1. Harmony, ES6, ES2015...

JS keeps evolving!

1. Let and Const

Up until now, the only way to declare a variable in JavaScript was to use the keyword var, but var can get us into trouble. Var is scoped either globally or hoisted to the top of the function. Let and const are block-specific!

* use let when you plan to reassign new values to a variable, and
* use const when you don’t plan on reassigning new values to a variable.

Try ditching var as a keyword.

1. Quiz: Using Let and Const (1-1)
2. Template Literals

Template literals are essentially string literals that include embedded expressions.

Denoted with backticks ( `` ) instead of single quotes ( '' ) or double quotes ( "" ), template literals can contain placeholders which are represented using ${expression}. This makes it much easier to build strings.

let message = `${student.name} please see ${teacher.name} in ${teacher.room} to pick up your report card.`;

Convert to Template Literal:

const myName = 'Brian';

const greeting = `Hello, my name is ${myName}.`;

console.log(greeting);

Template literals also preserve newlines as part of the string, no need for “/n”.

1. Quiz: Build an HTML Fragment (1-2)

const cheetah = {

name: 'Cheetah',

scientificName: 'Acinonyx jubatus',

lifespan: '10-12 years',

speed: '68-75 mph',

diet: 'carnivore',

summary: 'Fastest mammal on land, the cheetah can reach speeds of 60 or perhaps even 70 miles (97 or 113 kilometers) an hour over short distances. It usually chases its prey at only about half that speed, however. After a chase, a cheetah needs half an hour to catch its breath before it can eat.',

fact: 'Cheetahs have “tear marks” that run from the inside corners of their eyes down to the outside edges of their mouth.'

};

// creates an animal trading card

function createAnimalTradingCardHTML(animal) {

const cardHTML = `<div class="card">

<h3 class="name"> ${animal.name} </h3>

<img src="${animal.name}.jpg" alt="${animal.name}" class="picture">

<div class="description">

<p class="fact"> ${animal.fact} </p>

<ul class="details">

<li><span class="bold">Scientific Name</span>: ${animal.scientificName} </li>

<li><span class="bold">Average Lifespan</span>: ${animal.lifespan} </li>

<li><span class="bold">Average Speed</span>: ${animal.speed} </li>

<li><span class="bold">Diet</span>: ${animal.diet} </li>

</ul>

<p class="brief"> ${animal.summary} </p> +

</div>

</div>`;

return cardHTML;

}

console.log(createAnimalTradingCardHTML(cheetah));

1. Destructuring

In ES6, you can extract data from arrays and objects into distinct variables using destructuring.

1. Quiz: Destructuring Arrays (1-3)

/\*

\* Programming Quiz: Destructuring Arrays (1-3)

\*

\* Use destructuring to initialize the variables `one`, `two`, and `three`

\* with the colors from the `things` array.

\*/

const things = ['red', 'basketball', 'paperclip', 'green', 'computer', 'earth', 'udacity', 'blue', 'dogs'];

const [one,,, two,,,, three] = things;

const colors = `List of Colors

1. ${one}

2. ${two}

3. ${three}`;

console.log(colors);

1. Object Literal Shorthand
2. Lesson 1 Checkup
3. Iteration

ES6 has an iterable interface that allows JS object to define or customize their iteration behavior.

For…of loop need these new iteration interfaces.

1. Family of For Loops

Really the biggest downside of a for loop is having to keep track of the counter and exit condition. The for...in loop improves upon the weaknesses of the for loop by eliminating the counting logic and exit condition. You still have to deal with the issue of using an index to access the values of the array. For...in loops loop over all enumerable properties, this means if you add any additional properties to the array's prototype, then those properties will also appear in the loop. However, forEach() is actually an array method, so it can only be used exclusively with arrays. There is also no way to stop or break a forEach loop. If you need that type of behavior in your loop, you’ll have to use a basic for loop.

1. For...of Loop

The for...of loop is used to loop over any type of data that is iterable.

const digits = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];

for (const digit of digits) {

console.log(digit);

}

1. Quiz: Writing a For...of Loop (1-4)

const days = ['sunday', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday'];

// your code goes here

for (let day of days) {

console.log(day.charAt(0).toUpperCase() + day.slice(1));

}

1. Spread... Operator

The spread operator, written with three consecutive dots ( ... ), is new in ES6 and gives you the ability to expand, or spread, iterable objects into multiple elements.

const primes = new Set([2, 3, 5, 7, 11, 13, 17, 19, 23, 29]);

console.log(...primes);

const fruits = ["apples", "bananas", "pears"];

const vegetables = ["corn", "potatoes", "carrots"];

const produce = [...fruits, ...vegetables];

console.log(produce);

// [ 'apples', 'bananas', 'pears', 'corn', 'potatoes', 'carrots' ]

1. ...Rest Parameter

If you can use the spread operator to spread an array into multiple elements, then certainly there should be a way to bundle multiple elements back into an array, right?

const order = [20.17, 18.67, 1.50, "cheese", "eggs", "milk", "bread"];

const [total, subtotal, tax, ...items] = order;

console.log(total, subtotal, tax, items);

Prints: 20.17 18.67 1.5 ["cheese", "eggs", "milk", "bread"]

1. Quiz: Using the Rest Parameter (1-5)

function average(...numbers) {

let length = numbers.length;

if (length < 1) {

return 0;

}

let sum = 0;

for (let number of numbers) {

sum += number;

}

return sum / length;

}

1. Lesson 1 Summary

Yay we made progress!