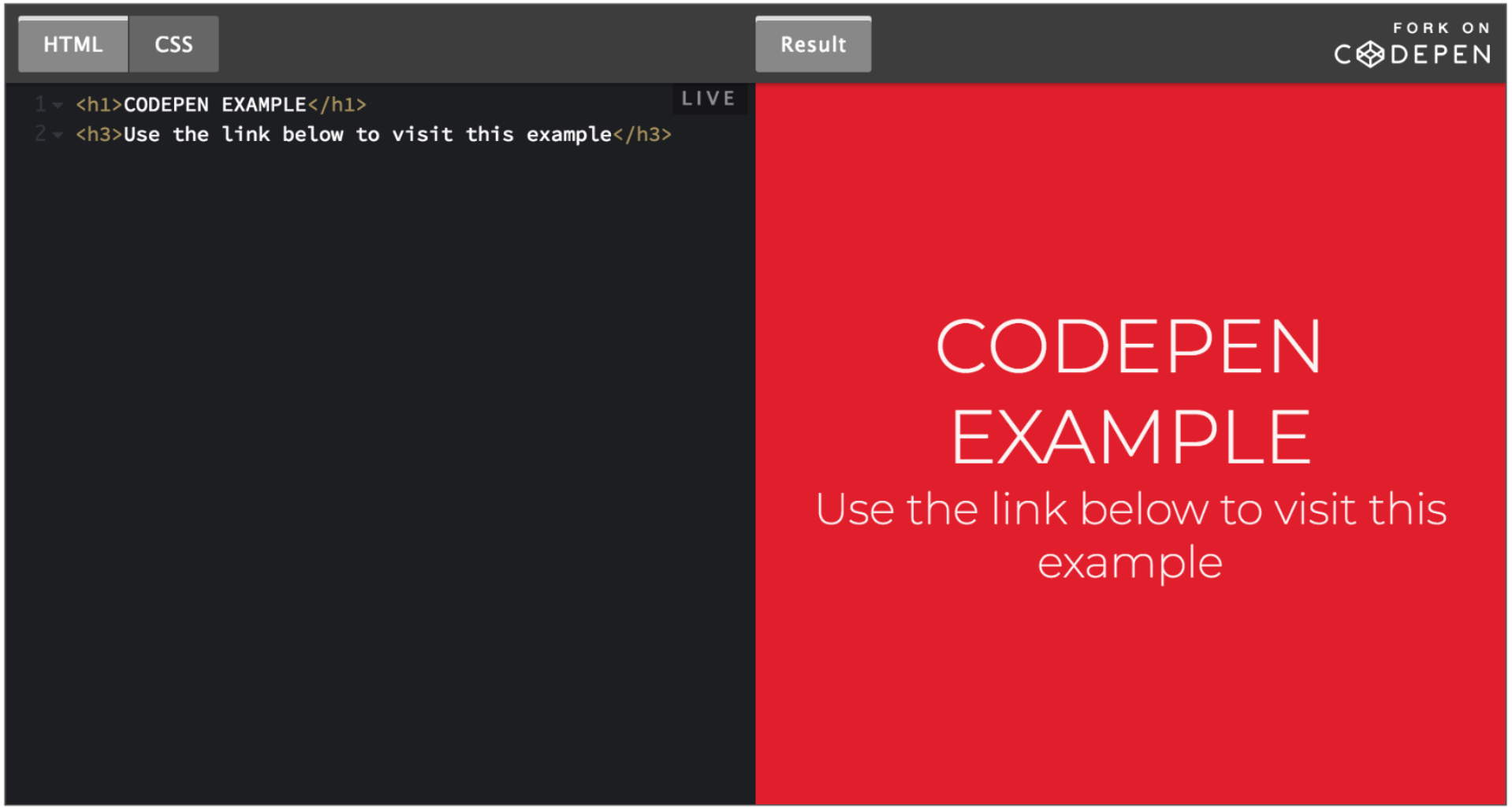
FEWD WEEK 7 • CLASS 12:

