# CSS3
## ANIMATION

## What is CSS3 Animation?

The CSS **animation property** can be used to animate an elements properties (color, background-color, height, width etc) from one state to another.

## How do animations work?

CSS animation requires **two components**.

## 1. @keyframes at-rule

Each animation needs to be defined with a @keyframes at-rule.

```
@keyframes animation-name
{

}
```

Each @keyframes at-rule **defines what should happen** at specific moments during the animation. For example, 0% is the beginning of the animation and 100% is the end.

```
@keyframes animation-name
{
    0% { background-color: #001F3F; }
    50% { background-color: #aa2255; }
    100% { background-color: #FF4136; }
}
```

The **“from” and “to” keywords** can also be used to define the beginning and end of the animation.

```css
@keyframes animation-name
{
    from { background: red; }
    to { background: yellow; }
}
```

## 2. Animation

For the animation to work, you must bind the animation to an element.

Each @keyframes is bound to the element either by the shorthand **animation property**, or its eight sub-properties.

The **animation property** and/or sub-properties define how the keyframes should be manipulated.

```
/* animation property */
.element
{
    width: 100%;
    height: 100%;
    animation: animation-name 5s infinite;
}
```

```css
/* animation sub-properties */
.element
{
    width: 100%;
    height: 100%;
    animation-name: animation-name;
    animation-duration: 5s;
    animation-iteration-count: infinite;
}
```

## animation
## sub-properties

The **animation sub-properties** are:

## animation-name

## animation-name

Specifies the name of the @keyframes at-rule describing the animation's keyframes.

The **animation-name value** must appear directly after the @keyframes keyword.

```
@keyframes button-push
{
    0% { background-color: #001F3F; }
    100% { background-color: #FF4136; }
}
```

The **animation-name value** must also be referenced either in the animation property or the animation-name sub-property.

```css
/* animation-name with property */
.element
{
    animation: button-push 5s infinite;
}
```

```css
/* animation-name with sub-property */
.element
{
    animation-name: button-push;
    animation-duration: 5s;
    animation-iteration-count: infinite;
}
```

## animation-duration

## animation-duration

Configures the length of time that an animation should take to complete one cycle.

```css
.element
{
    animation-duration: 300ms;
}
```

The **animation-duration value** is defined in seconds (s) or milliseconds (ms). The Default value is “0”

**Note:** If the animation-duration property is not specified, the animation will have no effect, because the default value is 0.

## animation-delay

## animation-delay

Configures the delay between the time the element is loaded and the beginning of the animation sequence.

```css
.element
{
    animation-delay: 1.5s;
}
```

The **animation-delay value** is defined in seconds (s) or milliseconds (ms). The Default value is “0”

## animation-direction

## animation-direction

Configures whether or not the animation should alternate direction on each run through the sequence or reset to the start point and repeat itself.

```
.element
{
    animation-direction: ;
}
```

The **normal value** means that the animation should be played as normal. The normal value is the default value.

```css
.element
{
    animation-direction: normal;
}
```

The **reverse value** means that the animation should play in reverse direction.

```css
.element
{
    animation-direction: reverse;
}
```

The **alternate value** means that the animation will be played as normal every odd time (1,3,5,etc.) and in reverse direction every even time (2,4,6,etc.) .

```css
.element
{
    animation-direction: alternate;
}
```

The **alternate-reverse value** means that the animation will be played reverse direction every odd time (1,3,5,etc.) and in normal direction every even time (2,4,6,etc.).

```
.element
{
    animation-direction: alternate-reverse;
}
```

## animation-iteration-count

## animation-iteration-count

Configures the number of times the animation should repeat; you can specify infinite to repeat the animation indefinitely.

```css
.element
{
    animation-iteration-count: ;
}
```

The **default value** is “1”. But any number value can be used.

```css
.element
{
    animation-iteration-count: 1;
}
```

The **infinite value** specifies that the animation should be played infinite times (for ever).

```css
.element
{
    animation-iteration-count: infinite;
}
```

