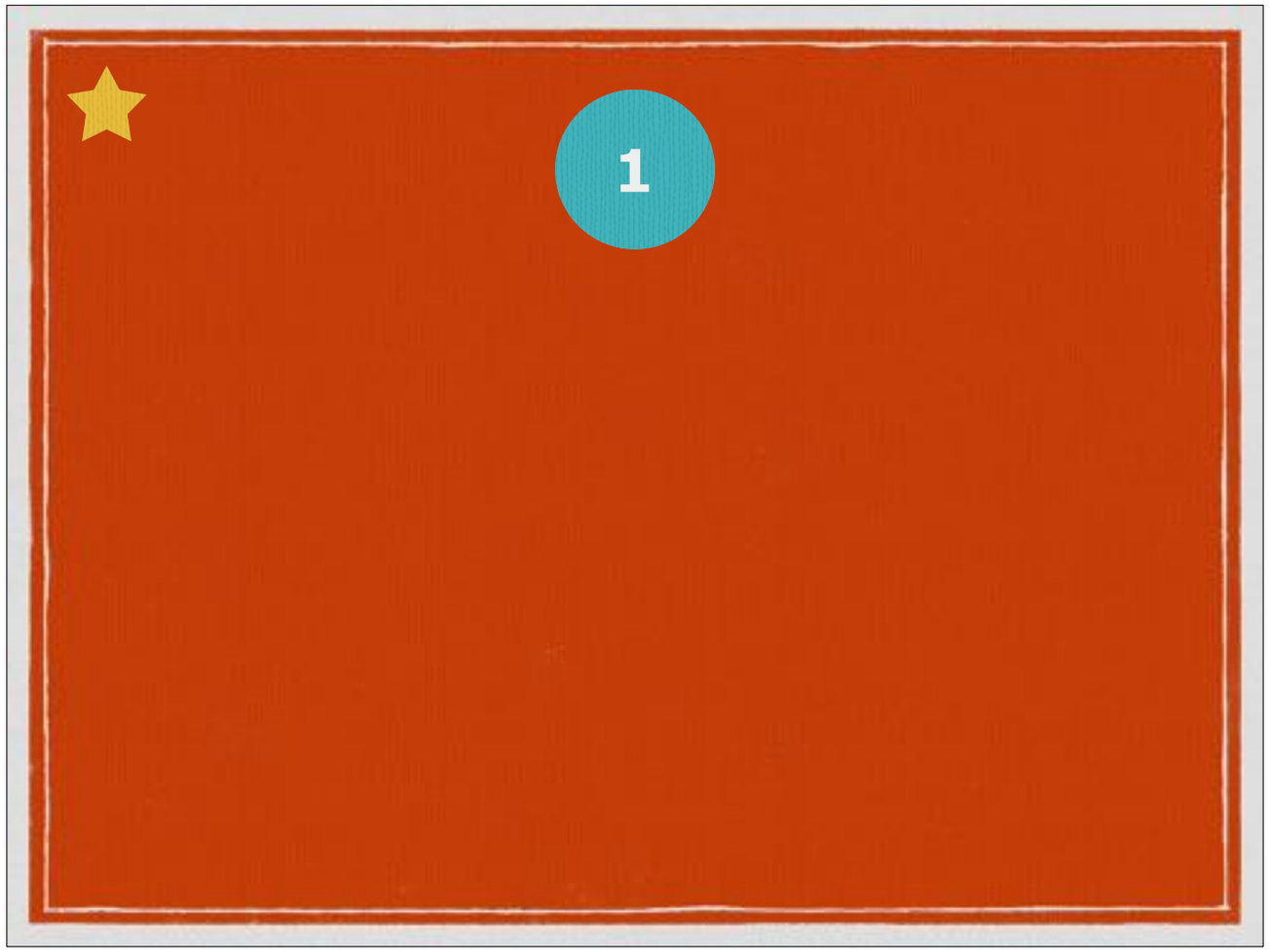
¡interfeud!

the game that gets YOU hired

Prep:
https://www.interviewcake.com/big-o-notation-time-and-space-complexity
https://justin.abrah.ms/computer-science/big-o-notation-explained.html
http://bigocheatsheet.com
https://gist.github.com/glebec/05c483c77f925ca8edef


1

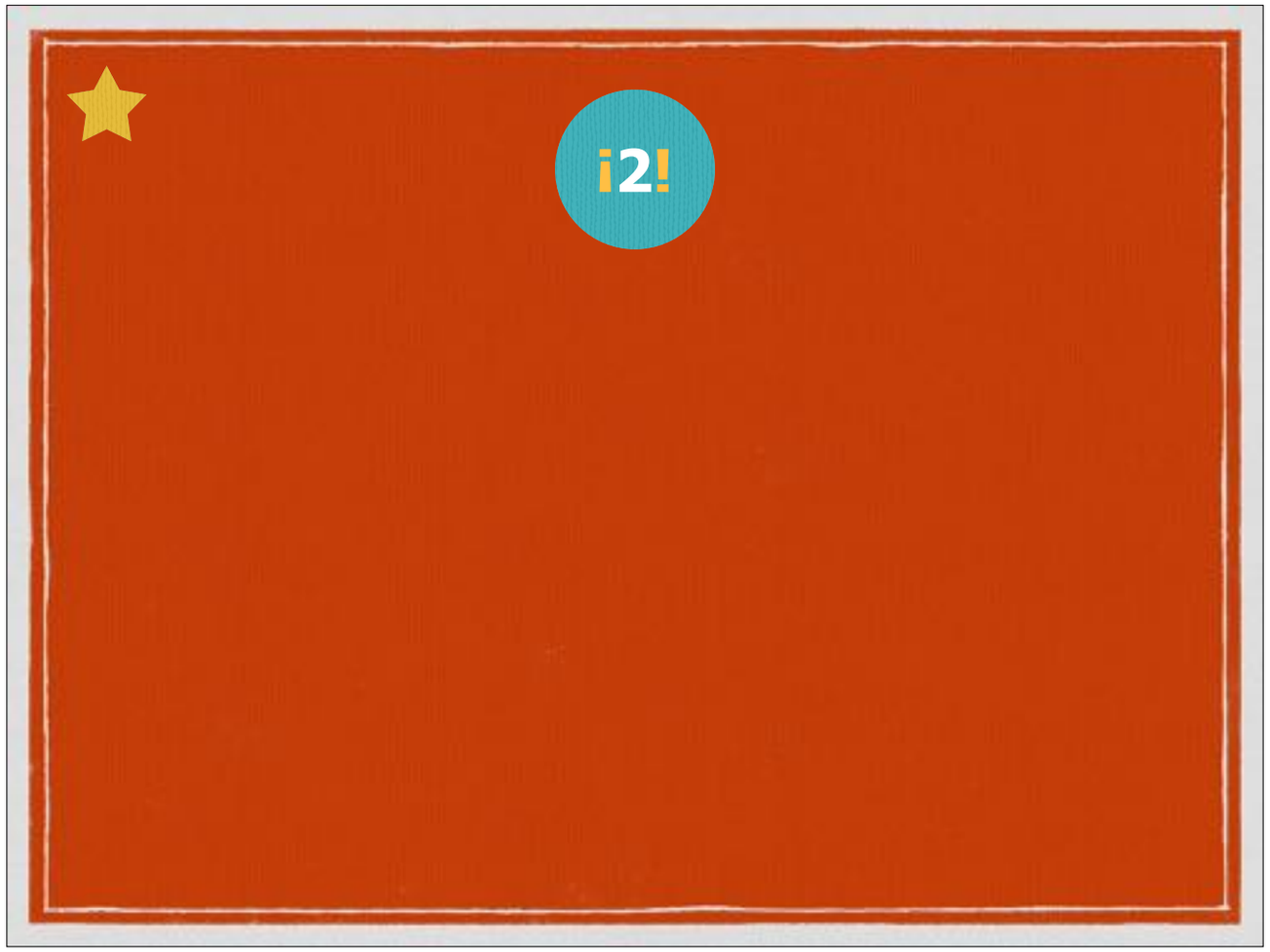
1

What two things do recursive functions need?

1

What two things do recursive functions need?

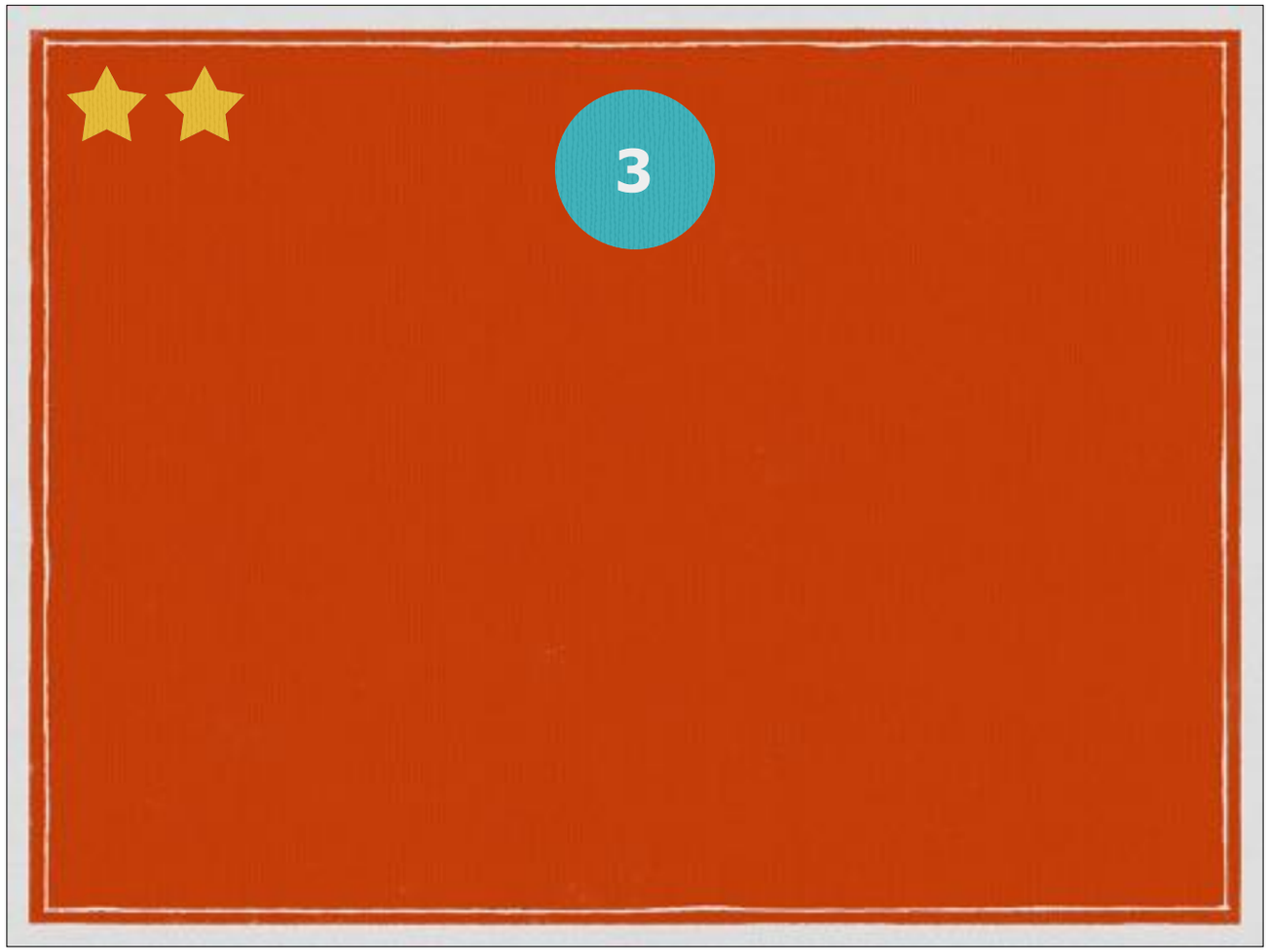
A base case and a recursive case.


