(Notes) Objects

Question 1: Delete keyword in Object
Explanation: The code uses the `delete` keyword to delete a property from an object, but since the property is immediately returned afterward, the delete operation has no effect, and the property still exists in the object.

Computed Properties
Explanation: Computed properties allow for dynamic property names in object literals, using expressions inside square brackets. This feature is useful for creating object properties based on variable values or expressions.

Looping in Object
Explanation: The `for...in` loop in JavaScript iterates over all enumerable properties of an object, allowing access to both keys and values, which can be useful for various operations involving objects.

Question 2: Output
Explanation: The code attempts to create an object with duplicate keys (`a`). In JavaScript, object keys are unique, so when a key is repeated, the last occurrence overwrites the earlier ones. Therefore, `console.log(obj);` will output:

{ a: 'three', b: 'two' }

Question 3: Create a function multiplyByTwo(obj)
Explanation: The function `multiplyByTwo(obj)` multiplies all numeric property values of the provided object `obj` by 2, using a `for...in` loop to iterate over the object's properties and check for numeric values.

Question 4: Output (Important)
Explanation: JavaScript automatically converts object keys to strings. In this code, both `b` and `c` are converted to `[object Object]` (string representation of an object), so `a[b]` and `a[c]` refer to the same property. The final value is **456**.

Question 5: JSON.Stringify and JSON.parse
Explanation: `JSON.stringify` converts a JavaScript object to a JSON string, while `JSON.parse` converts a JSON string back to a JavaScript object. These methods are commonly used for data serialization and deserialization.

Question 6: Output
Explanation: The spread operator (`...`) in an array literal spreads the characters of the string `"Lydia"` into individual array elements, resulting in:

['L', 'y', 'd', 'i', 'a']

Question 7: Output
Explanation: The spread operator (`...`) in object literals spreads the properties of one object into another, creating a new object with combined properties. The output will be:

{ name: 'Lydia', age: 21, admin: true }

Question 8: Output
Explanation: `JSON.stringify` can take an array of properties to include in the JSON string. In this case, only the properties `'level'` and `'health'` will be included in the output.

Question 9: Output
Explanation: The `perimeter` method uses an arrow function `(() =>)` that doesn't have its own `this` context, leading to `this.radius` being undefined and resulting in **NaN (Not a Number)** when called.

Destructuring in object
Explanation: Destructuring allows extracting specific properties from an object. In this code:

const { fullName: { firstName } } = user;

It extracts the `firstName` property from `user.fullName`.

Question 11: Output
Explanation: The function `getItems` has a syntax error because the rest parameter (`...args`) should be the last parameter in the function's parameter list. It will throw a **syntax error** when executed.

Question 12: Output
Explanation: Objects in JavaScript are assigned by reference, so changing `c.greeting` also changes `d.greeting` since `d` references the same object as `c`.

Question 13: Output
Explanation: In JavaScript, objects are compared by reference, so:

{a:1} == {a:1}
{a:1} === {a:1}

will both return **false** since they are separate objects in memory.

Question 14: Output
Explanation: Arrays and objects are reference types in JavaScript, so setting `person` to `null` does not affect the reference stored in `members`, resulting in the original object being logged.

Question 15: Output
Explanation: The `multiply` function doubles the `number` property of the passed object. When called without arguments, it uses a default object `{ ...value }`, which leads to doubling the value each time it's called.
