Advanced CSS

Agenda

- What is an animation and how to add some to your webpage
- How to make sure that your animation works on every browser
- transition
- call to action
- hover
- duration
- timing
- delay
- vendor prefixes
- transform
- rotate
- skew

Goals

- Create richer web application experiences via CSS animations, transitions and more.

- We will examine:
  - Transitions
  - Transforms

Guiding our Users

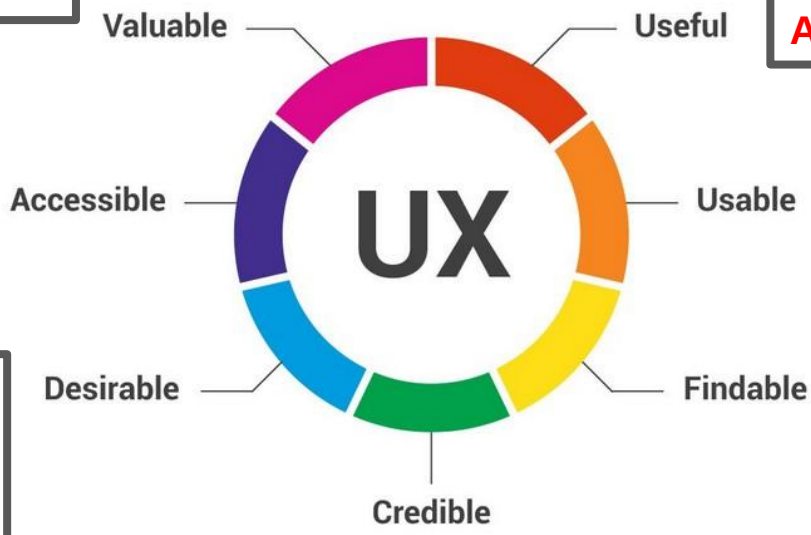
We want to have a good UX

UX = User Experience

Now it looks amazing!
vs
Ok, so the border is red…

Yes, I just need to click
vs
I tried to type but the screen kept scrolling

Yes, I want it!
vs
I don't need all these buttons



This filter option helps
vs
And now what…

Wow, you just drag it
vs
So how does it work?

Oh, here it is
vs
How could I know it is hiding in the sub menu

Guiding our Users

Often our site has a flow.

Some steps the user has to take in a specific order.

Do you have examples?

1. Filling a form.
2. Register
3. Create an album

We want to help the user complete his task!

Guiding our Users

We guide our users online

with the design tools we have at our disposal:

- Color

- Font

- ***Motion***

- And more…

Guiding our Users

When building a website, keep in mind that users should be guided.

Your site should be so intuitive, that even a drunk person could navigate through it easily.

Animations and transitions are often strong Call to Actions, that help guide the user through our app/site work flow.

Transitions 1

Transitions cause changes to a property and take place over a period of time.

Let's break it down

- What?
  - color, position, size…
- Time = the time it takes the change to take place.

