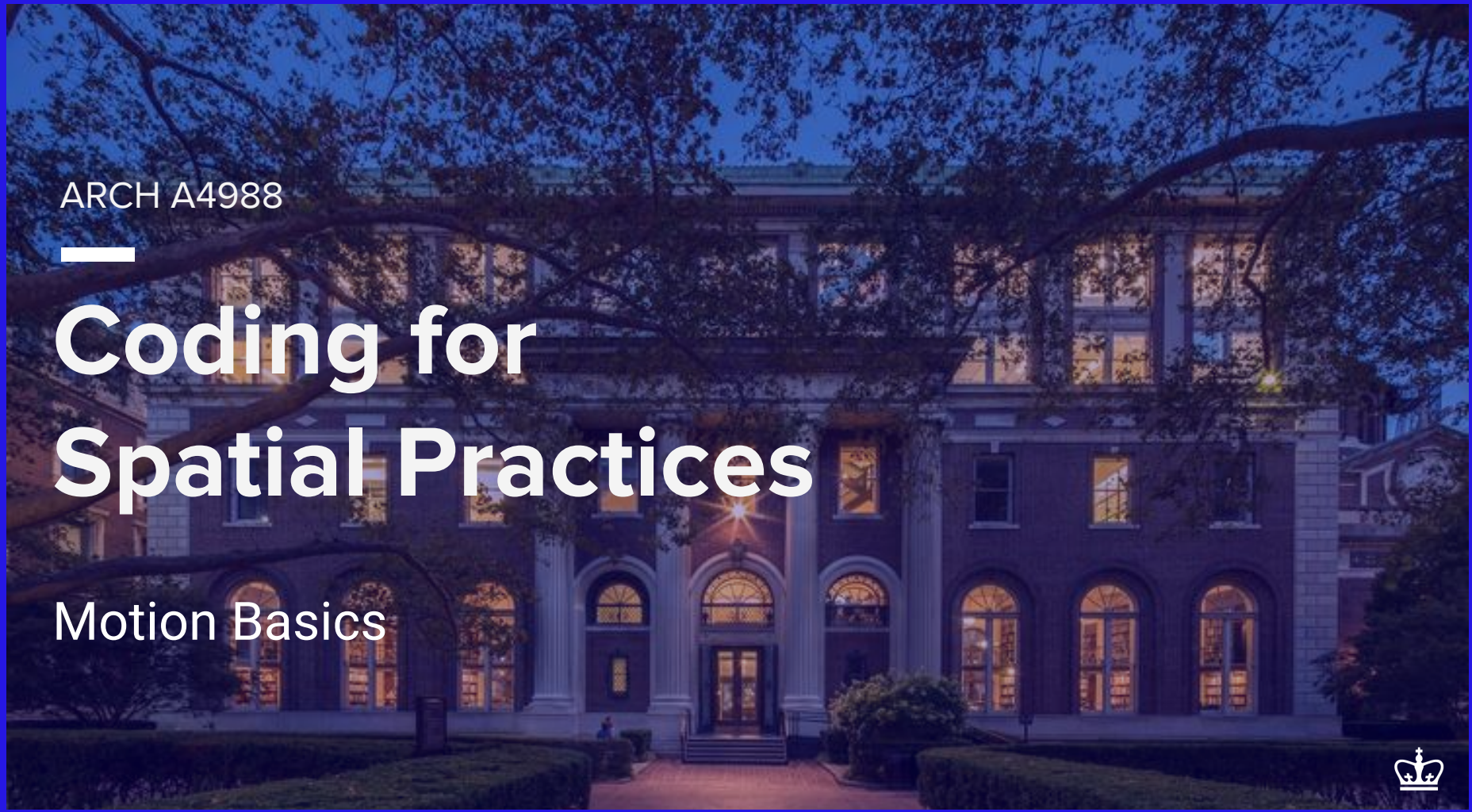


ARCH A4988

Coding for Spatial Practices

Motion Basics

Start Recording

Learning Objectives

Giving your website some motion

1. Use properties such as transition and transform to change element properties
2. Describe the purpose and syntax of css @keyframes
3. List and describe the purpose of the animation properties.

Agenda

Review

Motion Basics

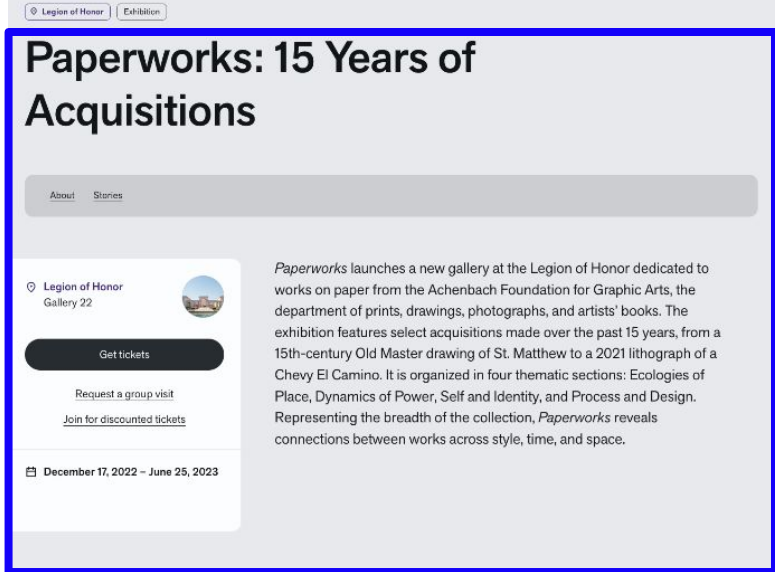
Lab Time

Review

Landing Page

Project 01

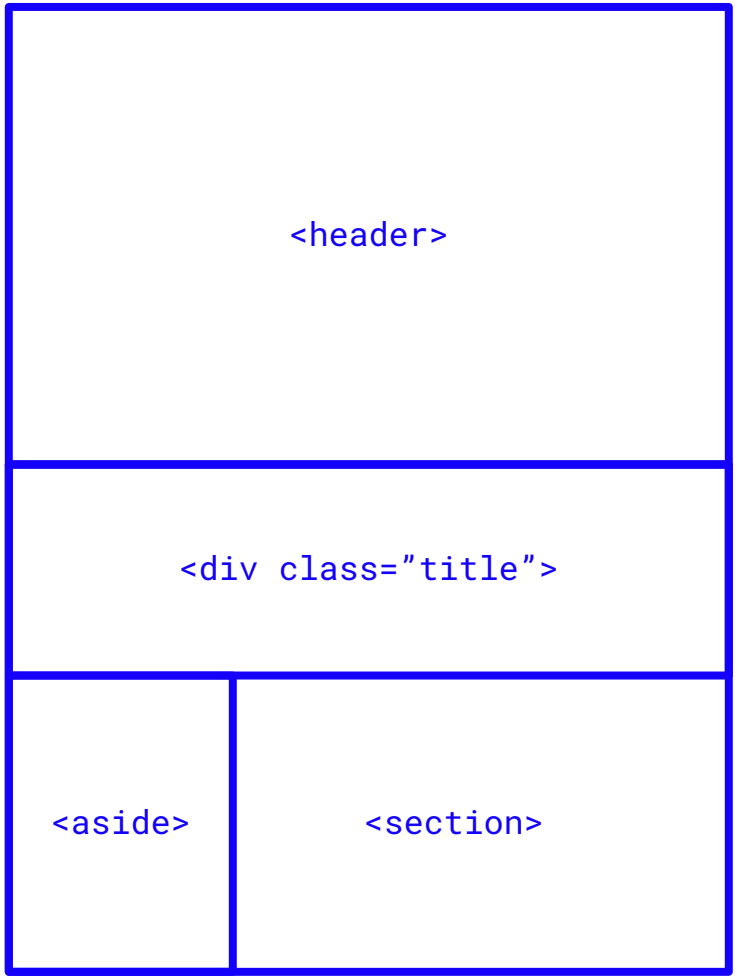
1. Select the page you want to replicate.
2. Create a wireframe for the layout
3. Drop in the content (images, text etc)