Arrays & Loops

https://slides.com/jennifermeade/fewd-7-12/live

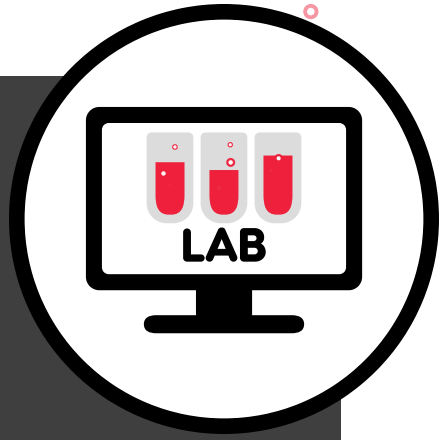
QUICK REVIEW

JQUERY WARMUP



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

WARMUP SOLUTION




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

  // Listen for clicks on div elements
  $('div').click(function(){
    // Get the value of the text and store in a variable
    const val = $(this).text();

    if (val == 20) {
      // If the val is 20, set the background to red
      $(this).css('background', 'red');
    } else if (val > 20) {
      // If the val is greater than 20, set the background to green
      $(this).css('background', 'green');
    } else {
      // If the val is neither 20 or more than 20, set the background to blue
      $(this).css('background', 'blue');
    }
  });

});
```

OBJECTIVES

- Understand how and when to use the append and prepend methods in jQuery.
- Understand the structure and use of arrays and objects in Javascript
- Use jQuery methods to "walk the DOM"

APPEND & PREPEND METHODS

ADDING TO THE DOM

- So far, we've used the **text** and **html** methods to replace content in the DOM.
- What if we want to just add to the content that is already there? That's where **append** and **prepend** come in handy...

APPEND METHOD

```
// Add an li element to the end of a list with append

$('ol').append('<li>Adding a new list item to the end!<li>');
```

- The `.append()` method will append new content to the element(s) that match the selector adding the new content after any existing contents.
- You must give it html (or text).

PREPEND METHOD

```
// Add an li element to the beginning of a list with prepend

$('ol').prepend('<li>Adding a new list item to the end!<li>');
```

- The `.prepend()` method will prepend new content to the element(s) that match the selector adding the new content before any existing contents.
- You must give it html (or text).

WHAT WILL HAPPEN?

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>jQuery</title>
  </head>
  <body>
    <div class="project">
      <img src="img/project1.png">
    </div>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $('.project')
        .prepend('<h4>Client:</h4><h2>First Bank</h2>')
        .append('<h4>Project Date:</h4><p>May 2018</p>');
    </script>
  </body>
</html>
```

THE RESULT

```
$('.project').prepend('<h4>Client:</h4><h2>First Bank</h2>');
```

```
<div class="project">
  <h4>Client:</h4>
  <h2>First Bank</h2>
  <img src="img/project1.png">
  <h4>Project Date:</h4>
  <p>May 2018</p>
</div>
```

```
$('.project').append('<h4>Project Date:</h4><p>May 2018</p>');
```

USING TEMPLATE LITERALS

```
let projectname = $(this).text();
const template = `
  <div>
    <h2>${projectName}</h2>
    <p class="summary">Here's the project summary</p>
  </div>
`
$('.pop-up').append(template);
```

- Backticks can be used to surround strings instead of single or double quotes.
- Can span multiple lines.
- Insert variables or expressions with: `${  }`

ARRAYS IN JAVASCRIPT

WHAT ARE ARRAYS

Arrays hold multiple pieces of related data. They are kind of like lists that we can assign to a variable.

ARRAY SYNTAX

```
const locations = ["New York", "Boston", "Atlanta"];

const scores = [5482, 6366, 4192, 671];

const lists = [['Buy wine', 'Pick up dogs'],
               ['Send emails', 'Make presentation']];

const shoppingList = []; /* creates an empty array */
```

- Arrays are surrounded by square brackets.
- Individual pieces of data (known as **array elements**) are separated by commas (no comma after the last element in the array).