¡2!

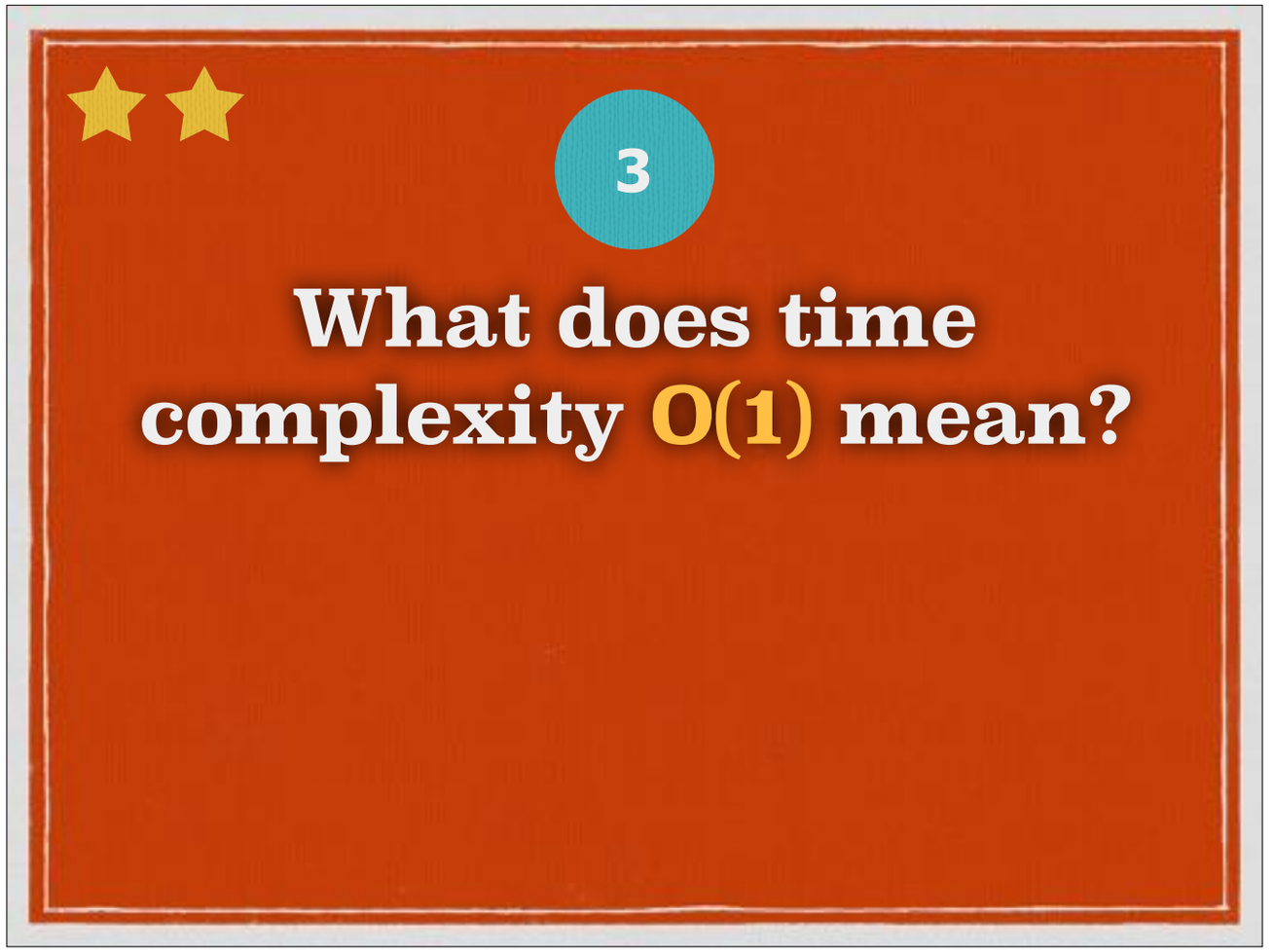
Apart from running (theoretically) infinitely, how might a recursive function blow the stack?

Apart from running (theoretically) infinitely, how might a recursive function blow the stack?

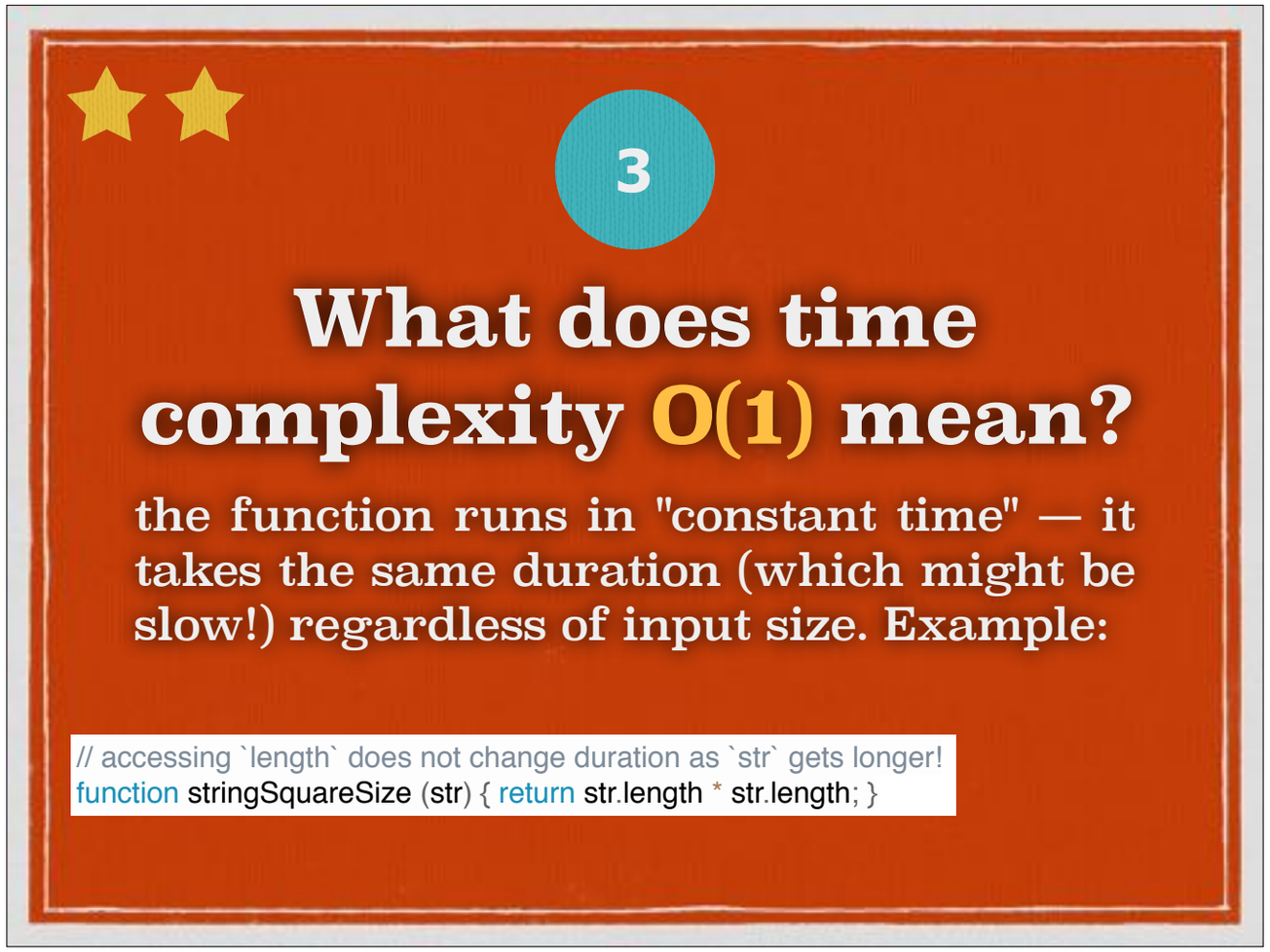
If recursive calls are made sufficiently more frequently than base cases are resolved, the stack can overflow.