- We may know it from creating our first PowerPoint presentations

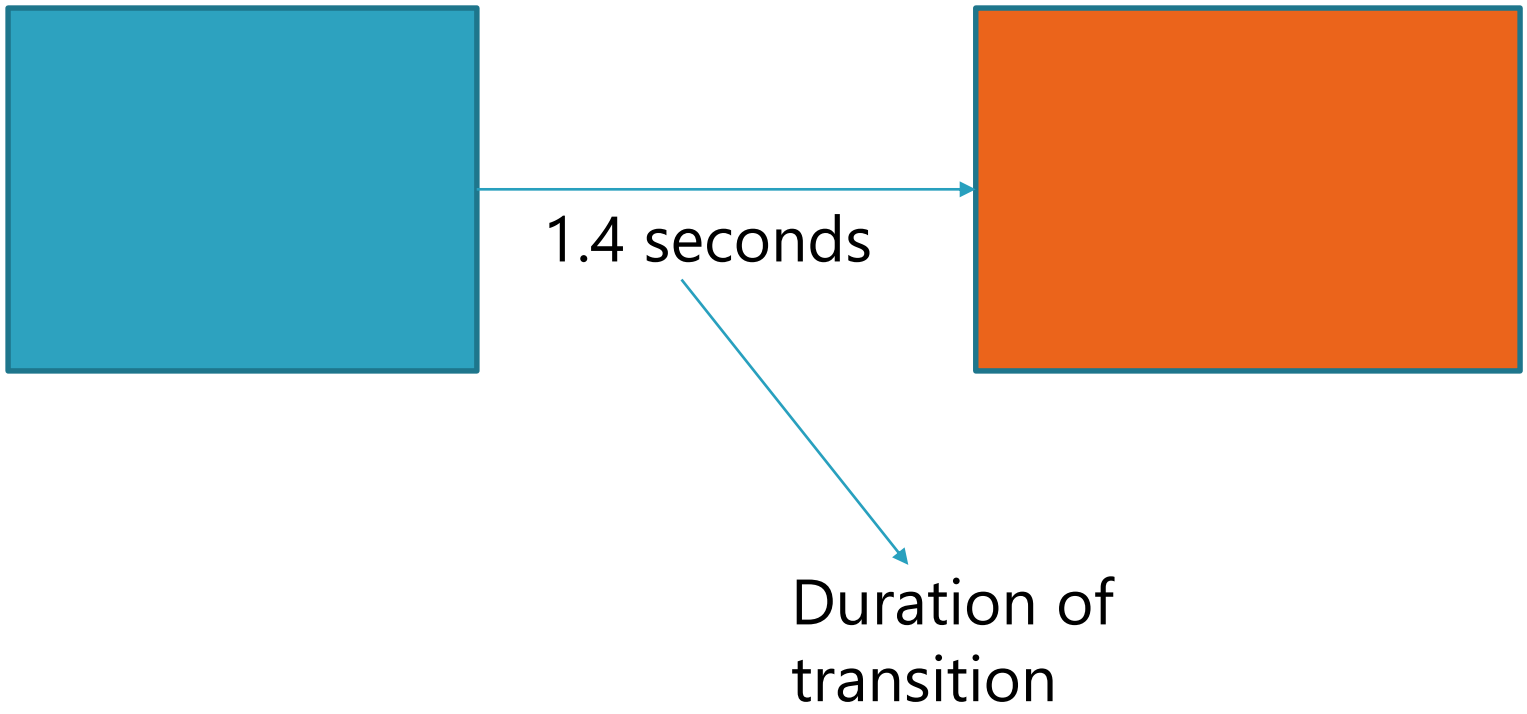
Transitions 1

Example:

We can create a transition on the color property of a .

The time it takes for the transitions from beginning until end, is referred to as ***duration***.

Transitions 1



Call to Action

A prominent call to action helps the user navigate in the site easily.

Example:
We want the user to purchase a product, sign up for a service, etc. The purchase button (for example) should be large and obvious.

Solution:
We can use CSS transitions to make purchase button more enticing.

Simple Example: hover

Our Task: when the user hovers over our button, the button should change it's color.

This is how it looks like

This is the markup:

```
<button class="box-hover">
    <h2>BUY!</h2>
</button>
```

This is the CSS:

```
.box-hover {
  width: 100px;
  height: 50px;
  background: bisque;
}

.box-hover:hover {
  background: #f44336;
  color: white;
}
```

Css Pseudo Class

Definition:
A CSS ***pseudo-class*** is a keyword added to a selector that specifies a special **state** of the selected element(s).

Example:
For example, :hover can be used to change a button's color when the user hovers over it.

Syntax:

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

Example:

```
.box:hover {
        background: coral;
}
```

When the element is in that state, the css rules will apply!

Css Pseudo Class

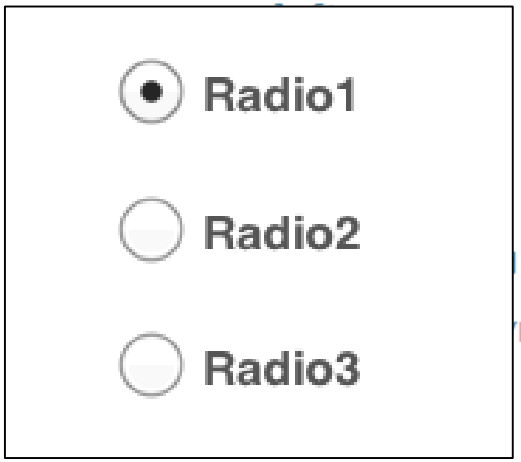
Index of standard pseudo-classes

:active
:any
:any-link
:checked
:default
:dir()
:disabled
:empty
:enabled
:first
:first-child
:first-of-type
:fullscreen
:focus

