FEWD WEEK 5 • CLASS 10:

Javascript Basics

https://slides.com/jennifermeade/fewd-5-10/live

QUICK REVIEW

ELEMENT TAG SELECTORS

To select all of the elements on the page with the same tag, we use the tag name

```
<body>
  <h1 id="title">This is a heading on the page</h1>
  <p class="red big">This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <div class="big">
    <p class="red">This paragraph is a child of a div.</p>
  </div>
</body>
```

```
$('p') //Selects all of the paragraphs
```

CLASS SELECTORS

To select all of the elements on the page with the same class, we use the class name preceded by a dot (.)

```
<body>
  <h1 id="title">This is a heading on the page</h1>
  <p class="red big">This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <div class="big">
    <p class="red">This paragraph is a child of a div.</p>
  </div>
</body>
```

```
$('.big') //Selects the elements with a class of big
```

ID SELECTORS

To select an element on a page by its id, we use the id name preceded by the pound symbol (#)

```
<body>
  <h1 id="title">This is a heading on the page</h1>
  <p class="red big">This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <div class="big">
    <p class="red">This paragraph is a child of a div.</p>
  </div>
</body>
```

```
$('#title') //Selects the element with an id of title
```

DESCENDANT SELECTORS

To select elements that are descendants of specific elements, we use combinators. The space is for descendants, and > is for child elements.

```
<body>
  <h1 id="title">This is a heading on the page</h1>
  <p class="red big">This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <div class="big">
    <p class="red">This paragraph is a child of a div.</p>
  </div>
</body>
```

```
$('div > p') //Selects p elements that are children of divs
```

