webpack

BABEL

Co to jest Babel

Babel to narzędzie pozwalające transpilować (kompilować) kod JavaScript w jakiejś wersji (także rozszerzoną składnię JSX i metajęzyk TypeScript) do kodu JavaScript w innej wersji. Najczęść "wspólczesny" JavaScript zamieniamy na wersję "starszą" JavaScript, która sprawia, że nasza strona/aplikacja będzie dostępna także w starszych przeglądarkach.

ES6+ -> ES5 ; JSX -> ES5 ; TypeScript -> ES5

(składnia, API)

Skąd nazwa Babel?

Nazwa od jakiejś ryby, której nazwa pochodzi od wieży babel

- jeśli od wieży babel, to wymowa (w uproszczeniu ;)), będzie mniej więcej wyglądała tak: “bejbyl”.
- ale jakiś czas temu by rozwiać wątpliwości twórcy babel przyjęli wymowę, która była bardziej popularna a mianowicie “babul” (od babble - bełkot)

Jak działa babel - składnia (syntax)

Wiele przeglądarek może nie rozpoznać nowej składni (ES6) np. składni klas (class), nowych zmiennych (let, const) czy funkcji strzałkowych (()=> {}). W istocie te elementy nie wprowadzają, poza składnią, nic czego nie dałoby się napisać za pomocą starszej wersji JavaScript (ES5). Babel pozwala więc zamienić kod napisany w nowej składni (ES6+) na składnie starszą (ES5), przy czym nie zmienia to w żaden sposób działania napisanego przez nas programu. Dzięki temu przeglądarki, które nie rozumieją nowej składni będą mogły uruchomić nasz program.

Babel zamieni ES6+ na ES5. Pozwala też transpilować składnię JSX (React) oraz TypeScript do “czystego JavaScript”

Zobaczmy w praktyce...

Jak działa babel - polyfill

Jednak nowy (ES6 i kolejne) JavaScript to nie tylko nowa składnia. To także nowe rozwiązania, czyli nowe obiekty i funkcje (określamy je jako API). Tutaj nie da się po prostu zamienić składni na starą, bo są to zupełnie nowe rozwiązania (nie mają więc nawet “starej” składni). Trzeba dostarczyć te nowe rozwiązania (obiekty, funkcje) środowisku uruchomieniowym.

Po co polyfill? Przeglądarka może nie rozpoznać wielu nowych rozwiązań (wielu elementów API), np. w postaci promisów czy map. W tym miejscu pojawia się narzędzie, które nazywamy polyfill.

Jak działa babel - polyfill

Polyfill to kod, który dostarczamy przeglądarce (dotyczy to nie tylko JavaScript ale też innych rozwiązań np. CSS-a). Zadaniem narzędzia babel polyfill jest dostarczyć przeglądarce brakujące API (brakujące rozwiązania - obiekty i funkcje), tak by przeglądarka rozpoznała składnie i potrafiła wykonać dostarczony kod.

Zobaczmy w praktyce...

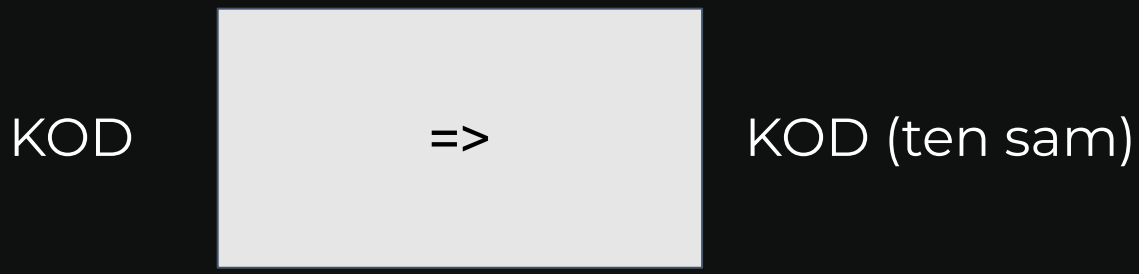
Babel bez webpacka? Oczywiscie!

Babel z webpackiem? Oczywiscie!

@babel/core

npm install @babel/core --save-dev

Podstawowe “maszyna” Babel (rdzeń jak sugeruje nazwa). Dzięki niej możliwe jest uruchomienie kompilatora Babel, wprowadzenie danych wejściowych, analiza, przekształcenie i zwrócenie kodu po transpilacji (kompilacji). Przy czym samo uruchomieniu Babel nie zrobi nic z dostarczonym kodem.



@babel/cli

npm install @babel/cli --save-dev

Dzięki Babel CLI możemy wykonywać polecenia Babel w wierszu poleceń. O ile używamy Babel z naszym webpackiem to nie musimy instalować tej paczki.

przykładowo w wierszu poleceń:

```
npx babel index.js --watch --out-file indexES5.js
```

@babel/plugin-proposal-class-properties (przykład pluginu)

npm install @babel/plugin-proposal-class-properties --save-dev

Sercem Babela są pluginy. Plugin może odpowiadać za jakiś drobny obszar (zakres) np. funkcje strzałkowe. Plugin jest instrukcją obsługi dla Babel co zrobić z danym elementem kodu (jak go zmienić).

Przykładowo plugin znajdujący się w tytule odpowiada za możliwość pisania bezpośrednio w klasach także właściwości, a nie tylko metod.

@babel/preset-env

Presety to po prostu zbiory pluginów - przygotowane dla deweloperów podstawowe ustawienie (gotowce). Oczywiście można je rozszerzać instalując kolejne pluginy (a także inne presety).

npm install --save-dev @babel/preset-env (transpiluje ES6+ na ES5, czasami jest niewystarczający i wymaga doinstalowania dodatkowych pluginów)

npm install --save-dev @babel/preset-react (zamienia składnię JSX na JavaScript)

npm install --save-dev @babel/preset-typescript (zamienia TypeScript na JavaScript)

@babel/polyfill

Dodaje APIs ES6+ do środowiska, w którym nie są one dostępne. Nie chodzi tu o składnię, a o możliwości.

Zwróćmy uwagę, że instalujemy babel/polyfill jako wersję produkcyjną.

METAFORA: Polyfill można traktować jako wiadro z kodem źródłowym API. Jeśli polyfill rozpozna że czegoś brakuje (że dane rozwiązanie nie zadziała w danej przeglądarce czy ES5) , czyli że w docelowym środowisku nie ma danego elementu API np. promisów (powiedzmy że przeglądarka ma dziurę), to ją zaszpachluje (polyfill dostarczy to API).

npm install --save @babel/polyfill (--save nie jest potrzebne)

babel-loader

Ten loader pozwala na transpilację kodu JavaScript w webpacku za pomocą Babel.

by używać babel-loader w projekcie musi się znaleźć także babel/core i odpowiednie pluginy (webpack sugeruje preset-env)

npm install -D babel-loader @babel/core @babel/preset-env webpack

+ konfiguracja

domyślnie do ES5 (cały kod)

Ale czasem nie ma potrzeby...

Możemy określić na czym nam zależy, tzn. wskazać zakres przeglądarek na których nasz projekt ma być dostępny. To znaczy jeśli okaże się, że wszystkie wskazane przeglądarki rozpoznają już klasy to nie ma sensu zmieniać ich składni na ES5, podobnie jak nie ma po co dodawać dodatkowego kodu jeśli wszystkie wskazane przeglądarki mają już zaimplementowane Promisy.

/ package.json */*

"browserslist": "> 0.3%, not dead",