1. **Jaki jest główny cel Angulara? / What is the main purpose of Angular?**
   PL: Angular to framework do budowy aplikacji internetowych, szczególnie jednostronicowych (SPA). Jego głównym celem jest ułatwienie tworzenia złożonych interfejsów użytkownika z zachowaniem wydajności i skalowalności.
   EN: Angular is a framework for building web applications, especially Single Page Applications (SPA). Its main goal is to simplify the creation of complex user interfaces while maintaining performance and scalability.
2. **Czym są dyrektywy w Angular i jakie są ich rodzaje? / What are directives in Angular and what types exist?**
   PL: Dyrektywy to specjalne instrukcje dodawane do elementów HTML, które umożliwiają zmianę wyglądu lub zachowania tych elementów. Istnieją trzy typy dyrektyw: strukturalne, atrybutowe i komponenty.
   EN: Directives are special instructions added to HTML elements to modify their appearance or behavior. There are three types of directives: structural, attribute, and components.
3. **Wyjaśnij wiązanie danych i jego różne typy. / Explain data binding and its types.**
   PL: Wiązanie danych to sposób synchronizacji danych między widokiem (UI) a logiką aplikacji. W Angular można używać interpolacji, wiązania właściwości (Property Binding), wiązania zdarzeń (Event Binding) oraz wiązania dwukierunkowego (Two-Way Binding).
   EN: Data binding is a way to synchronize data between the view (UI) and the application logic. In Angular, you can use interpolation, property binding, event binding, and two-way binding.
4. **Jakie są podstawowe komponenty Angular? / What are the basic components of Angular?**
   PL: Podstawowe komponenty to komponenty, moduły, usługi i dyrektywy.
   EN: The core components are components, modules, services, and directives.
5. **Jaka jest różnica między AngularJS a Angular? / What is the difference between AngularJS and Angular?**
   PL: AngularJS to starsza wersja napisana w JavaScript, a Angular (od wersji 2+) to nowoczesna wersja napisana w TypeScript, z lepszą wydajnością i modularnością.
   EN: AngularJS is the older version written in JavaScript, while Angular (from version 2+) is a modern version written in TypeScript with better performance and modularity.
6. **Czym są komponenty i moduły w Angular? / What are components and modules in Angular?**
   PL: Komponenty są podstawowymi elementami interfejsu użytkownika, a moduły grupują komponenty, dyrektywy i usługi w logiczne jednostki.
   EN: Components are the building blocks of the UI, while modules group components, directives, and services into logical units.
7. **Czym są dekoratory w Angular? / What are decorators in Angular?**
   PL: Dekoratory to funkcje, które dodają metadane do klas, pomagając Angularowi rozpoznać, jak mają być interpretowane (np. `@Component`, `@Injectable`).
   EN: Decorators are functions that add metadata to classes, helping Angular recognize how they should be interpreted (e.g., `@Component`, `@Injectable`).