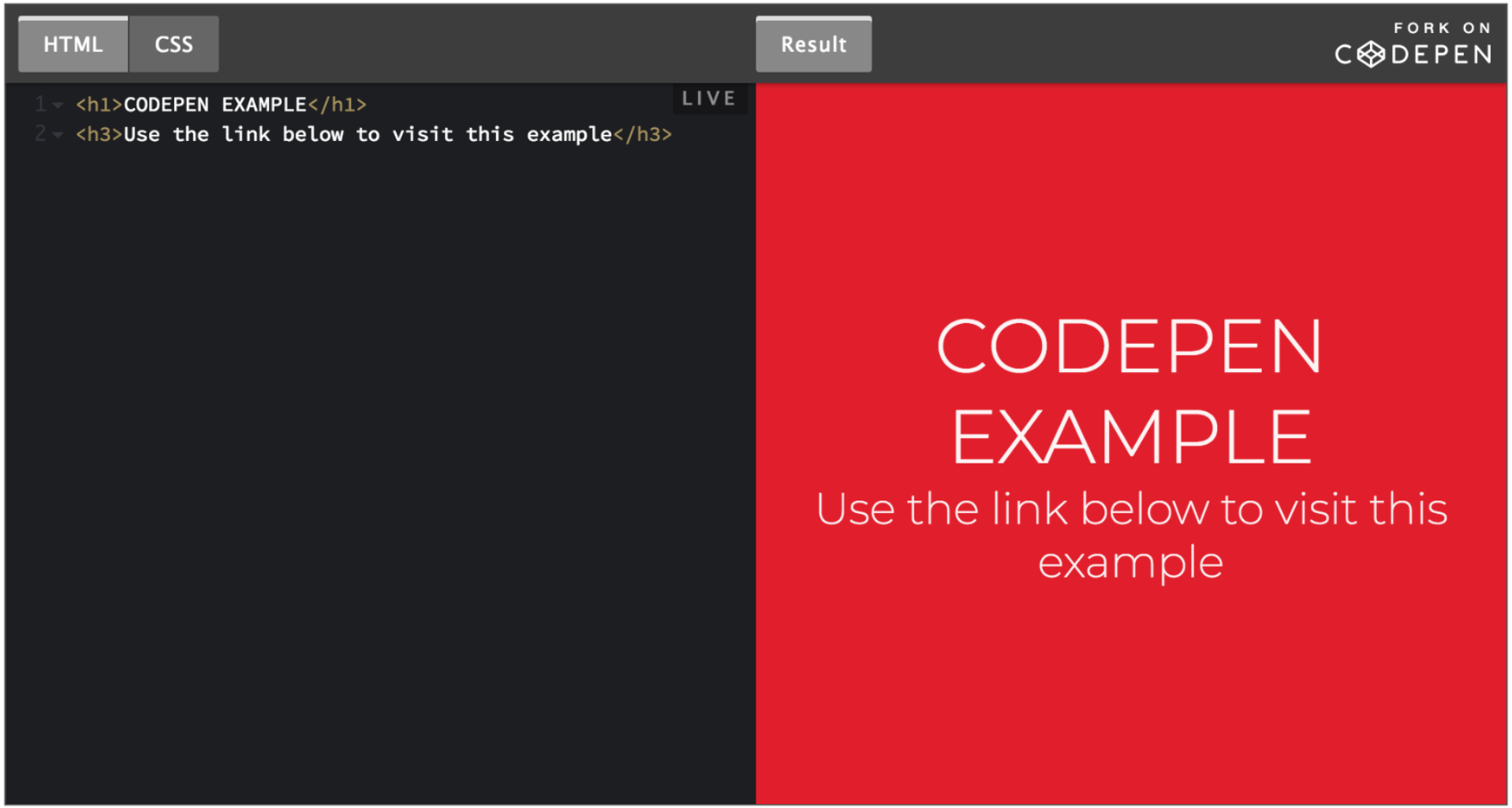
GROUPS OF SELECTORS

As in CSS, we can use a comma to group multiple selectors together. The comma acts like an **OR**.

```
<body>
  <h1 id="title">This is a heading on the page</h1>
  <p class="red big">This is a paragraph.</p>
  <p>This is another paragraph.</p>
  <div class="big">
    <p class="red">This paragraph is a child of a div.</p>
  </div>
</body>
```

```
$('.red, .big') //Selects elements with a red OR big class
```

CODEPEN



https://codepen.io/jme11/pen/OaMMRV

OBJECTIVES

- Use jQuery to add effects and animation
- Create and use variables to store strings, numbers and boolean values
- Use jQuery methods as both a getter and setter
- Chain methods to write more efficient code
- Understand and use the Javascript keyword **this**

JQUERY EFFECTS

JQUERY SHOW AND HIDE METHODS

```
$('div').show();
$('div').hide();
$('div').toggle();
```

- The show method displays an element that has been hidden with hide, toggle or display: none.
- The hide method sets an element's display property to none.
- The toggle method shows an element if it's hidden and hides it if it's displayed.

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <button>Toggle Me</button>
    <div style="height: 100px; width: 100px; background-color: red;"></div>

    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $(document).ready(function(){
        $('button').click(function(){
          $('div').toggle();
        });
      });
    </script>
  </body>
</html>
```

JQUERY FADE METHODS

```
$('p').fadeIn('slow');

$('div').fadeOut('fast');

$('h1').fadeToggle(5000);
```

- The fade methods fade an element in or out
- Using the hide method or setting the display property of the element to none, sets up an element for the fadeIn method
- You can control the speed by passing the method: fast, slow or a specific speed in milliseconds

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <h1 style="display: none;">My Title</h1>

    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $(document).ready(function(){
        $('h1').fadeIn(1000);
      });
    </script>
  </body>
</html>
```

JQUERY SLIDE METHODS

```
$('p').slideDown('slow'); //shows a hidden element

$('div').slideUp('fast'); //hides an element

$('h1').slideToggle(5000);
```

- jQuery effects can also be used to animate elements on or off the page
- Using the hide method or setting the display property of the element to none, sets up an element for the fadeIn method
- You can control the speed by passing the method: fast, slow or a specific speed in milliseconds

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <button>Toggle</button>
    <div style="height: 100px; width: 100px; background-color: red;"></div>

