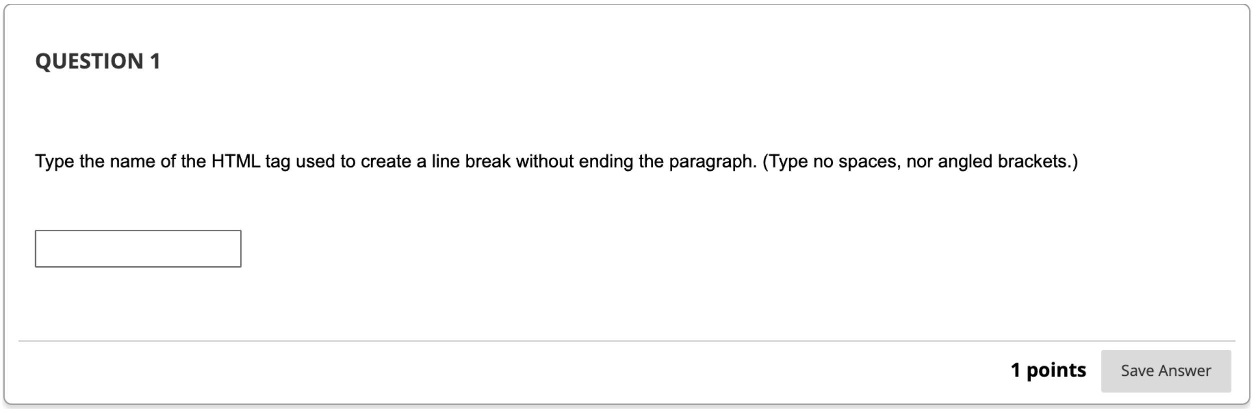
MSc Mock: ChatGBT answers:

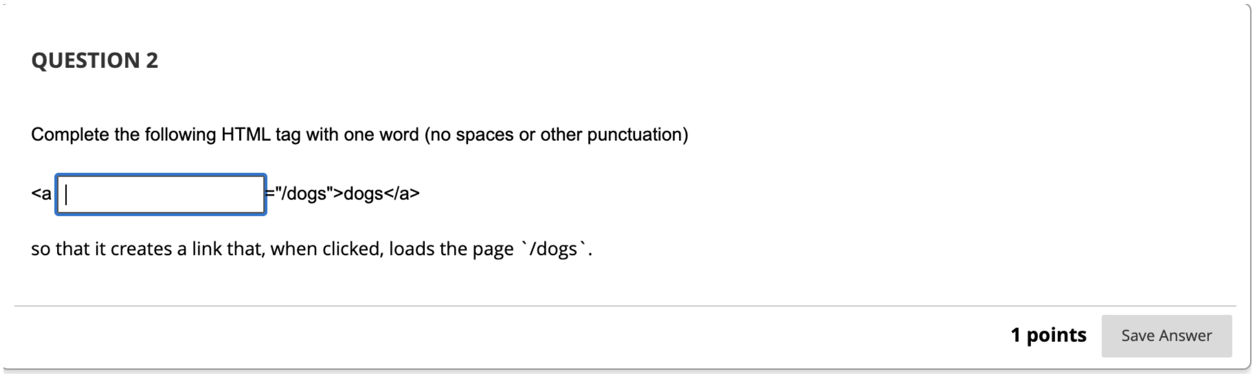


QUESTION 1

Type the name of the HTML tag used to create a line break without ending the paragraph. (Type no spaces, nor angled brackets.)

1 points

The HTML tag used to create a line break without ending the paragraph is "br".



QUESTION 2

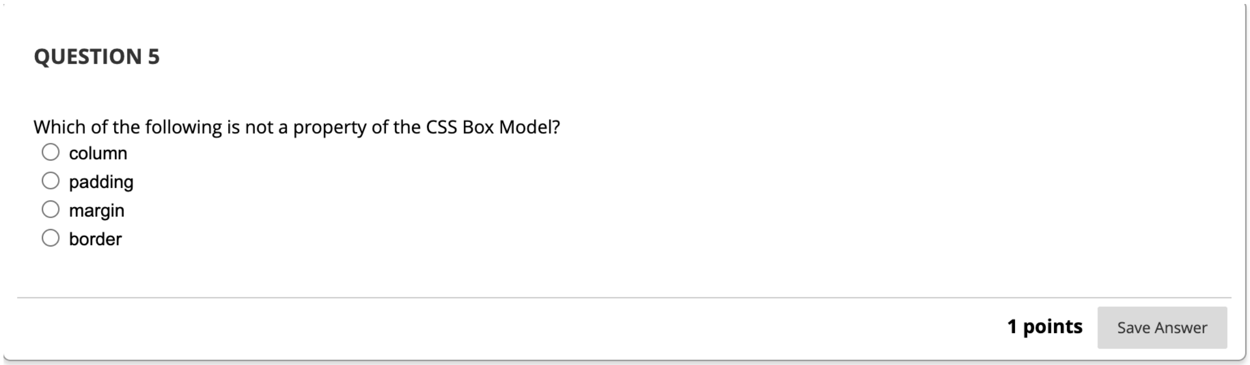
Complete the following HTML tag with one word (no spaces or other punctuation)

<a ="/dogs">dogs</a>

so that it creates a link that, when clicked, loads the page `/dogs`.

