
FEWD

FINAL PROJECTS

AGENDA



- Debugging
- Carousel Lab Pt. 2
- Advanced CSS Positioning
- Transitions
- Transformations
- Animations

LEARNING OBJECTIVES

- Identify and differentiate between different CSS positioning techniques
- Familiarity with how animations and transitions can be used in CSS
- Understand how animation can still be controlled using JS
- Know the different ways to debug code and how to apply the concepts

FEWD

REVIEW

SYNTAX — DECLARING A FUNCTION

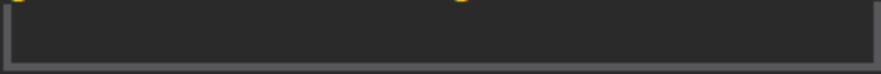
```
function pickADescriptiveName() {  
    // Series of statements to execute  
}
```

The diagram illustrates the syntax of a function declaration. It features three labels with brackets pointing to specific parts of the code: 'Keyword' points to 'function', 'Name' points to 'pickADescriptiveName()', and 'Code block' points to the entire function declaration structure, including the opening and closing curly braces and the comment.

SYNTAX — CALLING A FUNCTION

- ▶ To run the code in a function, we 'call' the function by using the function name followed by parenthesis.

```
pickADescriptiveName();
```



Function name

SYNTAX — DECLARING A FUNCTION (WITH PARAMETERS)

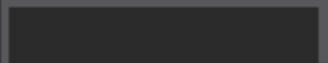
Parameters

```
function multiply(param1, param2) {  
  return param1 * param2;  
}
```

We can use these parameters like variables from within our function

SYNTAX — CALLING A FUNCTION (WITH ARGUMENTS)

Arguments



multiply(350, 140)

The diagram illustrates the syntax of a function call. The word "multiply" is in white, and the numbers "350" and "140" are in yellow. A horizontal line with vertical end caps is positioned above the numbers, with the word "Arguments" in yellow centered above it, indicating that the numbers are the arguments passed to the function.

ARRAYS

STORING LISTS OF VALUES

- An array can be used to **store a list of values in a single variable**
- Holds an ordered collection of values
- Can hold numbers, strings, even other arrays!
- Good for things like a grocery list, a list of states, or any other list

DECLARING ARRAYS

```
var descriptiveNameHere = [item1, item2, item3];
```

ARRAYS - INDEXING

- Each item in an array has an **index**, by which you can access that item.
- The first item has an index of **0**, the second item 1, the third item 2, etc.

0. Milk

1. Eggs

2. Frosted Flakes

3. Salami

4. Juice

ARRAYS - ACCESSING ITEMS BY INDEX

- Each item in an array has an **index**, by which you can access that item.
- The first item has an index of **0**, the second item 1, the third item 2, etc.

```
var myArray = [5, true, 2, 'Hello']
```

0 1 2 3

The diagram consists of four yellow arrows pointing upwards from the numbers 0, 1, 2, and 3 to the corresponding elements in the array: 5, true, 2, and 'Hello'.

ARRAYS — ACCESSING ITEMS IN AN ARRAY

Accessing items in array:

`myArray[1]` → `true`

`myArray[2]` → `2`

`myArray[0]` → `5`

`myArray[3]` → `'Hello'`

```
var myArray = [5, true, 2, 'Hello']
```

ARRAYS - ADDING A VALUE/REPLACING A VALUE

INSERTING A NEW VALUE

- We can insert new values into any space in the array using the positions index.

```
myArray[1] = 'Hello';
```

UPDATING VALUES

- If there's already an item at that position, it will be replaced with the new value.

```
var myArr = [65, 'hello', true];  
myArr[1] = 'goodbye';  
// myArr[1] now holds 'goodbye' instead of 'hello'
```

ARRAYS - LENGTH

- ▶ We can use the `.length` property to find out how many items are in an array

```
var shapes = ['circle', 'triangle', 'square'];
```

```
shapes.length;
```

 => 3

- ▶ Accessing the last element in an array:

```
console.log(shapes[shapes.length-1]);
```

 => Prints 'square'

FEWD

‘THIS’ KEYWORD

THE KEYWORD 'THIS'

this refers to whatever you *selected* with jQuery

```
$('p').on('click', function(){  
    $(this).fadeOut(500);  
});
```




Notice — no quotes around this!

JQUERY — SELECTING ELEMENTS

Selector

```
$('li').addClass('selected');
```

A diagram consisting of a horizontal line with a vertical line extending upwards from its center, forming a bracket shape. This bracket is positioned above the text 'li' in the code snippet, pointing to it.