    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $(document).ready(function(){
        $('button').click(function(){
          $('div').slideToggle('slow');
        });
      });
    </script>
  </body>
</html>
```

COMPLETE FUNCTION

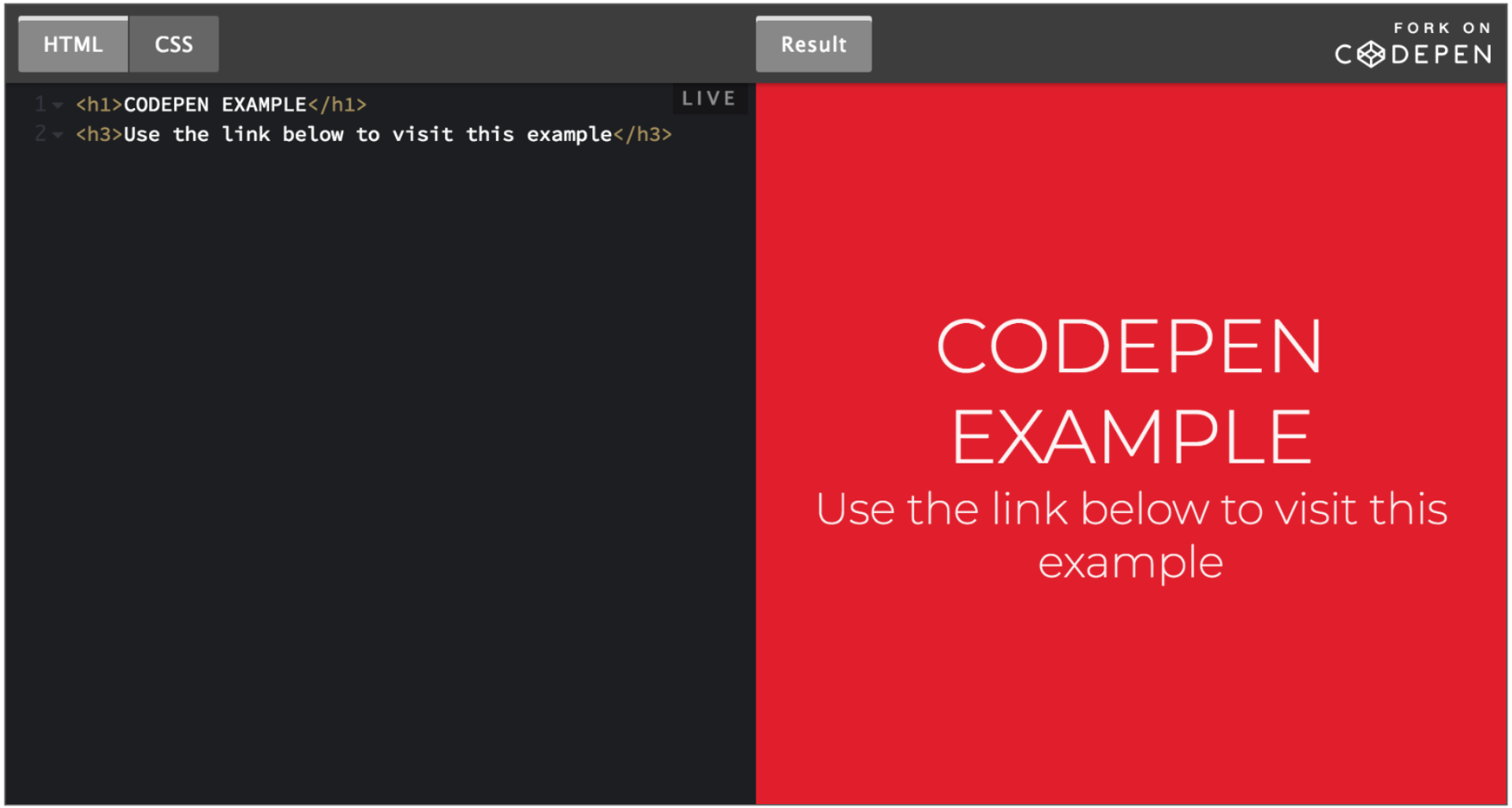
- These effects also take a second value, which is an anonymous function (like the one we use with event handlers).
- It runs the code inside of it only when the animation is completed!

Javascript code is said to be *asynchronous*, so the code will run as soon as it is encountered by the Javascript engine. The complete function gives you more control to ensure that your animation completes before some other code is run.

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <button>Toggle</button>
    <div style="height: 100px; width: 100px; background-color: red;"></div>
    <p></p>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $(document).ready(function(){
        $('div').hide();
        $('button').click(function(){
          $('div').slideDown(1000, function(){
            $('p').text('Done!')
          });
        });
      });
    </script>
  </body>
</html>
```

SLIDING MENU LAB



https://codepen.io/jme11/pen/OBjGxB

SLIDING MENU SOLUTION


```
$(document).ready(function(){

  // 1. Hide the menu
  $('#site-menu').hide();

  // 2. Listen for clicks on the button
  $('button').click(function(){

    // 3. Inside the click handler function, toggle the site-menu
    $('#site-menu').slideToggle();

  });

});
```

ANIMATING WITH JQUERY

- Animations in CSS will always be more efficient than in Javascript
- Javascript can give you more control over the animation, allowing you to pause, reverse, add intermittent delays, etc.