Project 01

1. Select the page you want to replicate.
2. Create a wireframe for the layout
3. Drop in the content (images, text etc)

```
<div class="flex-container">
  <header></header>
  <div class=”title”></div>
  <main>
    <aside></aside>
    <section></section>
  </main>
</div>
```



Steps to achieve a multi-column layout

1. MAKE SURE ALL THE CONTENT FOR EACH COLUMN HAS ONE WRAPPER AROUND IT IN YOUR HTML
2. WRAP THE COLUMNS WITH A PARENT CONTAINER, THE "FLEX CONTAINER"
3. ADD DISPLAY: FLEX; TO FLEX-CONTAINER
4. GIVE EACH COLUMN A WIDTH IN PERCENTAGES
5. OPTIONAL: USE JUSTIFY-CONTENT ON FLEX CONTAINER TO DETERMINE COLUMN SPACING

Class Exercise

In the past, designers and developers had to create separate sites for desktop, tablet and mobile so they would have a different version for each of these. Then, as more devices came out, with different sized screens there was a need to shift the thinking away from screen size to thinking about a design that looks good regardless of the device you happen to be using.

For this exercise, we're going to layout Lina Bo Bardi's Houses or Museums? using a mobile-first approach. When designing for mobile first, since it has the most limitations (1 > screen size, 2 > bandwidth), it forces you to prioritize content. So the mobile-first approach is sometimes considered a content-first approach.

Class Exercise

Using media-queries, your website should resize intentionally (see Step 02 for breakpoints). Finally, design one of the breakpoints to change the appearance of your web page when it's printed on a paper. For example, you can specify one font for the screen version and another for the print version.

Step 1: Use Lina Bo Bardi’s Houses or Museums? essay.

Class Exercise

Houses Or Museums?

Lina Bo Bardi, 1958

What should come first, houses or museums?

Everything at once: the houses, the schools, the museums, the libraries. Urban Planning cannot ignore cultural issues. If in the construction of new neighbourhoods, new housing forms the basis of the city plan (and by housing we also mean the market, the schools and the public services like the hospital and the post offices), the planning of a city cannot overlook two key public buildings that still today are considered an intellectual luxury: the Museum and the Library.

Museum? What is a museum?

In everyday life, when we want to describe a person, thing or idea that is outdated, not practical or useful, we often say 'they belong in a museum'. The expression is a clear indicator of the place museums occupy in

Class Exercise

Step 2: Adapt the text to **three** of the following 5 media queries:

- @media screen and (max-width: 1024px)
- @media screen and (max-width: 768px)
- @media screen and (max-width: 600px)
- @media screen and (max-width: 480px)
- @media screen and (max-width: 320px) *to view this width, use mobile mode in the Developer Tools

Class Exercise

Step 3: Properties you may consider changing:

- display: display, z-index
- position: top, bottom, left, right
- scale: width, height, max-width, max-height
- style: background, border, box-shadow, opacity, filter, color
- font: font-size, line-height, font-family
- padding: top, bottom, left, right
- others: content, mix-blend-mode

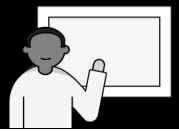
2D Transforms

Animation Basics

2D Transforms

Transforms are static — immediately changing the targeted element and causing it to stay that way. There are four main types of transformations:

- rotate,
- translate,
- scale and
- skew.



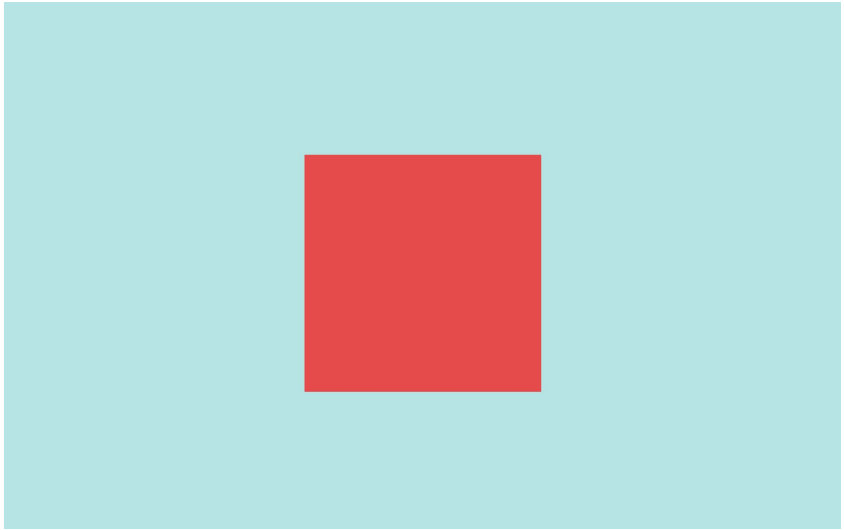
Guided Walk-Through:

2D Transforms

Let’s apply the four main types of transformations to this square: rotate, translate, scale and skew.

```
<div class="square"></div>
```

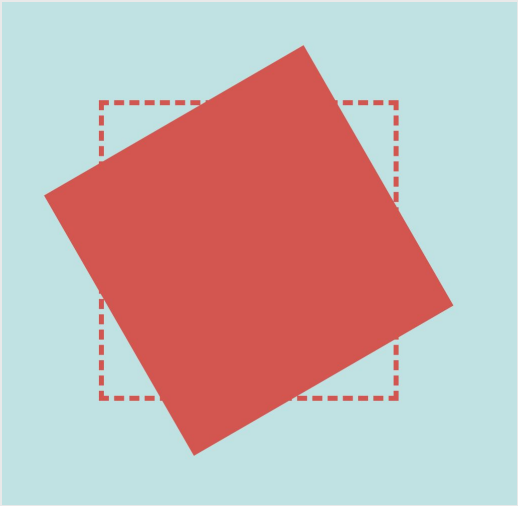
```
.square {
  width: 150px;
  height: 150px;
  background: tomato;
}
```



Animation Basics

2D Transforms + Rotate

The rotate value provides the ability to rotate an element from 0 to 360 degrees. Using a positive value will rotate an element clockwise, and using a negative value will rotate the element counterclockwise.