THIS



EXERCISE

KEY OBJECTIVE

- ▶ Practice applying the `this` keyword

TYPE OF EXERCISE

- ▶ Individual/Partner

TIMING

6 min

1. Follow the instructions in `starter_code_lesson_12 > this > js > main.js`

FEWD

PREVENT DEFAULT

JQUERY METHODS — THE EVENT OBJECT

- ▶ The event object has properties and methods that tell you more about the event that took place.
- ▶ By using the preventDefault method, the default action of the event will not be triggered.

Parameter name

```
$('#li').on('eventGoesHere', function(e) {  
    e.preventDefault();  
});
```

Use that name in the function and use dot notation to access its properties and methods.

**CREATE
EVENT
LISTENERS**

FEWD

ADVANCED CSS POSITIONING REVIEW

STATIC POSITIONING

- This is the normal flow of the document, the **default**
- Elements render in order, as they appear in the document flow.

```
.my-class {  
  position: static;  
}
```

RELATIVE POSITIONING

- ▶ Relative positioning moves an element *relative to where it would have been in normal flow*.
- ▶ For example, "left: 20px" adds 20px to an element's **left** position
- ▶ Creates a *coordinate system for child elements*.

```
.my-class {  
  position: relative;  
  top: 20px;  
  left: 30%;  
}
```

ABSOLUTE POSITIONING

- ▶ When the *position* property is given a value of *absolute*, an element is taken out of the normal flow of the document.
- ▶ This element no longer affects the position of other elements on the page (they act like it's not there).
- ▶ You can add the *right*, *top*, *left* and *bottom* properties to specify where the element should appear relative to its first positioned (not static) ancestor element

```
.my-class {  
  position: absolute;  
  top: 0;  
  left: 500px;  
}
```

FIXED POSITIONING

- ▶ When the *position* property is given a value of *fixed*, the element is positioned in relation to *the browser window*
- ▶ When the user scrolls down the page, it stays in the same place.
- ▶ You can add the *right*, *top*, *left* and *bottom* properties to specify where the element should appear in relation to the browser window.

```
.my-class {  
  position: fixed;  
  top: 0;  
  left: 500px;  
}
```

OVERLAPPING ELEMENTS — Z-INDEX

- When using relative, fixed or absolute positioning, elements can overlap.
- When elements overlap, the elements that appear later in the HTML code sit on top of those that appear earlier in the page.
- If you want to control which elements are layered on top of each other, you can use the z-index property.
- This property takes a number — the higher the number the closer that element is to the front.
- Similar to 'bring to front' and 'send to back' in programs like *Adobe Illustrator*.

```
.my-class {  
  z-index: 10;  
}
```

FEWD

CSS POSITIONING & ANIMATION

FEWD

TRANSITIONS

TRANSITIONS

- Provide a way to control animation speed when changing properties
- Instead of having property changes take effect immediately, you can have them take place over a period of time.

```
.example {  
  transition: [transition-property] [transition-duration] [transition-timing-function] [transition-delay];  
}
```

TRANSITIONS



PROPERTY

Which properties
to animate



DURATION

How long the
transition will last



**TIMING
FUNCTION**

How the transition
will run

TRANSITIONS - TRANSITION-PROPERTY

- ▶ Can specify a specific property to transition or "all" to transition all properties
- ▶ Default: all

```
div {  
  transition: opacity 0.5s;  
}
```

```
div {  
  transition: all 0.5s;  
}
```

```
div {  
  transition: height 0.5s;  
}
```

```
.example {  
  transition: [transition-property] [transition-duration] [timing-function] [transition-delay];  
}
```

TRANSITIONS - TRANSITION-DURATION

- ▶ A time value, defined in seconds or milliseconds

```
div {  
  transition: all 0.5s;  
}
```

```
div {  
  transition: all 350ms;  
}
```

```
div {  
  transition: all 3s;  
}
```

```
.example {  
  transition: [transition-property] [transition-duration] [timing-function] [transition-delay];  
}
```

TRANSITIONS

- ▶ Describes how a transition will proceed over its duration, allowing a transition to change speed during its course.
- ▶ Timing functions: ease, linear, ease-in, ease-out, ease-in-out

```
div {  
  transition: opacity 0.5s ease;  
}
```

```
div {  
  transition: opacity 0.5s ease-in-out;  
}
```

```
.example {  
  transition: [transition-property] [transition-duration] [timing-function] [transition-delay];  
}
```

transition timing: [W3 Schools](#)