ANIMATE METHOD

```
$('button').click(function() {
  $('div').animate({
    opacity: 0.25,
    left: '+=50', // adds 50 to the current position
    height: 'toggle'
  }, 5000);
});
```

- Notice the slightly different syntax here
- The first parameter is an object.
- The object format is:

```
{property: value, property: value}
```

VARIABLES IN JAVASCRIPT

WHAT IS A VARIABLE?

- Variables are containers for bits of information that we can reuse in our programs
- We give variables names so that we can access them and get at the data inside of them.

CREATING VARIABLES

```
const x = 10;
var y = 5;
let z = 8;
```

- Variables are ***declared*** using the keyword `var`, `let` or `const` in Javascript followed by a name.
- Variables are ***assigned*** when they are given a value.

Variables names can contain: letters, numbers, the underscore (_) and the dollar sign ($), but cannot begin with a number.

VAR, LET AND CONST

- `var`: Older format supported in all browsers
- `let`: Newer format. Tells Javascript that the content of the variable is meant to be changed (called reassigned)
- `const`: Newer format. Tells Javascript that the variable is a constant. It won't be changed! This kind of variable **needs to be declared and given its value in one statement**.

BEST PRACTICE: Use const unless your value is going to change. If it will change, use let.

ASSIGNING & REASSIGNING

```
let firstName; // declaration

firstName = 'Jennifer';  // assignment

let lastName = 'Maloney'; // declaration and assignment

lastName = 'Meade'; // reassignment
```

- A variable is assigned with the `=` known as the assignment operator.
- A variable can be reassigned (unless it was declared with const)
- When reassigning a variable, no keyword is used

GETTING AT THE DATA

```
let name = 'Fred';

$('.name').text(name);

let age = 21;

age = age + 10;

console.log(age);
```