8. **Co to jest metadane lub adnotacje w Angular? / What are metadata or annotations in Angular?**
   PL: Metadane dostarczają informacji o klasach lub funkcjach, pomagając Angularowi w ich odpowiednim interpretowaniu i przetwarzaniu.
   EN: Metadata provides information about classes or functions, helping Angular interpret and process them correctly.
9. **Czym są szablony w Angular? / What are templates in Angular?**
   PL: Szablony to struktury HTML powiązane z komponentami, które określają, jak mają być renderowane dane.
   EN: Templates are HTML structures associated with components that define how data should be rendered.
10. **Co to jest SPA i jak je zaimplementować w Angular? / What is SPA and how do you implement it in Angular?**
    PL: SPA (Single Page Application) to aplikacja, która działa w jednej stronie HTML, dynamicznie zmieniając treść bez przeładowywania strony. W Angular można to osiągnąć, konfigurując odpowiednio routing.
    EN: SPA (Single Page Application) is an application that operates on a single HTML page, dynamically updating content without reloading the page. In Angular, this can be achieved by configuring routing properly.
11. **Wyjaśnij znaczenie routingu w Angular i jak go zaimplementować. / Explain the importance of routing in Angular and how to implement it.**
    PL: Routing umożliwia nawigację między widokami. W Angular konfigurujemy go, definiując trasy w `AppRoutingModule`.
    EN: Routing allows navigation between views. In Angular, it is configured by defining routes in the `AppRoutingModule`.
12. **Co to jest leniwe ładowanie w Angular? / What is lazy loading in Angular?**
    PL: Leniwe ładowanie to technika ładowania modułów tylko wtedy, gdy są potrzebne, co poprawia czas ładowania aplikacji.
    EN: Lazy loading is a technique that loads modules only when needed, improving the application's loading time.
13. **Jak zaimplementować leniwe ładowanie w Angular? / How do you implement lazy loading in Angular?**
    PL: W konfiguracji routingu używamy właściwości `loadChildren`, aby ładować moduły asynchronicznie.
    EN: In the routing configuration, we use the `loadChildren` property to load modules asynchronously.
14. **Co to jest Node.js? / What is Node.js?**
    PL: Node.js to środowisko uruchomieniowe JavaScript pozwalające na uruchamianie JavaScript po stronie serwera.
    EN: Node.js is a JavaScript runtime that allows running JavaScript on the server side.
15. **Czym jest NPM? / What is NPM?**
    PL: NPM (Node Package Manager) to menedżer pakietów, który ułatwia zarządzanie zależnościami i bibliotekami w projektach Node.js.

EN: NPM (Node Package Manager) is a package manager that simplifies managing dependencies and libraries in Node.js projects.

16. **Dlaczego folder node_modules jest ważny? / Why is the node_modules folder important?**
    PL: Folder `node_modules` przechowuje wszystkie pakiety zainstalowane przez NPM, które są niezbędne do działania aplikacji.
    EN: The `node_modules` folder stores all packages installed by NPM, which are essential for the application to run.
17. **Co to jest package.json? / What is package.json?**
    PL: `package.json` to plik konfiguracji, który zawiera informacje o projekcie, jego zależnościach, wersjach oraz skryptach.
    EN: `package.json` is a configuration file that contains information about the project, its dependencies, versions, and scripts.
18. **Czym jest TypeScript? / What is TypeScript?**
    PL: TypeScript to superset JavaScript, który dodaje typowanie statyczne i dodatkowe funkcje.
    EN: TypeScript is a superset of JavaScript that adds static typing and additional features.
19. **Dlaczego potrzebujemy Angular CLI? / Why do we need Angular CLI?**
    PL: Angular CLI to narzędzie do zarządzania projektami Angular, które ułatwia ich tworzenie, testowanie i budowanie.
    EN: Angular CLI is a tool for managing Angular projects that simplifies creating, testing, and building applications.
20. **Co to są usługi w Angular? / What are services in Angular?**
    PL: Usługi to klasy, które przechowują logikę biznesową i mogą być wstrzykiwane do różnych komponentów.
    EN: Services are classes that store business logic and can be injected into various components.
21. **Kiedy używasz projekcji treści? / When would you use content projection?**
    PL: Projekcja treści jest używana, gdy chcemy wstrzyknąć dynamiczne treści do komponentu.
    EN: Content projection is used when we want to inject dynamic content into a component.
22. **Wyjaśnij sloty projekcji treści w Angular. / Explain content projection slots in Angular.**
    PL: Sloty pozwalają na umieszczanie różnych części treści w określonych miejscach komponentu za pomocą `ng-content`.
    EN: Slots allow placing different parts of content in specific locations within a component using `ng-content`.
23. **Dlaczego potrzebujemy ViewChild i ViewChildren w Angular? / Why do we need ViewChild and ViewChildren in Angular?**
    PL: `ViewChild` i `ViewChildren` umożliwiają dostęp do dzieci komponentów lub elementów DOM z poziomu logiki komponentu.

EN: `ViewChild` and `ViewChildren` allow access to child components or DOM elements from within the component's logic.

