Instrukcja 3

1. Cel ćwiczenia

Obecnie strony WWW, celem przyciągnięcia jak największej liczby odwiedzających, stają się coraz bardziej multimedialne. Naszym celem będzie uatrakcyjnienie naszej strony – stworzymy własne animacje.

2. Teoria

Strony multimedialne nie są niczym nowym. Od czasu popularyzacji tych dokumentów kolejni projektanci i inżynierowie tworzą technologie i rozwiązania pozwalające na opracowywanie w pełni funkcjonalnych, a przy okazji interaktywnych stron WWW. Kto dzisiaj wyobraża sobie brak stron, na których możemy przeglądać filmy, odsłuchiwać utwory muzyczne, przeglądać mapy oraz dokonywać wirtualnych wycieczek po świecie bądź zagrać w gry bezpośrednio z przeglądarki.

Pierwszymi rozwiązaniami pozwalającymi na dodawanie animacji oraz pisanie gier działających w przeglądarce były odpowiednio Flash i Dreamweaver. Obie technologie nie były pierwotnie zakładane jako stricte dla przeglądarek – więcej o nich (głównie o technologii Flash) można przeczytać w dodatkowych materiałach (odnośniki na końcu instrukcji). Oczywiście technologie te nie działają w przeglądarce od razu – najpierw musi zostać pobrana i zainstalowana odpowiednia wtyczka. Ponadto technologie te, wraz z ich rozwojem, stały się coraz cięższe do zastosowań internetowych oraz bardziej podatne na ataki (większa funkcjonalność – więcej dość niebezpiecznych luk do wykorzystania przez atakujących).

W chwili projektowania piątej wersji HTML inżynierowie uwzględnili możliwość dodawania treści multimedialnych – audio, wideo, animacji czy renderowania grafiki – bez konieczności instalowania dodatkowych wtyczek. Zadaniem tym obarczone zostały silniki renderujące dokumenty HTML, czyli przeglądarki. Ponieważ sam język HTML nie jest językiem programowania, rolę wykonywania instrukcji warunkowych, skoków czy poszczególnych zadań otrzymał język JavaScript, stając się od tego momentu nieodzowną częścią stron WWW (do wersji 5 był jedynie dodatkiem, bez którego według specyfikacji renderowane strony miały działać tak samo jak z nim).

Nie tylko JavaScript pozwala na tworzenie animacji na stronach WWW. Jeżeli nasza animacja nie jest skomplikowana i polega np. jedynie na przesuwaniu, powiększaniu bądź innym prostym przemieszczeniom to możemy do jej wykonania użyć mechanizmów CSS. Za animację elementów HTML odpowiada właściwość transition, natomiast za animowanie elementu na początku załadowania strony/cykliczne animowanie poszczególnych elementów (ze zmianą kolorów, skali, umiejscowienia, itp.) odpowiada właściwość animation. Druga właściwość wymaga ponadto utworzenia tzw. ramek kluczowych, które pozwalają na tworzenie pojedynczego przejścia z jednego stanu elementu do innego.

Ostatnim z multimedialnych aspektów stron WWW jest możliwość załączania do nich np. map wskazujących na lokalizację firmy, ciekawego miejsca bądź zachęcających do wirtualnej wędrówki przez konkretne lokalizacje.

2.1 Przejścia CSS

Jeden z podstawowych typów animacji na stronach WWW. Jego zastosowanie nie jest szerokie – pozwala bowiem jedynie na animację modyfikacji elementów dostępnych na stronie (bądź pojawiających się na stronie).

Najprostszą wersją animacji CSS jest zastosowanie właściwości transition. Jej składnia wygląda następująco:

transition: <właściwość_wyzwalacz_css> <czas_animacji> <charakterystyka_przejścia> <opóźnienie_startu>

gdzie:
- właściwość_wyzwalacz_css – właściwość znacznika, przy zmianie której uruchomiona zostanie animacja przejścia. Przykładowo właściwością tą może być szerokość (width), margines (margin, margin-left, itp.) kolor (color) itd.
- czas_animacji – czas, w jakim wskazany wcześniej parametr zmieni swoją wartość. Im większa wartość tym dłużej będzie np. zwiększana/zmniejszana szerokość znacznika czy jego kolor (tonalne przechodzenie jednego koloru w drugi). Podawany w sekundach
- charakterystyka_przejścia – wybór charakterystyki krzywej przyspieszenia przejścia. Domyślnym ustawieniem jest przejście liniowe (linear), jednak do wybory jest łącznie jedenaście możiwości (np. ease-in, ease-out)
- opóźnienie_startu – opcjonalne opóźnienie wystąpienie przejścia. Jeżeli ostanie ustawione, to animacja przejścia (a tym samym zmiana wskazanego parametru) rozpocznie się po wskazanym czasie. Podawany w sekundach.

