Techniki multimedialne

Rysowanie z wykorzystaniem łączności P2P w JavaScript

Michał Wereszczak

L7, 3EF-DI

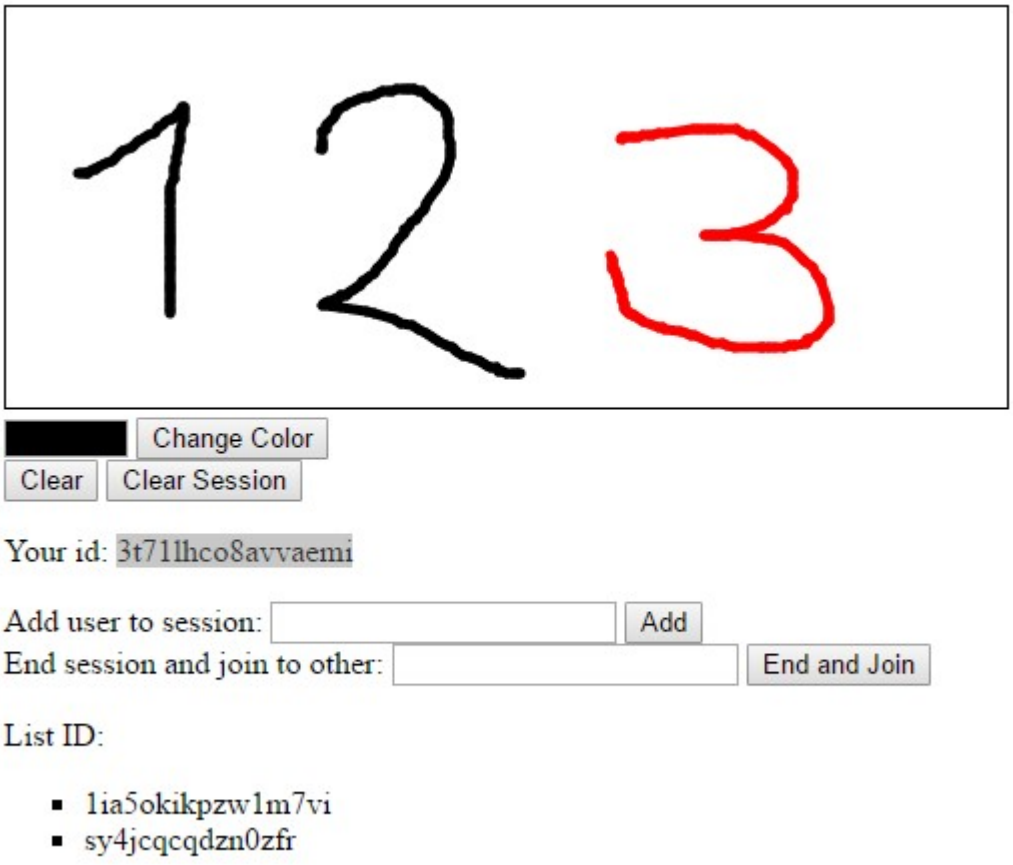
1. Opis projektu

Celem projektu było napisanie aplikacji webowej, wykorzystującej łączność p2p w JavaScript. W celu pokazania możliwości p2p zostało zrealizowane rysowania na wspólnym obszarze przez użytkowników połączonych ze sobą. W aplikacji jest możliwość tworzenia sesji, dołączania do niej, zapraszania użytkowników, opuszczania.

Aplikacja składa się ze strony html oraz skryptu JavaScript. Każdy użytkownik ma taką samą aplikację i ma takie same prawa.

Dokumentacja kodu została wygenerowana przez jsdoc i umieszczona wraz z projektem pod adresem:

https://github.com/mwereszczak/drawing



Rysunek 1 - Zrzut ekranu aplikacji.

2. Post-Mortem

Podczas realizacji projektu pojawiło się kilka problemów, które wstrzymały prace na jakiś czas.

