Do sprawdzenia dalej: webpack, rollup, esbuild, microbundle i może browserify (opis niżej)

Bundler	Resolve import	Resolve isomorphic import	Resolve require	Conditional import	Global consts	Built-in code optimizer	Notes	Score
Webpack	yes	yes	yes	with ifdef-loader	with DefinePlugin	yes		Do sprawdzenia
Rollup.js	@rollup/plugin-node-resolve	@rollup/plugin-node-resolve + browser flag	@rollup/plugin-commonjs + transformMixedEsModules flag	with rollup-plugin-jscc	with rollup-plugin-consts	yes		Do sprawdzenia
Browserify	yes	yes	yes	no?	no?	Brak	Dobre do portu typowo Node-owego kodu na browsera, ale sam browserify nie ma nawet mumifikacji - do dodatkowych zmian trzeba zewnętrzne toole	Trudno powiedzieć czy to się sprawdzi dla całego drzewa. Możemy sprawdzić dalej ale trzeba będzie obudować dodatkowymi toolami.
parcel	with includeNodeModules in package.json	with includeNodeModules in package.json	yes	?	?	Słaby, nie eliminuje martwego kodu z if (stała) - może to kwestia konfiguracji?		Nie udało mi się na szybko skonfigurować aby usuwało serwerowy kod tak jak chcemy. Możemy jeszcze pogrzebać w konfigu.
fuse-box	yes	yes	yes	?	?	Słaby, nie eliminuje martwego kodu z if (stała) - może to kwestia konfiguracji?		Nie udało mi się na szybko skonfigurować aby usuwało serwerowy kod tak jak chcemy. Możemy jeszcze pogrzebać w konfigu.
esbuild	yes	yes	yes	?	yes esbuild --define: DEBUG=true ...	yes Wygląda, że nie działa propagacja stałych. if (false) - usunięta const x=false; If (x) - zostaje	Bardzo szybki - paczka npm to tylko wrapper na natywną binarkę	Według mnie warte sprawdzenia, bo jest bardzo szybkie, ale ponieważ to jest pisane w jakimś natywnym języku to może być ryzyko, że jak kiedyś pojawią się problemy to może być wolniej/trudniej je na bieżąco rozwiązać
vite	yes	yes	no?	?	yes pole define w vite.config.js	yes	Buduje index.html / przetwarza JS zaimportowany w <script type="module" src="..."> Dodaje dużo swojego gruzu - może być trudny w dostosowaniu do naszego kodu	Nacisk na zarządzanie całą aplikacją z plikiem index.html. Dostarcza swój serwer www. Według mnie raczej nie dla nas.
microbundle	yes	yes	no?	?	yes microbundle --define DEBUG=true ...	yes		Według mnie warte sprawdzenia.
snowpack	na domyślnych ustawieniach nie rozwiązuje - nie znalazłem na szybko	na domyślnych ustawieniach podąża ścieżką servera (main w package.json) - nie znalazłem na szybko czy i jak wymusić target browser	na domyślnych ustawieniach nie rozwiązuje - nie znalazłem na szybko	?	?	?	na spodzie używa esbuild, webpack lub rollup Do build trafiają wszystkie pliki z projektu - mam wrażenie, że trudno nad tym zapanować i dostosować do naszych potrzeb	Nie udało mi się na szybko skonfigurować tego aby rozwiązywało require, import i usuwało serwerowy kod tak jak chcemy. Zorientowany raczej jako manager do całej aplikacji - raczej nie dla nas.

	lasso	na domyślnych ustawieniach nie rozwiązuje - nie znalazłem na szybko	na domyślnych ustawieniach nie rozwiązuje - nie znalazłem na szybko	na domyślnych ustawieniach nie rozwiązuje - nie znalazłem na szybko	?	?	?	Wygląda podobnie do snowpack Do build trafiają wszystkie pliki z projektu - mam wrażenie, że trudno nad tym zapanować i dostosować do naszych potrzeb	Nie udało mi się na szybko skonfigurować tego aby rozwiązywało require, import i usuwało serwerowy kod tak jak chcemy. Zorientowany raczej jako manager do całej aplikacji - raczej nie dla nas.
	poi	yes	yes	yes	?	?	?	"Poi has been deprecated, please migrate to Vite"	Przestarzałe (zastąpione przez vite)
	@carbon/bundler	?	?	?	?	?	?	Wygląda jak duży moloch. Wymaga python2.	Raczej nie dla nas

TODO: Watch source code?
TODO: Load code part on-the-fly (code splitting)

1.Większość bundlerów rozumie import isomorficznej paczki (main/browser w package.json), ale wymaga jawnej deklaracji paczki w dependencies (package.json) oraz jej instalacji do node_modules. Mam wrażenie, że to jest powszechnie akceptowany standard.

2. Warunkowe require - bundlery, które mają eliminacje zdechlego kodu usuwają kod serwerowy jeśli jest w fałszywym warunku if (stała). Część bundlerów ma możliwość globalnego zdefiniowana stałej w swoim konfigu. PROBLEM: Taki kod nie działa sam z siebie na serwerze bez przepuszczenia przez bundler - brakuje stałej.

3. Warunkowy import - wydaje mi się, że to karkołomny pomysł. Importy muszą być w globalnym namespacie - działają statycznie (w czasie kompilacji, nie w czasie uruchomienia) podobnie jak importy w pascalu, moduli, adzie itd. Do niektórych bundlerów da się dodrutować preprocesor #ifdef, ale - PROBLEM: Kod z #ifdef jest nieprzenośny, trudny w debugowaniu i serwer wymaga bundlera do pracy.
Sprawdziłem to rozwiązanie tylko dla 3 pierwszych bundlerow. Dalej uznałem, że te rozwiąania są mocno przyspawane do danego bundlera, wymagają sporo czasu żeby je znaleźć a i tak nie wiadomo czy będziemy tego używać. W razie potrzeby możemy do tego wrócić dla wybranego bundlera.

4. Raczej na pewno możemy olać - poi (oficjalnie przestarzały) i carbon (moloch wymagający pythona2)

5. Niektóre bundlery wyglądają jakby kładły nacisk na zarządzanie całym drzewem/aplikacją a nie tylko robienei bundli. Wydaje mi się, że to nie do końca to czego szukamy: vite, snowpack, lasso

6. Parcel i fuse nie udało mi się w pełni skonfigurować żeby robiły to co chcemy, ale wydaje mi się, że to jest do zrobienia. Na tym etapie nie porafię powiedzieć czy one mają jakąś przewagę nad resztą.

7. Browserify potrafi sprawić że paczki typowo nodowe zaczynają działać na browserze (sprawdziliśmy to też sami w calculli). Ale żeby mieć mumifikacje i usunięcie kodu serwerowego potrzebujemy obudować to zewnętrznymi toolami - trudno powiedzieć czy to będzie miało przewagę na jakimś etapie.

8. Te bundlery udało mi się w miarę łatwo skonfigurować aby robiły to co chcemy na tym etapie: rollup, webpack, esbuild, microbundle.