:last-of-type

:required
:right
:root
:scope
:target
:valid
:visited

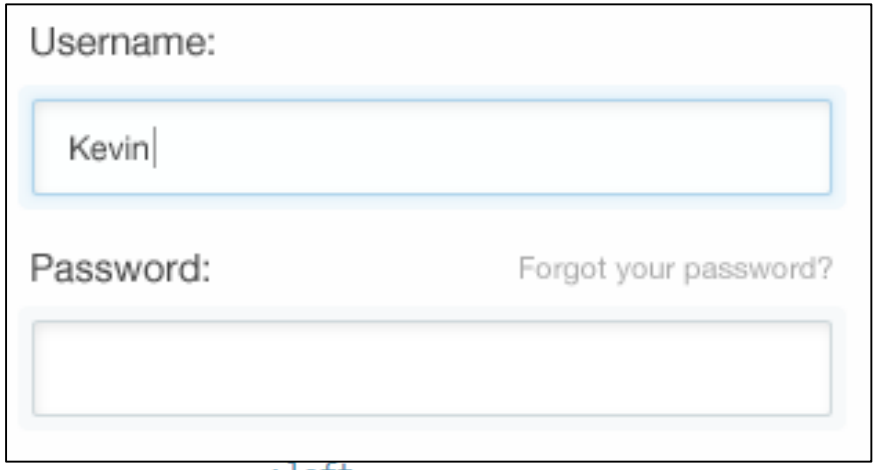
Represents the state of any radio (<input type="radio">), checkbox (<input type="checkbox">), or option (<option> in a <select>) element that is checked or toggled.



Css Pseudo Class

Index of standard pseudo-classes

:active
:any
:any-link
:checked
:default
:dir()
:disabled
:empty
:enabled
:first
:first-child
:first-of-type
:fullscreen
:focus



:left

Represents an element (such as a form input) that has received focus. It is generally triggered when the user clicks or taps on an element or selects it with the keyboard's "tab" key.

:only-child
:only-of-type
:optional
:out-of-range
:read-only
:read-write
:required
:right
:root
:scope
:target
:valid
:visited

Css Pseudo Class

Default State / initial state

```
.box {
    background: violet;
}
```

Hover State

```
.box:hover {
    background: coral;
}
```

Let's slow it down

But the change happened too fast.

We want it to be more gradual.

Jumping into the Code

a more gradual change

Jumping into the Code

The transition is added to the starting state.

Adding transition to starting state.

Duration

You can transition multiple comma-separated properties

Call to Action Tips

```
1  .btn {
2      background-color: #00A0D6;
3      color: #FFFFFF;
4      /*transition: background-color 0.4s, color 0.4s;*/
5      transition: all 0.4s;
6  }
7
8  .btn:hover {
9      background-color: #007DA7;
10     color: #E3E3E3;
11 }
12
```

Use the *all* keyword to transition every changing property.
Be careful though!
Any property that can animate, will!

All about Transitions

So far we used only 2 values for the transition:

- css-property (like color)
- duration.

But in fact transition has 4 properties we can specify.

All about Transitions

From W3C:

We have seen this kind of shorthand for border, padding, etc.

Border - Shorthand Property

As you can see from the examples above, there are m

To shorten the code, it is also possible to specify all th

The `border` property is a shorthand property for the

- `border-width`
- `border-style` (required)
- `border-color`

Example

```
p {
    border: 5px solid red;
}
```

Result:

The CSS3 transition properties can be specified one by one, like this:

Example

```
div {
    transition-property: width;
    transition-duration: 2s;
    transition-timing-function: linear;
    transition-delay: 1s;
}
```

Try it Yourself »

or by using the shorthand property `transition`:

Example

```
div {
    transition: width 2s linear 1s;
}
```

Try it Yourself »

All about Transitions

Which one do you prefer?

