Narzędzia internetowe – laboratorium

HTML5 – Geolokalizacja oraz WebStorage

Cel zajęć

Celem laboratorium jest poznanie nowych możliwości HTML5 takich jak geolokalizacja, magazyn sieciowy *sessionStorage* oraz *localStorage*.

Zadanie 1

Sprawdź działanie kodu przedstawionego na listingu 1. Następnie zmodyfikuj go tak, aby wyświetlał nie tylko szerokość ale również długość geograficzną, na której się znajdujesz.

Listing 1. Przykładowy kod strony HTML wykorzystującej geolokalizację

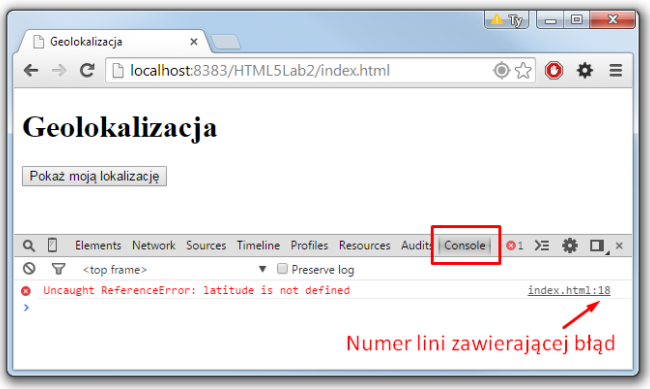
```
<!DOCTYPE html>
<html>
    <head>
        <title>Geolokalizacja</title>
        <meta charset="UTF-8">
        <script>
            function showLocation(position) {
                var latitude = position.coords.latitude;
                var output = document.getElementById("geo");
                output.innerHTML = "<p>Szerokość geograficzna: " + latitude + "</p>";
            }

            function errorHandler(error) {
                var output = document.getElementById("geo");
                switch (error.code) {
                    case error.PERMISSION_DENIED:
                        output.innerHTML = "Użytkownik nie udostępnił danych.";
                        break;
                    case error.POSITION_UNAVAILABLE:
                        output.innerHTML = "Dane lokalizacyjne niedostępne.";
                        break;
                    case error.TIMEOUT:
                        output.innerHTML = "Przekroczono czas żądania.";
                        break;
                    case error.UNKNOWN_ERROR:
                        output.innerHTML = "Wystąpił nieznany błąd.";
                        break;
                }
            }

            function getLocation() {
                if (navigator.geolocation) {
                    var options = {timeout: 60000};
                    navigator.geolocation.getCurrentPosition(
                            showLocation,
                            errorHandler,
                            options);
                } else { alert("Twoja przeglądarka nie wspiera geolokalizacji!");}
            }
        </script>
```

```
    </head>
    <body>
        <h1>Geolokalizacja</h1>
        <div id="geo"></div>
        <p><button onclick="getLocation()">Pokaż moją lokalizację</button></p>
    </body>
</html>
```

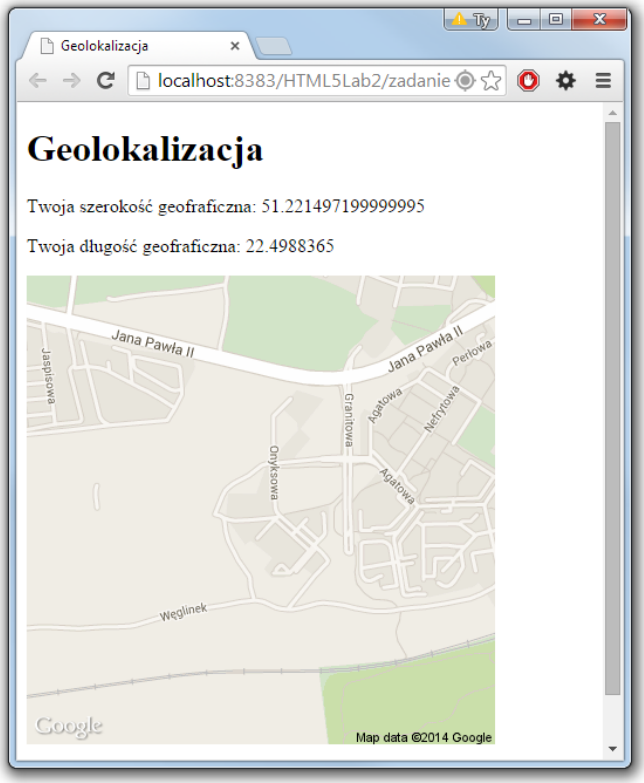
W przypadku trudności skorzystaj z konsoli dostępnej w przeglądarce *Chrome* (rys. 1) lub *Mozilla Firefox* do znalezienia błędu działania skryptu JS.