TRANSITIONS

- ▶ Length of time before the transition starts

```
div {  
  transition: background-color 0.5s ease 2s;  
}
```

```
.example {  
  transition: [transition-property] [transition-duration] [timing-function] [transition-delay];  
}
```

MORE FUN WITH TRANSITIONS

Fun CSS button styles: [Creative buttons](#)

Icon hover effects: [Icon Hover Effects](#)

Modal dialogue effects (advanced): [Dialogue Effects](#)

FEWD

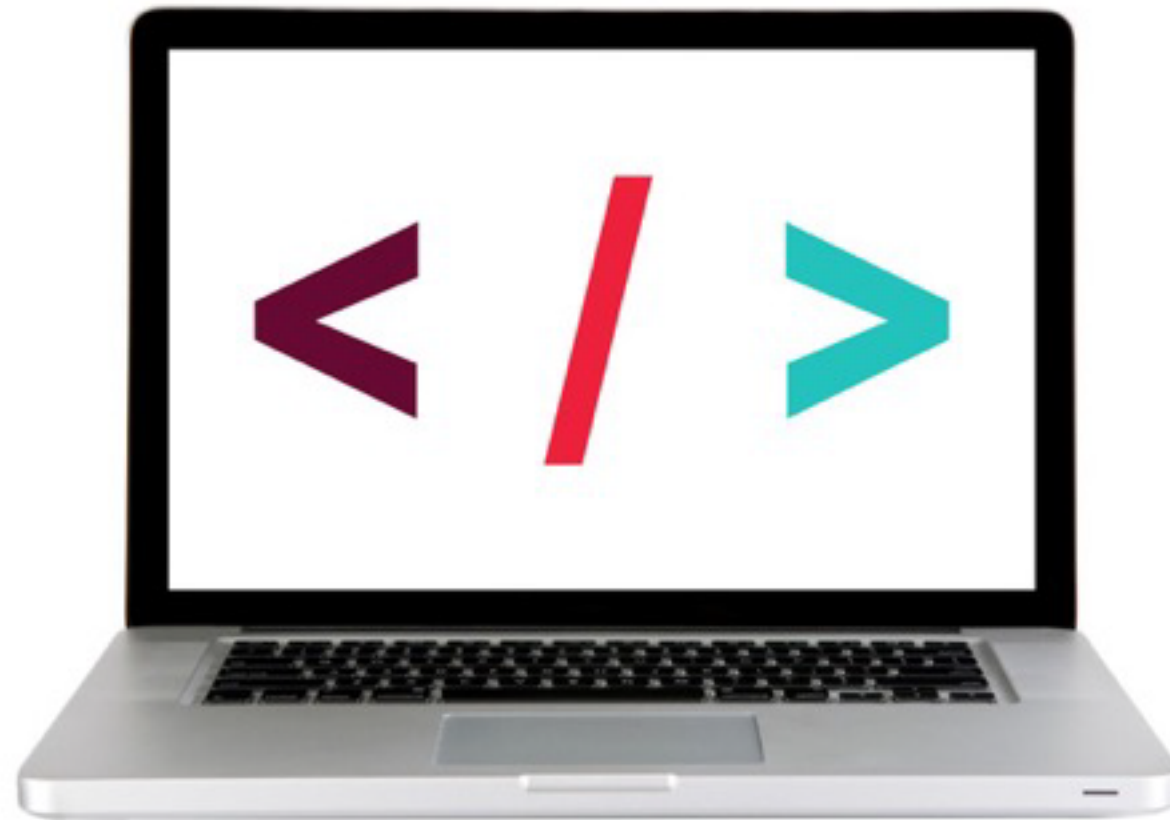
TRIGGERING TRANSITIONS

TRIGGERING TRANSITIONS

There are two ways to trigger CSS transitions:

