Instrukcja 4

1. Cel ćwiczenia

W niektórych wypadkach pisanie profesjonalnego kodu JavaScript może okazać się uciążliwe – szczególnie gdy programista potrzebuje bardzo szybko uzyskać określony efekt (np. animację elementu) bez konieczności dostosowywania go dla wszystkich przeglądarek. W tym celu można użyć biblioteki jQuery, która znacznie zmniejsza objętość kodu i przyspiesza proces tworzenia gotowego rozwiązania.

2. Teoria

Język JavaScript posiada bardzo dobrą dokumentację. Niestety jej objętość jest bardzo duża i znalezienie interesującego nas elementu może okazać się niezwykle żmudne. Innym mankamentem języka jest jego ogólnikowość – uzyskanie pożądanego efektu może wiązać się z koniecznością utworzenia dość rozbudowanego kodu. O ile doświadczeni programiści nie będą mieli z powyższymi problemami większych problemów, o tyle nowicjusze mogą, po dłuższych poszukiwaniach, stworzyć kod, który będzie nieefektywny. Nieefektywność kodu często spowodowana jest brakiem zrozumienia mechanizmów wykonywania instrukcji, nie wykorzystywaniem uchwytów do elementów oraz nieefektywne zarządzanie zmiennymi. Innym problemem, przynajmniej w przypadku niektórych funkcji, jest niezgodność ze wszystkimi przeglądarkami. Często chcąc wykonać jedną operację musimy ręcznie dodawać minimum 2 warunki pozwalające na poprawną obsługę kodu w różnych przeglądarkach (np. Edge i Firefox).

Ze względu na powyżej wymienione problemy sporo projektantów stron WWW wykorzystuje bibliotekę jQuery. Biblioteka jest zaliczana do tzw. lekkich. Oznacza to, że jej wpływ na działanie tworzonej aplikacji HTML nie powinna znacznie jej spowalniać. Ponadto sam jej kod nie jest zbyt duży – ok 100 kB, dzięki czemu pobranie go z serwera nie zajmuje dużo czasu. Dodatkowo biblioteka pozwala na pisanie własnych bądź wykorzystywanie przygotowanych przez innych użytkowników wtyczek. Wtyczki pozwalają na rozszerzenie funkcjonalności biblioteki w naszym własnym kodzie. Wtyczki pozwalają na zmniejszenie wielkości samej biblioteki, a tym samym na skrócenie jej ładowania i przekładają się na jej szybsze działanie.

INFORMACJA: W sieci można znaleźć informacje jakoby jQuery działało ze wszystkimi przeglądarkami dostępnymi na rynku, włączając w to Microsoft Internet Explorer 6. Nie jest to do końca prawdą. Od wersji 3 zostało porzucone wsparcie dla wersji 6,7 oraz 8 (najmniej zgodne z jakimkolwiek standardem). Działanie to zoptymalizowało i nieco przyspieszyło działanie biblioteki (brak obsługi dodatkowych warunków).

2.1 Ogólne zastosowanie

jQuery z założenia ma upraszczać wieloliniowy kod JavaScript na prostszy, najczęściej jednoliniowy. Głównym obszarem zastosowania biblioteki jest:
- operacje na DOM – znaczne uproszczenie odwołania się do konkretnego elementu HTML. Za pomocą jednej funkcji pytającej możemy odwołać się do pojedynczego elementu (bez znaczenia na to, czy operujemy na identyfikatorach czy np. klasach bądź znacznikach).
- operacje na CSS – pozwala poprzez metodę css() na dodawanie, modyfikowanie i usuwanie właściwości CSS. Ponadto ułatwia zarządzanie klasami CSS dla danego elementu poprzez możliwość usuwania wskazanej klasy (w JavaScript brak odpowiednika)
- obsługa zdarzeń – niezależna od wersji obsługa wszystkich zdarzeń HTML. Obecnie, ze względu na dodanie nowej metody przechwytującej zdarzenia (capture) lepszym rozwiązaniem jest używanie metody JavaScript addEventListener (aczkolwiek nowa metoda JS dostępna jest w nowszych przeglądarkach; dla zgodności lepiej jest wykorzystać mechanizm jQuery)