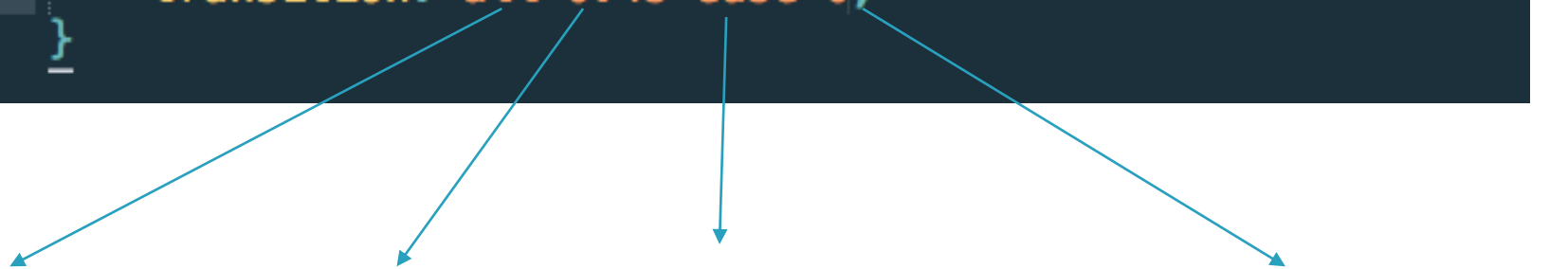
From MDN:

The shorthand CSS syntax is written as follows:

```
1 | div {
2 |     transition: <property> <duration> <timing-function> <delay>;
3 | }
```

All about Transitions

```
1  .btn {
2      background-color: #00A0D6;
3      color: #FFFFFF;
4      /*transition: background-color 0.4s, color 0.4s;*/
5      transition: all 0.4s ease 0;
6  }
7
```



Property

duration

Timing function
(defaults to ease)

Delay
(defaults to 0)

CSS Vendor Prefixes

CSS vendor prefixes, also sometime known as CSS browser prefixes, are a way for browser makers to add support for new CSS features before those features are fully supported in all browsers.

This may be done during a sort of testing and experimentation period where the browser manufacturer is determining exactly how these new CSS features will be implemented.

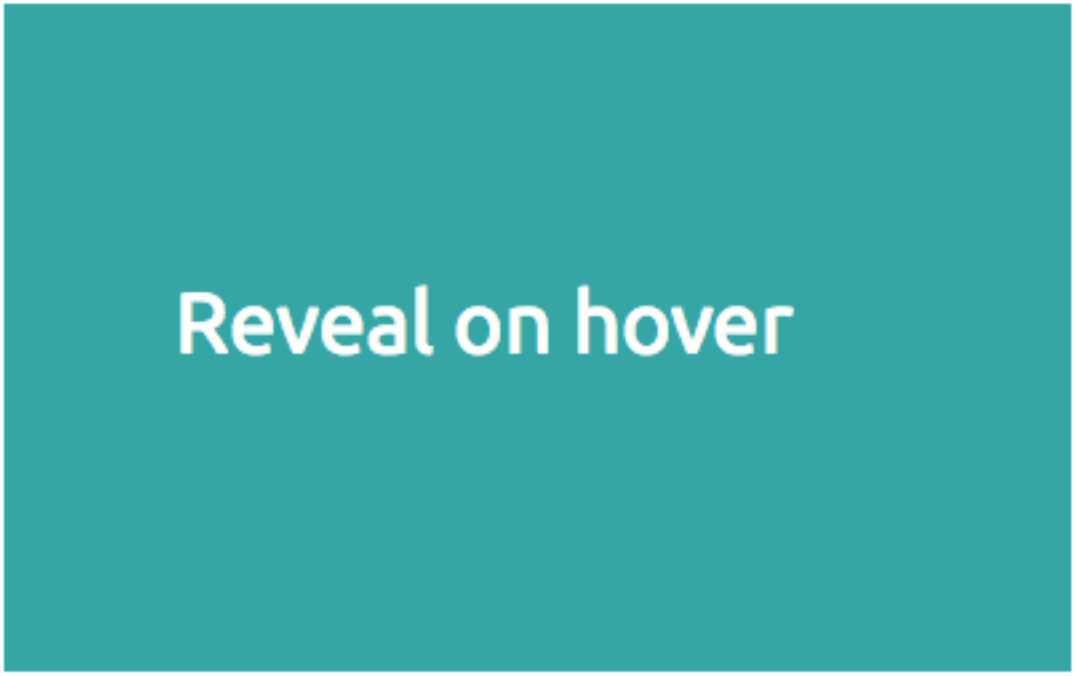
```
13  .btn {
14      -webkit-transition: background-color .4s;
15      -moz-transition: background-color .4s;
16      -ms-transition: background-color .4s;
17      -o-transition: background-color .4s;
18      transition: background-color .4s;
19  }
```

Transition in Hidden Content Example

Revealing hidden content onto the screen is another common use for transitions.

