Chapter 16 – Exercise to acustom ourselves to ES6 syntax.

What does each of the following do? What new feature(s) does it use?

1.

**var** customer **= {** name: "Foo" **}**

**var** card **= {** amount: 7**,** product: "Bar"**,** unitprice: 42 **}**

**var** message **= `**Hello $**{**customer**.**name**},**

want to buy $**{**card**.**amount**}** $**{**card**.**product**}** **for**

a total **of** $**{**card**.**amount **\*** card**.**unitprice**}** bucks?**`**

**2.**

let z =1;
let x = 1;
function foo(x, y=3, z=5) { console.log(x, y, z);}
foo(2)
function bar(x, y=3) { console.log(x, y, z);}
bar(2);

3. const lowPrimes = [2, 3, 5, 7, 11];
 let [p1, ,p3] = lowPrimes; //Note the empty space on the left side & the square brackets
 console.log(p3, p1);
 [p1, p3] = [p3, p1] //Note the square brackets

 console.log(p3, p1);

4.
 oval = {Pres:'Biden', VP:'Harris'}
 let {Pres, VP} = oval; //Note use of braces
 console.log(Pres, VP);

5. let node = {
     type: "Identifier",
     name: "foo"
 },
 type = "Literal",
 name = 5;

 ({ type, name } = node); //Note the use of the parentheses
 //This example is from <https://learning.oreilly.com/library/view/understanding-ecmascript-6/9781492017509/xhtml/ch05.xhtml#ch05lev2sec01>

6. let person = {
 name: 'Christina',
 age: 25
}
let { name: firstName, age: yearsOld } = person;

### //This example is from Isaak's book [Get Programming with JavaScript Next](https://learning.oreilly.com/library/view/get-programming-with/9781617294204/) <https://learning.oreilly.com/library/view/get-programming-with/9781617294204/kindle_split_021.html#ch11>

7.

|  |  |
| --- | --- |
|  | function margin() { const left=1, right=2, top=3, bottom=4; return { left, right, top, bottom };}const { left, bottom } = margin();console.log(left, bottom); // 1 4 |

NOTEL More complex examples of nested destructurin*g* may be found at
<https://kula.blog/learnjs/destructuring_5/>  very clear example.

Other examples are at:
<https://stackoverflow.com/questions/54293147/destructuring-nested-objects-in-javascript-destructure-second-level-parent-and>

<https://stackoverflow.com/questions/38808980/destructure-an-object-parameter-two-levels>

Also Chapter 5 of Zakas' book *Understanding ECMAScript6*

<https://learning.oreilly.com/library/view/understanding-ecmascript-6/9781492017509/xhtml/ch05.xhtml#ch05lev2sec04>

8. //From the Zakas book – Lesson 15
function countKids(...allMyChildren) {
 return `You have ${allMyChildren.length} children!`;
}

countKids('Talan', 'Jonathan');

function family(spouse, ...kids) {
 return `You are married to ${spouse} with ${kids.length} kids`;
}

family('Christina', 'Talan', 'Jonathan');

//Note that the values are assigned to the parameters in the usual way and anything left over is put into …theArray which must be at the end.

Please also note that this syntax allows you to pass in an array of unspecified length.

9. //Also from Zakas book

{

 const originalProcess = imageDoctor.process; //step 1

 imageDoctor.process = function(...args) { //step 2

 console.log('imageDoctor processing', args); //step 3

 return originalProcess.apply(imageDoctor, args); //step 4

 }

}

If you are not sure, here is what the steps do:
1. First get a reference to the original method.

2 Define a new function that gathers all the arguments.

3 Inject your logging.

4 Return the result of the original function invoked with args

10.
 a = [1, 2];
 b = [10, 20];
 console.log(...a, ...b);

11. var str1 = "foo";
 var str2 = […str1]
 What is str2?

12. Zakas' comment : "Closely related to rest parameters is the spread operator. Whereas rest parameters allow you to specify that multiple independent arguments should be combined into an array, the spread operator allows you to specify an array that should be split and passed in as separate arguments to a function."

13. Some built in functions expect parameters which are **not** in an array. The spread operator will help here. Consider the example in <https://adrianmejia.com/overview-of-javascript-es6-features-a-k-a-ecmascript-6-and-es2015/#Rest-parameters>

 Math.max(2, 100, 6): //100
 Math.max([2, 100, 6]); //NaN

Now we can say
 Math.max(…[2, 100, 6]); //100

Note: Classic JS code would use apply() to convert the array into the listed parameters:
 Math.max.apply(Math, [2, 100, 6])