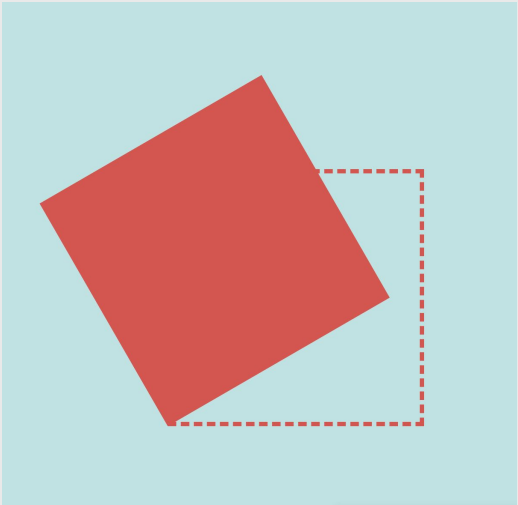


```
.square {
  transform: rotate(-30deg);
}
```

Animation Basics

2D Transforms + Rotate

The rotate value provides the ability to rotate an element from 0 to 360 degrees. Using a positive value will rotate an element clockwise, and using a negative value will rotate the element counterclockwise.

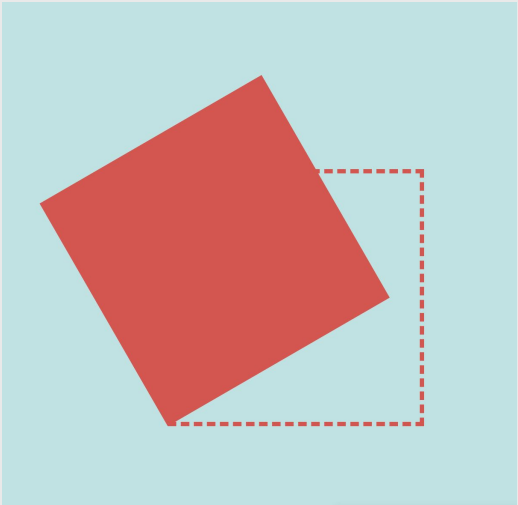


```
.square{
  transform: rotate(-30deg);
  transform-origin: bottom left;
}
```

Animation Basics

2D Transforms + Rotate

The rotate value provides the ability to rotate an element from 0 to 360 degrees. Using a positive value will rotate an element clockwise, and using a negative value will rotate the element counterclockwise.

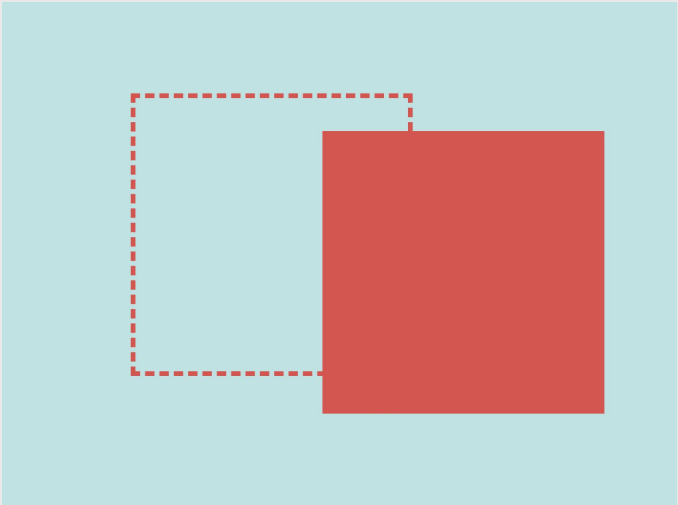


```
.square{
  transform: rotate(-30deg);
  transform-origin: 0 100%;
}
```

Animation Basics

2D Transforms + Translate

The translate value works a bit like that of relative positioning, pushing and pulling an element in different directions.

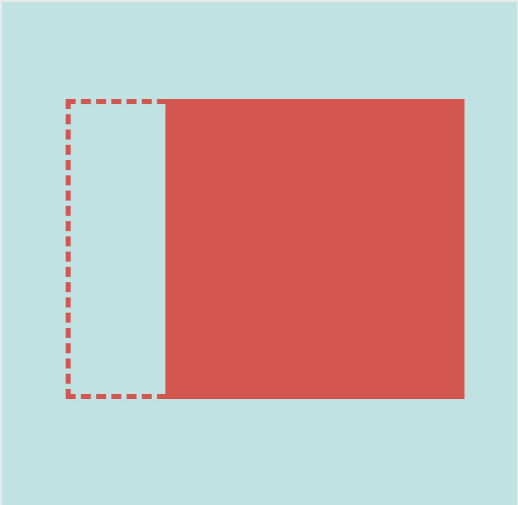


```
.square{
 transform: translate(40px 20px);
}
```

Animation Basics

2D Transforms + TranslateX

Using the translateX value will change the position of an element on the horizontal axis.

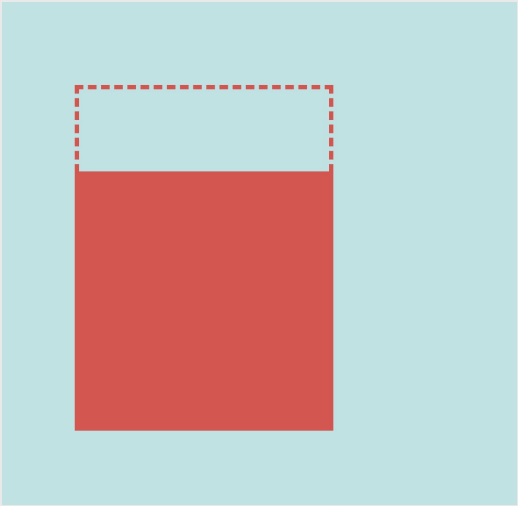


```
.square{
 transform: translateX(100px);
}
```

Animation Basics

2D Transforms + TranslateY

Using the translateY value will change the position of an element on the vertical axis.

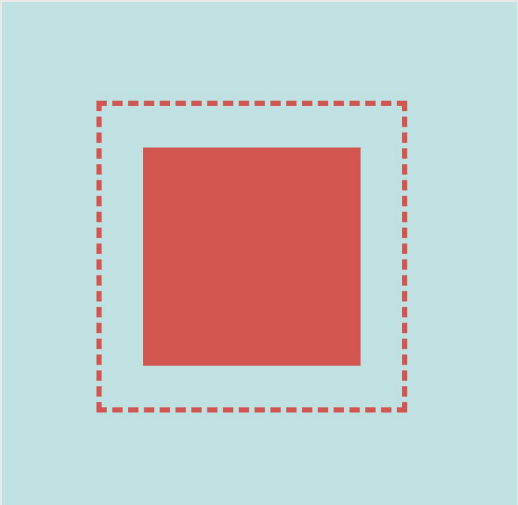


```
.square{
 transform: translateY(100px);
}
```

Animation Basics

2D Transforms > Scale

Using the scale value within the transform property allows you to change the appeared size of an element. It's also possible to scale only the height or width of an element using the scaleX and scaleY values.