GETTING AT STUFF IN AN ARRAY

```
const fruits = ["🍏", "🍊", "🍋", "🍇", "🍓", "🍑"];
                 0    1    2    3    4    5

console.log( fruits[4] ); // returns 🍓
console.log( fruits[0] ); // returns 🍏
```

- Array elements are **indexed** meaning they are assigned a number starting with 0.
- Individual elements are accessed with the name of the variable followed by the number of the element inside square brackets.

TRY SOME OTHERS

```
let years = [1980, 1969, 2000, 2001, 2011, 2018];

let cities = ["Boston", "Paris", "London", "Frankfurt"];

let months = ["jan", "feb", "mar", "apr", "may", "jun"];
```

- 2000 == years[2]
- "Boston" == cities[0]
- months[1] == "feb"

SETTING ARRAY ELEMENTS

We can set values in an array the same way as we access them to retrieve values.

```
let fruits = ["🍏", "🍊", "🍋"];

fruits[2] = "banana";

console.log(fruits); /* outputs:  ["🍏", "🍊", "banana"] */
```

EMPTY INDICES

Arrays can have "empty" indices.

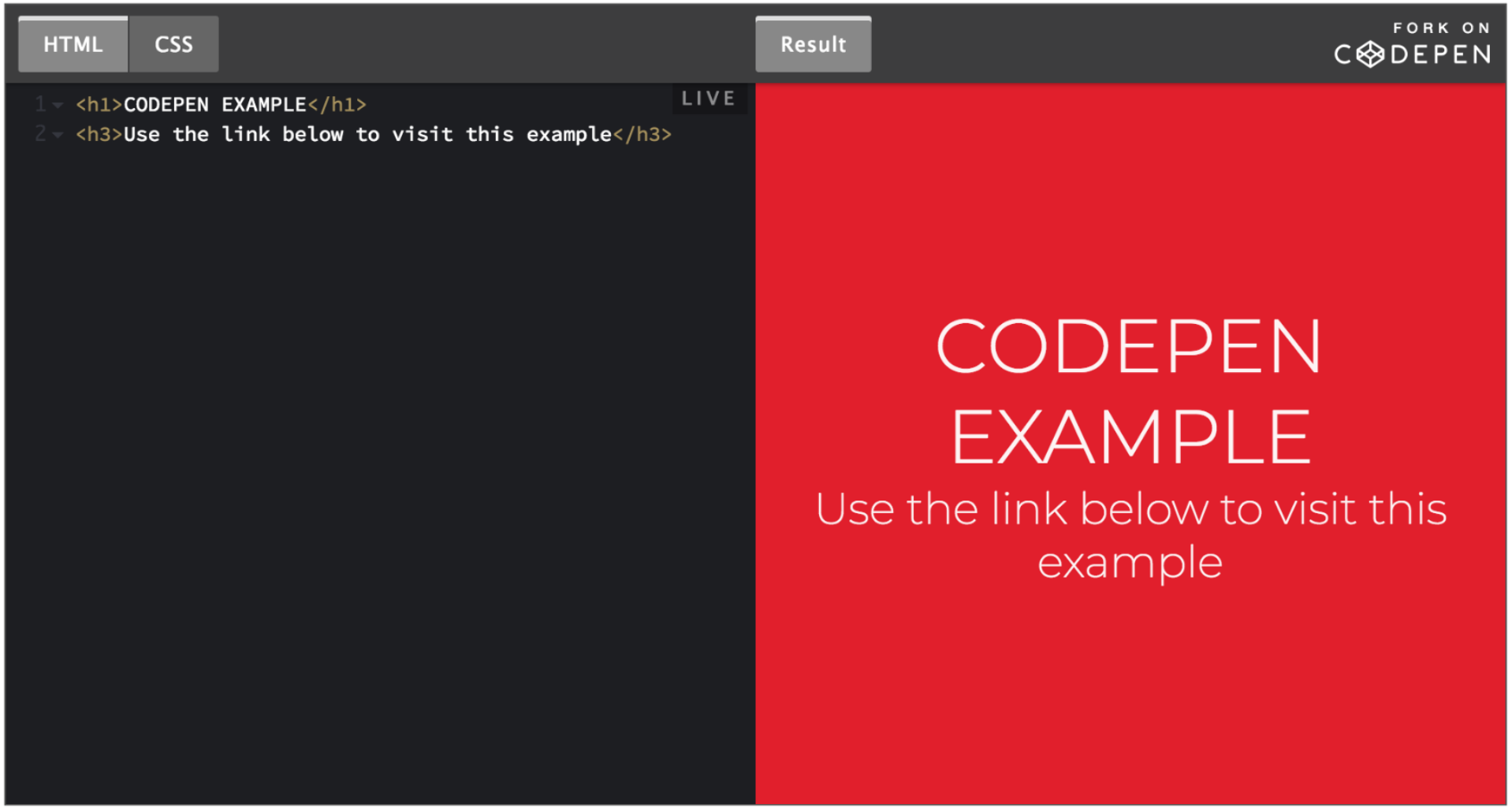
```
let fruits = ["🍏", "🍊", "🍋"];

fruits[4] = "🍌";

console.log(fruits); /* outputs:  ["🍏", "🍊", "🍋", empty, "🍌"] */

console.log(fruits[3]); /* outputs:  undefined */
```