- obsługa animacji i przejść – ze względu na różny stopień obsługi animacji i przejść w poszczególnych przeglądarkach wielu projektantów stron WWW nadal wykorzystuje wbudowane funkcje jQuery do tworzenie efektownych animacji oraz zmian stanu elementów HTML.
- obsługa AJAX – jQuery pozwala na prostszą obsługę asynchronicznej wymiany danych pomiędzy serwerem a klientem. Wykorzystywać można komunikację z innymi skryptami JS, PHP, ASP, JSP itd. jQuery pozwala na obsługę komunikacji AJAX nawet w starszych wersjach przeglądarki.
- dodatkowe funkcje narzędziowe – biblioteka posiada w sobie szereg dodatkowych metod i właściwości, które ułatwiają, testowanie właściwości elementów, zarządzanie danymi JS (dataset) oraz pozwalają zamieniać obiekty jQuery na tablice JavaScript czy dowolny obiekt JavaScript (głównie jQuery) na szeregową reprezentację danych (zmienna=wartość&zm2=wartosc2 → postać szczególnie przydatna do zapytań AJAX przy wykorzystaniu zmiennych GET)

2.2 Używanie jQuery

Biblioteka jQuery, w przeciwieństwie do samego parsera JavaScript, nie jest dostarczana wraz z przeglądarką. Oznacza to, że jeżeli chcemy z niej skorzystać to musimy najpierw się do niej odwołać.

W sieci możemy znaleźć dwie linie biblioteki. Oto one:
- **Core** – pełna wersja jQuery (bez wtyczek). To właśnie ta biblioteka powinna być stosowana w większości przypadków.
- **Migrate** – wersja biblioteki zachowująca zgodność z elementami starszej wersji biblioteki, które zostały przebudowane (bądź najczęściej usunięte) z wersji Core. Stosowanie tej biblioteki powinno znaleźć zastosowanie jedynie na poziomie programistycznym. Każdy fragment przestarzałego kodu jQuery zostanie zakomunikowany w konsoli programistycznej z zaleceniami poprawy. Ważne jest by używać wersji migrate z dokładnie taką samą wersją jQuery Core! W przeciwnym wypadku działanie tej biblioteki może być niepożądane! Ponadto trzeba pamiętać, że najpierw musi zostać załadowana biblioteka jQuery, a dopiero w następnej kolejności biblioteka migrate.

UWAGA: Ponieważ największe zmiany dotyczyły wersji 3.0.0 dlatego wersja migrate dotyczy tylko tej wersji biblioteki. Jeżeli korzystaliśmy wcześniej z biblioteki 2.x.x to pierwszym krokiem powinno być skorzystanie z wersji 3.0.0 wraz z biblioteką migrate 3.0.0. Po tej operacji powinniśmy dopiero zamienić wersję biblioteki na najnowszą z linii 3 (na dzień pisania materiału to 3.2.1).

Każda z linii posiada dodatkowo kilka „wersji”:
- **uncompressed** – standardowy kod JavaScript, czytelny dla nas. Zawiera wszystkie znaki (w tym puste), ewentualne komentarze oraz przyjazne nazwy funkcji. To rozwiązanie polecane jest dla programistów, którzy chcą zmienić kod biblioteki i/lub zrozumieć działanie jej pewnych funkcji.
- **minified-** wersja przeznaczona do wersji produkcyjnej strony. Nie zawiera komentarzy, przyjaznych nazw ani pustych znaków. Dzięki temu wersja ta jest znacznie mniejsza od pozostałych
- **slim** – odchudzona, jeszcze szybsza wersja biblioteki. W przeciwieństwie do pełnej wersji nie zawiera obsługi AJAX, animacji jQuery, metody each i load, parsowania XML oraz wsparcia dla przestarzałego kodu.
- **slim minified** – wersja slim dostosowana dla użytkownika końcowego

