Session 6

Mohamed Emary

March 24, 2024

Some CSS Properties

Box Sizing

The `box-sizing` property allows us to include the padding and border in an element's total width and height.

```
div {
  box-sizing: border-box;
}
```

So if your element width is 100px, and you give it a 5px border instead of the 100px getting wider to 110px (not 105px because of left and right width), the 100px stays 100px and the 10px border is added inside the 100px so the actual content width is 80px.

The default value for `box-sizing` is `content-box`.

Note that `box-sizing` does not include margin, only padding and border.

So how to solve this margin issue?

To solve this we can put the element inside a container with a `border-box` box-sizing, then we give the container a padding. So the outer space for the element is the padding of the container.

Hover

To make an element interact with the mouse hover, we can use the `:hover` pseudo-class.

Pseudo-classes are keywords added to a selector that specifies a **special state of the selected elements**. Pseudo-classes are used with **a colon : behind them**.

Remember that it's only one colon : for pseudo-classes, not two because **two colons :: are used for pseudo-elements**.

In this example when we hover on the button, the background color changes to red.

```
button:hover {
  background-color: red;
}
```

The general form when dealing with pseudo-classes is:

```
selector:pseudo-class {
  property: value;
}
```

You can even make the hover on one element affect another element.

```
button:hover + p {
  color: red;
}
```

Transition

To make a transition effect on an element, we can use the `transition` property.

Transition makes element changes state smoothly over a specified **duration**.

To make a transition effect on an element we need to specify 3 things:

- `transition-property` - The property we want to transition (ex: width).
- `transition-duration` - The duration of the transition (ex: 5s).
- `transition-timing-function` - The timing function. (**optional**)
- `transition-delay` - The delay before the transition starts (ex: 2s). (**optional**)

For example if we have an element that changes width on `:hover` we can make it transition smoothly in 5 seconds like this:

```
div{
  width: 100px;
  transition-property: width;
  transition-duration: 5s;
  transition-timing-function: ease-in;
}

div:hover {
  width: 300px;
}
```

The `transition-timing-function` is used to specify the speed curve of the transition effect. The default value is `ease`.

`transition-timing-function` can have the following values:

Value	Description
`cubic-bezier`	a timing function that allows you to specify your own values
`ease` *Default*	Specifies a transition effect with a slow start, then fast, then end slowly (equivalent to `cubic-bezier(0.25,0.1,0.25,1)`)
`linear`	Specifies a transition effect with the same speed from start to end (equivalent to `cubic-bezier(0,0,1,1)`)

Value	Description
`ease-in`	Specifies a transition effect with a slow start (equivalent to `cubic-bezier(0.42,0,1,1)`)
`ease-out`	Specifies a transition effect with a slow end (equivalent to `cubic-bezier(0,0,0.58,1)`)
`ease-in-out`	Specifies a transition effect with a slow start and end (equivalent to `cubic-bezier(0.42,0,0.58,1)`)

The function `cubic-bezier` takes 4 prameters:

- `x1`: The x-coordinate of the first control point
- `y1`: The y-coordinate of the first control point
- `x2`: The x-coordinate of the second control point
- `y2`: The y-coordinate of the second control point

It's hard to define the transition curve using `cubic-bezier` so you can use this website to help you.

Also this website will help you better understand each value for `transition-timing-function`.

Most of the time we don't specify each property separately, instead we use the shorthand `transition` property.

It takes the following values:

```
transition: property duration timing-function delay;
```

It's important to keep the order `duration` and `delay` values since both take time values.

you can also specify transition effect for more than one property in the same line.

```
transition: width ease-in 2s, height 4s, background-color 1s;
```

If all properties have the same duration you can specify it once.

```
transition: all 2s;
```

This will make all properties transition in 2 seconds.

for example you can use it in a code like this:

```
div {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: all 2s;
}
div:hover {
  width: 300px;
  height: 300px;
  background-color: red;
}
```

This will make the `width`, `height`, and `background-color` transition take 2 seconds.

You can even ignore `all` and just use the duration value and this will make all properties transition in the same duration specified inside `transition`.

Notice that in the last code, the transition effect was specified inside `div` and not inside `div:hover`. This is because we want our transition effect to be applied to the element itself and not only in hover state.

Transform

Transform is a CSS that allows us to move elements. It can take the following values:

- `rotate()` - rotates an element. It can take a value in degrees like `rotate(45deg)`.
- `rotateX()` - rotates an element around its X-axis.
- `rotateY()` - rotates an element around its Y-axis.
- `scale()` - scales an element. It can take two values like `scale(2,2)` which is the scale factor for the width and height.
- `scaleX()` - scales an element horizontally (width).
- `scaleY()` - scales an element vertically (height).
- `skew()` - skews an element. It can take two values like `skew(30deg, 20deg)` which is the skew factor for the horizontal and vertical axis.
- `skewX()` - skews an element horizontally.
- `skewY()` - skews an element vertically.
- `translate()` - moves an element. It can take two values like `translate(50px, 100px)` which is the distance to move the element horizontally and vertically.
- `translateX()` - moves an element horizontally.
- `translateY()` - moves an element vertically.

The transform by default happens around the center of the element, but you can change the origin of the transform using `transform-origin`, for example you can make it `top right` or `bottom left`.

You can also apply `transition` to the transform property to make the transform effect smooth.

```
div {
  transition: transform 4s;
}

div:hover {
  transform: rotate(360deg);
}
```

With `skew` you can make pretty designs inside your website. Search for skew web design.

You can use negative values in each one of these functions

You can apply more than a `transform` function to an element but it should be on the same line and separated by a space like `transform: rotate(45deg) translate(50px, 100px);`, otherwise the last transform will override the previous ones.

For example using `transform: rotate(45deg);` then using `transform: translate(50px, 100px);` the translate will override the rotate so you will not see the rotation effect.

Overflow

The `overflow` property specifies what happens if content overflows an element's box. For example a text inside a `div` that is too much to fit inside the `div` so it overflows.

Using this property you can control the overflow of the content in four ways:

- `visible` - The overflow is not clipped. It renders outside the element's box. (*default*)
- `hidden` - The overflow is clipped, and the rest of the content will be invisible.
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content.
- `auto` - Similar to `scroll`, but it adds a scrollbar only when necessary.

You can also control the overflow for each direction separately using `overflow-x` and `overflow-y`.

With `overflow` property we can solve some issues we faced before:

The first issue is **Margin Collapse**.

Overflow & Margin Collapse

We already know margin collapse from **Session 4**, but a quick reminder:
Margin Collapse happens when two margins touch, they collapse into a single margin.
This problem happens only with top and bottom margins.

To solve margin collapse problem:

- Use `padding` on the parent container, instead of `margin` on the child container.
- Use `border` on the parent container.
- Use `overflow: auto;` on the parent container.

The second is float related issues:

Overflow & Float

We know from **Session 4** that float layout has two issues:

1. Floating elements are removed from the normal flow of the document, so parent element no longer contains the floated element. Example downside is if the container has a background color, it will not expand to contain the floating element.
2. The last floating element have to be cleared, otherwise it will affect the layout of the next element.

We can solve the first issue only using float
just give the parent container `overflow: auto;` property.

IFrame

IFrame allows us to embed another HTML page inside our current HTML page.

The syntax for IFrame is:

```
<iframe src="URL"></iframe>
```

You can also specify the width and height of the IFrame.

```
<iframe src="URL" width="500" height="500"></iframe>
```

You mostly will find embed option in the share menu of many websites like YouTube and Google Maps.

Summary

In this session, we covered several **CSS properties** like:

- **Box Sizing**: This property allows us to include the padding and border in an element's total width and height by giving it `border-box` value. The default value is `content-box` and it does not include margin.
- **Hover**: This pseudo-class allows an element to interact with the mouse hover. Pseudo-Classes are used with a colon : behind them.
- **Transition**: This property allows for smooth state changes over a specified duration. It requires the specification of the transition property, duration, timing function (optional), and delay (optional).
- **Transform**: This property allows us to move elements. It can take several values such as rotate, scale, skew, and translate. The transform origin can be changed using `transform-origin`.
- **Overflow**: This property specifies what happens if content overflows an element's box. It can be set to visible, hidden, scroll, or auto.
    - We also discussed how to solve **margin collapse** and **float** related issues using `overflow: auto;` property on the parent container.

We have also covered the **IFrame** concept which allows us to embed another HTML page inside our current HTML page.