**1. Pokazywanie ukrytego elementu**

Stwórz projekt z ukrytym elementem HTML (np. <div> z display:none w CSS), a także

przyciskiem (<button>). Przypisz na kliknięcie przycisku funkcję, która pokaże ukryty

element, gdy jest niewidoczny i ukryje go, gdy jest widoczny. Podczas zmiany stanu

widoczności tego elementu, zmień również tekst przycisku np. z “*Pokaż treść*” na “*Ukryj treść*” i na odwrót.

**2. Walidator formularza**

Stwórz prosty walidator formularza, który zawierał będzie pola <input> o typach “text”,

“email”, “number” oraz element <textarea>. Dodaj do elementu <form> atrybut

novalidate, aby wyłączyć domyślną walidację przeglądarki. Od Ciebie zależy, czy chcesz wyświetlać komunikaty o błędach czy tylko podświetlać niepoprawnie uzupełnione pola np. na czerwono. Za poprawnie uzupełnione pole <input> o typie “text” lub pole <textarea> uznajemy takie, które ma wpisany przynajmniej jeden znak. W przypadku pola o typie “email” sprawdź czy zawiera ono znak **@**, a w przypadku pola o typie “number” czy podana wartość jest liczbą (pamiętaj, że DOM zwróci Ci zawsze wartość o typie String, więc musisz znaleźć sposób, jak sprawdzić czy string ten zawiera wyłącznie liczbę).

**Uwaga:** Jeśli stworzysz pole <input> o typie “number” to w niektórych przeglądarkach nie będzie możliwości wpisania innych znaków niż liczby. Mimo to napisz logikę, która taką walidację wykona, a aby móc wprowadzić inne znaki niż tylko liczby, zmień na czas testów atrybut type pola <input> z “number” na “text”.

**3. Odliczanie od 10 do 0**

Stwórz projekt, który po uruchomieniu odpowiedniej funkcji, pozwoli na odliczanie

**od 10 do 0**. Wszystkie wartości powinny być wyświetlane na stronie, a czas pomiędzy zmianą wartości powinien wynosić **1 sekundę**. Choć cały Twój kod może być podzielony na wiele funkcji, jedna z nich powinna uruchamiać proces odliczania. Przy wywołaniu tej funkcji, daj możliwość przekazania innej funkcji jako argument. Przekazaną funkcję wywołaj, gdy licznik osiągnie wartość **0**. Istotą przekazywania jednej funkcji do drugiej (w tym przypadku nazwalibyśmy ją funkcją

**callback**) jest to, aby dać użytkownikowi naszego kodu pewną gotową funkcjonalność (odliczanie od 10 do 0), ale także możliwość dodania czegoś od siebie, tj. wykonania własnej funkcji po zakończeniu odliczania. W przekazanej funkcji możesz wpisać po prostu console.log(“Odliczanie zakończone!”).

**4. Dziedziczenie z klasy EventEmitter**

Przygotowany pod adresem http://pastebin.com/YEBncx0d kod zmodyfikuj tak, aby obiekty tworzone z klasy Database mogły korzystać z wszystkich metody klasy EventEmitter. Na chwilę obecną, podany kod wygeneruje błąd, gdyż klasa Database nie zawiera metody on oraz emit. Skorzystaj z dziedziczenia prototypowego, aby klasą nadrzędną dla Database stała się klasa EventEmitter. Zadanie to wymaga od Ciebie dopisania wyłącznie kilku linijek kodu.

**5. Funkcja debounce**

Stwórz funkcję o nazwie debounce, która przyjmie przy wywołaniu dwa argumenty.

Pierwszy z nich to inna **funkcja** do późniejszego wywołania, a drugi to **czas** w

milisekundach. Po takim wywołaniu, funkcja ta powinna zwrócić **nową funkcję**, którą

można zapisać np. w zmiennej. Następnie tę nową funkcję będzie można wielokrotnie wywoływać (a ona powinna wywołać pierwotnie przekazaną funkcję), jednak jeśli czas pomiędzy poszczególnymi wywołaniami będzie krótszy, niż podany wcześniej (wspomniany argument z czasem w milisekundach), to funkcja nie powinna nic zrobić, ale ustawić licznik na kolejne **Xms** i dopiero wtedy się wywołać.

Rozwiązania typu **debounce** stosuje się np. przy obsłudze zdarzenia scroll w

przeglądarkach, które wywoływane jest bardzo wiele razy. Jeśli nie chcemy, aby nasza funkcja negatywnie wpływała na wydajność podczas scrollowania, a wystarczy, że wywoła się np. 100ms po zakończeniu scrollowania, wtedy stosujemy funkcję debounce. Innym przykładem jest np. zdarzenie resize, które też jest wielokrotnie wywoływane podczas skalowania okna przeglądarki, a nam może zależeć, aby wywołać pewien kod dopiero wtedy, gdy taka akcja się zakończy.

Całość przetestować możesz z użyciem tego kodu: https://pastebin.com/J0BJVqtR. Twoim zadaniem jest dopisanie funkcji debounce, aby ten kod zaczął działać.