1 points

The word to complete the HTML tag is "href". This attribute specifies the URL of the page the link goes to.



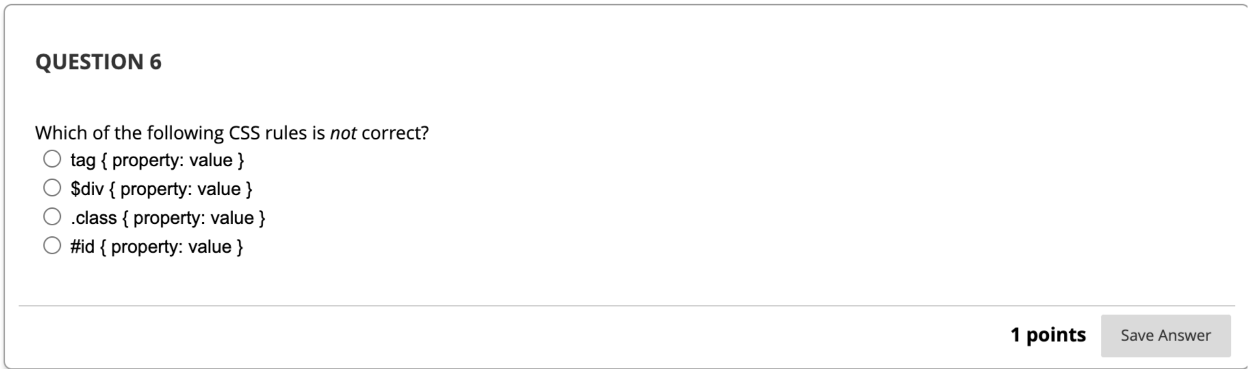
QUESTION 5

Which of the following is not a property of the CSS Box Model?

○ column
○ padding
○ margin
○ border

1 points Save Answer

The property that is not a part of the CSS Box Model is "column". The CSS Box Model consists of padding, margin, and border, but not "column".



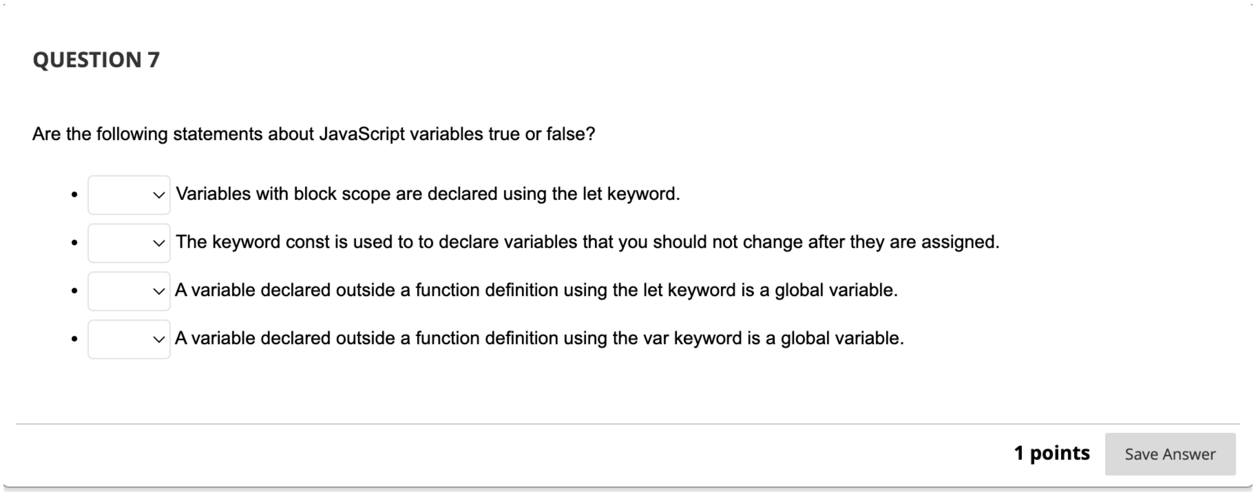
QUESTION 6

Which of the following CSS rules is *not* correct?