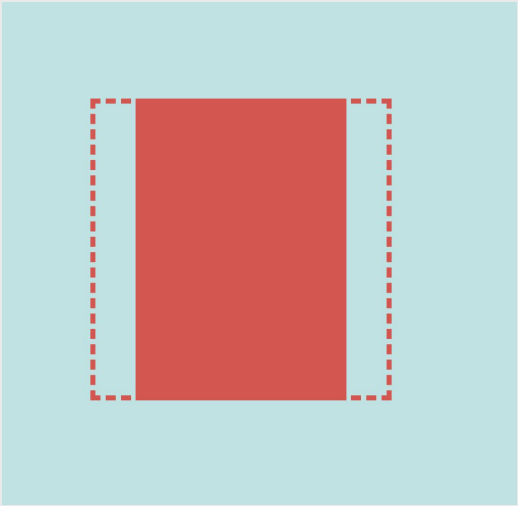


```
.square{
 transform: scale(0.7);
}
```

Animation Basics

2D Transforms > ScaleX

Using the scale value within the transform property allows you to change the appeared size of an element. It's also possible to scale only the height or width of an element using the scaleX and scaleY values.

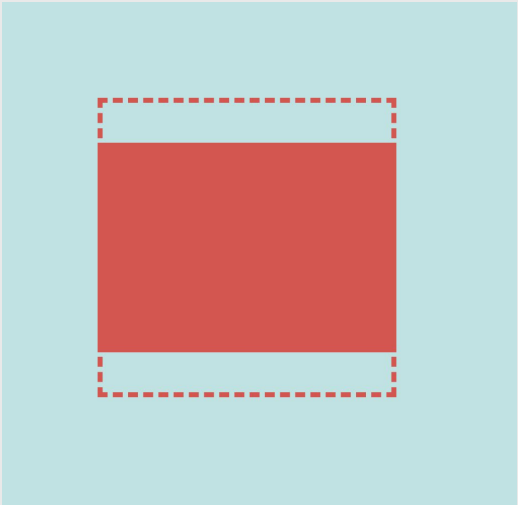


```
.square{
 transform: scaleX(0.7);
}
```

Animation Basics

2D Transforms > ScaleY

Using the scale value within the transform property allows you to change the appeared size of an element. It's also possible to scale only the height or width of an element using the scaleX and scaleY values.



```
.square{
 transform: scaleY(0.7);
}
```

Animation Basics

2D Transforms > Skew

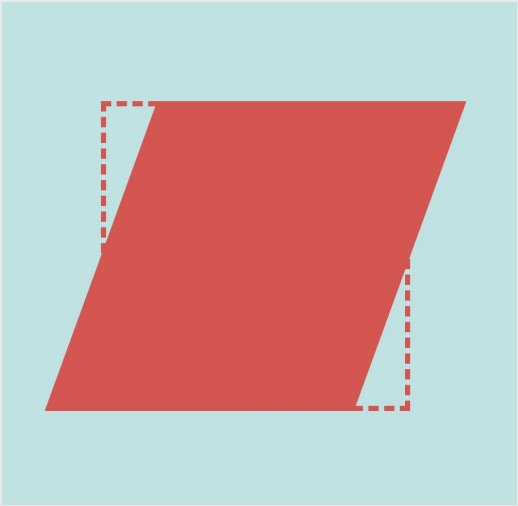
Skew, is used to distort elements on the horizontal axis, vertical axis, or both. To distort an element on both axes the skew value is used, declaring the x axis value first, followed by a comma, and then the y axis value.

```
.square{
 transform: skew(15deg, 15deg);
}
```

Animation Basics

2D Transforms > Skew

Skew, is used to distort elements on the horizontal axis, vertical axis, or both. To distort an element on both axes the skew value is used, declaring the x axis value first, followed by a comma, and then the y axis value.

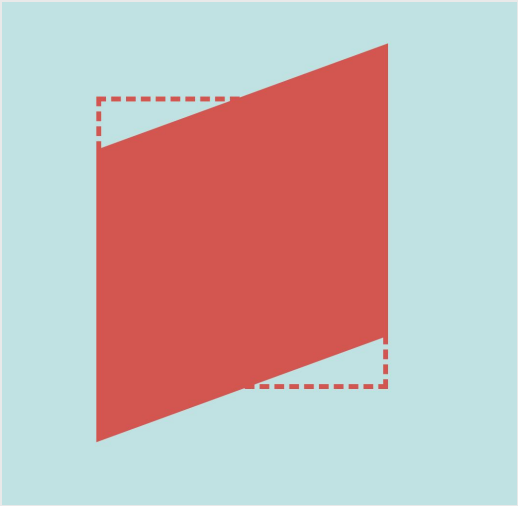


```
.square{
 transform: skewX(-20deg);
}
```

Animation Basics

2D Transforms > SkewY

Skew, is used to distort elements on the horizontal axis, vertical axis, or both. To distort an element on both axes the skew value is used, declaring the x axis value first, followed by a comma, and then the y axis value.



```
.square{
 transform: skewY(-20deg);
}
```

Animation Basics

Combining Transforms

- Separate each property with a space.
- Transforms are applied in the order listed.

```
.square{
 transform: scale(0.7, 1.5) rotate(30deg)
skewY(15deg) translate(200px, 20%);
}
```

Transitions

Animation Basics

Transitions

Alter the appearance and behavior of an element over a specified duration.

There are two ways to trigger CSS transitions:

- Using the **:hover** CSS pseudo-class
- Adding a class with JavaScript

```
.button {
  color: white;
  background-color: rgb(28, 40, 126);
  transition-property: background-color;
  transition-duration: 0.3s;
  transition-timing-function: ease-in;
  transition-delay: 0.2s;
}

.button:hover{
  background-color: tomato;
}
```

Animation Basics

Transitions > Timing Units

Seconds (s) or milliseconds (ms)

```
.button {
  color: white;
  background-color: rgb(28, 40, 126);
  transition-property: background-color;
  transition-duration: 0.3s;
  transition-timing-function: ease-in;
  transition-delay: 0.2s;
}

.button:hover{
  background-color: tomato;
}
```

Animation Basics

Transitions > Timing Units

Seconds (s) or milliseconds (ms)

```
.button {
  color: white;
  background-color: rgb(28, 40, 126);
  transition-property: background-color;
  transition-duration: 0.3s;
  transition-timing-function: ease-in;
  transition-delay: 0.2s;
}

.button:hover{
  background-color: tomato;
}
```

Animation Basics

Transitions

Common Timing Functions

linear
ease
ease-in
ease-in-out
ease-out
cubic

```
.button {
  color: white;
  background-color: rgb(28, 40, 126);
  transition-property: background-color;
  transition-duration: 0.3s;
  transition-timing-function: ease-in;
  transition-delay: 0.2s;
}

.button:hover{
  background-color: tomato;
}
```