Pierwszym z nich było uruchomienie komunikacji między przeglądarkami aby można było wysłać komunikat. Rozwiązanie problemu pojawiło się po około dwóch godzinach, gdy bardziej zapoznałem się z przykładami i dokumentacją biblioteki PeerJS.

Kolejną rzeczą ważną odnotowania jest konieczność sprawdzenia czy biblioteka wykorzystuje serwer do komunikacji się przeglądarek. Z analizy pakietów przeprowadzonej programem Wireshark można stwierdzić, że biblioteka wykorzystuje serwer pośredniczący tzw. broker. Spowodowało to konieczność rozbudowy projektu o coś co wykorzystuje PeerJS.

Kolejnym problemem jak to przy używaniu nowych rzeczy było uruchomienie rysowania w przeglądarce za pomocą myszki. Problem podobny do pierwszego, rozwiązany w podobnym czasie.

Trzecim problemem było połączenie tych dwóch elementów nad którymi wcześniej pracowałem. Mianowicie wysyłanie do drugiej przeglądarki danych umożliwiających zobaczenie na drugiej przeglądarce tego co jest rysowane na pierwszej. Samo zapakowanie i odbieranie danych w JSON nie było problemem. Dane przechodziło zbyt wolno i często były gubione. Pierwszym pomysłem było paczkowanie danych, lecz nie przyniosło pożądanych rezultatów. Działo się tak ponieważ za każdym podczas ruchu myszki było nawiązywane nowe połączenie. Rozwiązaniem problemu było nawiązywanie połączenia tylko wtedy gdy go nie ma lub zostało przerwane.

Problemem nad którym spędziłem najwięcej czasu w stosunku do trudności był dostęp do listy id, do których miały być wysyłane dane. Używając obiektowego modelu dokumentu HTML i metoda języka JavaScript chciałem zaktualizować listę na stronie. Metoda document.getElementById(‘’listId”) zwracała null. Spowodowane było to niezamknięciem cudzysłowia w znaczniku ul. Mała nieuwaga spowodowała chwilowy zastój.

Kolejnym napotkanym problemem było przy dodawaniu użytkownika do sesji. Pierwotnie połączenie było nawiązywane przy rozpoczęciu rysowania, stąd początkowy fragment nie został wysyłany. Rozwiązaniem tego było nawiązywanie połączenia już przy dodaniu użytkownika do sesji.

Ostatnim problemem było to, żeby podczas dołączania użytkownika do sesji synchronizacja przebiegła pomyślnie, aby każdy użytkownik wiedział o każdym i wszyscy widzieli to samo.

Najpoważniejszym problemem, którego rozwiązanie zajęło najwięcej czasu było nawiązywanie połączenia za każdym razem od nowa, natomiast najbardziej czasochłonnym poprawianie synchronizacji

3. Wykres Gantta

Czas zadania	Treść zadania
2h	Poszukiwanie odpowiedniej biblioteki do realizacji komunikacji p2p.
2h	Wykonanie pierwszej strony z implementacją znalezionej biblioteki.
1h	Poszukiwanie odpowiedniego przykładu realizującego rysowanie w przeglądarce i uruchomienie go.
3h	Połączenie p2p z rysowaniem.
1h	Dodanie zmiany koloru i czyszczenia obszaru rysowania.
2h	Zwiększenie liczby użytkowników, mogących się połączyć.
12h	Stworzenie synchronizacji między użytkownikami.
4h	Zmiana sposoby przechowywania śladów użytkowników.
3h	Testy aplikacji.
2h	Uzupełnienie kodu o komentarze i wygenerowanie dokumentacji.
1h	Przygotowanie Post-Mortem.

Razem 33h

4. Wnioski

Projekt udało się zrealizować w około 33 godziny, choć pewnie zajęło to trochę więcej czasu. W aplikacji zrealizowano synchronizację na podstawie różnych rodzajów komunikatów, które są opisane w dokumentacji kodu w opisie metody: preparePeerJ S(). Aplikacja nie obsługuje błędów typu: utrata połączenia z użytkownikiem, gdyż na to zabrakło już czasu. Aplikacja działa poprawnie i realizuje założone funkcje.