Przykład:

```
<style>
div {
  width: 200px;
  height: 200px;
  overflow: hidden;
  background: blue;
  transition: width 5s ease-out 1s;
}
<style>
<div><p>Kliknij mnie!</p></div>
<script>
document.getElementsByTagName('div')[0].addEventListener('click',
                                    function(){
                                            this.style.width = "600px";
                                    });
</script>
```

Powyższy kod spowoduje wyświetlenie niebieskiego kwadratu, który po kliknięciu zwiększy swój rozmiar do 600 pikseli. Ponieważ dodana została właściwość przejścia, rozpoczęcie zmiany szerokości będzie opóźnione o 1 sekundę, po czym będzie ono trwało przez 5 sekund. Do tego krzywa przejścia będzie miała dodatkowy efekt spowolnienia animacji przy jej końcu.

Należy pamiętać, że w ramach właściwości transition możemy przypisywać kilka właściwości, a każda z nich może mieć swoje, niezależne parametry. Aby jednocześnie utworzyć efekt przejścia dla wysokości i szerokości właściwości należy zapisać w ten oto sposób;

transition: width 5s ease-out 1s, height 2s linear 1.2s;

Proszę zauważyć, że wysokość ma opóźnienie wynoszące 1.2 sekundy (1200 milisekund).

2.2 Animacje CSS

Kolejnym sposobem animowania elementów HTML jest utworzenie dla nich odpowiednich klatek kluczowych. Animacje mają tę przewagę nad przejściami, że działają niezależnie od zmiany parametrów jakichkolwiek znaczników w dokumencie. Programista ma możliwość zadecydowania kiedy dana animacja wystąpi, jak długo potrwa oraz czy jest jednorazowa bądź cykliczna.

Postać właściwości animacji jest następująca:

animation: <nazwa_klatki> <czas_trwania> <charakterystyka_przejścia> <opóźnienie_startu> <ilość_wystąpień> <kierunek> <stan_po_animacji> <stan_odtwarzania>

gdzie:

- nazwa_klatki – nazwa ramek kluczowych (zadeklarowanych w CSS), która ma zostać zastosowana do animacji
- czas_trwania – analogicznie do przejść – określa czas w jakim ma wykonać się animacja
- charakterystyka_przejścia – analogicznie do przejść – określa funkcję czasową, wedle której następuje animacja
- opóźnienie_startu – analogicznie do przejść – określa czas, po upływie którego rozpocznie się animacja
- ilość_wystąpień – określa liczbę powtórzeń wskazanej animacji. Domyślnie animacja odtwarzana jest jeden raz; jeżeli wpiszemy pod nią wartość infinite to animacja będzie się wykonywać bez przerwy
- kierunek – określa kierunek animacji (domyślnie wartość normal)
- stan_po_animacji – określa jak ma się zachować obiekt animowany po jej zakończeniu. Domyślnie zostaje on na swojej pozycji. Można go jednak cofnąć do wartości pierwotnej (sprzed animacji)
- stan_odtwarzania – określa czy animacja jest zatrzymana czy działająca

Dodatkowo należy również definiować ramkę/ramki kluczowe. Wzór ramek:

```
@keyframes <nazwa> {
	from {/* ustawienia początkowe modyfikowanych właściwości */}
	to {/* ustawienia końcowe modyfikowanych właściwości */}
}
```

lub

```
@keyframes <nazwa> {
	0% {/* ustawienia właściwości dla początku animacji */}
	5% {/* ustawienia właściwości przy wykonanych 5% czasu całej animacji */}
	6% {/* ustawienia właściwości przy wykonanych 6% czasu całej animacji */}
	…..
	100% {/* ustawienia właściwości na końcu animacji */}
}
```

gdzie:
- nazwa – nazwa ramek kluczowych animacji. Nazwa ta podawana jest we właściwości animation
- {} - ciało ramki kluczowej, w której umieszczane są selektory animacji. Jak można wywnioskować z powyższego wzoru minimalna ilość selektorów to 2. Zestaw ramek może mieć większą ilość selektorów – wszystko zależy od naszych potrzeb. Należy jednak pamiętać, że każdy selektor musi posiadać właściwości CSS, które mają podlegać modyfikacji w danym selektorze!