LENGTH PROPERTY

To find out how many elements are in your array, you use the length property.

```
let fruits = ["🍏", "🍊", "🍋"];

console.log(fruits.length)  /* outputs:  3 */

fruits[4] = "🍌"; /* add banana in the 4th index */

console.log(fruits.length); /* outputs:  5 */
```

ARRAY PRACTICE



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

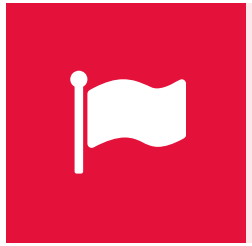
WHAT'S THE BIG DEAL

Arrays and objects are the cornerstones of nearly all programs in Javascript. Let's look at how powerful they can be with loops...

LOOPS

FOR LOOPS IN JAVASCRIPT

- Loops allow us to repeat a set of code statements a specific number of times
- Like if statements, loops are kind of control flow for our programs
- In Javascript there are several different kinds of loops, but we're going to focus on the **for-of** loop
- The for-of loop makes it super easy to **iterate** over an iterable object like an array



The for-of loop is **not** supported in <= Internet Explorer 11 (only IE Edge).

FOR-OF LOOP SYNTAX

```
const names = ['Markus','David','Kelly','Ann','Dina'];

for (let name of names) {
  console.log(name);
}
```