Rys. 1. Sprawdzanie błędów JS w konsoli

Zadanie 2

Wykorzystując skrypt przygotowany w zadaniu 1, rozszerz stronę HTML o statyczną mapę (rys. 1) w oparciu o API Google Maps (listing 2).



Rys. 2. Strona ze statyczna mapą

Zmienna **img_url** na listingu 2 zawiera adres URL do pliku graficznego wygenerowanego na podstawie podanych w adresie paramentów. Przekaż ten adres do znacznika **<img>.**

Listing 2. Generowanie mapy w oparciu o API Google Maps

```
var img_url = "http://maps.googleapis.com/maps/api/staticmap?center="
            + 51.22 + "," + 22.42     //szerokość i długość geograficzna
            + "&zoom=15&size=400x400&sensor=false";
```

Zadanie 3

Do strony przygotowanej w poprzednich zadaniach dodaj interaktywną mapę w oparciu o API Google Maps (listing 3).
Komercyjne strony wykorzystujące API powinny w ładować bibliotekę map w oparciu o unikalny klucz do uzyskania na stronie: https://developers.google.com/maps/documentation/javascript/tutorial#api_key. Natomiast API z listingu 3 wykorzystuje eksperymentalną wersję biblioteki.

Listing 3. Dodanie eksperymentalnej wersji biblioteki API Google Maps do strony

```
<script src="//maps.googleapis.com/maps/api/js?v=3.exp&sensor=true"></script>
```

Przykładowy kod został umieszczony na listingu 4. Wykorzystaj z dodatkowych opcji konfiguracji mapy przedstawionych a stronie dokumentacji map (w szczególności *Developer's Guide* oraz *Code Samples*): https://developers.google.com/maps/documentation/javascript/maptypes

Listing 4. Przykład kodu wykorzystania mapy

```
var mapOptions = {
           zoom: 12,
           center: new google.maps.LatLng(50.10, 22.50),
           mapTypeId: google.maps.MapTypeId.HYBRID
       };

var map = new google.maps.Map(document.getElementById('imap'), mapOptions);
```

Pamiętaj o dodaniu w sekcji **body** kontenera **div** o odpowiednim identyfikatorze. Dodatkowo , aby mapa się wyświetliła, konieczne jest ustawienie wysokości i szerokości tego kontenera **div**.

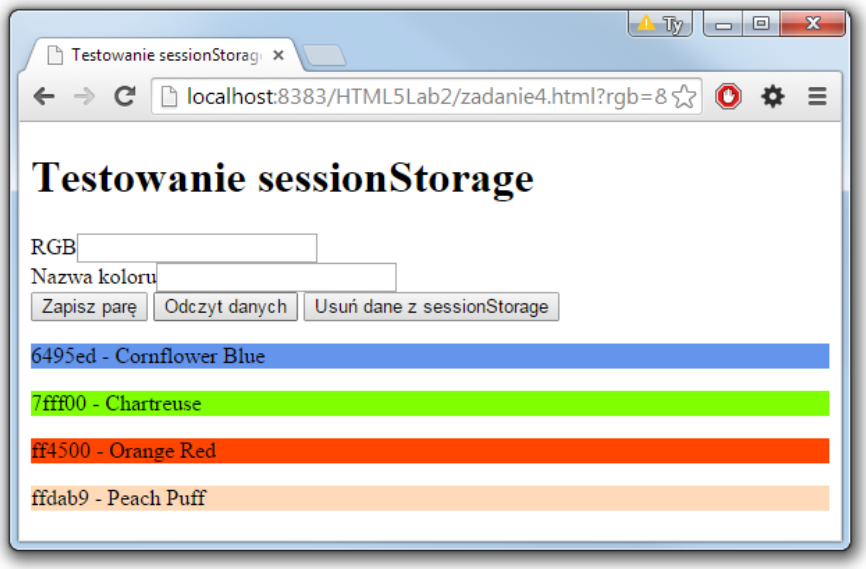
Zadanie 4

Stwórz formularz zawierający dwa pola tekstowe: RGB oraz nazwa koloru. Do formularza dodaj trzy przyciski wraz z kodem JS do ich obsługi (rys. 3):

- zapis danych do *sessionStorage*,
- odczyt wszystkich danych zapisanych w *sessionStorage* i wyświetli je pod formularzem,
- usunięcie wszystkich wprowadzonych danych z magazynu sieciowego *sessionStorage*.