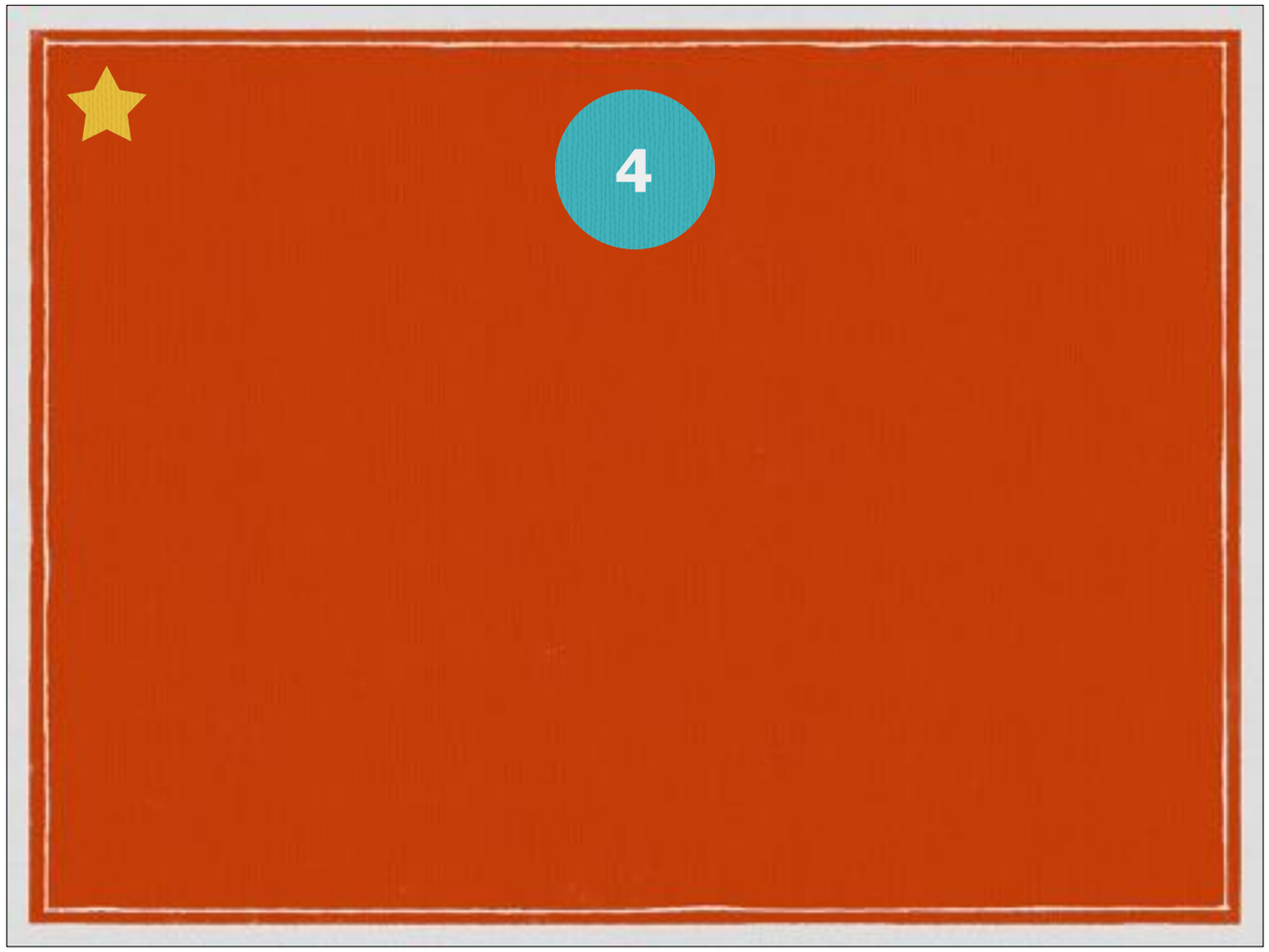
NOTE: I checked, this is true.



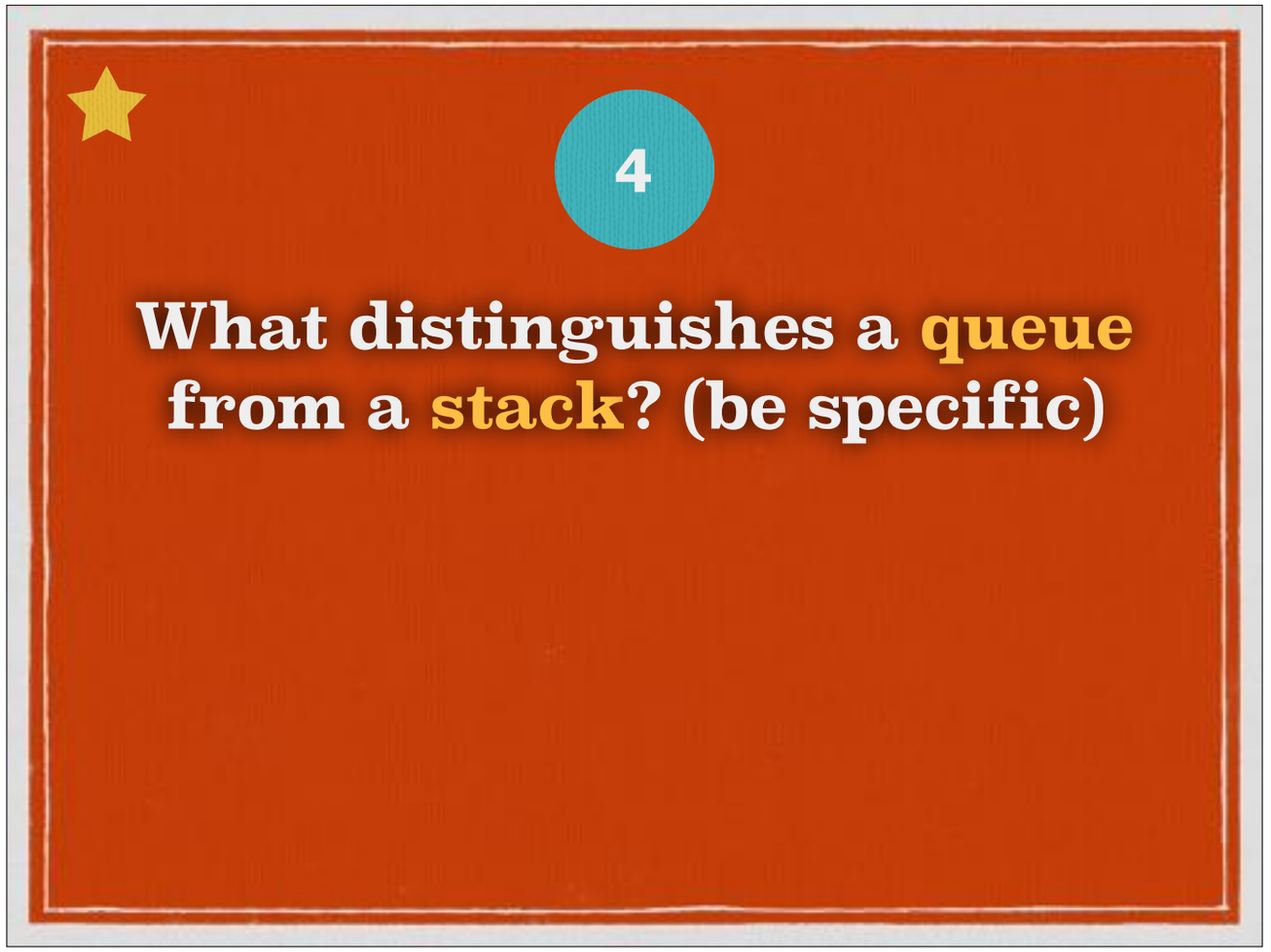
NOTE: I checked, this is true.



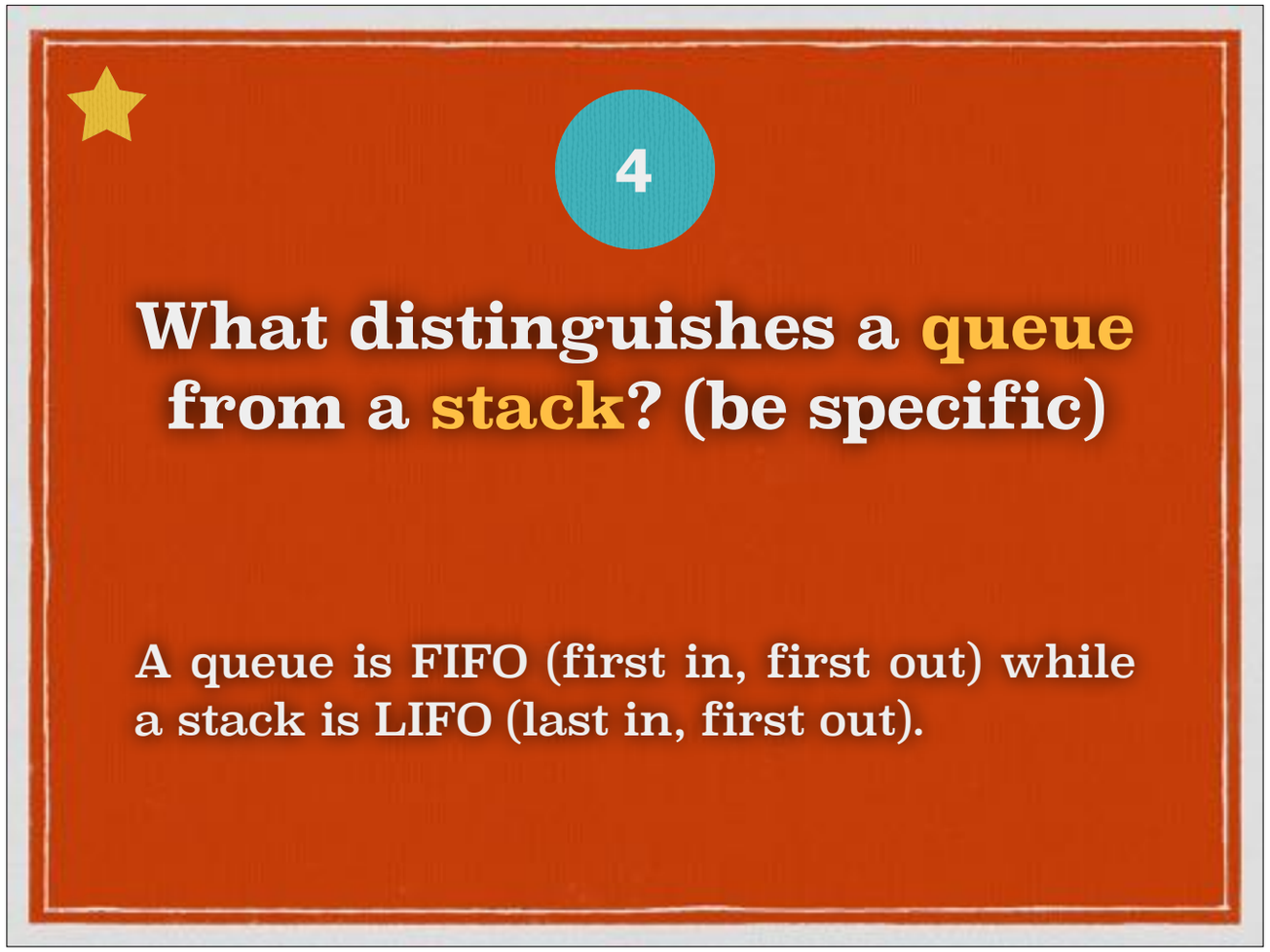
NOTE: I checked, this is true.



