bootcamp online

JAVASCRIPT

Witaj w czwartym tygodniu Bootcampu

Przed Tobą czwarty tydzień nauki w Bootcampie JavaScript. Mam nadzieję, że poprzedni tydzień był dla Ciebie ciekawy oraz zawierał wiele cennych informacji. Ten tydzień w całości poświęcony jest najpopularniejszej bibliotece JavaScript, która choć zadebiutowała w 2006 roku, to wciąż jest rozwijana - **jQuery**! Dowiesz się wszystkiego co jest Ci potrzebne by bardziej produktywnie pracować z Obiektowym Modelem Dokumentu, poznasz ciekawe techniki oraz dobre praktyki wykorzystania tej biblioteki.

Piotr Palarz

Piotr Palarz

--- Zadania na tydzień 4 ---

*Zadania domowe spakuj ZIPem i umieść na kanale **#prace_domowe** na Slacku.*
W przypadku zadań, gdzie niezbędny jest dodatkowy kod HTML i CSS, nie musisz się skupiać na tym, aby strona wyglądała pięknie. Jeśli masz czas i chęci, dopracuj swój projekt, ale najważniejszy pozostaje i tak kod JavaScript.

1. Selektory w jQuery

Podaj przykłady selektorów, które pozwolą znaleźć na stronie następujące elementy:

- wszystkie elementy `<div>`, posiadające zarówno klasę `grid` oraz klasę `grid-12`.

- wszystkie elementy `<a>`, których atrybut `href` zaczyna się na “http”, znajdujące się w elemencie o klasie `nav`

- wszystkie elementy `<input>`, których typ to `radio` lub `checkbox` oraz dodatkowo nie są aktualnie zaznaczone (`checked`).

- wyłącznie pierwszy element `<p>`, który jest pusty (nie zawiera dzieci) oraz znajduje się w elemencie `<div>` z identyfikatorem `text`

- wszystkie elementy z klasą `pagination-item`, które nie są elementem `<span>`

Uwaga: Możesz utworzyć prostą stronę w HTML, gdzie w jakiś sposób podświetlisz znalezione powyższymi selektorami elementy, ale możesz równie dobrze po prostu wypisać te selektory w pliku tekstowym, bez tworzenia dla tego przykładu specjalnej strony.

2. Pokazywanie elementów i animacje

Stwórz kontener z przykładową nawigacją, który domyślnie nie będzie widoczny. Może on znajdować się na samej górze strony lub w sidebarze. Następnie umieść na stronie przycisk typu “**hamburger menu**” ☰, po kliknięciu którego wysunie się wcześniej ukryte menu. Animacja jaką zastosujesz, zależy od Ciebie. Ponowne kliknięcie przycisku powinno chować nawigację.

3. Dynamiczne tworzenie elementów

Stwórz aplikację, która zawierać będzie prosty formularz z wyłącznie jednym polem `<input>`, a także przyciskiem `<button>`. Napisz kod tak, aby po wpisaniu informacji do pola i po wciśnięciu przycisku, wstawić wpisane informacje do nowego elementu `<li>`, a ten element do wcześniej wstawionego elementu `<ul>`. W ten sposób za każdym razem, gdy zostanie wpisane np. jakieś imię, pojawi się ono na stronie. Dodatkowo zadbaj o to, by jeśli nic nie zostało wpisane do pola, nie wstawiać na stronę pustego elementu `<li>`.

4. Praca z Ajax i JSON

Stwórz aplikację, która pozwoli po kliknięciu wybranego przycisku, np. "**Załaduj**", pobrać **Ajax**em dane typu **JSON** i wyświetlić je na stronie. Adres, pod który wyślesz zapytanie z użyciem jQuery to http://code.eduweb.pl/bootcamp/users/

Otrzymane dane wyświetl na stronie w formie nieuporządkowanej listy `<ul>`, a każdego użytkownika w tagu `<li>`. Z podanych danych wyłuskaj `name`, `username`, `email` oraz `phone`. Sformatuj je według własnego uznania.

Podpowiedź: Dane możesz umieścić w elemencie `<li>` jak tylko chcesz. Jeśli jednak chciałbyś poznać tzw. systemy templatingu, to polecam dodatkowo zapoznać się np. z Handlebars.js. Biblioteka ta pozwoli Ci utworzyć szablon HTML, do którego następnie przekażesz dane JSON, a w odpowiedzi otrzymasz gotowy kod HTML z wstawionymi w odpowiednie miejsca danymi. Taki kod pozostanie wstawić na stronę. Jak korzystać z Handlebars.js od A do Z dowiesz się z naszego wpisu na blogu pt. Kompendium wiedzy o Handlebars.js.