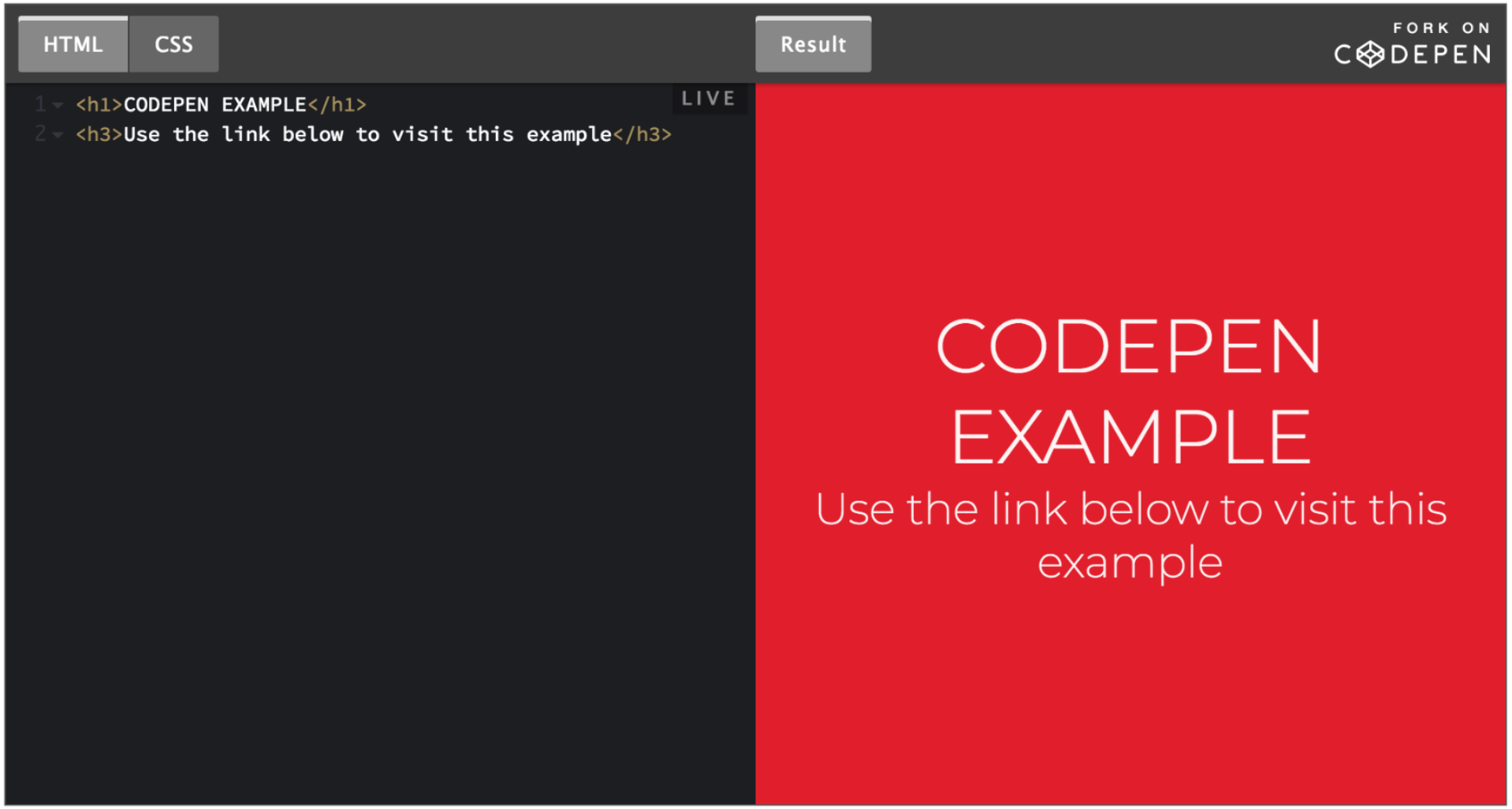
1. Use the **for** keyword followed by parentheses
2. Create a variable with **let** to store the current value of the array element.
3. Use the **of** keyword followed by an array to loop over.
4. The code inside the curly braces get run on each loop.

LOOPS & APPEND

```
const shoppingList = ['Coffee', 'Wine', 'Chocolate', 'Emergency Wine'];

for(let item of shoppingList) {

  $('ol').append(`<li>${item}</li>`);

}
```

- Together with the append method, loops can be used to construct a whole web page with Javascript alone.
- This concept is fundamental to popular Javascript frameworks like React and Angular

PHOTO GALLERY



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

PHOTO GALLERY SOLUTION

```
// Loop over the photoUrls array

// On each loop, store the current element in the url variable

for (let url of photoUrls) {

  //On each loop append a new img tag and set the src to the url variable

  $('#gallery').append(`<img src="${url}">`);

}
```