1. Using the :hover CSS pseudo-class
2. Adding a class with jQuery with mouseover

LET'S TAKE A CLOSER LOOK — TRIGGERING TRANSITIONS



ACTIVITY — BUTTON LAB



EXERCISE

KEY OBJECTIVE

- ▶ Practice using CSS transitions

TYPE OF EXERCISE

- ▶ Individual/Partner Lab

TIMING

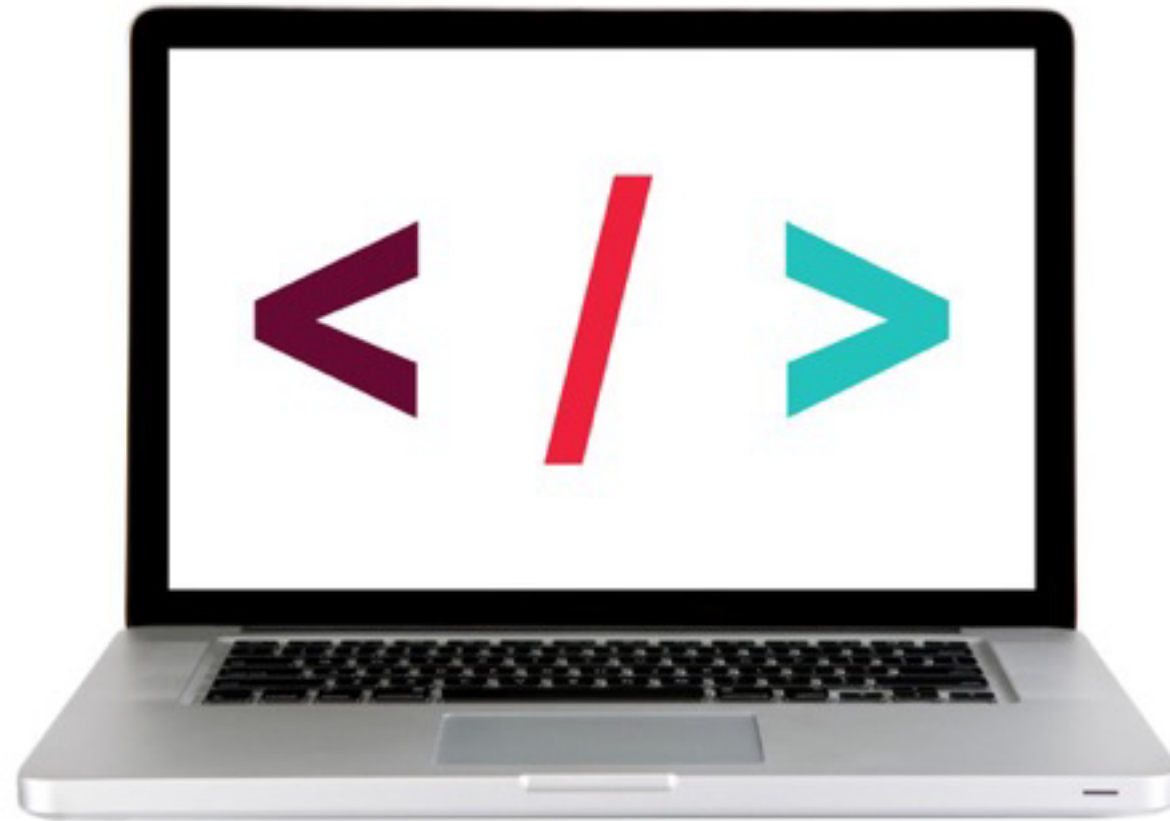
6 min

1. Add :hover styles and transition to the button:
lesson11_starter_code > [2] transition_button_lab

FEWD

TRANSFORMATIONS

LET'S TAKE A CLOSER LOOK — TRANSFORM



transform: [W3 Schools](#)
transform-origin: [W3 Schools](#)

ACTIVITY — HAMBURGER ICON



EXERCISE

KEY OBJECTIVE

- ▶ Practice using CSS transitions

TYPE OF EXERCISE

- ▶ Individual/Partner Lab

TIMING

10 min

1. Follow the instructions in `lesson11_starter_code > [3] transformation_lab`

FEWD

ANIMATIONS

KEYFRAME ANIMATIONS

- ▶ Keyframe animations allow developers to create smooth, maintainable animations that perform well and don't require tons of scripting

<https://www.impressivewebs.com/demo-files/css3-animated-scene/>

KEYFRAME ANIMATIONS — SYNTAX

1. Define custom animation

```
@-webkit-keyframes NAME-YOUR-ANIMATION {  
  0%   { opacity: 0; }  
  100% { opacity: 1; }  
}  
@keyframes NAME-YOUR-ANIMATION {  
  0%   { opacity: 0; }  
  100% { opacity: 1; }  
}
```

2. Assign using the animation property

```
#box {  
  -webkit-animation: NAME-YOUR-ANIMATION 5s infinite;  
  animation:        NAME-YOUR-ANIMATION 5s infinite;  
}
```

A simple tool to make sure you're including all the necessary browser prefixes: [please](#)

KEYFRAME ANIMATIONS — ANIMATION PROPERTY

Properties:

