WEEK 08

Chapter 8: CSS TRANSFORMATION AND TRANSITIONS

- The CSS3 transform property gives you unprecedented control over many more aspects of the element’s appearance example translate, rotate, scale, skew
- element’s appearance can be manipulated using **transform functions**
- **Translation functions** allow you to move elements left, right, up, or down

Translation

- Translation function, moves elements without impacting the flow of the document.
- The translate(x,y) function moves an element x from the left, and y from the top:

transform: translate(45px, -45px);

- to move an element vertically or horizontally, you can use the *translatex* or *translatey* functions respectively
- Transforms don’t work on inline elements.

Scaling

- The scale(x,y) function scales an element by the defined factors horizontally then vertically.
- As with *translate*, you can also use the *scaleX*(x) or *scaleY*(y) functions
- To declare multiple transformations, provide a space-separated list of transform functions.
- Transforming does not cause a reflow.

Rotation

- The rotate() function rotates an element around the point of origin by a specified angle value.

- Positive degrees goes clockwise and negative degrees goes anti-clockwise
- transforms are applied in the order provided.

Skew

- The skew(x,y) function specifies a skew along the x and y axes.
- As with translate and scale, there are axis-specific versions of the skew transform:
  skewX() and *skewY*().

Changing the Origin of the Transform

- the transform-origin property control the origin from which your transforms are applied

Transitions

- Transitions allow the values of CSS properties to change over time, essentially providing simple animations.
- to create a simple transition using only CSS:
  1. Declare the original state of the element in the default style declaration.
  2. Declare the final state of your transitioned element;
  3. Include the transition functions in your default style declaration using the transition properties.

transition-property

- The transition-property property specifies the properties to be transitioned.
- Any property changing from one value to another for which you can find a valid midpoint can be transitioned.
- It is important to include a pre-state and a post-state.

transition-duration

- The transition-duration property sets how long the transition will take

transition-timing-function

- The transition-timing-function controls the pace of the transition in detail
- common timing functions include the key terms ease, linear, ease-in, ease-out, or ease-in-out.

- You can also describe your timing function more precisely by defining your own cubic-bezier function.
  cubic-bezier(0, 0, 1, 1).

The transition-delay Property

- transition-delay property introduce a delay before the transition begins.
- Include the number of milliseconds (ms) or seconds (s) to delay the transition.

```
-webkit-transition-delay: 50ms;
transition-delay: 50ms;
```

all transition properties in action:

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

So simply:

```
.ad-ad2 h1 span {
transition: transform 0.2s ease-out 50ms;
}
```

- A *transitionend* event is fired upon completion of a CSS transition in both directions.
- If you have more than one property being transitioned, the *transitionend* event will fire multiple times.

Animations

- A **keyframe** is a snapshot that defines a starting or end point of any smooth transition.
- To create an animation, use the @keyframes rule

For an animation called *myAnimation*, the @keyframes rule would look like this:

```
@-webkit-keyframes myAnimation {
/* put animation keyframes here */
@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;
}
}
}
```

Animation Properties

- ***animation-name***: attach an animation to an element. no quotes around name because the name is an identifier and not a string.

- ***animation-duration***: the length of time an animation takes to complete one iteration the animation-duration *should* be considered required to animate an element.

- ***animation-timing-function*:** determines how the animation will progress over its duration.
  The options are: *ease*, *linear*, *ease-in, ease-out, easein-out, cubic-bezier() , step-start, step-end,* or a developer-defined number of steps with the *steps(number, direction)* function

- ***animation-iteration-count*:** how many times the animation will play through. If omitted, it will default to 1. You can use infinite for endless repetition.

- ***animation-direction*:** values includes *normal* (from 0% to 100%), *reverse* (from 100% to 0%) ,
  alternate (odd-numbered iterations), *alternate-reverse animation-direction* (alternate direction at every iteration but starts with reverse)

- ***animation-delay:*** how many milliseconds or seconds to wait before the browser begins the animation

- ***animation-fill-mode**:* defines what happens before the first animation iteration begins and after the last animation iteration concludes. values are *none, forwards, backwards*, or *both*.

- ***animation-play-state:*** defines the state of the animation. The available values are *running* and *paused.*

```
.verbose {
animation-name: appearDisappear;
animation-duration: 300ms;
animation-timing-function: ease-in;
animation-iteration-count: 1;
animation-direction: alternate;
animation-delay: 5s;
animation-fill-mode: backwards;
animation-play-state: running;
}
```

```
/* shorthand */
.concise {
animation: 300ms ease-in alternate 5s backwards appearDisappear;
}
```

Chapter 12 : Canvas, SVG, and Drag and Drop

Canvas

- we can draw shapes and lines, arcs and text, gradients and patterns with canvas.
  Creating a canvas:

```
<canvas>
Sorry! Your browser doesn’t support Canvas.
</canvas>
```

- Canvas has no default styling
- All drawing on the canvas happens via the Canvas JavaScript API.
- The context is the place where your drawing is rendered.
- Both *strokeStyle* and *fillStyle* saturate your brush with paint.
- *fillRect* and *strokeRect* methods take the X and Y coordinates where you want to begin drawing the fill or the stroke, and the width and height of the rectangle.
- Instead of *fillStyle*, you can use *CanvasGradient* or *CanvasPattern* object.
- To draw other shapes you should find the *path* of the shape
- **Paths** create a blueprint for your lines, arcs, and shapes.
- *startAngle* and *endAngle* represent the start and end angles along the circle’s circumference that you want to draw.
- We can use closePath method to close a method.
- Use can save a copy of what you have drawn using API’s toDataURL method

- We can also draw images onto the canvas element.
- You can redraw an image, but its not so different from the img tag
- You can make an image a black and out using *getImageData*
- With canvas, we can convert an image to a video

Accessibility Concerns

- A major downside of canvas in its current form is its lack of accessibility. The canvas does not create a DOM node.

SVG

- SVG stands for **Scalable Vector Graphics.**
- A file format that allows you to describe vector graphics using XML.
- The vector images preserve their quality even as you blow them up or shrink them.
- On SVG, *viewBox* attribute defines the starting location, width, and height of the SVG image.
- Instead of creating svg by hand, use *Inkscape*
- Raphaël12 is an open-source JavaScript library that makes drawing and animating with SVG much easier.

Canvas versus SVG

- Canvas allows for pixel manipulation.
- However canvas operates in the *immediate mode* i.e Every time you finish drawing a shape, the canvas no longer has access to that shape, because it won’t persist as an object that you can modify
- By contrast, SVG which uses *retained mode* meaning that the structure of the image is preserved in the document that describes it

Drag and Drop

- The Drag and Drop API allows elements to be defined as draggable and what happened when dragged and dropped.
- The API is unsupported by Android.
- Most common uses of the API are when dragging files (working in conjunction with File API).

- To add a drag and drop to your page:
    1. Set a draggable attribute to the HTML element
    2. Add an event listener for the *dragstart* event
    3. Add event listener for the *dragover* and *drop* events.

The DataTransfer Object

- The object allows to get and set data about elements that are being dragged.
- It allows:
    a. the type of data we're saving of the draggable element
    b. the value of the data itself
- JavaScript's *preventDefault* method prevents the default behavior after drop event have been fired.

QUESTIONS: