CSS Backgrounds, Animation & Sprites

Agenda

- CSS Backgrounds
- Background Properties
- CSS Animations
- Animation Sub-properties
- Keyframes
- Transforms
- Transitions
- CSS Sprites

Backgrounds

The CSS background properties are used to define the background effects for elements.

Css Background Properties :

- Background-color
- Background-image
- Background-repeat
- Background-position
- Background-attachment
- Background-size

Background-Color

The background-color property specifies the background color of an element.

Example-

```
body {
background-color : blue;
}
```

With CSS, a color is most often specified by:

a valid color name - like "red"
a HEX value - like "#ff0000"
an RGB value - like "rgb(255,0,0)"

Background-Image

The background-image property specifies an image to use as the background of an element. By default, the image is repeated so it covers the entire element.

Example-

```
body {
  background-image: url("paper.gif");
}
```

Background-Repeat

The background-repeat CSS property sets how background images are repeated. A background image can be repeated along the horizontal and vertical axes, or not repeated at all.

Example-

```
body {
  background-image: url("paper.gif");

  background-repeat: repeat|repeat-x|repeat-y|no-repeat|initial|inherit;
}
```

Property values-

- **repeat** - The background image is repeated both vertically and horizontally. The last image will be clipped if it does not fit. This is default. .
- **Repeat-x** -The background image is repeated only horizontally.
- **repeat-y** -The background image is repeated only vertically.

Background-Repeat Cont..

- **space -** The background-image is repeated as much as possible without clipping. The first and last images are pinned to either side of the element, and whitespace is distributed evenly between the images.
- **round** - The background-image is repeated and squished or stretched to fill the space (no gaps).
- **initial** - Sets this property to its default value.
- **Inherit** - Inherits this property from its parent element.
- **no-repeat** -The background-image is not repeated. The image will only be shown once.

Background-Position

The background-position CSS property sets the initial position for each background image.

By default, a background-image is placed at the top-left corner of an element.

Example-

```
body {
  background-image: url("paper.gif");
  background-position: center;
}
```

Background-Position Cont..

Property Values -

- **left , top, center ,bottom** (if you specify only one property then other will by default be center)
- **x% y%** - The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. Default value is: 0% 0%.
- **xpos ypos -** The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions.
- **Initial**- Sets this property to its default value.
- **Inherit -** Inherits this property from its parent element.

Background-Attachment

The background-attachment property sets whether a background image scrolls with the rest of the page, or is fixed.

By default, it is set to scroll.

Example-

```
body {
  background-image: url("paper.gif");

  background-attachment: fixed;
}
```

Background-Attachment Cont..

Property Values

- **scroll** - The background image will scroll with the page. This is default.
- **fixed** - The background image will not scroll with the page
- **local** - The background image will scroll with the element's contents
- **initial** - Sets this property to its default value.
- **inherit -** Inherits this property from its parent element.

Background-Size

The background-size property specifies the size of the background images.

Example-

```
body {
  background-image: url("paper.gif");

  background-size: cover;
}
```

Property Values -

- **auto-** Default value. The background image is displayed in its original size.

- **length -** Sets the width and height of the background image. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto".

Background-Size Cont..

- **percentage** - Sets the width and height of the background image in percent of the parent element. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto".
- **cover** - Resize the background image to cover the entire container, even if it has to stretch the image or cut a little bit off one of the edges.
- **contain** - Resize the background image to make sure the image is fully visible.
- **initial** - Sets this property to its default value.
- **inherit -** Inherits this property from its parent element.

Animations

CSS animations make it possible to animate transitions from one CSS style configuration to another.

You can change as many CSS properties you want, as many times you want.

Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation’s style, as well as possible intermediate waypoints.

Keyframes hold what styles the element will have at certain times.

How the animation works

To create a CSS animation sequence, you style the element you want to animate with the animation property or its sub-properties. This lets you configure the timing, duration, and other details of how the animation sequence should progress.

To get an animation to work, you must bind the animation to an element.

Animation Sub-properties

animation-name :- Specifies the name of the @keyframes animation

animation-duration :- Specifies how long time an animation should take to complete one cycle

animation-timing-function :- Specifies the speed curve of the animation

animation-delay :- Specifies a delay for the start of an animation

animation-iteration-count :- Specifies the number of times an animation should be played

animation-direction :- Specifies the direction in which an animation should be played

animation-fill-mode :- Specifies a style for the element when the animation is not playing

Animation-play-state :- Specifies whether the animation is running or paused

Keyframes

Once you've configured the animation's timing, you need to define the appearance of the animation. This is done by establishing two or more keyframes using the @keyframes at-rule. Each keyframe describes how the animated element should render at a given time during the animation sequence.

Example :-

```
p { animation-duration: 3s; animation-name: slidein; }

 @keyframes slidein {

from { margin-left: 100%; width: 300%; }

to { margin-left: 0%; width: 100%; }

}
```

Transforms

CSS transforms allow you to translate, rotate, scale, and skew elements.
A transformation is an effect that lets an element change shape, size and position.
CSS supports 2D and 3D transformations.

2D Transforms support - X & Y directions.

3D Transforms support X, Y & Z directions.

Transformation Methods

2D Transforms :-

translate() :- The translate() method moves an element from its current position. We can also use translateX and translateY for respective axis.

rotate () :- The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

scale () :- The scale() method increases or decreases the size of an element. We can also use scaleX and scaleY for respective axis.

skew () :- The skew() element performs a transformation which displaces each point of an element by a given angle in each direction. We can also use skewX and skewY for respective axis.

matrix () :- The matrix() method combines all the 2D transform methods into one.

Transformation Methods contd ..

3D Transforms :-

translate3d – gets an additional z-axis parameter.

scale3d – gets an additional z-axis parameter. There is also scaleZ function that scale the element only in the z-axis.

rotate3d – gets four parameters – x, y and z that define the [x y z] direction vector and a degree to rotate in that direction. There is also a rotateZ function that rotate the element in the z-axis.

matrix3d :– the same as the matrix function but now gets 16 parameters.

Transitions

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized.

Transition Properties

transition-delay :- Specifies a delay (in seconds) for the transition effect

transition-duration :- Specifies how many seconds or milliseconds a transition effect takes to complete

transition-property :- Specifies the name of the CSS property the transition effect is for

transition-timing-function :- Specifies the speed curve of the transition effect

Example

```
.box { border-style: solid; border-width: 1px; display: block; width: 100px; height: 100px; background-color: #0000FF; transition: width 2s, height 2s, background-color 2s, transform 2s; }

 .box:hover { background-color: #FFCCCC; width: 200px; height: 200px; transform: rotate(180deg); }
```

Sprites

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

How to use Sprites

```
.flags-canada, .flags-mexico, .flags-usa {
  background-image: url('../images/flags.png');
  background-repeat: no-repeat;
}

.flags-canada {
  height: 128px;
  background-position: -5px -5px;
}

.flags-usa {
  height: 135px;
  background-position: -5px -143px;
}

.flags-mexico {
  height: 147px;
  background-position: -5px -288px;
}
```

Thank You!