This adds personality and more information to user on actions like *hover*.

Transition in Hidden Content: Example



Let's reveal a description on hover

Reveal On Hover: Code

Positioning: Notice *a.box* and *.content* positioning:

```css
@import url(http://fonts.googleapis.com/css?family=Ubuntu);

a.box {
    color: #fff;
    background-color: #2f9b9b;
    display: block;
    width: 400px;
    height: 250px;
    overflow: hidden;
    position: relative;
    cursor: pointer;
    font-family: 'Ubuntu', sans-serif;
}

.content {
    position: absolute;
    bottom: 0;
    right: 0;
    top: 0;
    margin: auto;
    padding: 0 4em;
    z-index: 2;
    height: 3em;
    transition: all .2s ease;
}
```

```html
<a class="box">
    <div class="content">
        <h2>
            Reveal on hover
        </h2>
        <p class="description">
            Some description text here of the title that displays on hover
        </p>

    </div>
</a>
```

Because the *a.box* (the parent div) is positioned relative, and *.content* is positioned absolute, .content will be moved relative to parent

Questions?

Looking at our row of buttons

Let’s see another example: link

Looking at our row of buttons

Let's take a look at the CSS Rules of our buttons:

```
button {
  position: relative;
  height: 45px;
  width: 150px;
  margin: 10px 7px;
  padding: 5px 5px;
  font-weight: 700;
  font-size: 15px;
  letter-spacing: 2px;
  color: #383736;
  border: 2px #383736 solid;
  border-radius: 4px;
  text-transform: uppercase;
  outline: 0;
  overflow:hidden;
  background: none;
  z-index: 1;
  cursor: pointer;
  transition:         0.08s ease-in;
  -o-transition:      0.08s ease-in;
  -ms-transition:     0.08s ease-in;
  -moz-transition:    0.08s ease-in;
  -webkit-transition: 0.08s ease-in;
}
```

Notice the transition property:

All transitions will ease in over 0.08 seconds.

More hover and CSS3 transformations


```css
.highlight:hover {
  background-color: #FFffff;
  color: #3498db;
}
```

```
8
9   .fade:hover {
0     border: 0px;
1     color: #009999;
2     opacity: 0;
3   }
4
```

```css
.boom:hover {
  color: #009999;
  border: #009999;
  opacity: 0;
  -webkit-transform: scale(2, 2);
  -moz-transform: scale(2, 2);
  transform: scale(2, 2);
}
```

Transform: scale(x,y)

The element will increase or decrease *x* times its original width, and y times its original height.

More hover and CSS3 transformations


```
.rotate:hover {
  -webkit-transform: rotate(90deg);
  -moz-transform: rotate(90deg);
  transform: rotate(90deg);
}
```

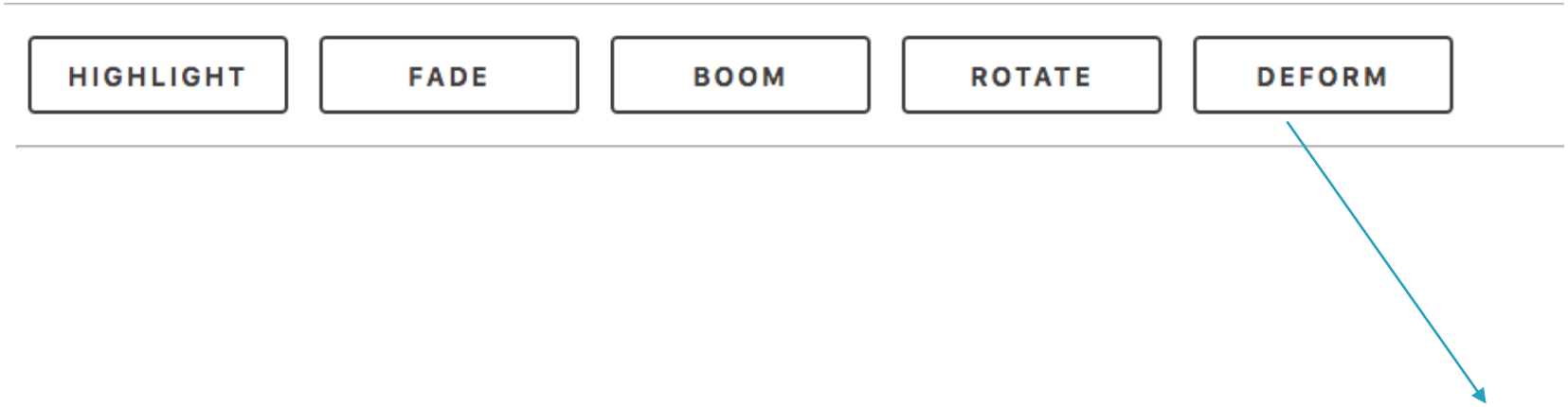
transform: rotate()