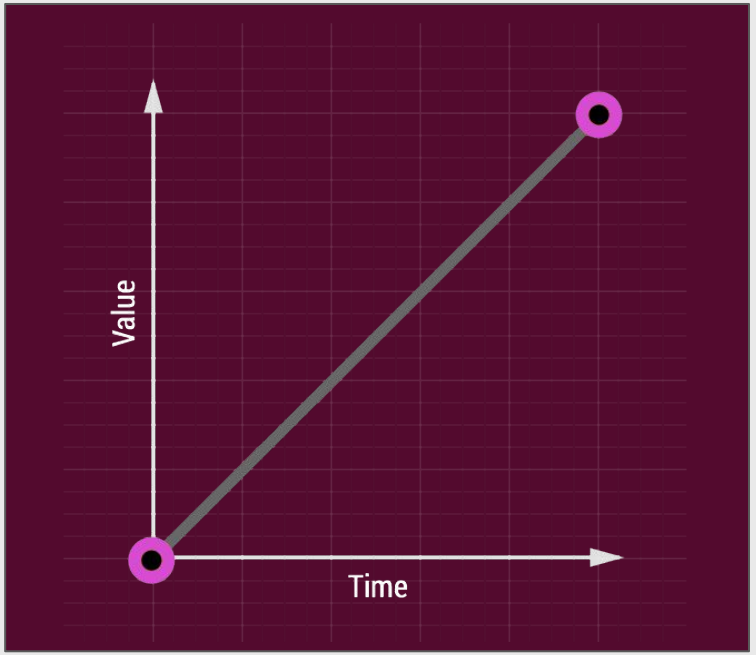
Animation Basics

Transitions

Common Timing Functions

`linear`
`ease`
`ease-in`
`ease-in-out`
`ease-out`
`cubic`

```
transition-timing-function: linear;
```



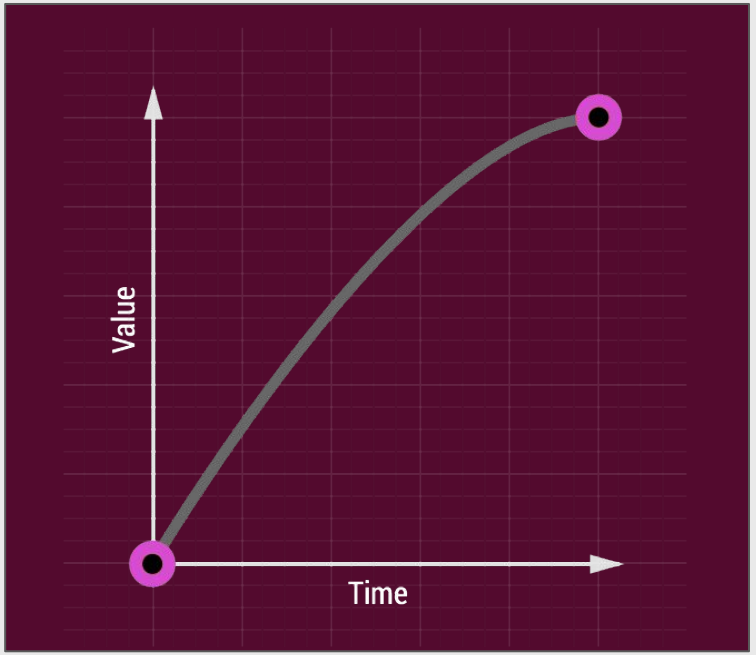
Animation Basics

Transitions

Common Timing Functions

linear
ease
ease-in
ease-in-out
ease-out
cubic

```
transition-timing-function: ease-out;
```



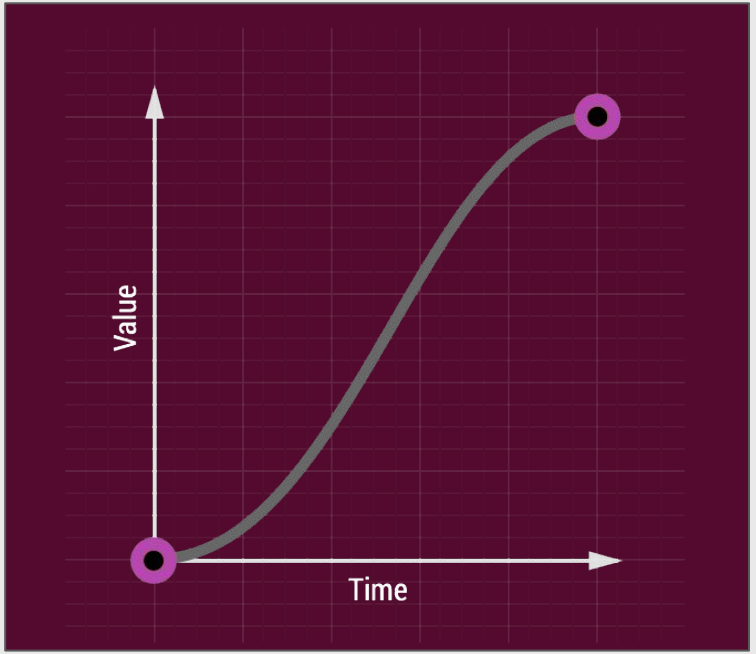
Animation Basics

Transitions

Common Timing Functions

linear
ease
ease-in
ease-in-out
ease-out
cubic

```
transition-timing-function: ease-in-out;
```



Animation Basics

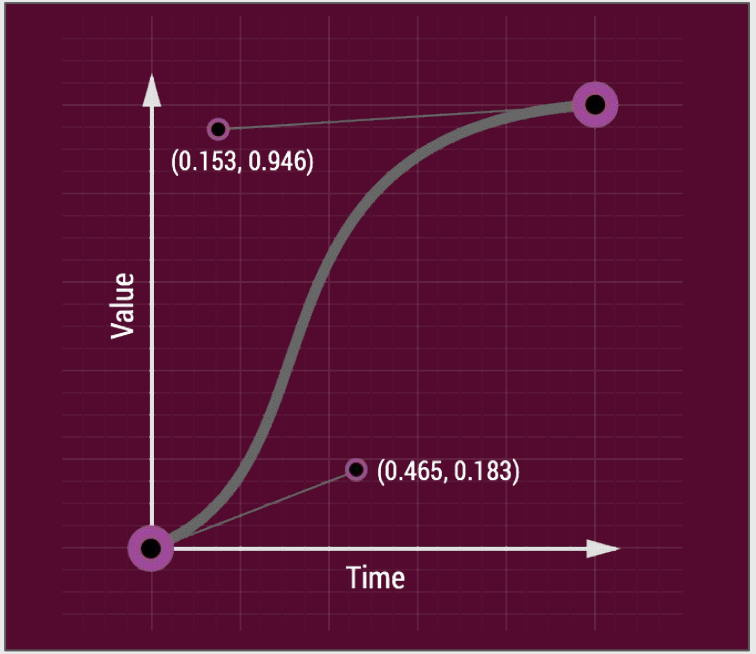
Transitions

Common Timing Functions

linear
ease
ease-in
ease-in-out
ease-out
cubic

Generate your own here.