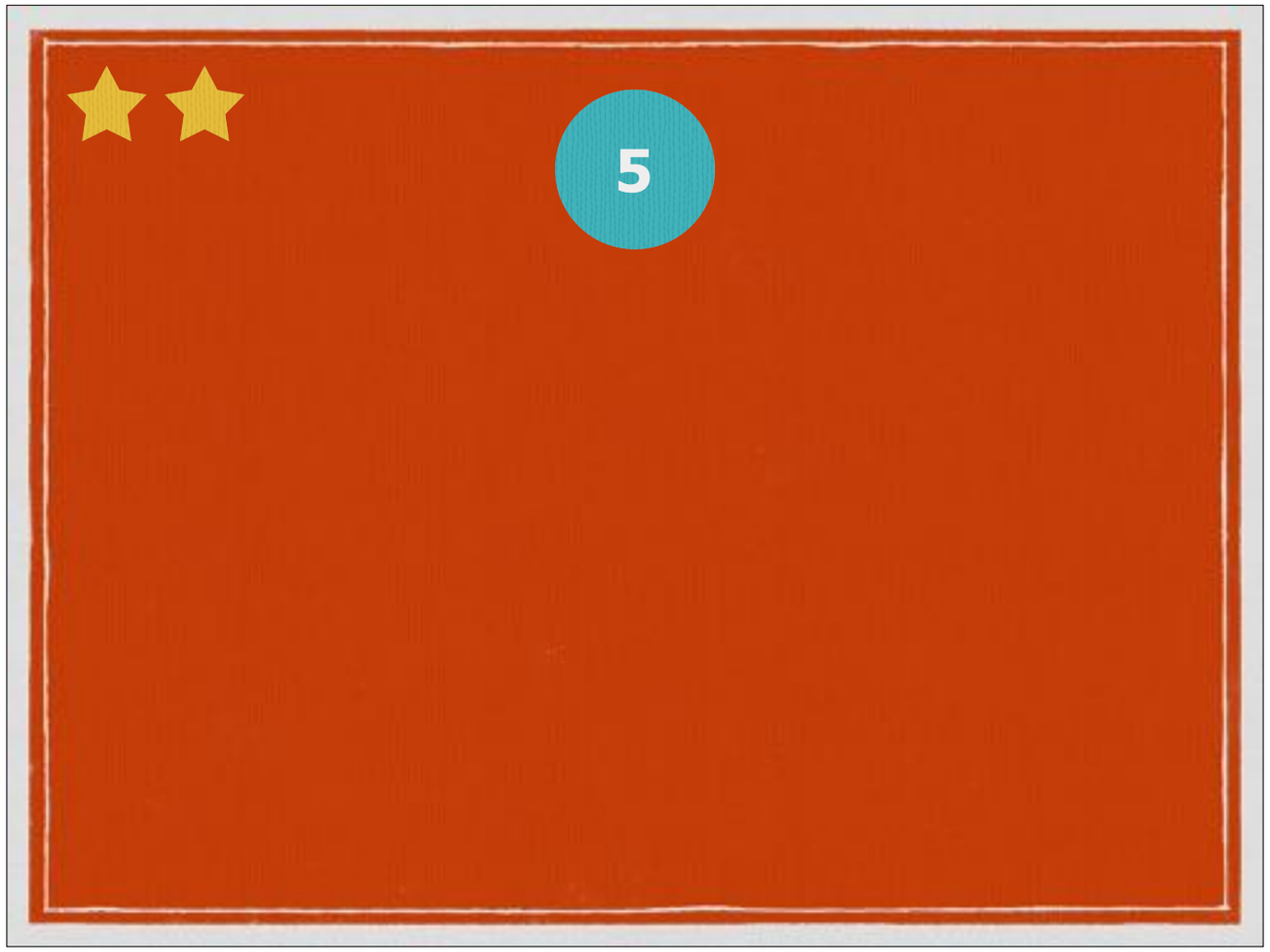
Rubric: have to say which is which.



Rubric: have to say which is which.



Rubric: have to say which is which.


5

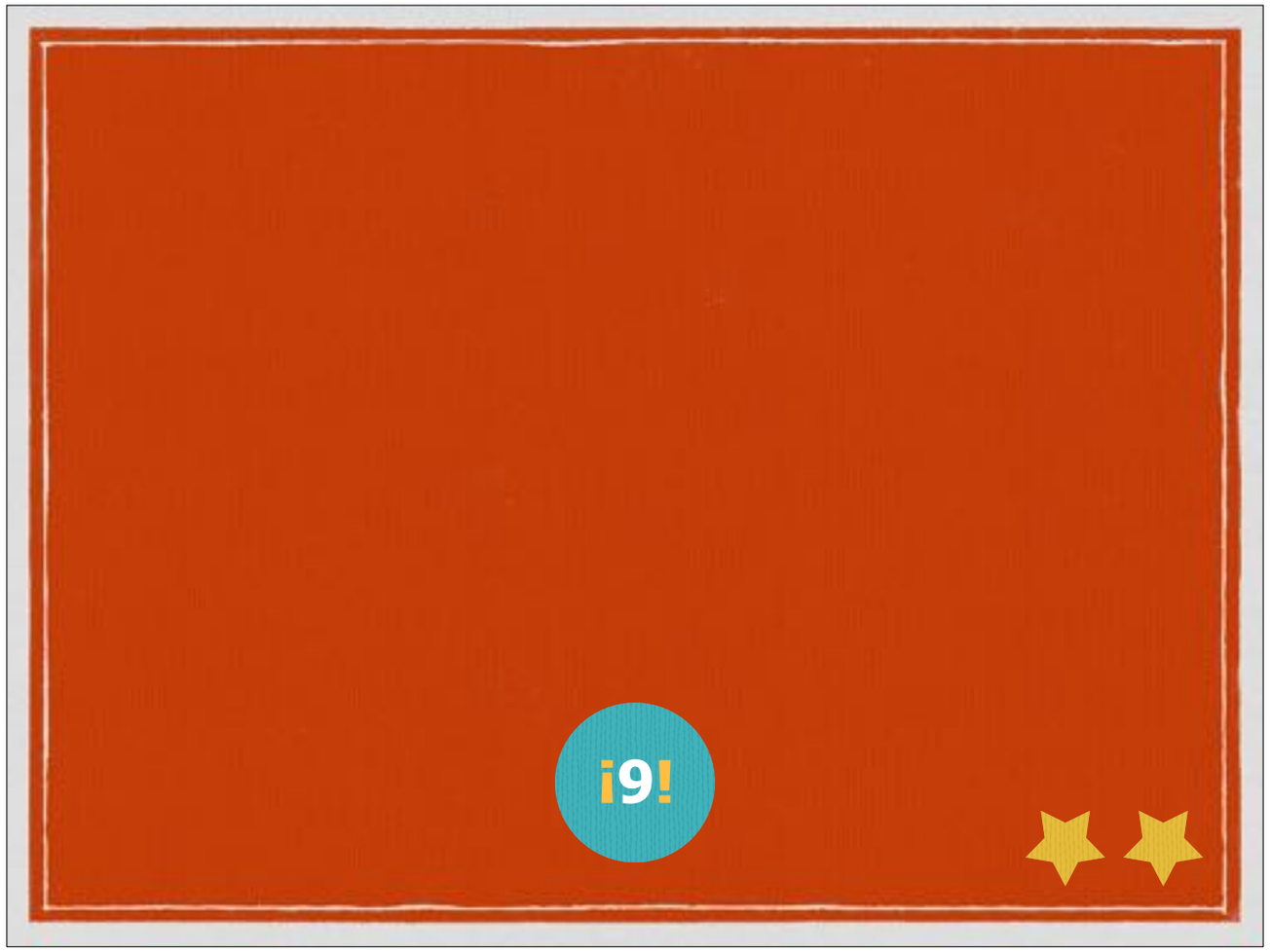
5

What is the difference between slice and splice?

5

What is the difference between slice and splice?

Slice returns a copy of an array from one index up to (but not including) another index. Splice modifies the original array by deleting elements at a starting index and then inserting elements.


¡9!

¡6!

What does splice return?

¡6!

What does splice return?