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

- How will we specify counter clockwise rotation?

Using negative values will rotate the element counter-clockwise.

More hover and CSS3 transformations




```
.deform:hover {
  -webkit-transform: skew(45deg, 45deg);
  -moz-transform: skew(45deg, 45deg);
  transform: skew(45deg, 45deg);
}
```

transform: skew()

- The skew() method skews an element out of view along the X and Y-axis by the given angles.

Positioning elements with transitions

- We saw:
  - Color change
  - Opacity change
  - Transform methods

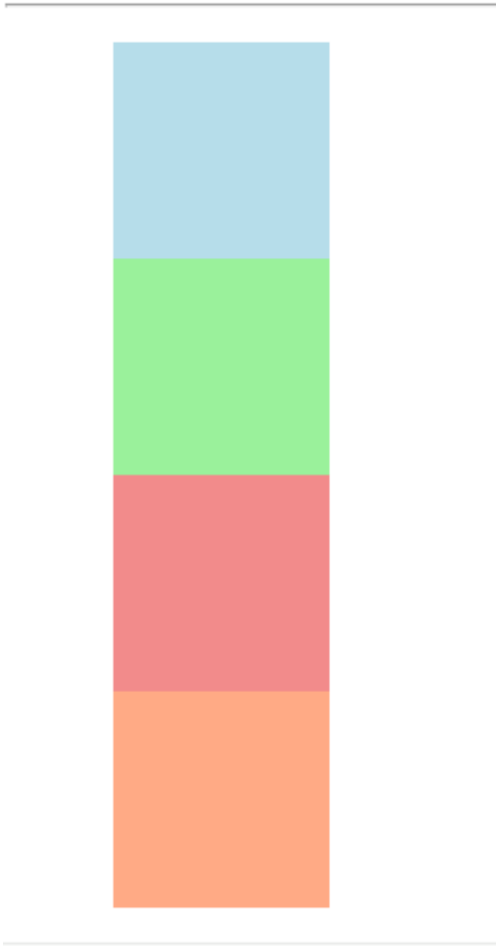
- Let's see some **position** change via transitions.

Positioning elements with transitions

Let's see an example of a funky menu that changes when hovered on.

Our funky menu

Now let's see it in action
in the browser



Our funky menu

- Let's examine the code and break it down ☺
- First, the markup (nothing special here…)

```
<ul class='menu'>
  <li>
    <div class='block blue'>
    </div>
    <div class='text'>
      YOLO
    </div>
  </li>
    <li>
    <div class='block green'>
    </div>
    <div class='text'>
      CARPE DIEM
    </div>
  </li>
   <li>
    <div class='block red'>
    </div>
    <div class='text'>
      HIPSTER TEXT HERE
    </div>
  </li>
  <li>
    <div class='block orange'>
    </div>
    <div class='text'>
      STUFF
    </div>
  </li>
</ul>
```

Our funky menu: Styling

- CSS

```css
.menu{
  list-style:none;
}

.menu li{
  position:relative;
  height:100px;
}

.blue{
  background:lightblue;
}

.red{
background:lightcoral;
}

.green{
background:lightgreen;
}

.orange{
background:lightsalmon;
}
```

Getting ready to use position absolute…!

Our funky menu: Styling

```
<ul class='menu'>
  <li>
    <div class='block blue'>
    </div>
    <div class='text'>
      YOLO
    </div>
  </li>
```

Remember *.menu li* is positioned relative.

```
.block{
  width:100px;
  height:100px;
  left:10px;
  position:absolute;
  -webkit-transition-duration: 0.3s;
  -moz-transition-duration: 0.3s;
  -o-transition-duration: 0.3s;
  transition-duration: 0.3s;
}

.text{
  -webkit-transition-duration: 0.3s;
  -moz-transition-duration: 0.3s;
  -o-transition-duration: 0.3s;
  transition-duration: 0.3s;
  height:height:100px;
  position:relative;
  padding:30px;
  font-family: sans-serif;
  color: #221F1F;
  text-shadow: 1px 1px 1px lightgray;
  font-size: 30px;
  left:40px;
  opacity:0;
}
```

On default we are hiding text.