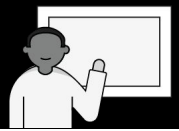
```
transition-timing-function:
cubic-bezier(0.465, 0.183, 0.153, 0.946);
```



Animation Basics

Shorthand

```
.button {
 color: white;
 background-color: tomato;
 transition: background-color 0.3s ease-in 0.2s;
}
```



Guided Walk-Through:

Transitions

Let's trigger a CSS transition using the **:hover** pseudo-class.

```
<div class="button">Button</div>
```

```
.button {
  width: 200px;
  height: 80px;
  background-color: red;
  text-align: center;
  line-height: 80px;
}
```

Animations

Animation Basics

Why Use Motion?

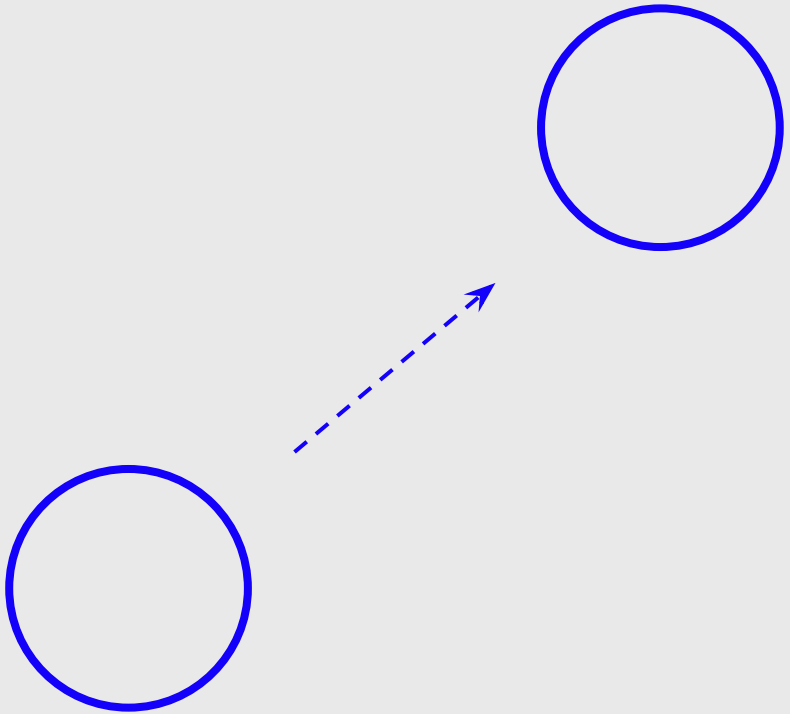
Motion for Feedback
Motion to Communicate State Change
Motion for Spatial Metaphors and Navigation
Motion as a Signifier
Attention Grabbing and Attention Hijacking

Animation Basics

Building Blocks of Animations:

CSS animations are made up of two basic building blocks:

- @keyframes
- animation properties



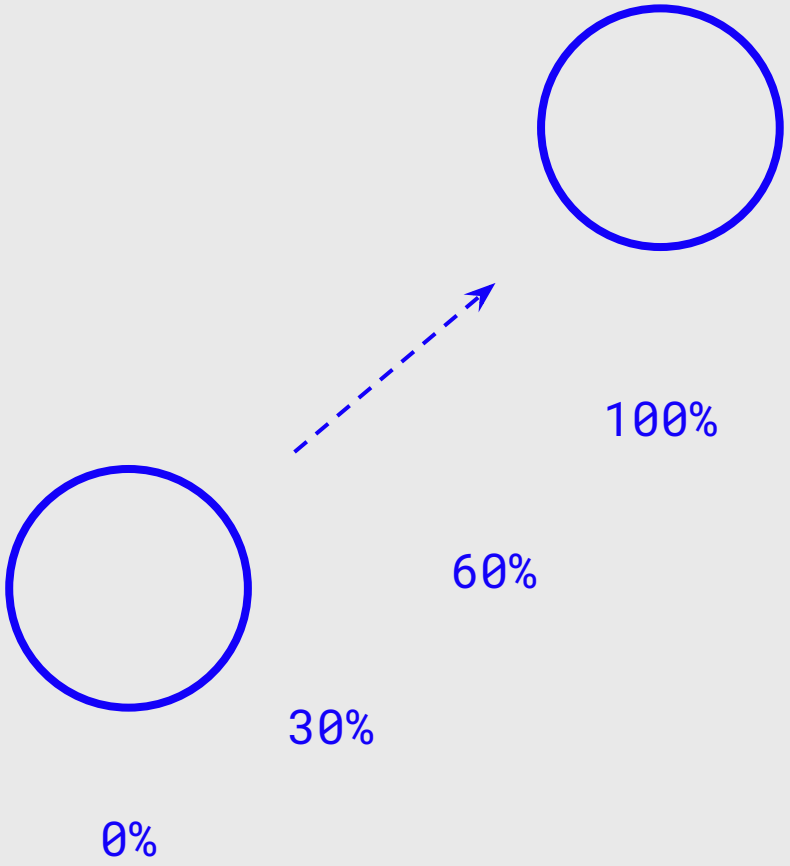
Animation Basics

Building Blocks of Animations:

CSS animations are made up of two basic building blocks:

- @keyframes
- animation properties

define what the animation looks like at each stage of the animation.



Animation Basics

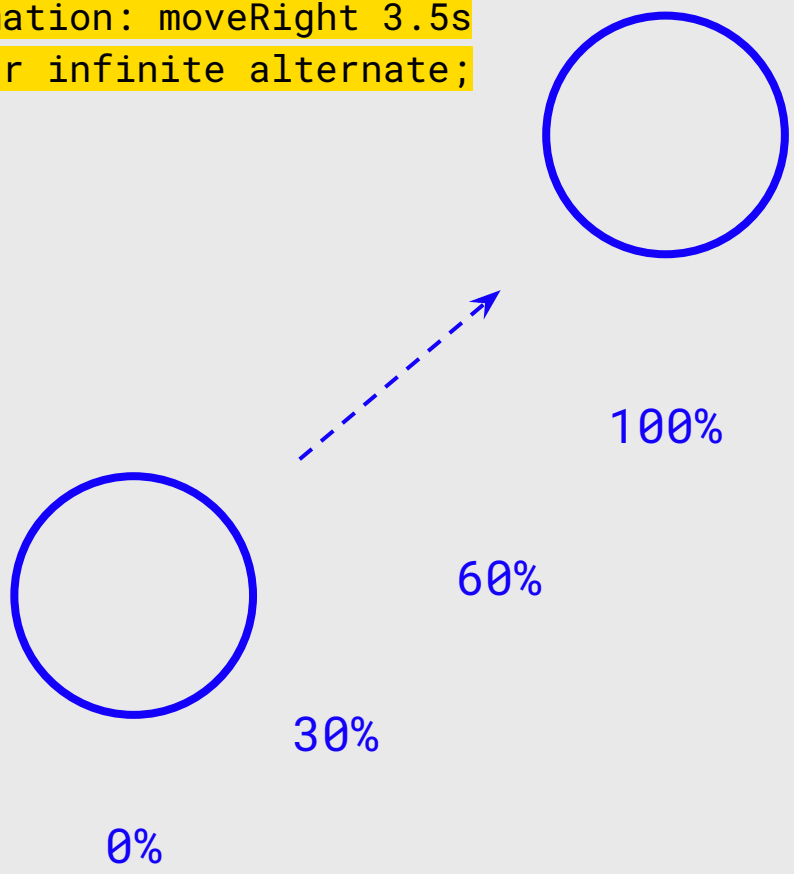
Building Blocks of Animations:

CSS animations are made up of two basic building blocks:

- @keyframes
- animation properties

assign the @keyframes to a specific CSS element and define how it is animated.

```
.circle {
 animation: moveRight 3.5s
linear infinite alternate;
}
```



Animations

Animation Properties

Each @keyframes is composed of:

- name of the animation
- stages of the animation
- CSS properties at each stage

```
@keyframes colors {
  0% {
     background-color: blue;
  }
  50% {
     background-color: yellow;
     color: rgba(200,155,20,0.8);
  }
  100% {
     background-color: green;
  }
}
```

Animations

Animation Properties

Each @keyframes is composed of:

- name of the animation
- stages of the animation
- CSS properties at each stage

```
@keyframes colors {
  0% {
     background-color: blue;
  }
  50% {
     background-color: yellow;
     color: rgba(200,155,20,0.8);
  }
  100% {
     background-color: green;
  }
}
```

Animations

Animation Properties

Each @keyframes is composed of:

- name of the animation
- stages of the animation
- CSS properties at each stage

```
@keyframes colors {
  0% {
    background-color: blue;
  }
  50% {
    background-color: yellow;
    color: rgba(200,155,20,0.8);
  }
  100% {
    background-color: green;
  }
}
```

Animations

Animation Properties

Each @keyframes is composed of:

- name of the animation
- stages of the animation
- CSS properties at each stage

```
@keyframes colors {
  0% {
     background-color: blue;
  }
  50% {
     background-color: yellow;
     color: rgba(200,155,20,0.8);
  }
  100% {
     background-color: green;
  }
}
```

Animations

Animation Properties

Once the @keyframes are defined, the animation properties must be added in order for your animation to function.

Animations

Animation Properties

Animation properties do two things:

- assign the @keyframes to the elements you want animated
- they define how it is animated.

Animations

Animation Properties

All the animation properties are added in the following order:

```
animation {
  [animation-name]
  [animation-duration]
  [animation-timing-function]
  [animation-delay]
  [animation-iteration-count]
  [animation-direction]
  [animation-play-state]
  [animation-fill-mode]
}
```

Animations

Animation Properties

The name of the animation, defined in the @keyframes.

```
.square {
  animation-name: bounceIn;
  animation-duration: 3.5s;
  animation-timing-function: linear;
  animation-delay: 0s;
  animation-iteration-count: 3;
  animation-direction: normal;
  animation-play-state: running;
  animation-fill-mode: none;
}
```

Animations

Animation Properties

These work like their transition properties

```
.square {
  animation-name: bounceIn;
  animation-duration: 3.5s;
  animation-timing-function: linear;
  animation-delay: 0s;
  animation-iteration-count: 3;
  animation-direction: normal;
  animation-play-state: running;
  animation-fill-mode: none;
}
```

Animations

Animation Properties

Specifies the number of times that the animation will play.

```
.square {
  animation-name: bounceIn;
  animation-duration: 3.5s;
  animation-timing-function: linear;
  animation-delay: 0s;
  animation-iteration-count: 3;
  animation-direction: normal;
  animation-play-state: running;
  animation-fill-mode: none;
}
```

Animations

Animation Properties

Specifies whether the animation should play forward, reverse, or in alternate cycles

```
.square {
  animation-name: bounceIn;
  animation-duration: 3.5s;
  animation-timing-function: linear;
  animation-delay: 0s;
  animation-iteration-count: 3;
  animation-direction: normal;
  animation-play-state: running;
  animation-fill-mode: none;
}
```

Animations

Animation Properties

Specifies whether the animation is playing or paused. Resuming a paused animation starts the animation where it was left off.

```
.square {
  animation-name: bounceIn;
  animation-duration: 3.5s;
  animation-timing-function: linear;
  animation-delay: 0s;
  animation-iteration-count: 3;
  animation-direction: normal;
  animation-play-state: running;
  animation-fill-mode: none;
}
```

Animations

Animation Properties

Specifies if the animation styles are visible before or after the animation plays.

```
.square {
  animation-name: bounceIn;
  animation-duration: 3.5s;
  animation-timing-function: linear;
  animation-delay: 0s;
  animation-iteration-count: 3;
  animation-direction: normal;
  animation-play-state: running;
  animation-fill-mode: none;
}
```

Animations

Animation Properties

What if we want to create an animation that causes an element to bounce into the browser?

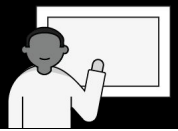
Animations

Animation Properties

First, we create a @keyframe with the animation name, bounceIn. Then, apply the animation rule to the CSS element you want animated.

```
@keyframes bounceIn {
  0% {
     transform: scale(0.1);
     opacity: 0;
  }
  60% {
     transform: scale(1.2);
     opacity: 1;
  }
  100% {
     transform: scale(1);
  }
}

.square {
  animation: bounceIn 3s ease-in-out;
}
```



Guided Walk-Through:

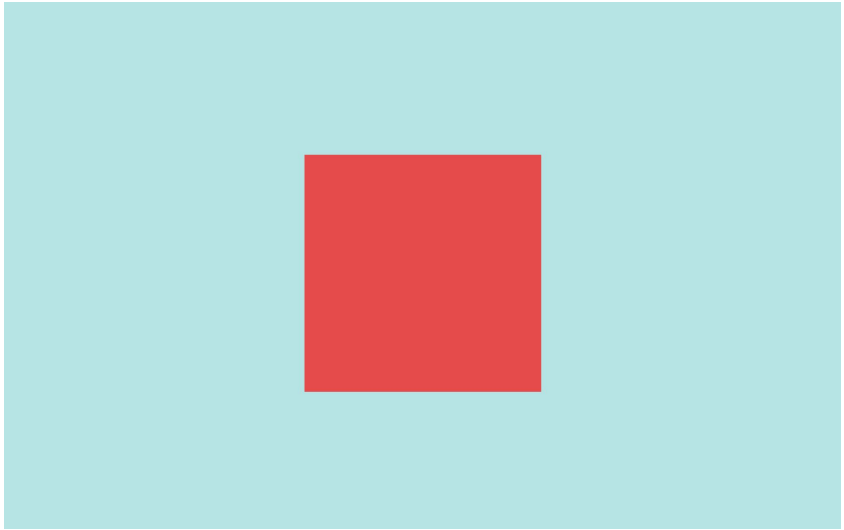
Animations

Create an animation that causes an element to slide across the screen.

```
<div class="square"></div>
```

```
.square {
  width: 150px;
  height: 150px;
  background: tomato;
}
```