Niezależnie od wersji, którą ostatecznie zdecydujemy się używać, musimy posiadać dostęp do kodu biblioteki. Kod można zdobyć na dwa sposoby:

- **pobrać plik jQuery** – dzięki temu rozwiązaniu biblioteka zawsze znajduje się na naszym serwerze/w naszym repozytorium. To z kolei pozwala na lepsze zarządzanie kodem oraz pozwala

mieć pewność, że strona zawsze załaduje się poprawnie (nigdy nie dojdzie do sytuacji, w której kod jQuery będzie niedostępny dla naszej witryny)
- korzystać z repozytoriów kodu firm trzecich (np. Google, Microsoft, code.jquery.com, cdnjs), tzw. CDN (Content Delivery Network). Rozwiązanie tego typu ma w zasadzie szereg zalet. Po pierwsze, nie musimy każdorazowo dołączać pliku jQuery do naszego projektu – prosty znacznik <script> ze źródłem załatwi sprawę. Po drugie mamy pewność, że zawsze otrzymamy najnowszą wersje jQuery. Niektóre źródła podają, że dodatkowym atutem tego rozwiązania jest szybciej ładująca się strona. Dzieje się tak dlatego, że ze stron takich jak Google czy Microsoft korzystamy na co dzień (jak zresztą większość użytkowników – wszak głównymi wyszukiwarkami są Google bądź Bing). To z kolei powoduje, że w naszej przeglądarce (w jej pamięci podręcznej) prawdopodobnie będzie znajdywać się biblioteka jQuery (ze względu na to, że obie wyżej wspomniane firmy wykorzystują ją na swoich stronach). Oczywiście należy mieć na uwadze, że ten aspekt jest wątpliwy – użytkownicy coraz częściej wyłączają pamięć podręczną bądź czyszczą co jakiś czas zawartość pamięci przeglądarki. Ponadto kilkadziesiąt kilobajtów nie przyśpieszy znacznie ładowania strony (co najwyżej o kilkanaście milisekund).

Repozytoria Google:

<script src="https://ajax.googleapis.com/ajax/libs/jquery/x.x.x/jquery.min.js"></script>

Repozytoria Microsoft:

<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-x.x.x.min.js"></script>

gdzie za x.x.x należy wstawić numer wersji biblioteki, który chcemy używać (np. 3.2.1)

2.3 Podstawy jQuery

Biblioteka sama w sobie jest definicją obiektu jQuery. Każde zapytanie wykonane przez jego konstruktor tworzy de facto nową instancję tego obiektu. Obiekt posiada swoje właściwości oraz metody, które możemy wywoływać w tzw. szeregu, co jest szczególnie korzystne dla programisty od strony wizualnej – większość kodu zajmuje przeważnie jedną linię.

Obecnie utworzenie obiektu jQuery może być wykonane na kilka sposobów. Najstarszym i najmniej konfliktowym zadeklarowanie obiektu jest po prostu wywołanie konstruktora

jQuery()

Dozwolone jest (a w zasadzie jest to najszybsza i najczęściej stosowana metoda) używanie znaku $ jako deklaracji zmiennej. Można go używać w następującej postaci:

$() - alias dla jQuery()
$ - ogólny uchwyt do obiektu (najczęściej stosowane do tworzenia operacji wykonywanych bezpośrednio o po załadowaniu strony)
$. - pozwala na wywołanie metody bądź właściwości obiektu jQuery teoretycznie bez bezpośredniej deklaracji jego samego

W dalszych przykładach wykorzystane zostaną znaki $. Pełną nazwę obiektu wykorzystywać należy jedynie w przypadku zachowania pełnej zgodności z bardzo starymi parserami JS.