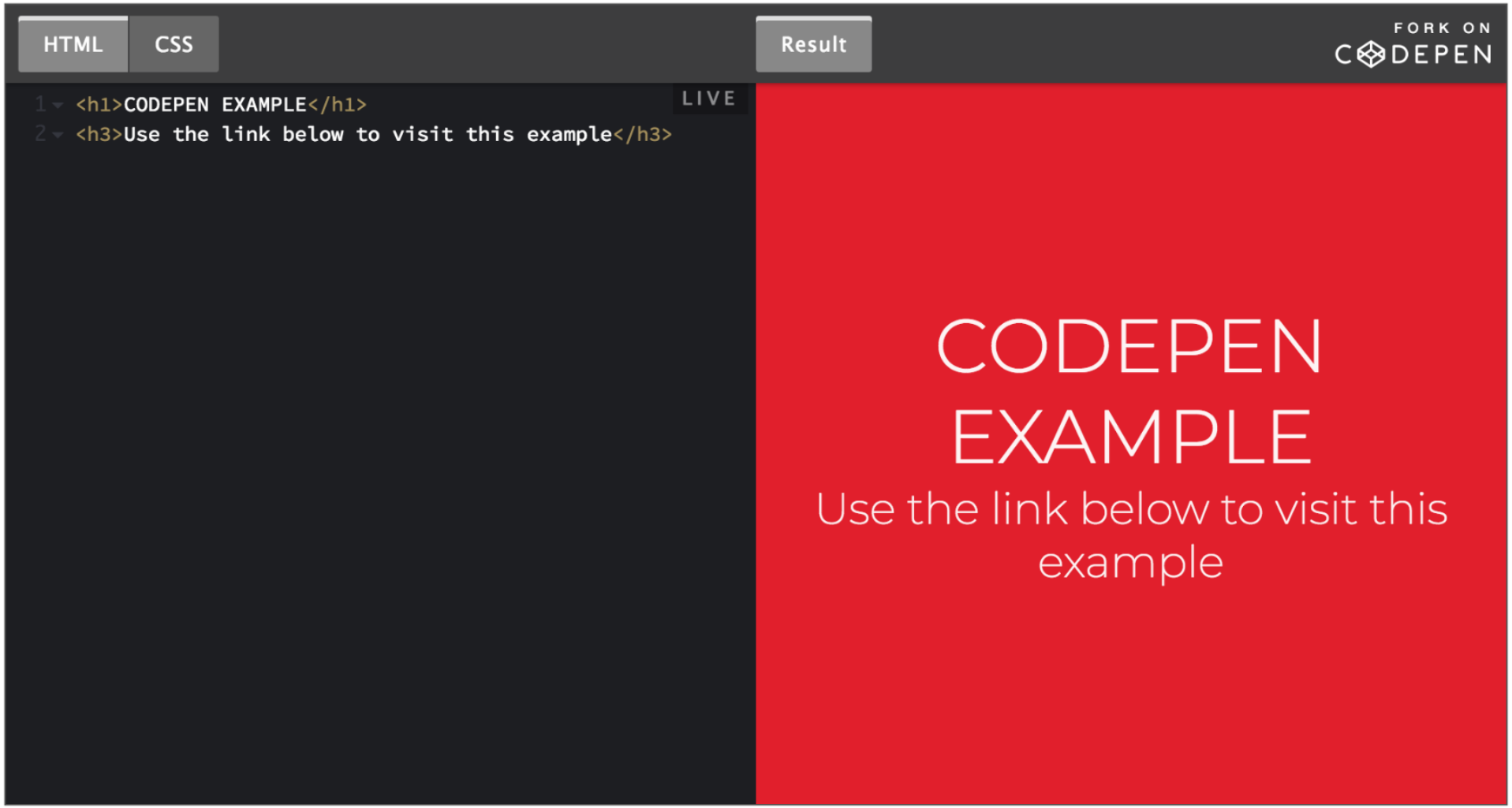
- To get at the data in a variable, use the variable name where you want the value
- We **never surround variable names with quotes**, that's how Javascript knows its a variable and not random text

WHAT GOES IN VARIABLES?

```
let age = 21; /* Number (integers and floating point) */
let height = 65.25; /* Number */
let balance = -20.66; /* Number */

let name = 'Jen'; /* String (any combination of characters) */
let dogs = '2'; /* String */

let tired = true; /* Boolean (true or false) */

var book; /* undefined (declared but not assigned)*/
let tickets = null; /* null (empty but not undefined) */

const $body = $('body') /* Objects */
```

There are 7 datatypes in Javascript. Symbols will not be covered in this course.