## animation-play-state

## animation-play-state

Lets you pause and resume the animation sequence.

```
.element
{
    animation-play-state:  ;
}
```

The **running value** specifies that the animation is running. This is the default value.

```css
.element
{
    animation-play-state: running;
}
```

The **paused value** specifies that the animation is paused.

```css
.element
{
    animation-play-state: paused;
}
```

## animation-timing-function

**animation-timing-function**
Configures the timing of the animation; that is, how the animation transitions through keyframes, by establishing acceleration curves.

```
.element
{
    animation-timing-function:   ;
}
```

The **ease value** specifies that the animation has a slow start, then fast, before it ends slowly. This is the default value.

```css
.element
{
    animation-timing-function: ease;
}
```

The **linear value** specifies that the animation has the same speed from start to end.

```css
.element
{
    animation-timing-function: linear;
}
```

The **ease-in value** specifies that the animation has a slow start.

```
.element
{
    animation-timing-function: ease-in;
}
```

The **ease-out value** specifies that the animation has a slow end.

```
.element
{
    animation-timing-function: ease-out;
}
```

The **ease-in-out value** specifies that the animation has both a slow start and a slow end.

```
.element
{
    animation-timing-function: ease-in-out;
}
```

The **cubic-bezier(n,n,n,n) value** allows you to define your own values in the cubic-bezier function. Possible values are numeric values from 0 to 1.

```
.element
{
    animation-timing-function:
        cubic-bezier(n,n,n,n);
}
```

## animation-fill-mode

## animation-fill-mode

By default, CSS animations do not affect the element until the first keyframe is “played”, and then stops once the last keyframe has completed. The animation-fill-mode property can override this behaviour.

```
.element
{
    animation-fill-mode:   ;
}
```

The **none value** means that the animation will not apply any styles to the target element before or after it is executing. This is the default value.

```css
.element
{
    animation-fill-mode: none;
}
```

The **forwards value** means that after the animation ends (determined by animation-iteration-count), the animation will apply the property values for the time the animation ended.

```css
.element
{
    animation-fill-mode: forwards;
}
```

The **backwards value** means that the animation will apply the property values defined in the keyframe that will start the first iteration of the animation, during the period defined by animation-delay.

```css
.element
{
    animation-fill-mode: backwards;
}
```

The **both value** means that the animation will follow the rules for both forwards and backwards. That is, it will extend the animation properties in both directions.

```css
.element
{
    animation-fill-mode: both;
}
```

