

Wydanie drugie

TypeScript

Skuteczne programowanie

83 sposoby ulepszania kodu TypeScript

Dan Vanderkam

*przekład: Joanna Zatorska
Marek Włodarz*

Spis treści

Przedmowa do drugiego wydania	xi
Przedmowa do pierwszego wydania	xxi
1. Poznajemy TypeScript	1
Element 1: Relacja między TypeScript a JavaScript.	1
Element 2: Które opcje TypeScript wykorzystujemy.	8
Element 3: Generowanie kodu jest niezależne od typów	12
Element 4: Przyzwyczaj się do strukturalnego typowania	19
Element 5: Ograniczanie użycia typu any.	24
2. System typowania TypeScript	29
Element 6: Używanie edytora do sprawdzania i eksploracji systemu typowania ...	29
Element 7: Typy jako zbiory wartości	35
Element 8: Ustalanie, czy symbol należy do przestrzeni typu czy do przestrzeni wartości.	43
Element 9: Deklarujemy typy zamiast stosować asercje typów	48
Element 10: Unikajmy typów opakowujących obiekty (String, Number, Boolean, Symbol, BigInt)	53
Element 11: Limity testowania dodatkowych właściwości.	56
Element 12: Stosujemy typy w całych wyrażeniach funkcyjnych, gdy jest to możliwe.	60
Element 13: Odróżniamy typ od interfejsu	64
Element 14: Używajmy readonly, aby uniknąć błędów związanych z mutowaniem	71
Element 15: Używajmy operacji typów i typów generycznych, aby uniknąć powtórzeń.	77
Element 16: Korzystajmy z sygnatur indeksów dla danych dynamicznych	87
Element 17: Unikanie liczbowych sygnatur indeksów	92

3. Wnioskowanie typów i analiza przepływu sterowania	97
Element 18: Unikajmy zaśmiecania kodu typami, które można wywnioskować ...	98
Element 19: Używajmy różnych zmiennych dla różnych typów	105
Element 20: Rozszerzanie typów	108
Element 21: Tworzenie całych obiektów od razu.	113
Element 22: Zawężanie typów	116
Element 23: Zachowajmy spójność w używaniu aliasów	121
Element 24: Jak wykorzystuje się kontekst podczas wnioskowania typów	125
Element 25: Ewolucja typu any	130
Element 26: Korzystajmy z konstrukcji funkcyjnych i bibliotek, aby ułatwić przepływ typów	134
Element 27: Używajmy funkcji asynchronicznych zamiast wywołań zwrotnych ..	138
Element 28: Używanie klas i rozwijania funkcji do tworzenia nowych punktów wnioskowania	144
4. Projektowanie typów	149
Element 29: Preferujmy typy, które zawsze reprezentują poprawne stany.	149
Element 30: Liberalne podejście do akceptowanych wartości i surowe do zwracanych danych	155
Element 31: Nie powtarzajmy informacji o typie w dokumentacji.	159
Element 32: Unikajmy dołączania null lub undefined w aliasach typu	162
Element 33: Przenośmy wartości null poza obręb swojego typu.	163
Element 34: Używajmy unii interfejsów zamiast interfejsów z uniami	167
Element 35: Preferujmy bardziej precyzyjne alternatywy typów łańcuchowych ..	171
Element 36: Używanie oddzielnego typu dla wartości specjalnych	176
Element 37: Ograniczanie użycia właściwości opcjonalnych.	179
Element 38: Unikajmy wielokrotnych parametrów o tym samym typie	183
Element 39: Preferujmy typy ujednolicone do modelowania różnic	185
Element 40: Używajmy typów niekompletnych zamiast nieprecyzyjnych.	187
Element 41: Nazywajmy typy zgodnie z językiem dziedziny swojego projektu ...	193
Element 42: Unikajmy typów opartych na danych anegdotycznych.	196
5. Niesolidność i typ any	201
Element 43: Używanie możliwie największego zakresu dla typów any	201
Element 44: Preferujmy warianty bardziej precyzyjne od zwykłego typu any	205
Element 45: Ukrywajmy nie gwarantujące bezpieczeństwa asercje typów w dobrze typowanych funkcjach	207