INFORMACJA: Niektóre biblioteki również mogą wykorzystywać znak dolara celem odnośnika do swoich obiektów. W tym wypadku jQuery posiada specjalną funkcję .noConflict(), która to

uwalnia znak dolara dla innych zastosowań. W tym wypadku możemy napisać własny odnośnik do obiektu, np. :

var jqjs = $.noConflictO);
jqjs(‘document’).ready(function() {/*kod wywołanej funkcji po załadowaniu dokumentu; w nim też musimy wykorzystywać nasz alias*/});

2.3.1 Reakcja na zmianę

Przykładowe operacje zmiany stanów wskazanych elementów jQuery:
- fade (wygaszanie) – zbiór metod pozwalających na wygaszanie (zanikanie) bądź rozjaśnianie (pokazanie) elementów obiektu jQuery.
- hide/show (ukrywanie i pokazywanie) – zbiór metod pozwalających na ukrywanie bądź pokazywanie elementów obiektu jQuery.
- slide (suwanie) – zbiór metod pozwalających na nadanie efektu suwania się elementów obiektu jQuery
- animate (animacja) – metoda pozwalająca na wykonanie animacji elementów obiektu jQuery. Animacja wykonuje się nieprzerwanie do określonego punktu, chyba że użyjemy metody stop().

Metody zmian można ze sobą łączyć znakiem wiązania kropki (.). Będą one wtedy wykonywane jedna po drugiej (wedle ciągu).

2.3.2 Operacje na elementach HTML

- **operacje ustaw/pobierz (set/get)** – pozwala na pobieranie tekstu, tekstu z kodem HTML, wartości pola (tylko elementy z atrybutem value) bądź wartości atrybutu elementów obiektu jQuery poprzez następujące metody: text(), html(), val() oraz attr(nazwa_atrybutu). Te same metody, jeżeli podamy w nich wartość atrybutu, zadziałają jak metody set. Wyjątkiem będzie tutaj metoda attr(), która będzie potrzebować dwóch parametrów – nazwy atrybutu oraz nowej wartości dla atrybutu. Jako drugi parametr można podać funkcję wywoływaną (tzw. callback).
- **operacje dodawania oraz usuwana elementów oraz zawartości** – dzięki metodom append(), prepend(), after(), before(), remove(), empty() możemy dodawać nowe elementy, wartości do już istniejących elementów, a także usuwać elementy bądź czyścić ich wartości.
- **operacje przemieszczania po drzewie elementów** – metody parent(), parents() oraz parentsUntil() pozwalają na przemieszczanie wsteczne po drzewie rodziców. Dwie ostatnie metody pozwalają na wybór rodziców, jakie chcemy przekazać do obiektu jQuery.

Metody children() oraz find() pozwalają na wynalezienie wszystkich dzieci elementów obiektu jQuery. Metody mogą przyjmować dodatkowe parametry, określające nazwy, identyfikatory czy klasy elementów wybieranych.

Metody wynajdywania elementów bliźniaczych – siblings(), next(), nextAll(), nextUntil(), prev(), prevAll(), prevUntil(). Działają one podobnie jak selektory rodziców oraz dzieci jednak operują one jedynie przestrzenią bieżących elementów obiektu jQuery.
- **operacje filtracji elementów obiektu jQuery –** w przypadku gdy nasz wybór może zawierać więcej niż 1 wynik (np. 20 elementów), opisywana biblioteka ułatwia odnalezienie właściwego obiektu docelowego poprzez następująca funkcje:
1) first()/last() - wybiera pierwszy/ostatni element ze wszystkich obecnych w utworzonym obiekcie
2) eq(numer) – wybiera element o indeksie wskazanym przez parametr numer (działa jak indeks w tablicy wyników)
3) filter(kryteria) – wybiera elementy, które spełniają kryteria wskazanego kryterium podanego w parametrze kryteria. Parametrem może być np. nazwa znacznika, nazwa klasy czy identyfikatora.