CODEPEN



https://codepen.io/jme11/pen/EparbP

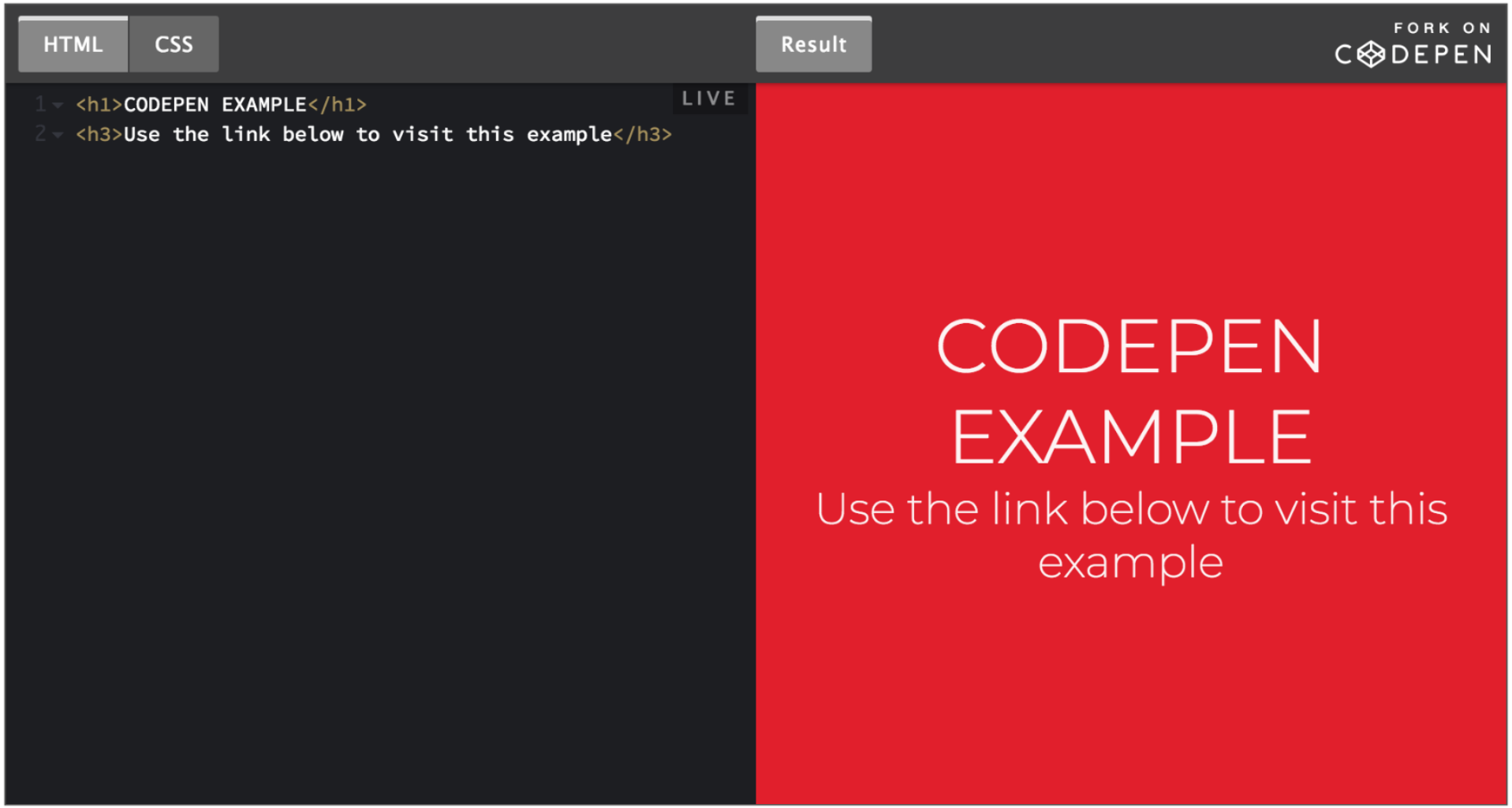
METHODS AS GETTERS

GETTERS VS SETTERS

- Setters set a value
- Getters return a value
- Many jQuery methods are both getters and setters

```
// with no value supplied, returns the current text

let name = $('#name').text();

// with no value supplied, returns the current background-color

let color = $('body').css('background-color');
```

COLOR VALUE FINDER



https://codepen.io/jme11/pen/PxZJXQ

CHAINING METHODS

❤ CHAINING METHODS ❤

- Chaining methods let us write more **efficient** and more concise code
- Makes us faster and helps reduce errors

Whenever Javascript encounters the end of a statement (denoted by a line-break or semicolon), it will promptly forget the element(s) it's been operating on. So, if you have multiple statements with the same selector, you're forcing the Javascript engine to go back and find those elements in the entire DOM each time!

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
  </head>
  <body>
    <h1 style="display:none;">My Title</h1>

    <!-- Where your content ends -->
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $('h1').text('My NEW Title').css('color','red').fadeIn(2000);
    </script>
  </body>
</html>
```

USING THIS

THIS

```
$('div').click(function(){
   //in here this refers to the div we clicked!

   $(this).css('background-color', 'red');
});
```

- The keyword ***this*** has a special meaning in Javascript.
- When we use it as a selector in jQuery, it refers only to the current element.
- Take note that this does **not** get quoted like other selectors.

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
    <style>
      div {height: 50px; cursor: pointer;}
    </style>
  </head>
  <body>
    <div>first div</div>
    <div>second div</div>
    <div>third div</div>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
       $('div').click(function(){
         $(this).text('you clicked me!');
       });
    </script>
  </body>
</html>
```

GO BUILD AWESOME THINGS!