5. Własny plugin dla spisu treści

Napisz własny plugin jQuery o nazwie `toc`, który umożliwi dynamiczne generowanie spisu treści na podstawie sekcji zawierających treść. *TOC* to skrót od *Table of Contents* i oznacza po prostu *Spis Treści*.

Przygotowałem dla Ciebie pliki startowe, od których powinieneś zacząć. Znajdziesz je tutaj. Zauważ, że plik `index.html` zawiera kilka sekcji. Każda z nich zawiera nagłówek `<h2>` oraz treść w postaci paragrafów `<p>`. Na samym dole strony znajdziesz podlinkowane

skrypty. Jest to biblioteka jQuery, a także plik `jquery.toc.js`, w którym powinieneś napisać cały kod pluginu. Poniżej tych skryptów znajduje się skrypt liniowy, wywołujący Twój plugin na zebranych elementach o klasie `section`. Na samej górze strony znajdziesz spis treści, zawarty w elemencie `<div>` z klasą `toc`.

Twoim zadaniem będzie usunięcie (ręczne, w kodzie) tego elementu, a następnie napisanie pluginu tak, aby taki element generował i wstawiał go dokładnie w to samo miejsce (podpowiedź: przed pierwszym elementem `<section>`). Treść odnośnika powinna odpowiadać treści nagłówka `<h2>` każdej sekcji. Dodatkowo stwórz mechanizm, który pozwoli po kliknięciu w wybrany odnośnik spisu treści, przenieść użytkownika do odpowiedniej sekcji (w kodzie, który wstawiłem na sztywno, celowo nie ma tego rozwiązania).

Uwaga: Jeśli masz czas i chęci, możesz dodać do swojego pluginu możliwość przekazywania opcji. Jedną z opcji mógłby być np. selektor nagłówka, według którego ma być generowana treść odnośnika w spisie treści, inną opcją np. klasa CSS, która powinna zostać dodana do elementu `<div>`, który w tym momencie ma klasę `toc`. Pomyśl, jakie jeszcze opcje można byłoby dodać, aby przyszli użytkownicy tego pluginu mogli go w łatwy sposób konfigurować pod swoje potrzeby.

Biblioteka jQuery

jQuery to najpopularniejsza biblioteka JavaScript. Czym jednak jest biblioteka w dowolnym języku programowania? Biblioteka to nic innego, jak kod napisany wcześniej, który adresuje różne problemy i ułatwia pracę z nimi.

Słowo “wcześniej” oznacza tutaj tylko tyle, że “później” znajduje się nasz kod. Bibliotekę zatem dołączamy do projektu, a następnie w swoim kodzie możemy korzystać z jej dobrodziejstw. Biblioteka jest po prostu zbiorem funkcji, z których następnie można korzystać. W świecie JavaScriptu często można spotkać nieprecyzyjne sformułowania, przez które początkujący twórcy stron internetowych sądzą, że jQuery jest językiem programowania. Nie jest to prawdą. Językiem programowania na potrzeby stron internetowych jest JavaScript i to w nim została napisana ta biblioteka.

Korzyści z używania jQuery

A jakie problemy rozwiązuje jQuery? Od samego jej początku, a więc od roku 2006, biblioteka jQuery ułatwia pracę z Obiektowym Modelem Dokumentu. I choć do tej pory poznałeś już natywne metody, które umożliwiają np. tworzenie nowych elementów, wstawianie ich na stronę, dodawanie do nich klas CSS czy przypisywanie obsługi zdarzeń, to z użyciem jQuery zrobisz to za pomocą mniejszej ilości kodu.

Wyobraź sobie sytuację, że potrzebujesz do 10 elementów z klasą `button` przypisać obsługę zdarzenia kliknięcia. O ile samo wyszukiwanie elementów jest dzisiaj bardzo łatwe, gdyż możesz skorzystać z zapisu `document.querySelectorAll(“.button”)`, to już przypisywanie zdarzeń wymaga nieco więcej wysiłku. W tym przypadku, po znalezieniu tych elementów na stronie, musiałbyś wykonać pętlę, a w niej do każdego elementu dodać obsługę zdarzenia za pomocą metody `addEventListener`. W jQuery zrobisz to za pomocą jednej linijki (bo dewiza jQuery brzmi: *write less, do more*). Oczywiście kod tej biblioteki “pod spodem” wykona podobną, do opisanej wyżej pracę, ale nie jest to już Twoim zmartwieniem.