Our funky menu: Styling

```
<ul class='menu'>
  <li>
    <div class='block blue'>
    </div>
    <div class='text'>
      YOLO
    </div>
  </li>
```

```
.menu li:hover .block{
  left:-10px;
}

.menu li:hover .text{
  left:65px;
  opacity:1;
}
```

On hover we are pushing the block 10px, to reveal the text.

On hover, we are revealing the text (set opacity:1).

Common features

- So far we have only focused on CSS3 built in animations and transformation capabilities.
- Many popular libraries leveraging JS exist.
- The most popular standalone CSS library for animations is animate.css
- Let's see some simple examples here

Bonus for later use: animate.css

Download the CSS and use it locally

```
<head>
    <link rel="stylesheet" href="animate.min.css">
</head>
```

Or use it with CDN:

```
<head>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/3.7.0/animate.min.css">
</head>
```

Example usage:

```
<h1 class="animated infinite bounce delay-2s">Example</h1>
```

Read more about it here

See some examples here

One Last Thing

Animations are cool!

(when they are gently applied)

Don’t abuse them!

Questions?

Summary

- You needed to understand:
  - Native CSS3 transformations and animations
  - How to use native CSS3 functionality to liven up our sites / apps, and guide your users.
- You need to remember:
  - The option to use CSS transitions.
- You need to be able to do:
  - Create CSS transitions
  - Use a CSS animations library.
- Further down the line:
  - When we learn DOM manipulation and JS, we will be exposed to many more interesting ways to make our pages more interactive

Cheat Sheet

css pseudo classes

- Syntax

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

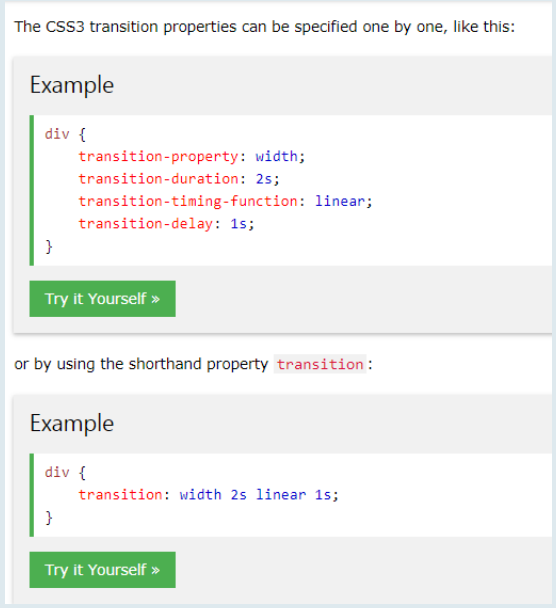
- Example

```
.box:hover {
        background: coral;
}
```

Transition

```
button {
  opacity: 0; /* initial value */
  /* add transition to the element */
  transition: opacity 1s ease-in 0.5s;
}

button:hover {
  opacity: 1; /* value at the end of change */
}
```


The CSS3 transition properties can be specified one by one, like this:

Example

```
div {
    transition-property: width;
    transition-duration: 2s;
    transition-timing-function: linear;
    transition-delay: 1s;
}
```

Try it Yourself »

or by using the shorthand property `transition`:

Example

```
div {
    transition: width 2s linear 1s;
}
```

Try it Yourself »


```
.fade:hover {
  border: 0px;
  color: #009999;
  opacity: 0;
}
```


```
.rotate:hover {
  -webkit-transform: rotate(90deg);
  -moz-transform: rotate(90deg);
  transform: rotate(90deg);
}
```


```
.boom:hover {
  color: #009999;
  border: #009999;
  opacity: 0;
  -webkit-transform: scale(2, 2);
  -moz-transform: scale(2, 2);
  transform: scale(2, 2);
}
```


```
.highlight:hover {
  background-color: #FFffff;
  color: #3498db;
}
```