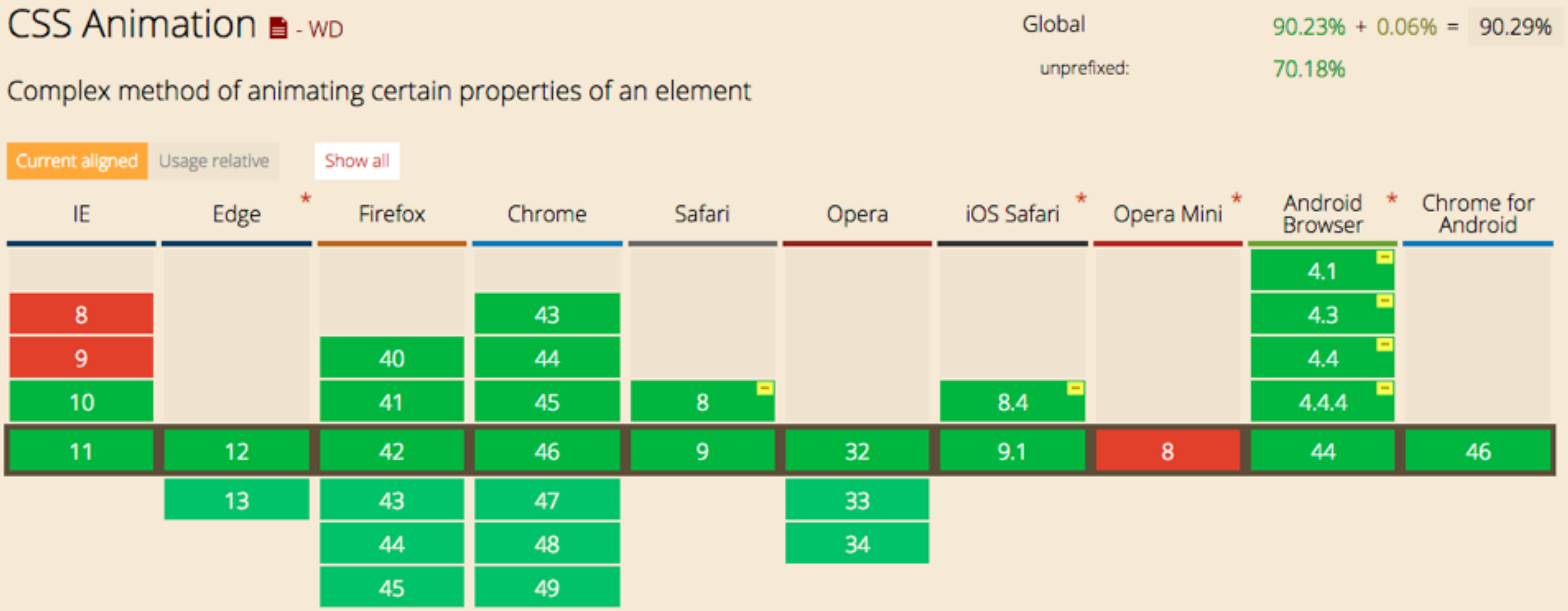
## animation shorthand

All of the sub-properties can be defined using a single **animation property**.

```css
.element
{
    animation:
        animation-name
        300ms
        ease
        1
        normal
        running
        1.5s
        none;
}
```

## browser support

CSS animations are supported by most modern browsers from IE10 and upwards.



CSS Animation - WD

Complex method of animating certain properties of an element

Global 90.23% + 0.06% = 90.29%

unprefixed: 70.18%

Current aligned | Usage relative | Show all

| IE | Edge * | Firefox | Chrome | Safari | Opera | iOS Safari * | Opera Mini * | Android Browser * | Chrome for Android |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 4.1 | |
| 8 | | | 43 | | | | | 4.3 | |
| 9 | | 40 | 44 | | | | | 4.4 | |
| 10 | | 41 | 45 | 8 | | 8.4 | | 4.4.4 | |
| 11 | 12 | 42 | 46 | 9 | 32 | 9.1 | 8 | 44 | 46 |
| | 13 | 43 | 47 | | 33 | | | | |
| | | 44 | 48 | | 34 | | | | |
| | | 45 | 49 | | | | | | |

Until recently, all @keyframes at-rules and animation properties had to be written twice - **once as “-webkit-” rules** and once as normal rules.

The @keyframes at-rules needed to be defined using **“@-webkit-keyframes”** before the normal @keyframes.

```
/* webkit keyframes defined first */
@-webkit-keyframes pulse
{
    from {left: 0px;}
    to {left: 200px;}
}

/* normal keyframes defined second */
@keyframes pulse
{
    from {left: 0px;}
    to {left: 200px;}
}
```

The animation property and all sub-properties also needed to be defined twice - with the **“-webkit-”** prefix and then without.

```css
.element
{
    -webkit-animation: pulse 5s infinite;
    animation: pulse 5s infinite;
}
```

However, many modern browsers now support @keyframes and animations **without the need for prefixes**.

**It’s up to you** whether you define animations with or without the -webkit- prefixes.

## Russ Weakley

Max Design

**Site:** maxdesign.com.au

**Twitter:** twitter.com/russmaxdesign

**Slideshare:** slideshare.net/maxdesign

**Linkedin:** linkedin.com/in/russweakley