24. **Co to jest zmienna referencyjna szablonu? / What is a template reference variable?**
    PL: Zmienna referencyjna to identyfikator przypisany do elementu w szablonie, który pozwala na dostęp do tego elementu.
    EN: A template reference variable is an identifier assigned to an element in a template, allowing access to that element.
25. **Wyjaśnij ContentChild i ContentChildren. / Explain ContentChild and ContentChildren.**
    PL: `ContentChild` i `ContentChildren` są używane do uzyskania dostępu do dzieci, które są projekcjonowane do komponentu za pomocą projekcji treści.
    EN: `ContentChild` and `ContentChildren` are used to access children projected into a component via content projection.
26. **Jaka jest różnica między ViewChild, ViewChildren, ContentChild a ContentChildren? / Differentiate between ViewChild, ViewChildren, ContentChild, and ContentChildren.**
    PL: `ViewChild` i `ViewChildren` odnoszą się do elementów bezpośrednio wewnątrz komponentu, a `ContentChild` i `ContentChildren` do elementów projekcjonowanych zewnętrznie.
    EN: `ViewChild` and `ViewChildren` refer to elements directly inside the component, while `ContentChild` and `ContentChildren` refer to externally projected elements.
27. **Co oznacza `{ static: true }` w ViewChild? / What is `{ static: true }` in ViewChild?**
    PL: `{ static: true }` określa, że element będzie dostępny w hooku `ngOnInit`, a nie tylko po pełnym renderowaniu widoku.
    EN: `{ static: true }` specifies that the element will be available in the `ngOnInit` hook, not just after the view has fully rendered.
28. **Jaka jest rola hooków cyklu życia komponentu w Angular? / What is the importance of Angular component hooks/life cycles?**
    PL: Hooki cyklu życia pozwalają na reakcję komponentów na zmiany w ich stanie i cyklu życia.
    EN: Lifecycle hooks allow components to respond to changes in their state and lifecycle.
29. **Wyjaśnij hooki cyklu życia w Angular. / Explain Angular life cycle hooks in detail.**
    PL: Najważniejsze hooki to: `ngOnInit`, `ngOnChanges`, `ngDoCheck`, `ngOnDestroy`, które pozwalają kontrolować zachowanie komponentu w różnych fazach.
    EN: The main hooks are `ngOnInit`, `ngOnChanges`, `ngDoCheck`, `ngOnDestroy`, which allow controlling component behavior at various stages.
30. **Czym się różni konstruktor od ngOnInit()? / Differentiate between constructor and ngOnInit().**
    PL: Konstruktor służy do inicjalizacji zależności, a `ngOnInit()` jest wywoływany po

utworzeniu komponentu i inicjalizacji danych.
EN: The constructor is for dependency initialization, while `ngOnInit()` is called after the component is created and data is initialized.

31. **Jak zaimplementować leniwe ładowanie w Angular? / How do you implement lazy loading in Angular?**
    PL: Poprzez konfigurację `loadChildren` w routingu, aby moduły były ładowane asynchronicznie.
    EN: By configuring `loadChildren` in routing to load modules asynchronously.
32. **Jak przekazujesz dane między komponentami? / How do you pass data between components?**
    PL: Można użyć `@Input` i `@Output` do przekazywania danych między komponentami nadrzędnymi i podrzędnymi.
    EN: You can use `@Input` and `@Output` to pass data between parent and child components.
33. **Co to są pipes w Angular? / What are pipes in Angular?**
    PL: Pipes to funkcje do przetwarzania danych w szablonach, np. formatowanie daty, waluty.
    EN: Pipes are functions for data processing in templates, like formatting dates or currency.
34. **Podaj przykłady wbudowanych pipes w Angular. / Can you give examples of inbuilt Angular pipes?**
    PL: Przykłady to `DatePipe`, `CurrencyPipe`, `UpperCasePipe`, `LowerCasePipe`.
    EN: Examples include `DatePipe`, `CurrencyPipe`, `UpperCasePipe`, `LowerCasePipe`.
35. **Jak napisać niestandardowy pipe? / How do you write a custom pipe?**
    PL: Tworzymy nową klasę z dekoratorem `@Pipe` i implementujemy metodę `transform`.
    EN: Create a new class with the `@Pipe` decorator and implement the `transform` method.
36. **Co to jest RxJs i dlaczego jest potrzebne? / What is RxJs and why is it needed?**
    PL: RxJs to biblioteka do reaktywnego programowania, która ułatwia obsługę asynchroniczności w Angular.
    EN: RxJs is a library for reactive programming that simplifies handling asynchronous operations in Angular.
37. **Co to są obserwowalne i obserwatorzy? / What are observables and observers?**
    PL: Obserwowalne to strumienie danych, które mogą być obserwowane przez obserwatorów, reagujących na zmiany.
    EN: Observables are data streams that can be observed by observers, reacting to changes.
38. **Czym jest strumień w RxJs? / What is a stream in RxJs?**
    PL: Strumień to sekwencja asynchronicznych zdarzeń, które można przetwarzać i subskrybować.
    EN: A stream is a sequence of asynchronous events that can be processed and subscribed to.