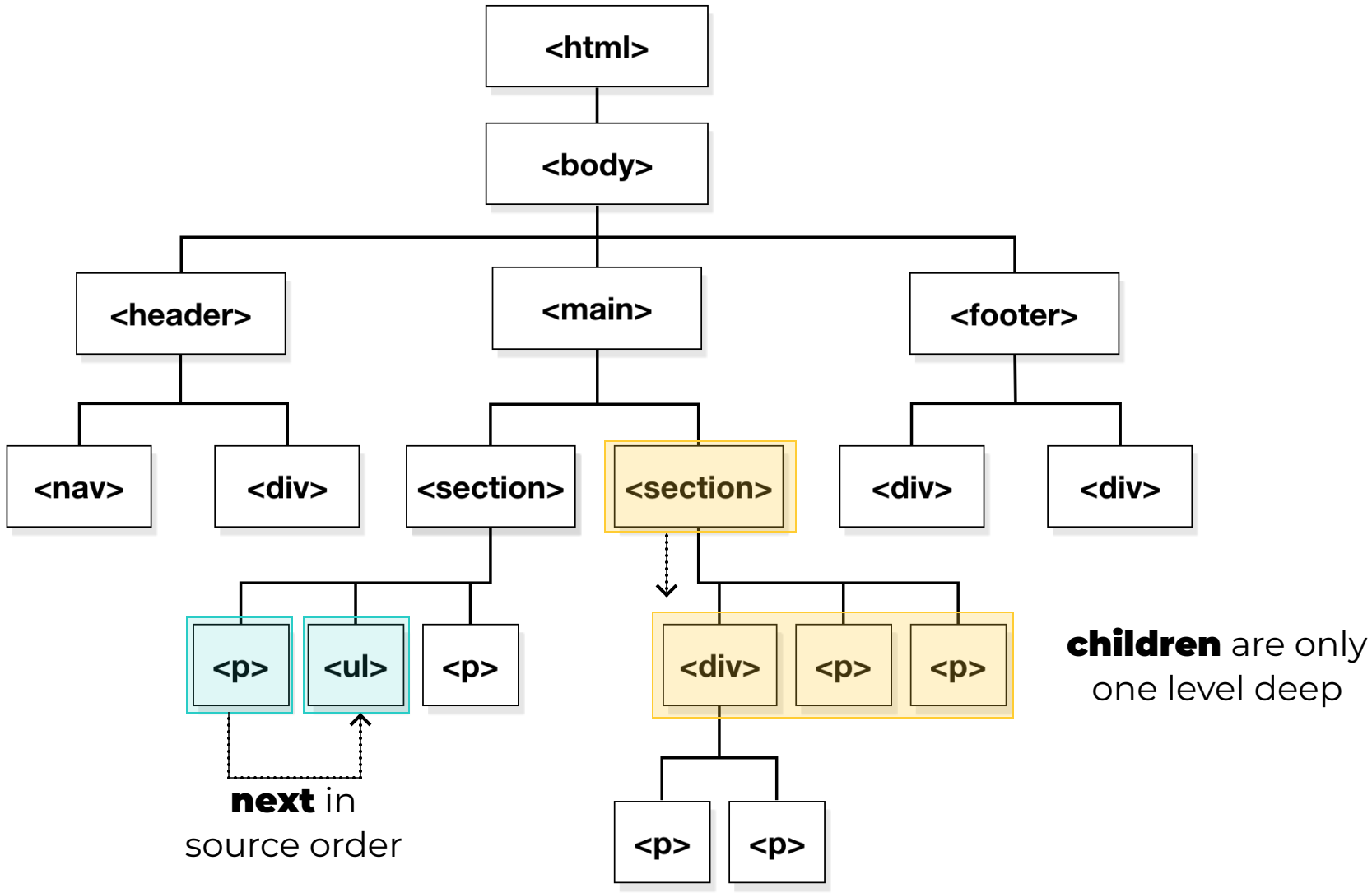
TRAVERSING THE DOM

MOVING AROUND THE DOM

- jQuery makes it easy for us to move between different elements on the page with its traversal method.
- We're going to take a look at several, but first let's look at the DOM again.