Przykład:

```
<style>
div {
  position: relative;
}
#anim {
  font-size: 70px;
  color: green;
  position: absolute;
  left: 100%;
  -webkit-animation: example 4s linear 0s 3 forwards;
  animation: example 4s linear 0s 3 forwards;
}
@-webkit-keyframes example {
  form {left: 100%; color: green;}
  to {left: 0%; color: red;}
}
@keyframes example {
  form {left: 100%; color: green;}
  to {left: 0%; color: red;}
}
</style>

<div>
<p id="anim">
Animacja!
</p>
</div>
```

W powyższym przykładzie animacja wykona się 3 razy (napis będzie przesuwał się od prawej do lewej), przy okazji zmieniając kolor z zielonego na czerwony. Proszę zauważyć, że w przypadku innego ustawienia właściwości left dla #anim animacja będzie wykonywać się od wskazanego w #anim miejsca (nie zaś od 100% jak to zostało zdefiniowane w ramce). W CSS użyty został prefiks -webkit- w związku z niekompatybilnością właściwości z niektórymi przeglądarkami (Safari aż do wersji 9, Google Chrome do wersji 43).

Animacje, podobnie do przejść pozwalają przypisać kilka grup ramek kluczowych. Przykład:

```
animation: example 4s linear 0s 3 forwards, example2 6s;
```

Po przecinku podane są ramki example2, których czas wykonania ma wynieść 6 sekund. W przeciwieństwie do pierwszej ramki zapisane pod tą ramką animacje wykonają się dokładnie raz, a sam element wróci do stanu sprzed animacji (brak odpowiedniej definicji właściwości).

Kod przejść i animacji można znaleźć pod adresem: https://jsfiddle.net/x8d6eqwn/

2.3 Transformacje CSS

Chociaż nie są one bezpośrednio związane z multimediami, transformacje mogą odegrać znaczącą rolę zarówno w animacji jak i przejściach. Pozwalają one bowiem zmieniać wygląd, układ czy

położenie elementów dokumentu HTML bez bezpośredniej zmiany ich pierwotnych wartości (stąd określenie transformacja zamiast modyfikacja).

Transformacje CSS charakteryzują się podobnymi właściwościami jak transformacje dokonywane w programach grafiki 2D/3D. CSS posiada 3 właściwości dotyczące przemian:

- transform – właściwość odpowiada za dokonywanie przemian na elementach HTML, dla których został przypisany selektor CSS. Przemian dokonuje się poprzez użycie odpowiednich funkcji, takich jak translate (przeniesienie), skew (przechylenie/pochylenie), scale (zmiana wielkości) itp. Transformacje mogą być dokonywane zarówno na płaszczyźnie 2D jak i 3D. Co ciekawe, w przypadku użycia opcji 3D (przeważnie polega ona na dodanie w funkcji 3 wymiaru oraz zmianie nazwy funkcji transformującej np. z scale na scale3d) nasza strona będzie działała znacznie płynniej i będzie mniej uciążliwa dla sprzętu użytkownika. Dzieje się tak dlatego, że przeglądarki wykorzystują w ich przypadku wsparcie sprzętowe GPU (o ile to możliwe).
- transform-origin – właściwość pozwala na na dokonanie transformacji w oparciu o zmieniony punkt startowy przemiany. Do dyspozycji mamy możliwość zmiany współrzędnych po osi X, Y oraz Z. Zmian możemy dokonywać z wykorzystaniem dowolnych jednostek – pikseli, centymetrów, procentów.
- transform-style – ustawienie tej właściwości pozwala na ustalenie renderowania wskazanych transformowanych elementów HTML w przestrzeni 3D. Dostępne są 4 możliwe wartości:
1) flat – elementy modyfikowane selektorem nie zachowują swojej pozycji 3D
2) preserve-3d – elementy zachowają swoją pozycję w przestrzeni 3D
3) initial – zmiana wcześniej ustawionej wartości na domyślną
4) inherit – element dziedziczy tę właściwość z elementu nadrzędnego (rodzica)

Ponieważ powyższe zagadnienie może być nieco zawiłe, najlepiej jest zapoznać się z przykładem jego zastosowania:

```
<style>
#one {
  margin-top: 100px;
  width:500px;
  height:400px;
  background: green;
  position: relative;
  transform: rotateY(-70deg);
}
#two {
  position: absolute;
  border: 2px solid orange;
  width: 300px;
  height: 100px;
  top: -20px;
  background: red;
    transform: rotate3d(0,1,0,45deg);
   transform-style: preserve-3d;
}
#four {
  position: absolute;
  width: 300px;
  height: 100px;
  border: 2px solid cyan;
  background: blue;
```

```
  top: 20px;
  transform: rotateY(-20deg);
  transform-style: preserve-3d;
}
#three {
  position: relative;
  width: 350px;
  height: 300px;
  background: yellow;
}

#five {
  position: absolute;
  border: 2px solid orange;
  width: 300px;
  height: 100px;
  top: -20px;
  background: red;
    transform: rotate3d(0,1,0,45deg);
}
#six {
  position: absolute;
  width: 300px;
  height: 100px;
  border: 2px solid cyan;
  background: blue;
  top: 20px;
  transform: rotateY(-20deg);
}
</style>
<div id='one'>
  <div id='two'>
    <div id='four'>

    </div>
  </div>
</div>
<div id='three'>
  <div id='five'>
    <div id='six'>

    </div>
  </div>
</div>
```

