Do sprawdzenia dalej: webpack, rollup, esbuild

Bundler	Built-in code splitting	Remove duplicated chunks	built-in watch mode	notes	score
webpack	yes	yes	yes	-	wszystkie trzy wyglądają na tym etapie dobrze - minimalne example działają
rollup	yes	yes	yes	-	
esbuild	yes	yes	yes	wyraźnie szybszy niż reszta - to natywna binarka	
microbundle	yes	no?	yes	nie udało mi się skonfigurować tego aby nie emitowało zduplikowanych chunków - ale może to da się zrobić	raczej bym odrzucił - wydaje mi się bardziej toporny / mniej dopracowany w porównaniu z resztą
browserify	no / another tool is needed	partial - only one common.js with factor-bundle tool	no / another tool is needed - watchify	dla mnie wygląda na starzejącą się technologię, którą użytkownicy próbują podtrzymywać przy życiu obudowując kolejnymi toolami	raczej bym odrzucił - są już nowsze narzędzia

1. Browserify wyraźnie odstaje od reszty bundlerów.
- nie obsługuje sam z siebie code splitting ani watch
- code-splitting może być częściowo zrealiowany przez factor-bundle, ale tylko na poziomie jednego wspólnego chunka common.js: https://stackoverflow.com/questions/21805308/sharing-common-code-across-pages-with-browserify
- watch może być zrealizowane też przez zewnętrzny tool watchify: https://www.npmjs.com/package/watchify

2. Wszystkie bundlery oprócz browserify obsługuja code splitting same z siebie:
- na pierwszy rzut oka wygląda, że wszystkie bundlery działają mniej więcej tak samo
- code splitting wymaga pisania kodu z użyciem import/export zamiast require
- jeśli bundler napotka import w kodzie to generuje dodatkowy plik tzw. chunk (plik js) , a w outpucie dla browsera generowany jest dynamiczny import('chunk.js')

3. Wsystkie bundlery oprócz browserify mają możliwość podania kilku entry pointów.
- wtedy powstaje kilka jawnie wskazanych outputów np. index1.js i index2.js
- większość bundlerów rozpoznaje sytuacje, kiedy dwa różne entry pointy importuje tę samą paczkę - wtedy oba korzystają z tego samego chunka
- wyjątek to microbundle - nie udało mi się skonfigurować tego tak aby powtarzające się chunki emitował tylko raz - ale być może to da się zrobić

4. Wszystkie bundlery oprocz browserify obsługują opcję --watch
- rebuilld odpalany jest zarówno kiedy zmienia się jawnie podany entry point jak i import, od którego zależy

5. Zauważyłem, że code splitting zachowuje się inaczej an różnych bundlerach
- importowane chunki w webpack i microbundle odpalane są za każdym razem kiedy ktoś je importuje - ten sam lib jset odpalany wiele razy
- w pozostałych bundlerach chunki odpalane są tylko za pierwszym razem
- być może to ma coś wspólnego z flagą sideEffects w packages.json w połączeniu z domyślnym zachowaniem bundlera ale nie zbadałem tego

6.Po odrzuceniu microbundle i browserify zostają webpack, rollup i esbuild. Wszystkie 3 wyglądają na tym etapie podobnie.

7. Być może bundlery mogą różnić się stopniem optymalizacji - to może wyjść na bardiej złożonych examplach. Oprócz tego esbuild wyróżnia się szybkością działania.