○ tag { property: value }
○ $div { property: value }
○ .class { property: value }
○ #id { property: value }

1 points Save Answer

The incorrect CSS rule among the options is "$div { property: value }". In CSS, selectors for elements are specified by their type (e.g., div, p) without any prefix, class selectors are prefixed with a period (.), and id selectors are prefixed with a hash (#). There is no "$" prefix used in standard CSS syntax.



QUESTION 7

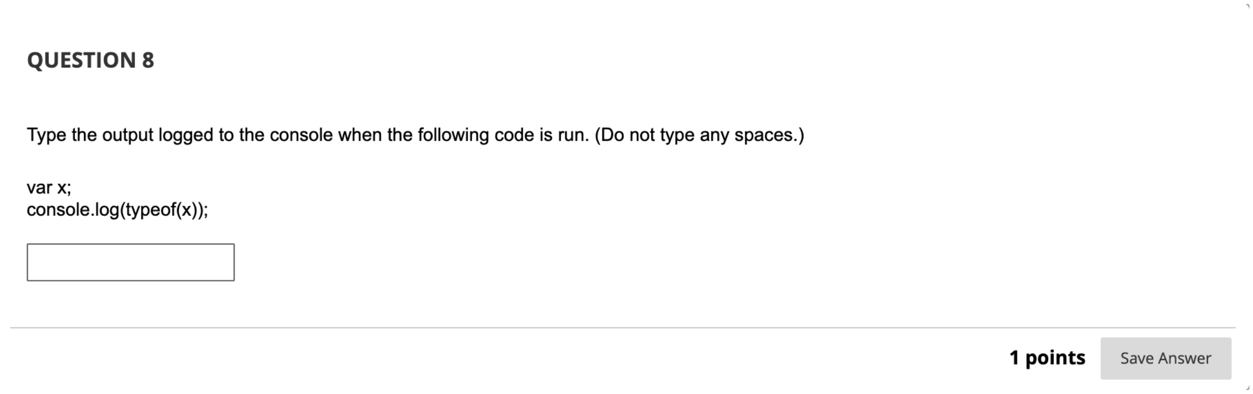
Are the following statements about JavaScript variables true or false?

- Variables with block scope are declared using the let keyword.
- The keyword const is used to to declare variables that you should not change after they are assigned.
- A variable declared outside a function definition using the let keyword is a global variable.
- A variable declared outside a function definition using the var keyword is a global variable.

The statement that is false in the list provided about JavaScript variables is:

- A variable declared outside a function definition using the `let` keyword is a global variable.

This statement is incorrect because a variable declared with the `let` keyword outside of a function, but within a script or a module, has a global scope only in that script or module, not universally as traditional global variables defined with `var`. If `let` is used in the global scope in a browser environment, it creates a script-scoped global variable, which is different from a traditional global variable defined with `var` that becomes a property of the global `window` object.



QUESTION 8

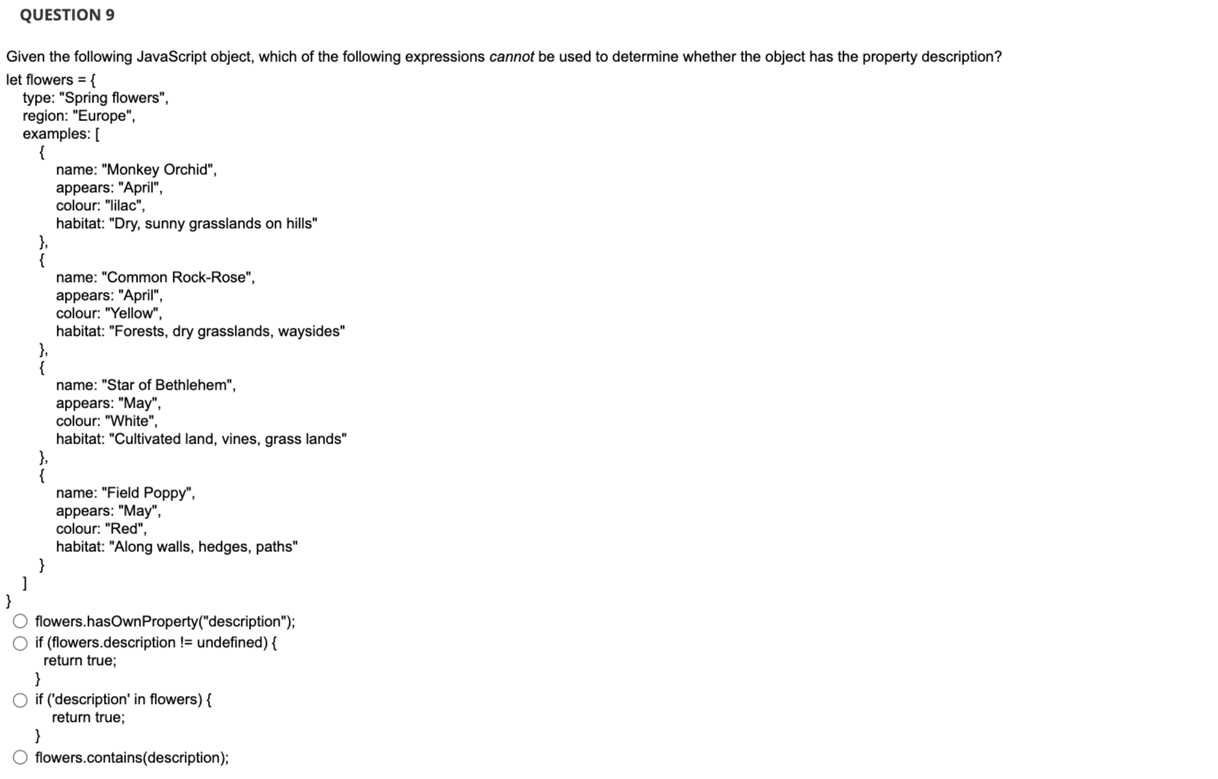
Type the output logged to the console when the following code is run. (Do not type any spaces.)

```
var x;
console.log(typeof(x));
```