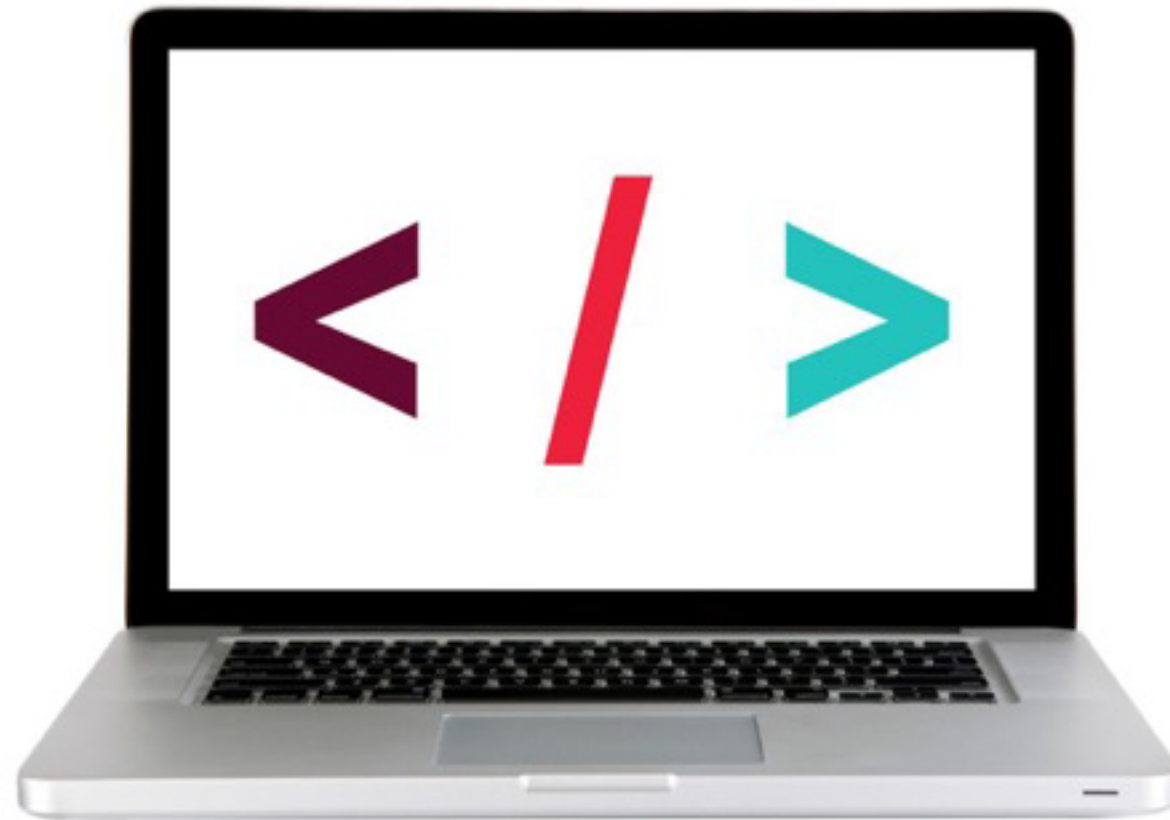
```
.box {  
  animation-name: bounce;  
  animation-duration: 4s;  
  animation-timing-function: ease-out;  
  animation-delay: 2s;  
  animation-iteration-count: 10;  
}
```

Shorthand:

```
.example {  
  animation: name duration timing-function delay iteration-count;  
}
```

```
.example {  
  animation: bounce 4s ease-out 2s 10;  
}
```

LET'S TAKE A CLOSER LOOK — TRIGGERING TRANSITIONS



Code along — Spinning Wheel

FEWD

LAB

LAB



ACTIVITY



EXERCISE

KEY OBJECTIVE

- ▶ Practice triggering CSS transitions with jQuery

TYPE OF EXERCISE

- ▶ Individual/Partner Lab

TIMING

10 min

Until 8:50

2. Add base styles to the page **lesson11_starter_code > [5] sidebar_lab**
3. Make sidebar interactive using jQuery and CSS transitions.

FEWD

HOMEWORK

IF YOU HAVEN'T DONE SO ALREADY

- FINISH FINAL HTML**
- UPLOAD BOILERPLATE TO GITHUB IN FINAL PROJECT REPOSITORY**

**HAVE CSS MOSTLY
READY THIS BY NEXT SUNDAY**

FINISH SIDEBAR NAV

FEWD

EXIT TICKETS