Kolejną niebywałą zaletą tej biblioteki jest fakt, że niweluje ona różnice w implementacji wielu mechanizmów w różnych przeglądarkach. I choć dzisiaj jest to rzecz ciut mniej dolegliwa niż kiedyś, to wciąż ma miejsce. Często bowiem bywa tak, że kod napisany z

użyciem natywnych API przeglądarek działa w jednej z nich, a nie działa w drugiej. Czasami działa, ale zwraca nieco inne rezultaty. Biblioteka jQuery na bieżąco adresuje wszystkie znalezione bugi tak, aby korzystając z jej metod w spójny sposób, dawały one identyczne rezultaty we wszystkich przeglądarkach internetowych.

jQuery i Ajax

Choć biblioteka jQuery kapitalnie sprawdza się w przypadku pracy z Obiektowym Modelem Dokumentu, to ma jeszcze wiele innych możliwości. Jedną z nich jest obsługa asynchronicznych żądań HTTP, czyli Ajax.

Widziałeś już jak pracować z Ajaxem za pomocą interfejsu `XMLHttpRequest` i choć początkowo wydaje się, że jest to dosyć łatwe, to aby tworzyć profesjonalne rozwiązania, należy zadbać o wiele innych aspektów. Dobrym przykładem może być tutaj obsługa błędów. Do obiektu utworzonego za pomocą `new XMLHttpRequest()` możemy przypisać zdarzenie `error`, które wykona się, kiedy pojawi się błąd. No właśnie, ale jaki błąd? Okazuje się, że wyłącznie błąd w komunikacji sieciowej z serwerem. Czy zatem jeśli serwer zwróci kod statusu 404, to wykona się nasza funkcja? **Nie**. I to jest bardzo istotne. Status serwera *404 Not found* czy *500 Internal Server Error* to wciąż poprawne statusy, z którymi przeglądarka otrzymuje pewne dane. Jak zatem je obsłużyć, jeśli chcemy wówczas wyświetlić dla użytkownika jakiś komunikat lub wykonać inne zadanie? Musimy te rzeczy sprawdzać "ręcznie". Wymaga to nie tylko większej ilości kodu, ale i wiedzy na temat tego, które statusy traktować jako błędy.

Dobra wiadomość jest taka, że korzystając z biblioteki jQuery, nie musimy się tym martwić. Jeśli przypiszemy funkcję obsługi błędu, to jQuery poprawnie wykona ją w wielu przypadkach. Podobnie będzie przy pracy z JSONem czy metodą JSONP. jQuery maksymalnie te zadania upraszcza, pozwalając Ci się skupić na pisaniu dobrego kodu.

Kiedy korzystać z biblioteki jQuery

Na koniec odpowiedzmy na pytanie: "*Kiedy korzystać z biblioteki jQuery, a kiedy nie?*". Kiedy poznasz już tę bibliotekę, praca z nią wciągnie Cię bez reszty. Z jednej strony to bardzo dobrze, ale z drugiej rodzi to pokusę, by jQuery wykorzystywać wszędzie. Kiedy jednak masz naprawdę niewielką porcję kodu JavaScript do napisania, aby dodać jakąś

funkcjonalność do strony internetowej, to nie zawsze dobrym pomysłem jest ładowanie wcześniej biblioteki jQuery. Dlaczego? Z powodu tego, że jest to kolejny zasób, który przeglądarka musi pobrać. Może to być dolegliwością szczególnie dla urządzeń mobilnych, które nie zawsze korzystają z szybkiego internetu.

Z tego powodu warto znać natywne API przeglądarek, by w wielu sytuacjach poradzić sobie bez użycia dodatkowej biblioteki. Jeśli jednak piszesz już troszkę bardziej zaawansowany kod, bez obaw dołączaj tę bibliotekę. W tym przypadku jest to po prostu kwestia rozsądku, czy warto dołączać do strony potężny kombajn, by wykorzystać wyłącznie jedną z dostępnych funkcji. Nie oznacza to jednak, że jeśli chciałbyś ułatwić sobie pracę z Ajaxem, a dołączanie jQuery wyłącznie do tego celu wyda Ci się przesadą, to powinieneś zawsze korzystać z natywnego API. W takich sytuacjach polecam Ci stronę Microjs, na której znajdziesz bardzo wiele maleńkich w rozmiarze bibliotek, adresujących jeden wybrany problem, np. pracę z Ajax czy DOM. Jeśli natomiast znajdziesz się w sytuacji, gdzie do strony dodajesz jakiś slider obrazów, który wymaga do poprawnego działania biblioteki jQuery, to skoro już ją dołączyłeś, to i w swoim kodzie bez obaw z jej metod korzystaj.

Mam nadzieję, że korzystanie z jQuery będzie dla Ciebie przyjemnością, zatem do dzieła!