WDD 330 Web Frontend Development II
Samuel Palacios
Week 08
June 08th, 2022

Readings Assignment Notes

1. Ch8: Transforms and Transitions

- ✓ **Transformations.** Using the "transform" directive from CSS you can rotate, scale, and skew any element on a page. These transformations allow you to do certain types of effects and animations.

- ✓ **Translation.** It allows to position any element in any (x, y) position. Using transitions, you can make an element move through various positions in a cycle, for example.

```
.ad-ad2 h1:hover span {
    color: #484848;
    transform: translateX(40px);
}
```

- ✓ **Scaling.** This type of transformation allows you to change the scale of an element. The natural scale of any element is 1, if we scale in values less than or greater than one, the size of the modified element will be different, achieving the desired transformation.

```
.ad-ad2 h1:hover span {
    color: #484848;
    transform: translateX(40px) scale(1.5);
}
```

- ✓ **Rotation.** This allows to rotate, very similar to the translation, but in degrees, about the same or central position of the element. The measure of rotation is given in degrees.

```
.ad-ad2 h1:hover span {
    color: #484848;
    transform: rotate(10deg) translateX(40px) scale(1.5);
}
```

- ✓ **Transitions.** This type of transformation is the one that allows you to bring all the previous ones together and consolidate the animations of web elements. In the

transitions, other types of transformation can be used, such as rotations, scales, and their time and frequency can be graduated.

```css
.ad-ad2 h1 span {
  transition-property: transform;
  transition-duration: 0.2s;
  transition-timing-function: ease-out;
  transition-delay: 50ms;
}
```

- ✓ **Multiple Transitions**. Two important variables are handled in transitions: time and frequency. The other aspect is the trigger, you must indicate what function or what elements in CSS will give life to the transition, you can also include delays, variations, etc.

```css
.ad-ad2 h1 span {
  transition-property: transform, color;
  transition-duration: 0.2s, 0.1s;
  transition-timing-function: ease-out, linear;
  transition-delay: 50ms;
}
```

- ✓ **Animations.** The principle to be able to implement it, are the transformations that we reviewed previously. The basis of CSS Animations is **Key Frames**. In short, a key frame is like a frame, like a video, in short, an animation.

```css
@keyframes moveRight {
  from {
    transform: translateX(-50%);
  }
  to {
    transform: translateX(50%);
  }
}

@keyframes appearDisappear {
  0%, 100% {
    opacity: 0;
  }
  20%, 80% {
    opacity: 1;
  }
}

@keyframes bgMove {
  100% {
    background-position: 120% 0;
  }
}
```

- ✓ To define the **key frames**, you can set other properties such as: the name of the animation (animation-name), the duration (animation-duration), progression (transition-timing-function), frequency or number of iterations (animation-iteration -count), direction and positioning (animation-direction), delays (animation-delay),

```
.ad-ad3 :after {
  content: '';
  width: 90px;
  height: 92px;
  background-image: url(../Images/bike_sprite.png);
  display: block;
  margin: auto;
}

@keyframes bike {
  0% {
    background-position: 0 0;
  }
  100% {
    background-position: -360px 0;
  }
}
```

2. Ch12: Canvas, SVG, and Drag and Drop

- ✓ **HTML5** was a revolution along with **CSS3**, ever since it came out and as time went on all browsers supported it, making animations, games, and anything else great just using HTML and CSS was easier than ever. HTML5 and CSS3 implemented new properties and functionality.

- ✓ There are 3 basic elements to power **animations**: Canvas API, SVG (vector images), and Drag and Drop abilities over HTML5.

- ✓ **Canvas** is an API that allows you to draw like on a canvas. With that API, all kinds of geometric figures can be drawn. For example, we can draw lines, text, rectangles, polygons, paths, etc.

```
var canvas = document.getElementById("myCanvas");
var context = canvas.getContext("2d");
context.strokeStyle = "red";
context.fillStyle = "rgba(0, 0, 255, 0.5)";
context.fillRect(10, 10, 100, 100);
context.strokeRect(10, 10, 100, 100);
```

- ✓ We can also manipulate images or use them to put them on the **canvas**. You can load any image resource, manipulate it, and render it on the canvas. The Canvas API is so powerful and so versatile that you can manipulate images and videos and add any shape or other image on top of it. For example, a video can be modified by overlaying text, the possibilities are endless.

```
function manipulateImage() {
    var canvas = document.getElementById("demo7");
    var context = canvas.getContext("2d");
    var image = document.getElementById("secondImage");
    context.drawImage(image, 68, 68);

    var imageData = context.getImageData(0, 0, 200, 200);

    var red, green, blue, greyscale;

    for (var i = 0; i < imageData.data.length; i += 4) {
      red = imageData.data[i];
      green = imageData.data[i + 1];
      blue = imageData.data[i + 2];
    }
  }
```

```
function makeVideoOldTimey() {
  var video = document.getElementById("video");
  var canvas = document.getElementById("canvasOverlay");
  var context = canvas.getContext("2d");

  video.addEventListener("play", function() {
    draw(video,context,canvas);
  }, false);

}
```

- ✓ **SVG** or scalable vectors graphics, is a file or fragment of code or XML tags that allows lines and geometric figures to be represented with great mathematical precision, so much so that it can represent almost any image. The advantage of these graphics is that they do not degrade when changing the scale or zoom, with this type of graphics the quality is not lost (since the intermediate points to display the image are calculated mathematically).

```svg
<svg xmlns="http://www.w3.org/2000/svg" viewbox="0 0 400 400">
  <desc>Drawing a rectangle</desc>
      <rect x="10" y="10" width="100" height="100"
            fill="blue" stroke="red" stroke-width="3"  />

</svg>
```

- ✓ Differences between **Canvas and SVG**, the first allows pixels to be manipulated in a better way and has a more immediate way of printing or drawing, only with the disadvantage that it can no longer be manipulated until the next rendering; SVG, on the other hand, works better with vectors and does not degrade with any change in scale.

- ✓ **Drag and drop**, now with HTML5 and CSS3 there is the "draggable" property, which together with an API for touch events, allows any web element to have this property. By having the ability to grab and drop, you can make various types of animations, or cover various types of needs and requirements.