Działanie powyższego kodu można sprawdzić pod adresem: https://jsfiddle.net/hrmLz7k8/

2.4 Animacje na płótnie JavaScript.

Ostatnim typem animacji, a zarazem najbardziej efektownym, jest animacja na płótnie JavaScript. Element ten został dodany do piątej wersji HTML i jego integralną częścią jest język skryptowy (bez JS element ten w zasadzie nie ma żadnego zastosowania).

Ideowo element ten ma zastąpić animacje wykonywane przy użyciu wtyczek zewnętrznych w technologii Flash czy DreamWeaver. Należy jednak pamiętać, że wspomniane technologie w głównej mierze potrafią tworzyć jedynie animacje na scenach 2D (Dreamweaver pierwotnie był tworzony jako technologia dla gier w przeglądarkach, jednak jego obecny udział w rynku jest znikomy; Flash dopiero niedawno wzbogacony został w przyspieszenie sprzętowe oraz sceny 3D) podczas gdy element Canvas pozwala na tworzenie w pełni trójwymiarowych scen (chociaż w perspektywie 2D – ale o tym za chwilę).

Najważniejszym aspektem działania płótna jest pełne wsparcie sprzętowe. Oznacza to, że tworzone na nim animacje nie obciążają procesora głównego (szczególnie ważne w przypadku stron WWW na telefonach i tabletach), a przynajmniej nie w większości przypadków.

Sam standard płótna, chociaż dobrze udokumentowany, niestety nie jest w pełni wspierany we wszystkich przeglądarkach w każdym aspekcie. Dlatego, chociaż możliwe jest używanie perspektywy w pełni trójwymiarowej (kontekst webgl, webgl2) to niestety nie wszystkie przeglądarki są w stanie te perspektywy wykorzystać (szczególnie urządzenia przenośne). Dlatego najczęściej wykorzystywany jest kontekst 2D; pozwala on na tworzenie animacji 3D jednak w perspektywie 2D. W przypadku tworzenia prostych animacji bądź reklam takie rozwiązanie jest wręcz idealne.

Ponieważ w sieci jest wiele samouczków oraz gotowych przykładów (wszystkie odnośniki w materiałach) toteż nie będą one poniżej przedstawiane (w zasadzie opis płótna jak i samego WebGL nadaje się na osobny materiał bądź książkę, nie zaś instrukcję do ćwiczeń).

3. Zadania do samodzielnego wykonania.

Rozwijamy projekt należy wzbogacić w opisane powyżej animacje. Priorytetem jest wykorzystanie mechanizmów animacji CSS – animacja logo, transformacje w menu bądź treści strony (podmiana zdjęć czy tekstu).

Elementem dodatkowym może być stworzenie prostego baneru reklamowego. Korzystając z przykładów zawartych na HTML5 Canvas Tutorial (adres poniżej) bez problemu można stworzyć prosty baner reklamowy zachęcający do przeczytania artykułu bądź odwiedzenia galerii na naszej stronie. Jeżeli strona nie ma przewidzianego miejsca na reklamy płótno można dodać na dodatkowej warstwie, która pojawiać się będzie jako okno przywitania (tzw. splash) bądź prosty instruktarz do strony (wyświetlany przy pierwszych odwiedzinach – ciasteczka/system sesyjny!).

MATERIAŁY I ODNOŚNIKI:

http://www.html5canvastutorials.com
https://www.tutorialspoint.com/webgl/html5_canvas_overview.htm
https://developer.mozilla.org/pl/docs/Web/API/WebGL_API/Tutorial/Getting_started_with_WebGL
http://cssdeck.com/labs/html5-canvas-3d-cubes
https://developer.mozilla.org/en-US/docs/Web/API/HTMLCanvasElement/getContext
https://www.w3schools.com/cssref/css3_pr_transform.asp
https://www.w3schools.com/cssref/css3_pr_animation.asp
https://developer.mozilla.org/en-US/docs/Web/CSS/perspective