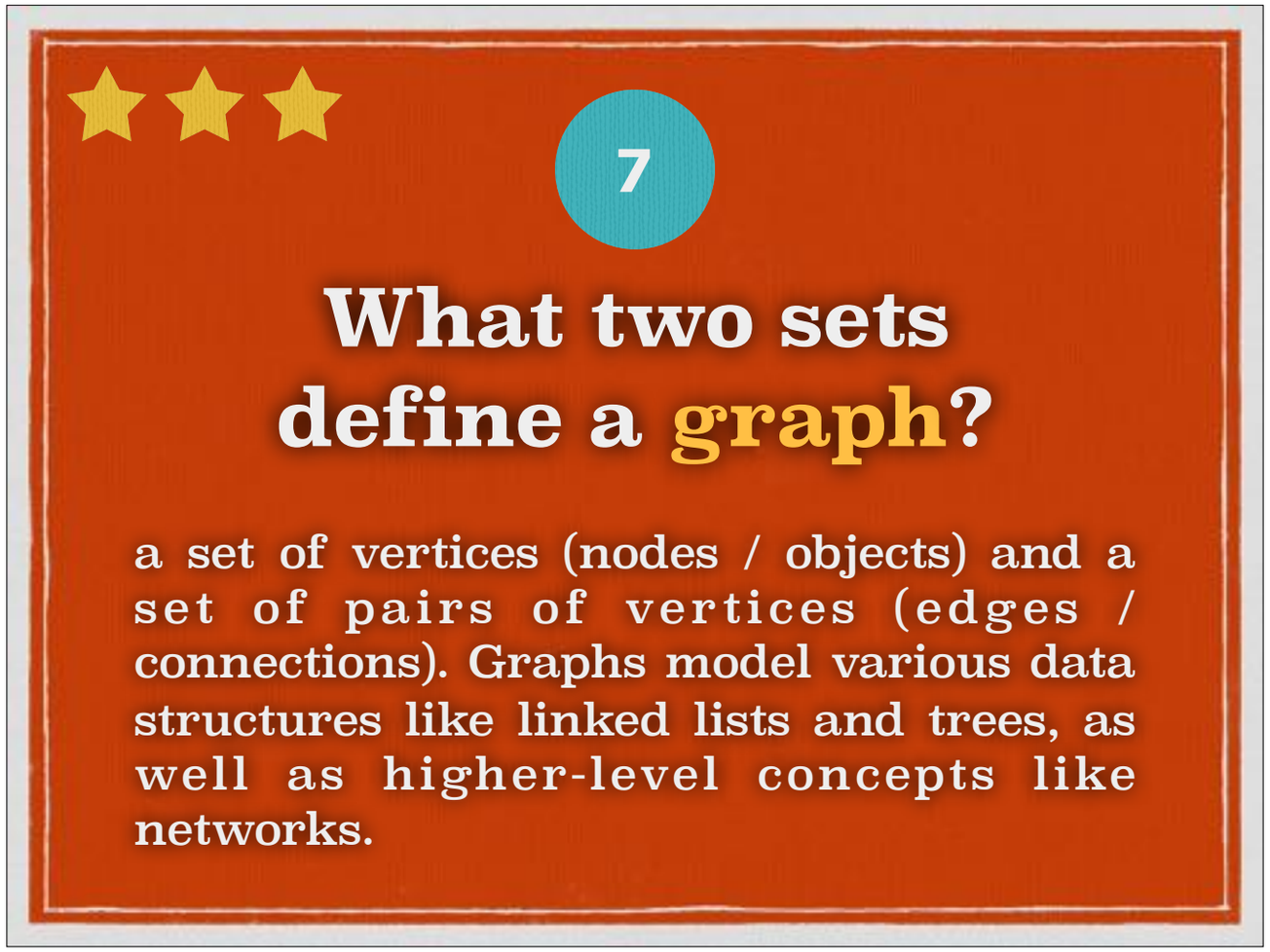
The deleted elements.

Note: the edges alone are not sufficient. { { n1 }, {} } is a graph of one vertex and no edges. Also, see disconnected graphs.

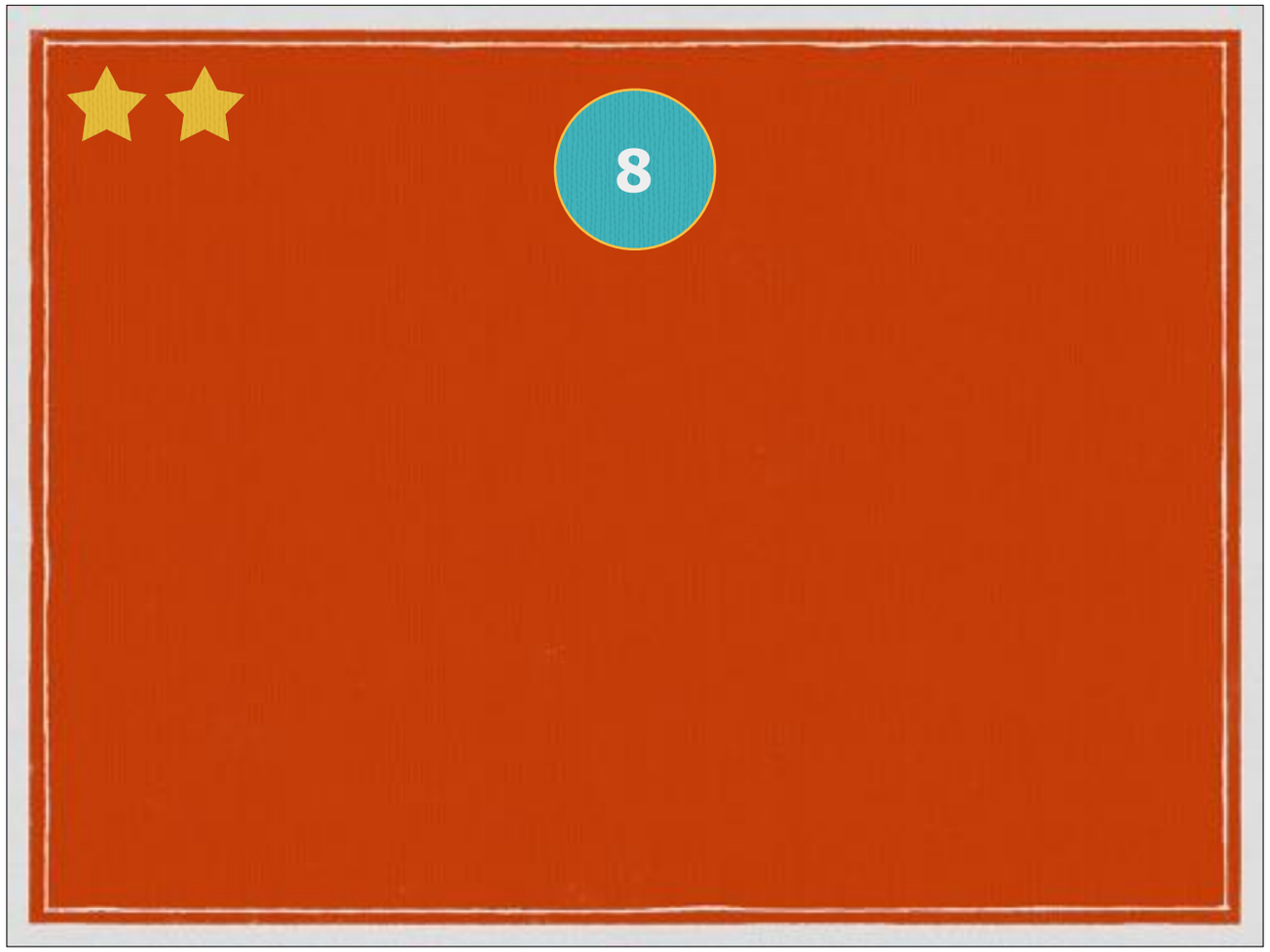
7

What two sets define a graph?

Note: the edges alone are not sufficient. { { n1 }, {} } is a graph of one vertex and no edges. Also, see disconnected graphs.



Note: the edges alone are not sufficient. { { n1 }, {} } is a graph of one vertex and no edges. Also, see disconnected graphs.


8

8

Rank the following O-notations from smallest to largest:

8

Rank the following O-notations from smallest to largest:

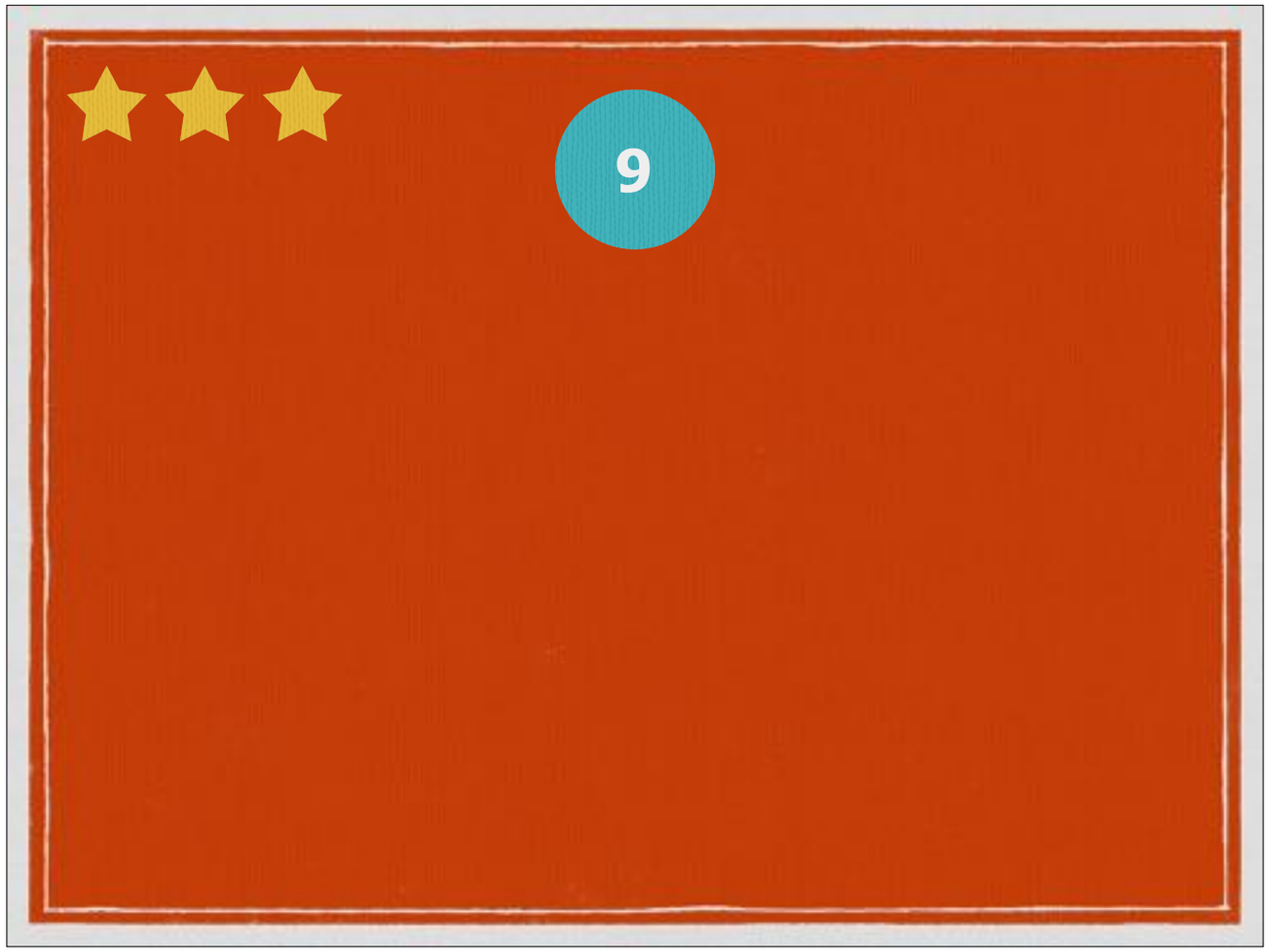
- n
- 2^n
- $\log(n)$
- n^2
- $n \cdot \log(n)$

8

Rank the following O-notations from smallest to largest:

- n
- 2^n
- log(n)
- n^2
- n·log(n)

Answer: log(n), n, n·log(n), n^2, 2^n


9

9

What is a stable sort?