Document Object Model

DOM: A WEBPAGE FAMILY TREE



"WALKING THE DOM"

- Walking the DOM is a really valuable skill to learn since we can't reasonably add ids to every element we might want to manipulate on our page.
- Combining our knowledge of the `this` keyword with a couple of handy traversal methods, we can easily and efficiently target virtually any element on the page.
- Let's take a look at some traversal methods...

https://api.jquery.com/category/traversing/

NEXT AND PREV METHODS

```
// Get the next element sibling in the source order
// Optionally, get the next sibling that matches the filter

$('selector').next('optional-filter');
```

- The `.next()` and `.prev()` methods get the next or previous sibling in the source order
- The optional filter takes any selector (just like the ones we've been using with the `$()` method).
- If a filter selector is provided, the method will get the next or previous sibling **only** if it matches the filter.

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>jQuery</title>
  </head>
  <body>
    <div id="start">First div</div>
    <div>Second div</div>
    <div id="last">Third div</div>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      $('#start').next().css('background-color', 'red');
      /* This one doesn't work because the previous sibling
         doesn't match the selector in the optional filter: */
      $('#last').prev('#start').css('background-color', 'blue');
    </script>
  </body>
</html>
```

SIBLINGS METHOD

```
// Get all of the siblings in both directions in the source order
// Optionally, only get the siblings that match a filter selector

$('selector').siblings('optional-filter');
```

- The `.siblings()` method is handy when you want to get all of the sibling elements in both directions in the source order.
- Optionally, select only those siblings which match a selector provided as a filter.

The siblings method will not return the current element. It only returns the siblings of the current element!

PARENT VS. PARENTS METHODS

```
// Get the immediate parent in the source order
// Optionally only returning the parent if it matches the filter.