Element 46: Korzystajmy z typu <code>unknown</code> zamiast <code>any</code> dla wartości nieznanego typu.	211
Element 47: Stosujmy bezpieczne pod kątem typów podejście do małego łątania	216
Element 48: Unikanie pułapek (nie)solidności	220
Element 49: Śledźmy pokrycie typami, aby zapobiec regresji bezpieczeństwa typów.	230
6. Generyki i programowanie na poziomie typów	233
Element 50: Traktujmy typy generyczne jak funkcje pomiędzy typami.	234
Element 51: Unikajmy niepotrzebnych parametrów typów.	240
Element 52: Używajmy typów warunkowych zamiast przeciążonych deklaracji ..	246
Element 53: Jak kontrolować rozkład unii przez typy warunkowe	249
Element 54: Używanie szablonów typów literałów do modelowania DSL i zależności pomiędzy ciągami.	254
Element 55: Piszmy testy swoich typów	261
Element 56: Zwróćmy uwagę na to, jak typy są pokazywane.	270
Element 57: Preferujmy typy generyczne z rekurencją ogonową	274
Element 58: Rozważmy generowanie kodu jako alternatywę dla złożonych typów	278
7. Przepisy dla TypeScript	283
Element 59: Używanie typów <code>never</code> do sprawdzania kompletności	283
Element 60: Iterowanie po obiektach	290
Element 62: Korzystajmy z typów mapowanych, aby zapewnić synchronizację wartości	293
Element 62: Używajmy parametrów resztowych i typów krotek do modelowania funkcji o zmiennej liczbie argumentów.	297
Element 63: Używajmy opcjonalnych właściwości <code>never</code> do modelowania alternatywy wykluczającej	300
Element 64: Rozważmy stosowanie „etykiet” dla typowania nominalnego.	303
8. Deklaracje typów i składnia <code>@types</code>	307
Element 65: Umieszczanie kodu TypeScript i <code>@types</code> w deklaracjach <code>devDependencies</code>	307
Element 66: Trzy numery wersji w deklaracjach typów	310
Element 67: Eksportujmy wszystkie typy obecne w publicznych API	314
Element 68: Używajmy TSDoc do tworzenia komentarzy API	315
Element 69: Zdefiniujmy typ dla <code>this</code> w wywołaniach zwrrotnych.	319

Element 70: Używajmy typów lustrzanych dla oddzielania zależności	323
Element 71: Używajmy rozszerzania modułu do ulepszania typów	326
9. Pisanie i uruchamianie kodu	331
Element 72: Korzystajmy z funkcjonalności standardu ECMAScript zamiast TypeScript.	331
Element 73: Korzystajmy z map kodu źródłowego do debugowania kodu TypeScript.	339
Element 74: Rekonstruowanie typów w czasie działania	346
Element 75: Hierarchia DOM	351
Element 76: Tworzenie dokładnego modelu środowiska	357
Element 77: Zależności pomiędzy sprawdzaniem typów a testami jednostkowymi.	360
Element 78: Zwracajmy uwagę na wydajność kompilatora	364
10. Modernizacja i migracja	373
Element 79: Pisanie nowoczesnego kodu JavaScript.	374
Element 80: Używajmy @ts-check oraz JSDoc do eksperymentowania z TypeScript	379
Element 81: Używajmy allowJs, aby połączyć kod TypeScript z JavaScript	383
Element 82: Przekształcajmy moduły po kolei, wędrując w górę grafu zależności	385
Element 83: Nie uważajmy migracji za zakończoną przed włączeniem opcji noImplicitAny	391
A. Odzworowanie elementów pomiędzy pierwszym i drugim wydaniem.	395
Indeks	399
O autorze	410