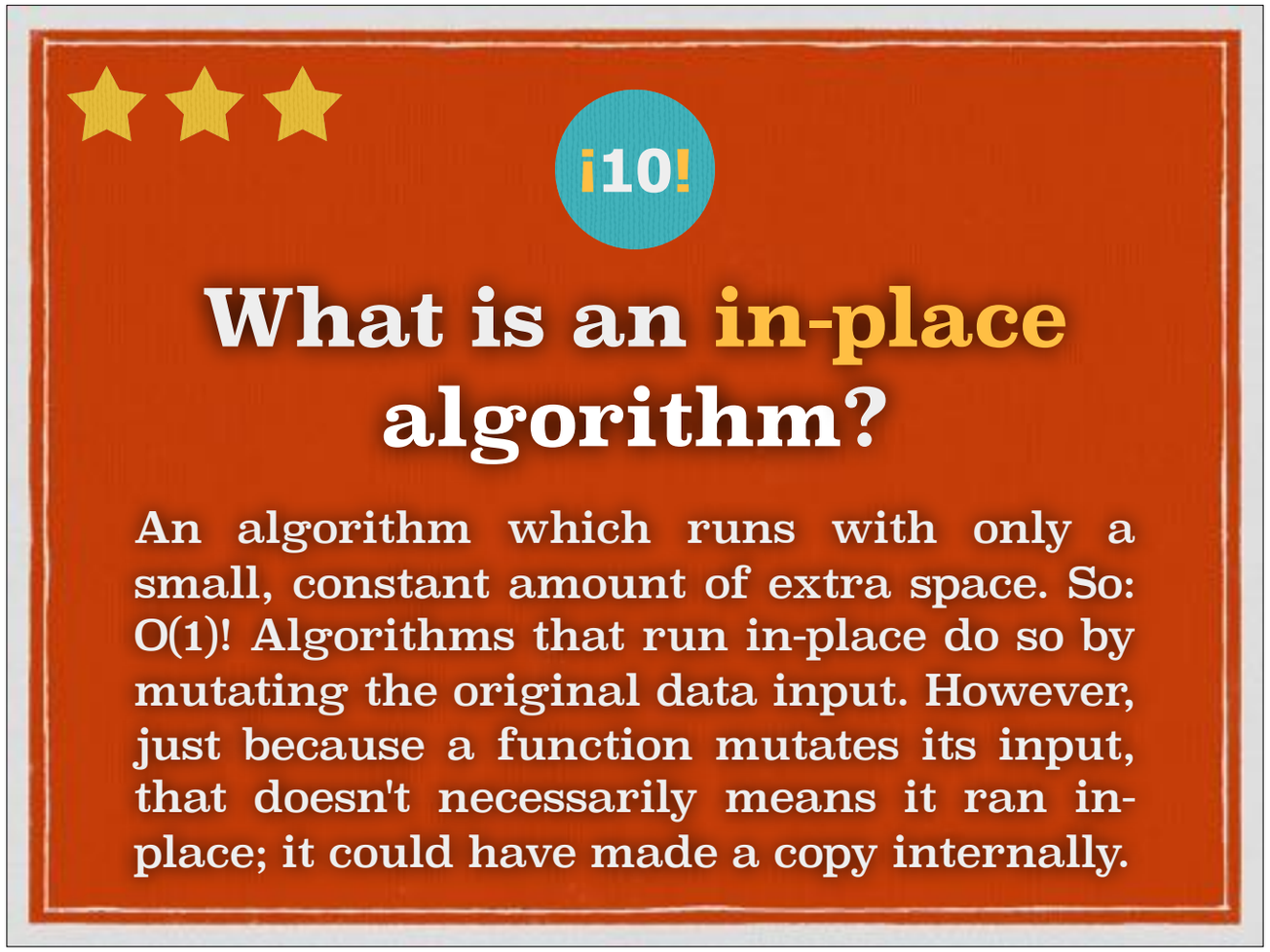
9

What is a stable sort?

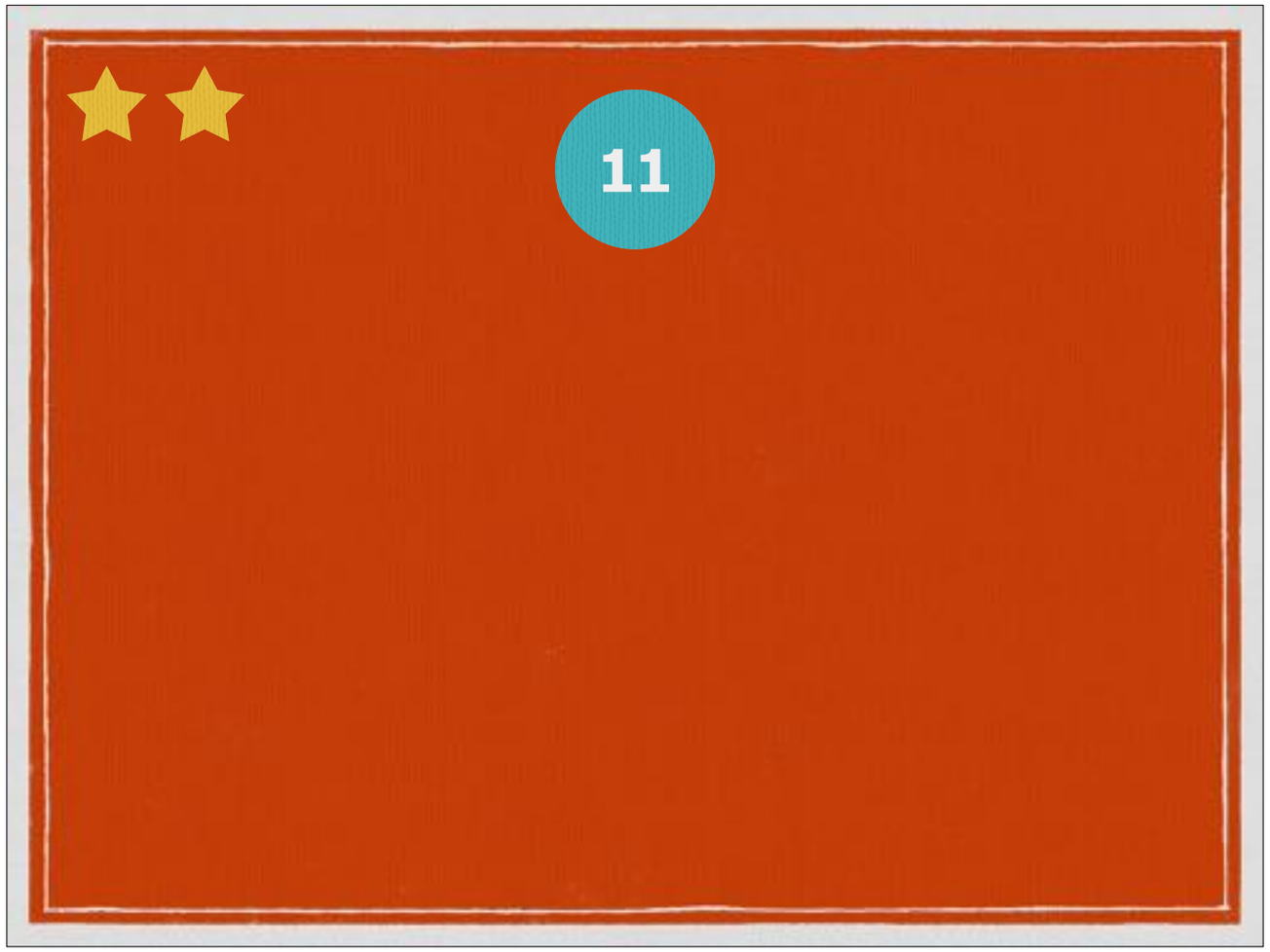
A sorting algorithm that maintains the existing order of "equal" values. Unstable sorts might swap elements that are considered to have the same value.

Common misconception: in-place means the output is changed. That is true, but not sufficient; some algorithms make a copy internally and then change the original. That's O(n) for space!

Common misconception: in-place means the output is changed. That is true, but not sufficient; some algorithms make a copy internally and then change the original. That's O(n) for space!