$('selector').parent('filter');
```

- The `.parent()` method gets only the immediate parent element.
- The `.parents()` method gets all of the ancestors of the current element.
- Both can be filtered by an optional selector, such that the result must match the supplied selector.

CHILDREN VS. FIND METHODS

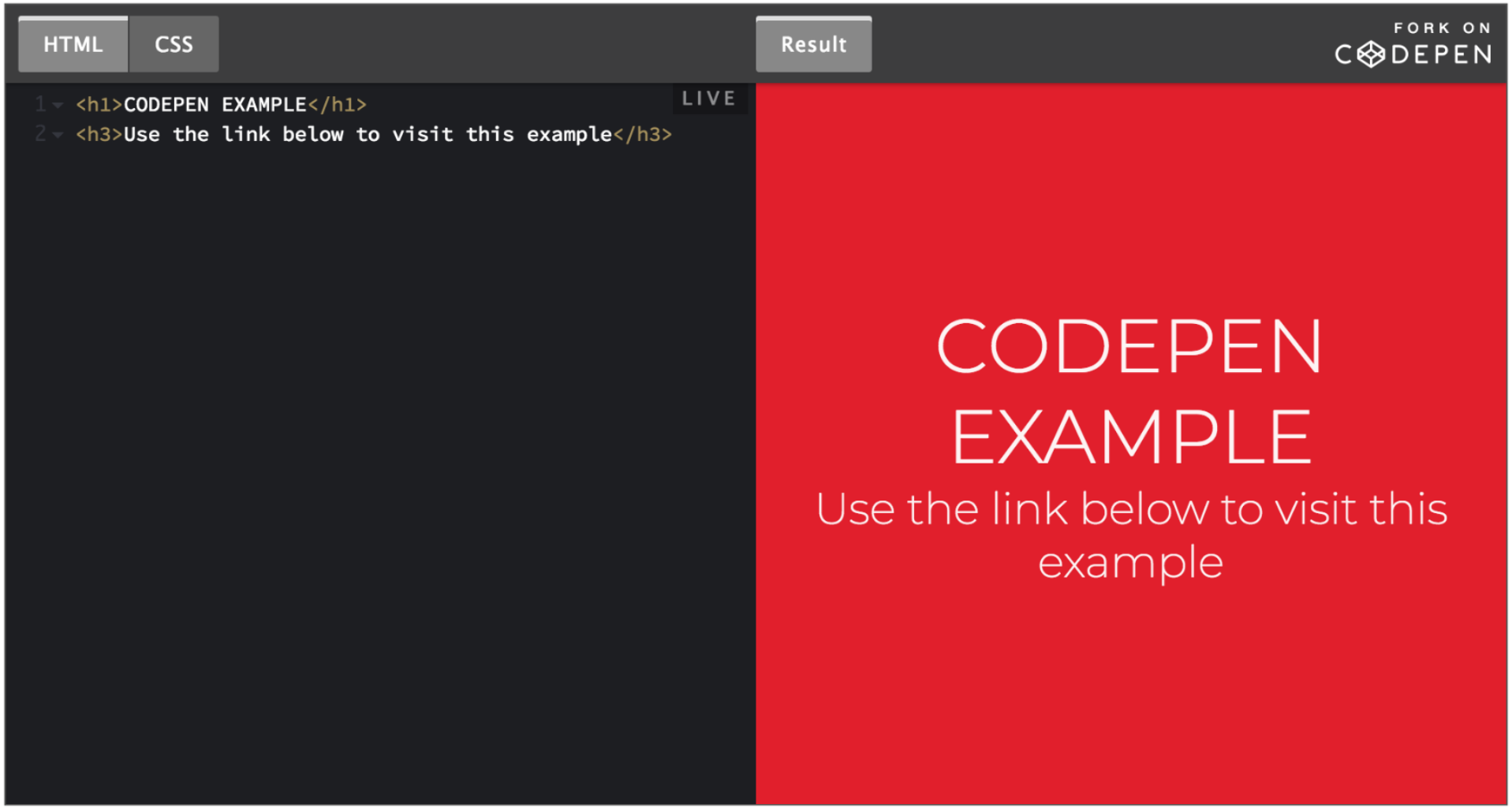
```
// Get the immediate parent in the source order
// Optionally only returning the parent if it matches the filter.

$('selector').parent('filter');
```

- The `.children()` method gets only the immediate children elements (one level down the tree).
- The `.find()` method gets all of the descendants of the current element.
- Both can be filtered by an optional selector, such that the result must match the supplied selector.

TRY IT!


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>jQuery</title>
  </head>
  <body>
    <div class="stop">First div</div>
    <div>Second div</div>
    <div>Third div</div>
    <div id="last">Fourth div</div>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script>
      // Makes the second and third divs red
      $('#last').prevUntil('.stop').css('background-color', 'red');
    </script>
  </body>
</html>
```

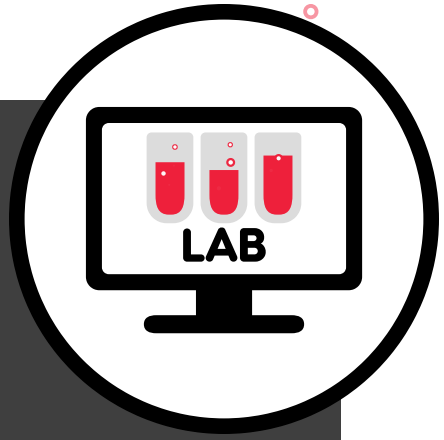
FAQS LAB



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

FAQS SOLUTION

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

  // Listen for clicks on elements with a class of question
  $('.question').click(function(){

    // Use this and next to target the corresponding answer
    $(this).next('.answer').slideToggle('fast');

  });

});
```



GO BUILD AWESOME THINGS!