When the JavaScript code provided is run:

```
javascriptCopy code
var x;
console.log(typeof(x));
```

The output logged to the console will be `undefined`. This is because the variable `x` is declared but not assigned any value, so its type is `undefined`.



QUESTION 9

Given the following JavaScript object, which of the following expressions *cannot* be used to determine whether the object has the property description?

```
let flowers = {
    type: "Spring flowers",
    region: "Europe",
    examples: [
        {
            name: "Monkey Orchid",
            appears: "April",
            colour: "lilac",
            habitat: "Dry, sunny grasslands on hills"
        },
        {
            name: "Common Rock-Rose",
            appears: "April",
            colour: "Yellow",
            habitat: "Forests, dry grasslands, waysides"
        },
        {
            name: "Star of Bethlehem",
            appears: "May",
            colour: "White",
            habitat: "Cultivated land, vines, grass lands"
        },
        {
            name: "Field Poppy",
            appears: "May",
            colour: "Red",
            habitat: "Along walls, hedges, paths"
        }
    ]
}
```

○ flowers.hasOwnProperty("description");

○ if (flowers.description != undefined) {
    return true;
  }

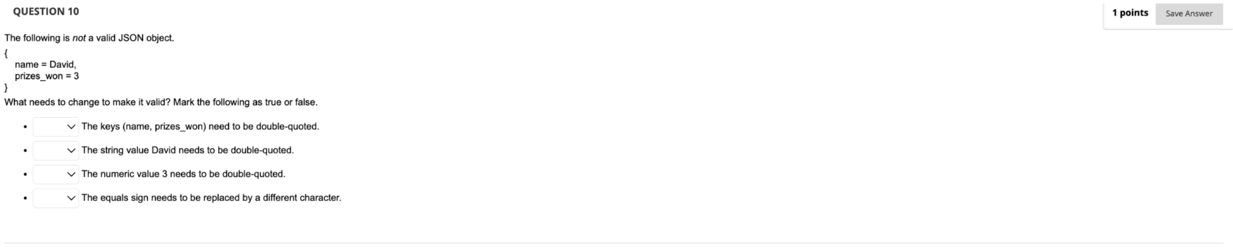
○ if ('description' in flowers) {
      return true;
  }

○ flowers.contains(description);

The expression that **cannot** be used to determine whether the JavaScript object `flowers` has the property `description` is:

- `flowers.contains(description);`

This expression is incorrect because the `contains` method does not exist for JavaScript objects. This method is typically used with strings or arrays, not object properties. The other expressions (`flowers.hasOwnProperty("description")`, `if (flowers.description != undefined)`, and `if ('description' in flowers)`) are all valid ways to check if the object has a property called `description`.



QUESTION 10

The following is *not* a valid JSON object.

```
{
   name = David,
   prizes_won = 3
}
```

What needs to change to make it valid? Mark the following as true or false.

- The keys (name, prizes_won) need to be double-quoted.
- The string value David needs to be double-quoted.
- The numeric value 3 needs to be double-quoted.
- The equals sign needs to be replaced by a different character.

In JSON syntax, it's important to format the keys and values correctly:

1. **True**: The keys (`name`, `prizes_won`) need to be double-quoted. JSON keys must always be in double quotes.
2. **True**: The string value `David` needs to be double-quoted. In JSON, string values must be enclosed in double quotes.
3. **False**: The numeric value `3` does not need to be double-quoted. In JSON, numbers are used as is, without quotes.
4. **True**: The equals sign needs to be replaced by a colon. In JSON, key-value pairs are separated by a colon, not an equals sign.

These changes will make the provided text a valid JSON object.