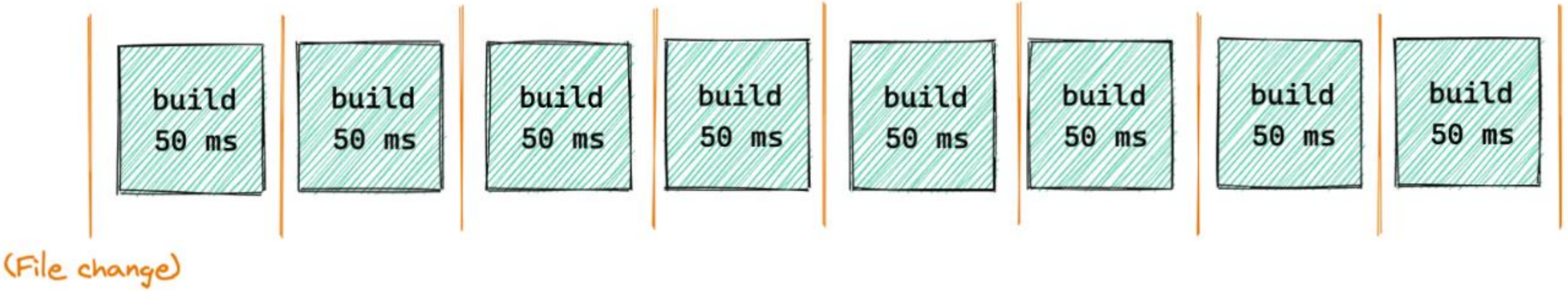
- По-бавен
- По-голям
- Подходящ за по-големи проекти

Като Volvo - малко тромово, но супер надеждно

Bundled (ex: Webpack)



Unbundled (Snowpack)

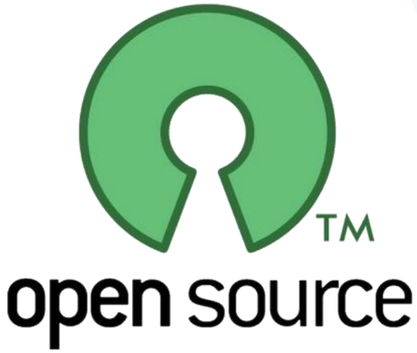


Parcel: Бърз и лесен уеб инструмент за изграждане

Нулева конфигурация, висока производителност

PARCEL

Какво е Parcel?



- Parcel е уеб инструмент за изграждане (bundler) с отворен код.
- Създаден през 2017 г. от Devon Govett.
- Цел: Опростяване на процеса на разработка чрез елиминиране на сложни конфигурации.
- Подходящ за малки и големи проекти – от прости страници до сложни приложения.

Основни характеристики

- Нулева конфигурация
- Бързина
- Вграден Dev сървър
- Автоматична оптимизация
- Поддръжка на много различни файлове

Предимства на Parcel

- Лесен за начинаещи
- Бърз старт на проекти
- Скалируемост
- Вградени инструменти
- Поддържа модерни технологии като React Server Components и MDX

Как работи Parcel

- Входна точка HTML/JS → index.html
- Bundling
- Оптимизация → minimize/tree-shake
- Изход → ./dist for prod

Parcel срещу конкуренцията

- Webpack
  - Изисква настройка
- Snowpack
  - Изисква конфигурация за модули
- Предимства на Parcel
  - Out of the box

Кога Parcel е най-добрият избор?

- Бързи прототипи
- Малки до средни проекти
- Екипи с фокус върху кода
- Модерни фреймуърки

Инсталация и начало

```
npm install parcel --save-dev

npx parcel index.html

npx parcel build index.html
```

Примерен проект с TypeScript и CSS

```html
<!DOCTYPE html>
<html lang="bg">
  <head>
    <title>Parcel с TypeScript и CSS</title>
  </head>
  <body>
    <h1 id="greeting">Здравей, Parcel!</h1>
    <script type="module" src="./app.ts"></script>
  </body>
</html>
```

```typescript
import "./style.css"; // Parcel magic

interface Greeting {
  message: string;
  color: string;
}

const greeting: Greeting = {
  message: "Parcel работи с TypeScript!",
  color: "blue",
};

const element = document.getElementById("greeting") as HTMLElement;
element.innerText = greeting.message;
element.style.color = greeting.color;

console.log(greeting.message);
```

```css
#greeting {
  font-family: Arial, sans-serif;
  font-size: 36px;
  text-align: center;
  padding: 20px;
  background-color: #f0f0f0;
  border-radius: 8px;
  transition: all 0.3s ease;
}

#greeting:hover {
  background-color: #e0e0ff;
  transform: scale(1.05);
}
```

Благодарим за вниманието!

Презентирали:

- *Десислава Добрева- 0MI0700129*
- *Поли Янева- 2MI0700143*
- *Севда Иванова- 3MI0700109*
- *Иван Стойнев- 6MI0700164*
- *Александър Тагамлицки- 2MI0700171*