**Defaults and destructuring arrays**

You can combine default function parameters with [destructuring](https://classroom.udacity.com/courses/ud356/lessons/42383e89-ac6a-491a-b7d0-198851287bbe/concepts/7c7be588-31e9-4c62-9dad-ecf23f943b19" \t "_blank) to create some pretty powerful functions!

**function** **createGrid**([width = 5, height = 5]) {

**return** `Generates a ${width} x ${height} grid`;

}

createGrid([]); *// Generates a 5 x 5 grid*

createGrid([2]); *// Generates a 2 x 5 grid*

createGrid([2, 3]); *// Generates a 2 x 3 grid*

createGrid([undefined, 3]); *// Generates a 5 x 3 grid*

***Returns:*** *Generates a 5 x 5 grid  
Generates a 2 x 5 grid  
Generates a 2 x 3 grid  
Generates a 5 x 3 grid*

The createGrid() function expects an array to be passed to it. It uses destructuring to set the first item in the array to the width and the second item to be the height. If the array is empty or if it has only one item in it, then the default parameters kick in and give the missing parameters a default value of 5.

There is a problem with this though, the following code will not work:

createGrid(); *// throws an error*

***Uncaught TypeError:****Cannot read property 'Symbol(Symbol.iterator)' of undefined*

This throws an error because createGrid() expects an array to be passed in that it will then destructure. Since the function was called without passing an array, it breaks. But, we can use default function parameters for this!

**function** **createGrid**([width = 5, height = 5] = []) {

**return** `Generates a ${width} x ${height} grid`;

}

See that new = [] in the function's parameter? If createGrid() is called without any argument then it will use this default empty array. And since the array is empty, there's nothing to destructure into width and height, so their default values will apply! So by adding = [] to give the entire parameter a default, the following code will now work:

createGrid(); *// Generates a 5 x 5 grid*

***Returns:****Generates a 5 x 5 grid*

**QUESTION 1 OF 2**

Take a look at the following code:

**function** **houseDescriptor**([houseColor = 'green', shutterColors = ['red']]) {

**return** `I have a ${houseColor} house with ${shutterColors.join(' and ')} shutters`;

}

Which of the following choices will run without throwing an error?

* 

houseDescriptor('red', ['white', 'gray', 'pink']);

* houseDescriptor(['green', ['white', 'gray', 'pink']]);
* 

houseDescriptor(['blue', 'purple']);

* houseDescriptor(['green']);

SUBMIT

**Defaults and destructuring objects**

Just like array destructuring with array defaults, a function can have an object be a default parameter and use object destructuring:

**function** **createSundae**({scoops = 1, toppings = ['Hot Fudge']}) {

**const** scoopText = scoops === 1 ? 'scoop' : 'scoops';

**return** `Your sundae has ${scoops} ${scoopText} with ${toppings.join(' and ')} toppings.`;

}

createSundae({}); *// Your sundae has 1 scoop with Hot Fudge toppings.*

createSundae({scoops: 2}); *// Your sundae has 2 scoops with Hot Fudge toppings.*

createSundae({scoops: 2, toppings: ['Sprinkles']}); *// Your sundae has 2 scoops with Sprinkles toppings.*

createSundae({toppings: ['Cookie Dough']}); *// Your sundae has 1 scoop with Cookie Dough toppings.*

***Returns:*** *Your sundae has 1 scoop with Hot Fudge toppings.  
Your sundae has 2 scoops with Hot Fudge toppings.  
Your sundae has 2 scoops with Sprinkles toppings.  
Your sundae has 1 scoop with Cookie Dough toppings.*

Just like the array example before, if you try calling the function without any arguments it won't work:

createSundae(); *// throws an error*

***Uncaught TypeError:****Cannot match against 'undefined' or 'null'.*

We can prevent this issue by providing a default object to the function:

**function** **createSundae**({scoops = 1, toppings = ['Hot Fudge']} = {}) {

**const** scoopText = scoops === 1 ? 'scoop' : 'scoops';

**return** `Your sundae has ${scoops} ${scoopText} with ${toppings.join(' and ')} toppings.`;

}

By adding an empty object as the default parameter in case no arguments are provided, calling the function without any arguments now works.

createSundae(); *// Your sundae has 1 scoop with Hot Fudge toppings.*

***Returns:****Your sundae has 1 scoop with Hot Fudge toppings.*

**QUESTION 2 OF 2**

Take a look at the following code:

**function** **houseDescriptor**({houseColor = 'green', shutterColors = ['red']} = {}) {

**return** `I have a ${houseColor} house with ${shutterColors.join(' and ')} shutters`;

}

Which of the following choices will run without throwing an error?

* houseDescriptor({houseColor: 'red', shutterColors: ['white', 'gray', 'pink']});
* houseDescriptor({houseColor: 'red'});
* houseDescriptor();
* houseDescriptor({shutterColors: ['orange', 'blue']});
* houseDescriptor({});

SUBMIT

**Array defaults vs. object defaults**

Default function parameters are a simple addition, but it makes our lives so much easier! One benefit of object defaults over array defaults is how they handle skipped options. Check this out:

**function** **createSundae**({scoops = 1, toppings = ['Hot Fudge']} = {}) { … }

...with the createSundae() function using object defaults with destructuring, if you want to use the default value for scoops but change the toppings, then all you need to do is pass in an object with toppings:

createSundae({toppings: ['Hot Fudge', 'Sprinkles', 'Caramel']});

Compare the above example with the same function that uses array defaults with destructuring.

**function** **createSundae**([scoops = 1, toppings = ['Hot Fudge']] = []) { … }

With this function setup, if you want to use the default number of scoops but change the toppings, you'd have to call your function a little...oddly:

createSundae([undefined, ['Hot Fudge', 'Sprinkles', 'Caramel']]);

Since arrays are positionally based, we have to pass undefined to "skip" over the first argument (and accept the default) to get to the second argument.

Unless you've got a strong reason to use array defaults with array destructuring, we recommend going with object defaults with object destructuring!