Przykładowy kod:

```
<style>
#tablabel {
  background: white;
  border: 1px solid black;
  overflow: hidden;
  text-align: center;
}
#windowbody {
  display: none;
  overflow: hidden;
  background: orange;
}
.box {
  float: left;
  width: 100px;
  height: 100px;
}
.box::after {
  content: '';
  display: table;
  clear: both;
}
</style>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<script>
$(function() {$("#tablabel").click( function() {$("#windowbody").slideToggle();});
   $('.box').click(function() {$('.box').fadeToggle();});
   $('#windowbody').find('p').click(function() {$('.box').fadeToggle(2000)});
 });
</script>
<div id="window">
  <div id="tablabel">
      <p>
      Kliknięcie spowoduje rozwinięcie reszty okna
      </p>
  </div>
  <div id="windowbody">
    <p>
      Reszta oka. Kliknij na napis by rozpocząć działanie przykładu.
    </p>
    <div class="box" style="background: blue;">
    </div>
     <div class="box" style="background: red;">
    </div>
     <div class="box" style="background: yellow;">
    </div>
     <div class="box" style="background: pink;">
    </div>
  </div>
</div>
```

Odnośnik do kodu jsfiddle: https://jsfiddle.net/b96pa4rh/

2.3.3 Obsługa AJAX

Obsługa metody AJAX nie wymaga tworzenia obiektu. Spowodowane jest to pełną autonomicznością działania mechanizmu komunikacji modułu XMLHttpRequest (który jest tutaj podstawą działania całej komunikacji). Wyjątek stanowi metoda load, która wywołana dla konkretnej zawartości obiektu jQuery może podmieniać w nim zawartość na aktualnie pozyskaną.

- load() - Metoda pobiera wskazany plik/wykonuje wskazany skrypt serwerowy i przesyła do elementu wywołującego odpowiedź. Może przyjąć do 3 parametrów, z czego tylko pierwszy jest obowiązkowy (adres pliku, do którego wysyłamy zapytanie). Drugim parametrem modą być dane, które przesyłamy do pliku. Ostatni parametr powala na wywołanie funkcji po odebraniu odpowiedzi z pliku.
- .get – działa podobnie do load(). pozwala na wysłanie żądania pod wskazany adres i odebrania od niego wyników. Przyjmuje jednak dwa obowiązkowe parametry – pierwszym z nich jest adres do pliku/skryptu, drugi natomiast to funkcja typu callback.
- .post – jej działanie jest podobne do metody load(). Różnicą jest możliwość podania 4 parametru, który pozwala określić jakiego typu danych oczekujemy od serwera. Domyślnie metoda próbuje zgadnąć jaki typ danych został zwrócony.

UWAGA: Funkcja typu callback ma konkretne parametry, które może przyjąć:
- data – dane odebrane z żądania (w zwróconym formacie)
- status – informacja o statusie żądania
- xhr – uchwyt do utworzonego przez metodę obiektu XMLHttpRequest

INFORMACJA: Proszę pamiętać, że jQuery posiada więcej metod. W niniejszym materiale przedstawione zostały jedynie najważniejsze z nich. Pełną dokumentację oraz przykłady można znaleźć np. na stronie projektu lub na stronach o technologii HTML (odnośniki w materiałach).

3. Zadania do wykonania

Należy wykorzystać bibliotekę jQuery w swoim projekcie. W projekcie powinna znaleźć przynajmniej obsługa żądań (load/post/get) po to, by strona nie musiała być każdorazowo przeładowywana przy zmianie podstron. Ponadto mile widziana byłaby galeria, która zarządzana byłaby przez obiekt jQuery.
Swoją stronę należy przystosować pod urządzenia przenośne. W tym celu można zapoznać się z jQueryMobile oraz znacznikiem @media CSS.

MATERIAŁY:
https://code.jquery.com/jquery/
https://cdnjs.com/libraries/jquery/
https://github.com/jquery/jquery-migrate
http://stackoverflow.com/questions/35424053/what-are-the-differences-between-normal-and-slim-package-of-jquery
http://jquery.com
https://www.w3schools.com/jquery/default.asp