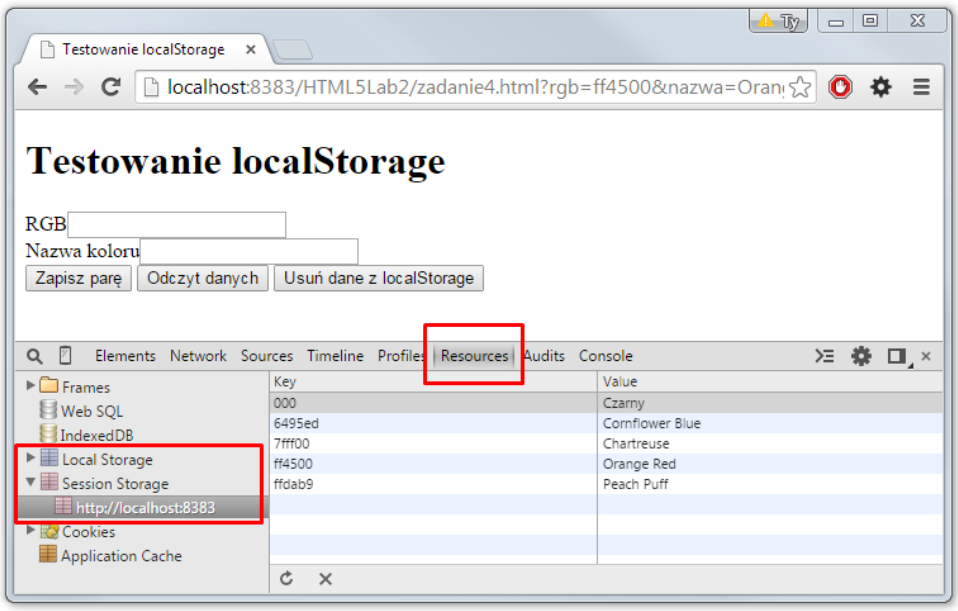
Pamiętaj o sprawdzeniu czy przeglądarka obsługuje *sessionStorage*. Wykorzystaj wszystkie metody obiektu *sessionStorage*:

- **getItem(key)** – zwraca wartość dla danego klucza lub **null** jeśli klucz nie istnieje,
- **setItem(key, value)** – zapisuje wartość dla danego klucza,
- **removeItem(key)** – usuwa klucz,
- **key(position)** – zwraca klucz do wartości umieszczonej pod określoną pozycją,
- **clear()** – usuwa wszystkie pary klucz-wartość.



Rys. 3. Przykład realizacji strony

Zapisane pary klucz-wartość możesz sprawdzić w przeglądarce *Chrome* (rys. 4) korzystając z opcji *„Zbadaj element”* i zakładki *„Resources”*.



Rys. 4. Dostęp do zapisanych wartości w *sessionStorage/localStorage*

Sprawdź działanie strony dla dwóch okien przeglądarki otwartych jednocześnie.

Zadanie 5

Utwórz stronę HTML zawierającą formularz z trzema kontrolkami:

- nazwa produktu,
- kolor,
- liczba sztuk.

Do formularza dodaj trzy przyciski wraz z kodem JS do ich obsługi:

- „*Zapisz produkt do koszyka*” - zapisuje dane z formularza do *localStorage*;

- *„Wyświetl koszyk”* – wyświetla listę wszystkich produktów w dodanych do koszyka na stronie w **postaci tabeli**, w przypadku braku produktów wyświetlany jest stosowny komunikat;
- *„Usuń wszystkie produktu”* – usuwa wszystkie elementy zapisane w *localStorage.*

Ponieważ formularz złożony jest z większej ilości danych niż para-klucz, zapisz wartości z formularza jako obiekt do *localStorage* (listing 5).

Listing 5. Zapis obiektu do *localStorage*

```
//tworzymy obiekt o odpowiednich atrybutach
var item = {};
item.rgb = "0c0c0c";
item.name = "Kapelusz myśliwski";

//zapisujemy i odczytujemy obiekt korzystając z metod klasy JSON
localStorage.setItem('item_1', JSON.stringify(item));
var retrieveItem = JSON.parse(localStorage.getItem('item'));
```

Możesz także sprawdzić przetwarzany obiekt korzystając z konsoli przeglądarki za pomocą polecenia **`console.log(item)`**.
Sprawdź działanie strony dla dwóch okien przeglądarki otwartych jednocześnie. Czy dwa odrębne okna mają dostęp do tego samego magazynu *localStorage*?

Zadanie 6

Zmodyfikuj tak stronę i skrypt utworzone w zadaniu 5, aby było możliwe usuwanie pojedynczych wybranych elementów z koszyka.