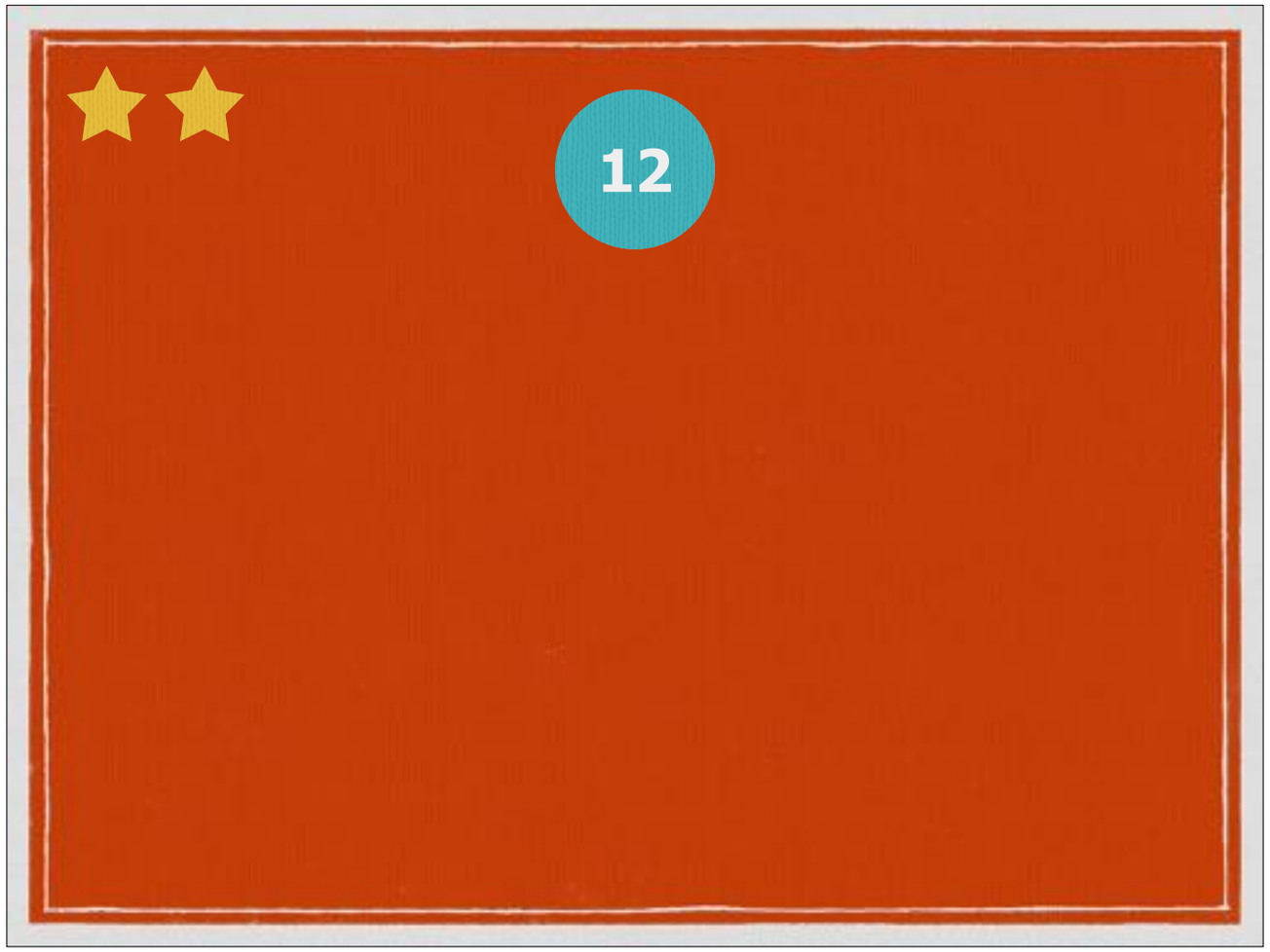
Common misconception: in-place means the output is changed. That is true, but not sufficient; some algorithms make a copy internally and then change the original. That's O(n) for space!



Caching a function output

Caching a function output

Caching a function output


12

12

What is the time complexity of this function?

```
function identifyDoubles (numArr) {
    return numArr.map(function(num){
        var numIsADouble = false;
        numArr.forEach(function(otherNum){
            if (num === 2*otherNum) numIsADouble = true;
        }
        return numIsADouble;
    });
}
```

12

What is the time complexity of this function?

```
function identifyDoubles (numArr) {
    return numArr.map(function(num){
        var numIsADouble = false;
        numArr.forEach(function(otherNum){
            if (num === 2*otherNum) numIsADouble = true;
        }
        return numIsADouble;
    });
}
```

O(n^2). `map` goes through n elements, and for each el, `forEach` goes through n again! n x n iterations is n^2.


13

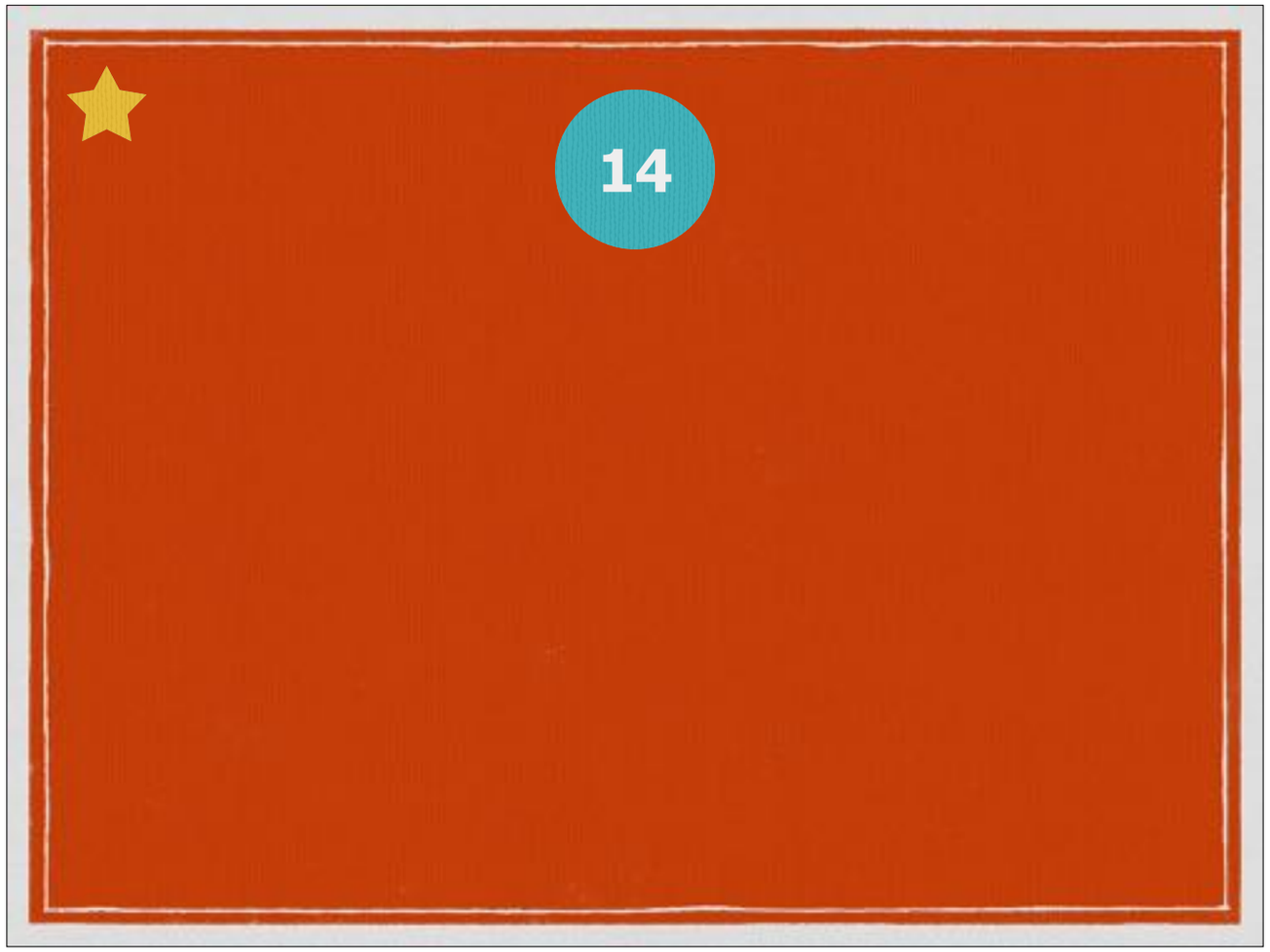
13

Describe the merge sort algorithm steps

13

Describe the merge sort algorithm steps

1. Split the input array in two
2. Recursively merge sort each half
3. (Base case: return a 1-el array)
4. Merge halves together by comparing pairs of head elements
5. Return sorted array
6. Profit



DOM selectors / JQuery are functions; also, accessing the `length` of a nodelist is actually a getter and has to be re-computed! Cache both.

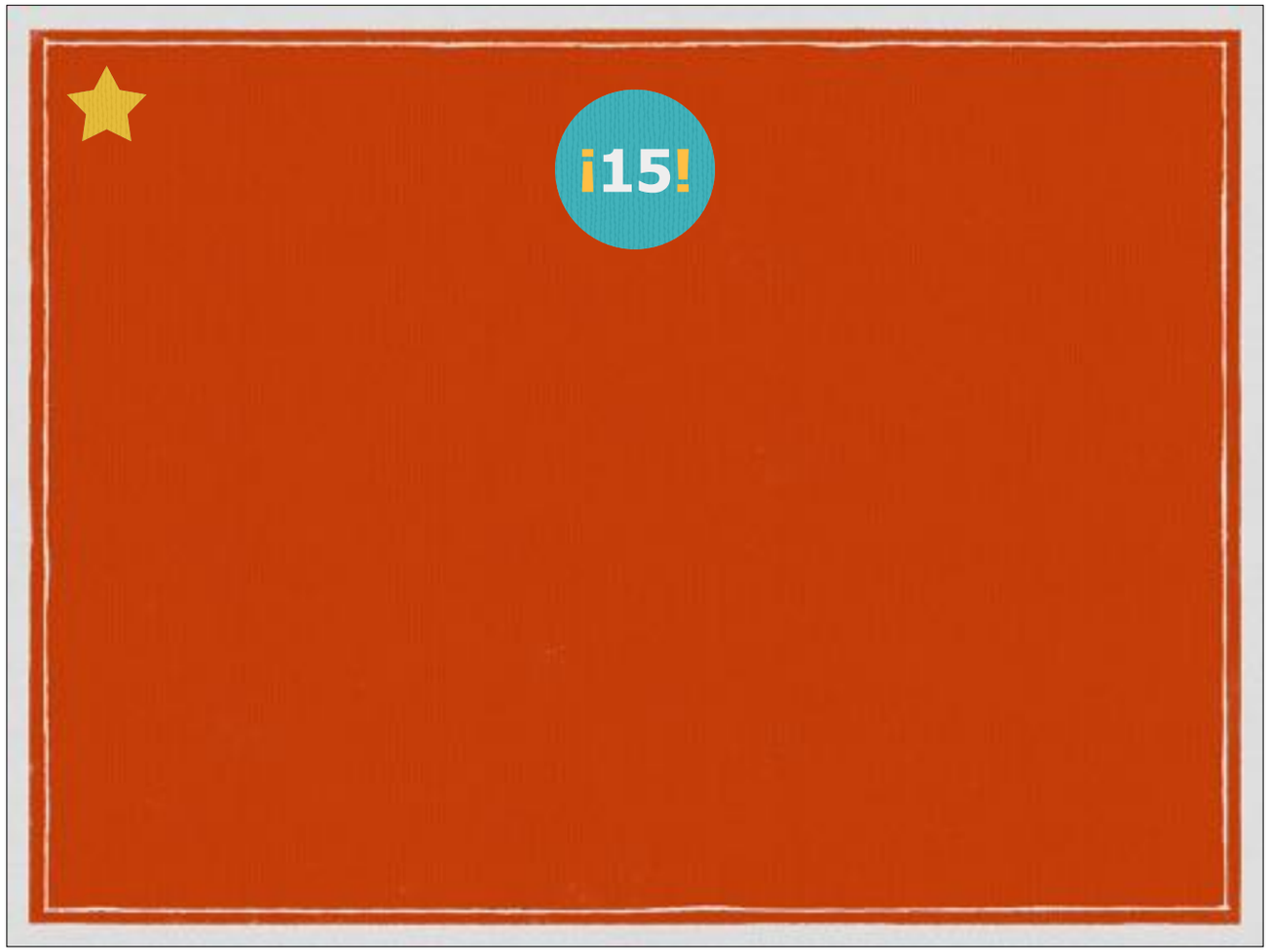
DOM selectors / JQuery are functions; also, accessing the `length` of a nodelist is actually a getter and has to be re-computed! Cache both.

