Rev. 25Jan2019

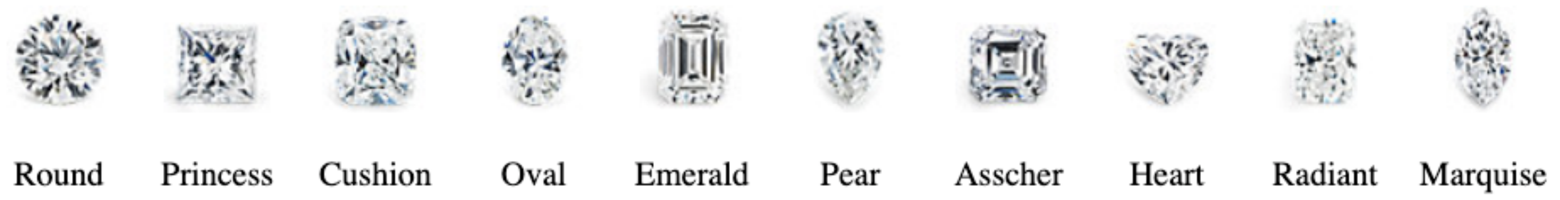
Exercise 33

DIAMOND SHAPES

Under normal circumstances, if I thought you might have been discouraged, I'd try to help you overcome that feeling at the end of the exercise. This time, it's different...

This is an *exceptionally* hard exercise. I will genuinely be amazed if you are able to complete it successfully without heavy reliance on `answer.js`. So, why am I asking you to undertake what will likely be a very frustrating task? Because sometimes, you just have to push yourself well past the point of comfort to uncover what you do know — and what you have yet to learn.

So with your promise that you won't give in to despondency, here's what we're aiming for:



I've provided the CSS for you. For the code, we're going to go through this *very* slowly.

Open `index.js`. You'll see that I've provided you with a variable called `diamonds`. Examine it. It is...

a. a JavaScript keyless object
b. an array, each member of which is a JavaScript object
c. a multi-level nested object
d. an array, each member of which is a nested array

When you've made your selection, go on to the next page.

The answer is:

b. an array, each member of which is a JavaScript object

You can tell `diamonds` is an array, as it starts out with opening and closing square brackets: `[]`

We expect a comma-delimited set of...something. What is that something? Here's the first one:

```
{
  name: 'Round',
  image: 'https://bnsec.bluenile.com/bluenile/is/image/bluenile/
    LD_ROUND_HERO_A_042318?$MODRN_DiaBar_slider_all$&rgn=1600,350,0,1500'
},
```

The value for image is a URL that points to an image on the web of that particular diamond shape

Opening and closing curly braces with key:value pairs? There's no mistaking that: it's an object — an object with two keys: `name` and `image`.

So, we see that `diamonds` is an array of objects.

We're going to call `forEach` on the `diamonds` array. In the last exercise, we learned that we must pass a function that we write into `forEach` — and that this function will be called once for each member of the array — and that each time it's called, it will be passed successive members of the array.

So, on the first pass, our passed-in function will be passed this:

```
{
  name: 'Round',
  image: 'https://bnsec.bluenile.com/bluenile/is/image/bluenile/
    LD_ROUND_HERO_A_042318?$MODRN_DiaBar_slider_all$&rgn=1600,350,0,1500'
}
```

The second pass will get the second object in the array — and so on, until all array members have been processed.

Here's a little trick I recommend: when you're setting out on some complex code (especially that you don't have experience with), *simplify* things as much as possible. How can we do that? By creating a very simple function to pass into `diamonds.forEach`.

The actual function we write is going to be more complex, so let's set that aside for the time being. Let's write a simple function that will write to the console. Let's call the function `printDiamondShape`. We can write the skeleton for this right now:

```
let printDiamondShape = (???) => {
  // function body here
}
```

How many times will this function be called?

a. 0
b. 4
c. 7
d. 10

The answer is **d. 10** (because there are 10 elements in the `diamonds` array)

And what is being passed in as an argument to `printDiamondShape`?

a. the diamonds array
b. each successive object in the array
c. two keys: name and image
d. the DOM

The answer is **b. each successive object in the array**

If this wasn't clear to you, review this exercise guide and, if need be, the one for Exercise 32. It's. important you understand that you're being passed one array member at a time — and, in this case, the array member is an object.

Once you understand that, you can fill in a little more of the skeleton:

```
let printDiamondShape = (diamondShapeObject) => {
  // function body here
}
```

So, we're being passed in an object that has this structure:

```
{
  name: 'Diamond Shape',
  image: 'image-url'
}
```

We know that we can access the *value* of a key:value pair by using this pattern: `objectName.key`.

But what is the `objectName`? It doesn't seem to have one! But actually it does — it's the name we provided as the argument into our `printDiamondShape` function: `diamondShapeObject`.

So, if we want to output the name, we would use this for our `printDiamondShape` function body:

```
console.log(diamondShapeObject.name)
```

The first pass through the function, `diamondShapeObject` references the first object in the array; the second pass through, `diamondShapeObject` refers to the *second* object in the array — and so forth until each object in the `diamonds` array is processed by our `printDiamondShape` function.

Now we can complete `printDiamondShape`:

```
let printDiamondShape = (diamondShapeObject) => {
  console.log(diamondShapeObject.name)
}
```

- ☐ Now, it's your turn: in `index.js`, write the `printDiamondShape` function, then call `diamonds.forEach`, passing in the `printDiamondShape` function.

- ☐ Run the repl and check the console tab.

INTERMISSION

If you got that working (and you understood it), major congratulations are in order. But we have more: we need to create HTML elements that will display each diamond shape in the diamonds array on the web page. But we'll pick that up in the next exercise. For now, enjoy the feeling of accomplishment. 33 lessons ago, you knew almost nothing. Now, you've written some sophisticated code. You genuinely should be proud of yourself.