### **Angular Defer**

Elementem podstawowym jest szablon reprezentowany przez następującą grupę bloków: **{#defer}...{/defer}**. Zależności (komponenty, dyrektywy itp.), do których odwołuje się odroczony blok, są ładowane leniwie. W rezultacie odroczone bloki są zawsze renderowane asynchronicznie. Obejmuje to wszystkie zależności w ramach odroczonego bloku, które obejmują komponenty, dyrektywy i potoki używane w ramach tych zależności.

@defer

Zawartość głównego @defer to sekcja treści, która jest leniwie ładowana. Nie będzie on początkowo renderowany, a cała zawartość pojawi się po spełnieniu określonego wyzwalacza lub warunku i pobraniu zależności. Domyślnie @defer jest wyzwalana, gdy stan przeglądarki staje się bezczynny .

@placeholder

Domyślnie bloki opóźniające nie renderują żadnej treści przed ich uruchomieniem. @placeholder to opcjonalny blok, który deklaruje zawartość wyświetlaną przed wyzwoleniem bloku @defer. Ta treść zastępcza zostaje zastąpiona treścią główną po zakończeniu ładowania. Można użyć dowolnej treści w sekcji @placeholder, w tym zwykłego kodu HTML, komponentów, dyrektyw i pipe.

Blok @placeholder opcjonalnie przyjmuje parametr (minimum [ms/s]) określający minimum czasu, przez jaki ten blok powinien być wyświetlany. Ten parametr ma na celu zapobieganie szybkiemu zniknięciu treści zastępczej w przypadku szybkiego pobrania odroczonych zależności. Licznik minimum czasu bloku @placeholder rozpoczyna się po zakończeniu początkowego renderowania @placeholder.

@loading

Blok @loading jest opcjonalnym blokiem, który umożliwia zadeklarowanie treści, która będzie wyświetlana podczas ładowania wszelkich odroczonych zależności. Można na przykład pokazać spinner ładujący.

Blok @loading akceptuje dwa opcjonalne parametry określające minimum czasu, przez jaki ten symbol zastępczy powinien być wyświetlany (minimum [ms/s]) oraz czas oczekiwania na rozpoczęcie ładowania przed wyświetleniem szablonu ładowania (after [ms/s]). Podobnie jak @placeholder, te parametry istnieją, aby zapobiec migotaniu treści w przypadku szybkiego pobrania odroczonych zależności. Zarówno liczniki, minimum jak i after dla @loading rozpoczynają się natychmiast po uruchomieniu ładowania.

@error

Blok @error umożliwia zadeklarowanie treści, która zostanie wyświetlona w przypadku niepowodzenia ładowania @defer. Blok @error jest opcjonalny.

**Wyzwalacze**

Po uruchomieniu bloku @defer zawartość zastępcza zostaje zastąpiona treścią ładowaną leniwie. Istnieją dwie opcje konfiguracji, kiedy ta zamiana jest wyzwalana: **on** i **when**.

**On** określa warunek wyzwalacza. Przykładem może być interakcja lub vieport.

Można zdefiniować wiele wyzwalaczy jednocześnie. Na przykład: on *interaction; on timer(5s)* oznacza, że ​​blokada odroczenia zostanie uruchomiona, jeśli użytkownik wejdzie w interakcję z @placeholder lub po 5 sekundach.

**When** określa warunek jako wyrażenie zwracające wartość logiczną. Kiedy to wyrażenie staje się prawdziwe, @placeholder jest zamieniany na leniwie ładowaną treść (co może być operacją asynchroniczną, jeśli konieczne jest pobranie zależności).

Można także użyć obu when i on razem w jednej instrukcji, zamiana zostanie uruchomiona, jeśli którykolwiek z warunków zostanie spełniony.

**on idle**

idle uruchomi @defr, gdy przeglądarka osiągnie stan bezczynności (wykryty za pomocą interfejsu requestIdleCallbackAPI). Jest to domyślne zachowanie w przypadku bloku odroczenia.

@defer (on idle) { <span>Browser has reached an idle state</span> }  
@placeholder { <span>Placeholder</span> }

**on viewport**

viewport uruchomi odroczony blok, gdy określona treść wejdzie do widocznego obszaru (za pomocą interfejsu IntersectionObserverAPI) . Może to być @placeholder lub odniesienie do elementu.

@defer (on viewport) { <app-red text="The block entered the viewport"/> }  
@placeholder { <span>Placeholder</span> }

Alternatywnie można określić zmienną odniesienia do szablonu w tym samym szablonie co @defer, w którym znajduje się element obserwowany przed wejściem do viewport. Ta zmienna jest przekazywana jako parametr wyzwalacza viewport.

<div #viewport>Hello!</div>  
@defer (on viewport(viewport)) { <app-red text="Hello entered the viewport"/> }  
@placeholder { <span>Placeholder</span> }

**on interaction**

interaction uruchomi@defr, gdy użytkownik wejdzie w interakcję z określonym elementem poprzez zdarzenia click lub keydown. Alternatywnie można określić zmienną odniesienia szablonu jako element wyzwalający interakcję. Ta zmienna jest przekazywana jako parametr wyzwalacza interakcji (jak w vieport powyżej).

@defer (on interaction) { <span>Clicked</span> }  
@placeholder { <span>Placeholder (click on it!)</span> }

**on hover**

hover uruchamia opóźnione ładowanie, gdy mysz najedzie na obszar wyzwalania. Wykorzystywane w tym celu zdarzenia to mouseenter i focusin.

@defer (on hover) { <span>Hovered</span> }  
@placeholder { <span>Placeholder (hover it!)</span> }

Alternatywnie możesz określić zmienną referencyjną szablonu jako element najechania. Ta zmienna jest przekazywana jako parametr wyzwalacza najechania myszką.

<div #greeting>Hello!</div>  
@defer (on hover(greeting)) { <span>Hovered Hello!</span> }  
@placeholder { <div>Calendar placeholder</div> }

**on immediate**

immediate natychmiast uruchamia @defer, co oznacza, że ​​gdy klient zakończy renderowanie, pobieranie odroczonej porcji rozpocznie się natychmiast.

@defer (on immediate) { <span>Immediate</span> }  
@placeholder { <span>Placeholder immediate)</span> }

**on timer**

timer(x) uruchomi się po określonym czasie. Czas trwania jest wymagany i można go określić w ms lub s.

@defer (on timer(5s)) { <span>Visible after 5s</span> }  
@placeholder { <span style="color: red">Placeholder</span> }

**Prefetch – pobieranie wstępne**

@defer pozwala określić warunki, w których ma zostać uruchomione pobieranie wstępne zależności za pomocą słowo kluczowego **prefetch**. Działa to podobnie do głównych warunków @defer i akceptuje when i/lub on przy deklaracji wyzwalacza.

Umożliwia to bardziej zaawansowane zachowania, takie jak rozpoczęcie wstępnego pobierania zasobów, zanim użytkownik faktycznie zobaczy blok odroczenia lub wejdzie z nim w interakcję, dzięki czemu zasoby będą szybciej dostępne.

W poniższym przykładzie pobieranie z wyprzedzeniem rozpoczyna się, gdy użytkownik najedzie myszką na placeholder, a zawartość bloku jest renderowana w wyniku interakcji.

@defer (on interaction; prefetch on hover) { <app-blue/> }  
@placeholder { <span>Placeholder (hover it, then click on it!)</span> }