39. **Jakie jest zastosowanie `subscribe` w RxJs? / What is the use of subscribe in RxJs?**
    PL: `Subscribe` pozwala nasłuchiwać danych z obserwowalnego strumienia i reagować na zmiany.
    EN: `Subscribe` allows you to listen to data from an observable stream and respond to changes.
40. **Jak anulować subskrypcję strumienia? / How do you unsubscribe from a stream?**
    PL: Używając metody `unsubscribe`, aby zapobiec wyciekom pamięci.
    EN: Use the `unsubscribe` method to prevent memory leaks.
41. **Co to są operatory w RxJs? / What are operators in RxJs?**
    PL: Operatory to funkcje do przekształcania danych w strumieniach, np. `map`, `filter`, `merge`.
    EN: Operators are functions for transforming data in streams, like `map`, `filter`, `merge`.
42. **Gdzie używałeś RxJs w Angular? / Where have you used RxJs in Angular?**
    PL: W Angular RxJs jest często używany do obsługi HTTP, formularzy reaktywnych i zarządzania stanem.
    EN: RxJs is commonly used in Angular for handling HTTP, reactive forms, and state management.
43. **Czym różni się RxJs od Promises? / Differentiate between RxJs and Promises.**
    PL: RxJs obsługuje wiele wartości w czasie, natomiast Promise tylko jedną wartość po zakończeniu operacji.
    EN: RxJs handles multiple values over time, while Promise handles only one value after the operation completes.
44. **Jak zainstalować RxJs? / How do you install RxJs?**
    PL: `npm install rxjs` instaluje RxJs w projekcie.
    EN: `npm install rxjs` installs RxJs in the project.
45. **Dlaczego RxJs jest określany jako push/reaktywny, a nie pull/imperatywny? / Why is RxJs called push/reactive and not pull/imperative?**
    PL: RxJs działa na zasadzie przepływu danych do obserwatora (push), a nie pobierania ich na żądanie (pull).
    EN: RxJs works by pushing data to the observer, rather than pulling it on demand.
46. **Podaj przykłady operatorów RxJs. / Name some RxJs operators.**
    PL: Przykłady to `map`, `filter`, `take`, `switchMap`.
    EN: Examples include `map`, `filter`, `take`, `switchMap`.
47. **Co to są interceptory w Angular? / What are interceptors in Angular?**
    PL: Interceptory przechwytują żądania HTTP przed ich wysłaniem lub po otrzymaniu odpowiedzi.
    EN: Interceptors intercept HTTP requests before they are sent or after receiving a response.
48. **Jak zaimplementować interceptory? / How do you implement interceptors?**
    PL: Tworzymy klasę z metodą `intercept`, która modyfikuje żądania i rejestrujemy ją w dostawcach.

EN: Create a class with an `intercept` method to modify requests and register it in providers.

49. **Jakie są zastosowania interceptorów i czy można je łączyć? / What are some uses of interceptors and can we combine them?**
    PL: Interceptory są używane do logowania, autoryzacji i obsługi błędów. Można je łączyć w kolejności rejestracji.
    EN: Interceptors are used for logging, authorization, and error handling. They can be combined in the registration order.
50. **Czym jest Node.js? / What is Node.js?**
    PL: Node.js to środowisko JavaScript działające po stronie serwera, które pozwala na tworzenie aplikacji sieciowych i serwerowych.
    EN: Node.js is a JavaScript environment that runs on the server side, allowing for the creation of network and server applications.