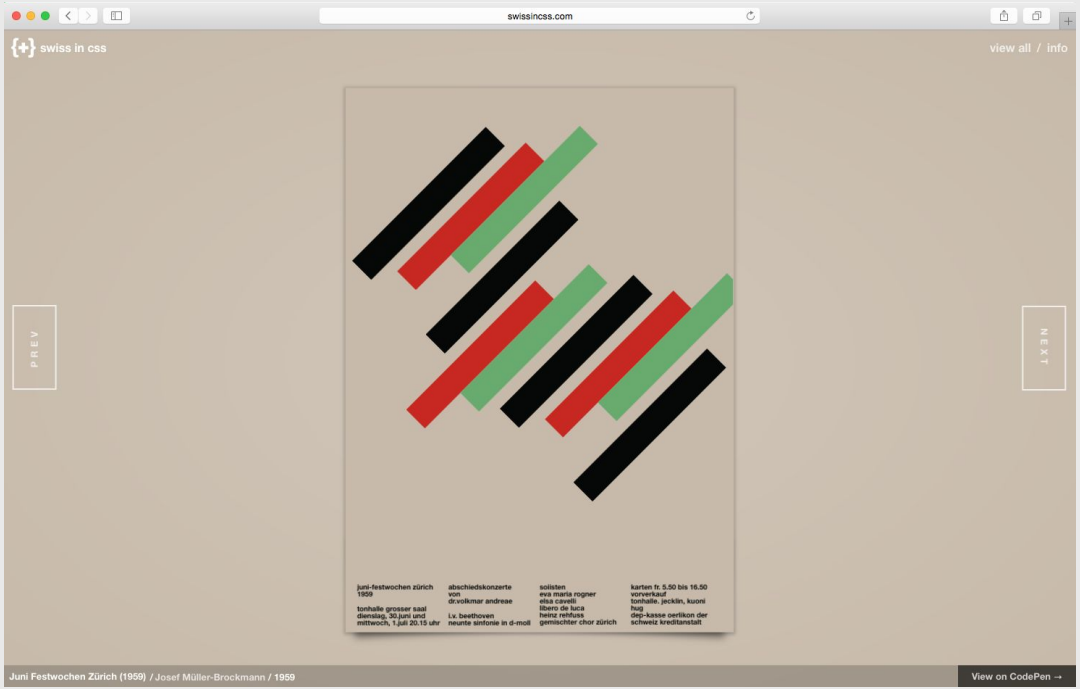
```
@keyframes slide {}
```



Examples

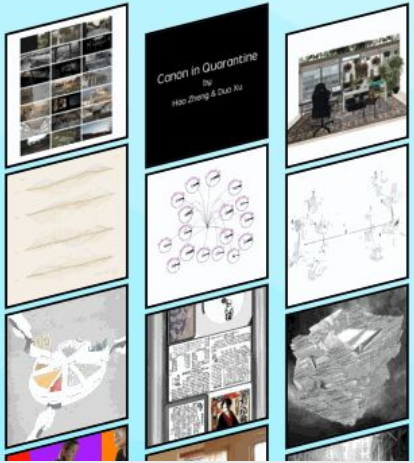
Animations

http://swissincss.com



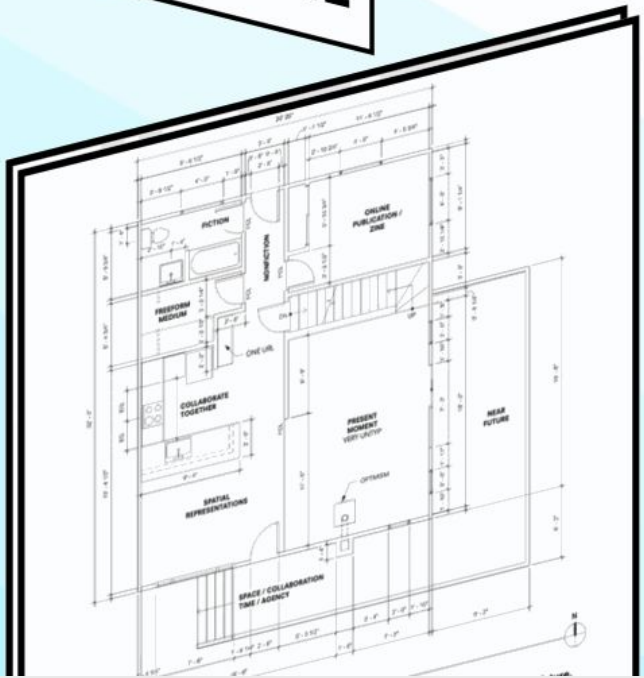
SPATIAL QUARANZINE

ABOUT SPATIAL QUARANZINE



Spatial Quaranzine is an ongoing online publication / zine by Columbia GSAPP students about our present moment and near future, using spatial and architectural representations. Spatial Quaranzine was made as part of Architectural Drawing and Representation 2, a class taught in Spring 2020 by Dan Taeyoung, Andrew Heumann, Lexi Tsien-Shiang, Quentin Yiu, and Violet Whitney.

Site Development by Andrew Heumann and Dan Taeyoung.



QUARANZINE ASSIGNMENT

Lab Time

Class Exercise

Using your new knowledge about how basic animations work, create a simple button. To this button, add a :hover style and write a transition rule that changes the background color over 2 seconds starting 1 second after the hover action.

Class Exercise

Using the flexbox navigation menu we worked on in class last week, add a **:hover** style and write a **@keyframe** animation that smoothly changes the background color of each menu item over 3 seconds. The animation should have four stages.

Update your HTML file to your Github repository, and add a link to the page on the Github homepage. Post the link in Slack with the channel #exercise06.

Project

Project 01

Deliverable << 02/28/2023 >>

Pick one of the following categories: e-commerce site, collection for a fashion design studio, architecture or interior design company, museum or art gallery.

You will be creating a **one-page website**.

Find a website that fits the category you have chosen. Create a wireframe influenced by this website you have selected.

See the following example:

Project 01

Deliverable << 02/28/2023 >>

Pick one of the following categories: e-commerce site, collection for a fashion design studio, architecture or interior design company, museum or art gallery.

Find a website that fits the category you have chosen. Create a wireframe influenced by this website you have selected.

Project 02

Deliverable << 02/28/2023 >>

Create the boilerplate. Collect the images and text that you will use in the site. Save these assets to a folder. Decide on the color scheme. Decide on the fonts that will be used (use two fonts maximum) and the sizes of the fonts for the different content types.

Set up the layout using flexbox. Add the content to the page: images, video and text.

Project 02

Deliverable << 02/28/2023 >>

Upload your final project to your Github repository, and add links to the project on your Github landing page. Post the link to the Projects tab in the Google Sheet.