14

How can this be optimized?

```
for (var i = 0, len = $('#myDiv').children.length; i < len; i++) {
    console.log( $('#myDiv').children[i].id );
}
var el = $('#myDiv');
var len = el.children.length;
for (var i = 0; i < len; i++) {
    console.log( el.children[i].id );
}
```

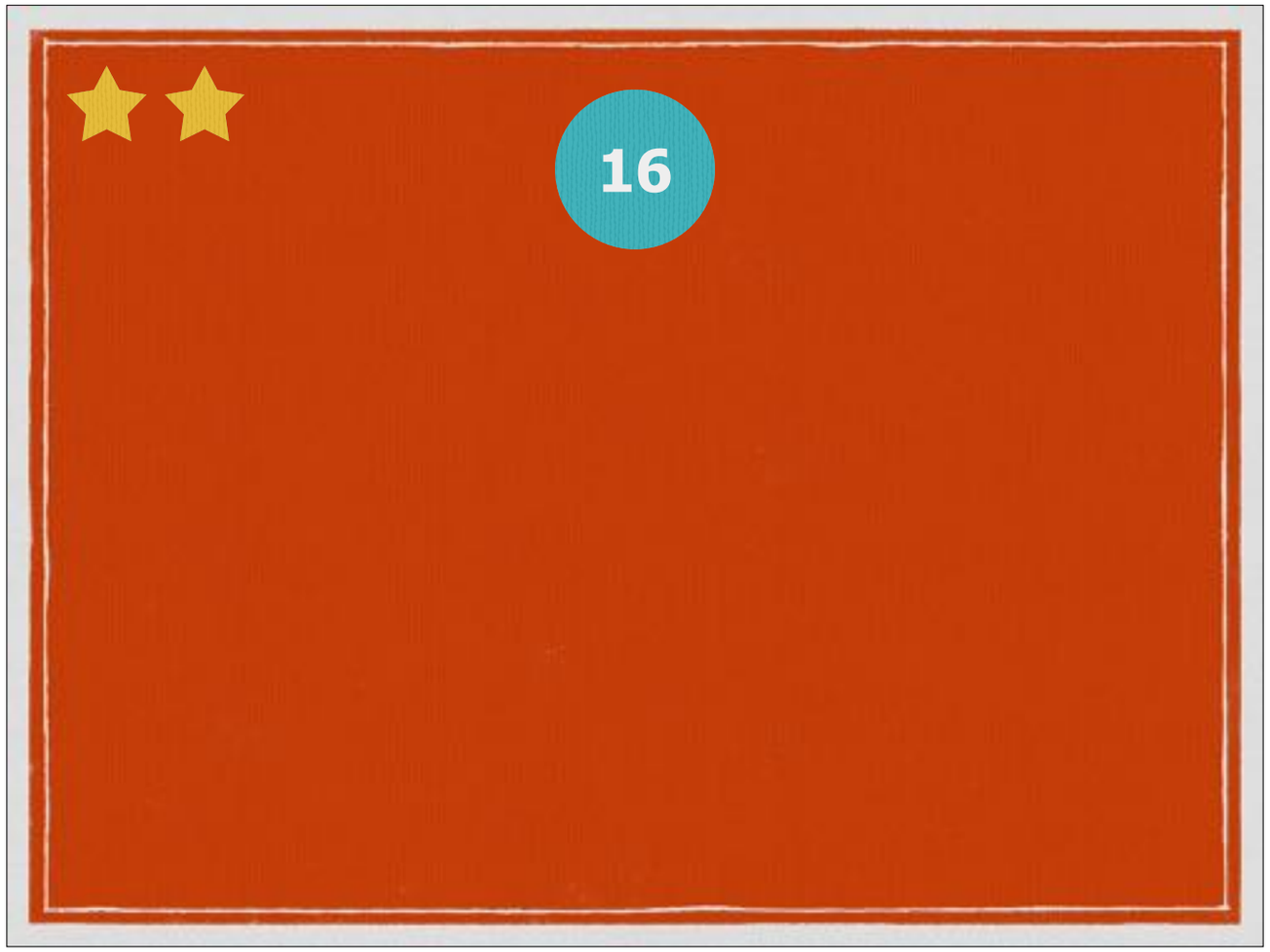
DOM selectors / JQuery are functions; also, accessing the `length` of a nodelist is actually a getter and has to be re-computed! Cache both.



forEach is slow — it invokes a function every loop! And regex literals essentially call new Regex() which is another function invocation every pass.

forEach is slow — it invokes a function every loop! And regex literals essentially call new Regex() which is another function invocation every pass.

forEach is slow — it invokes a function every loop! And regex literals essentially call new Regex() which is another function invocation every pass.


16

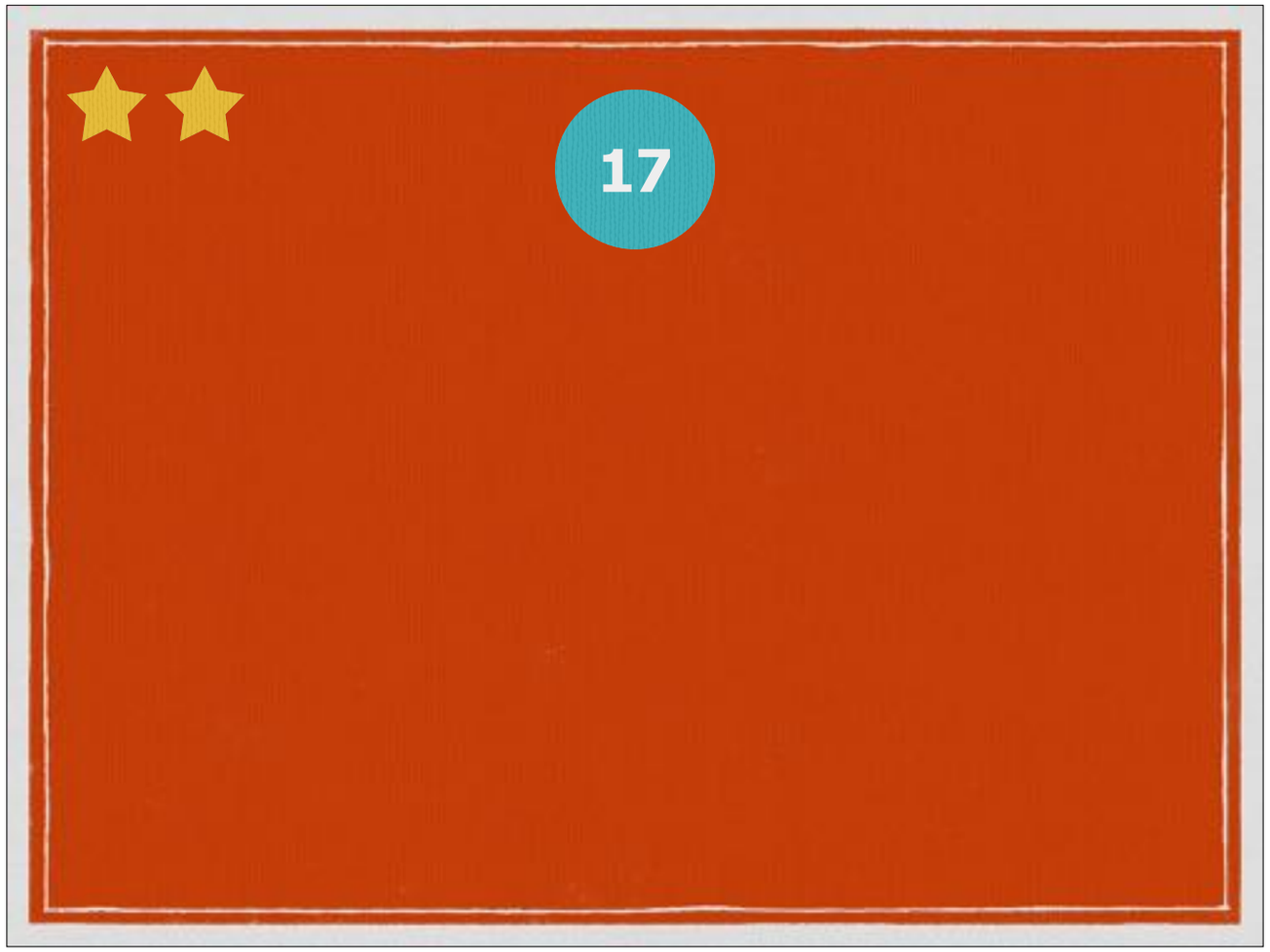
16

What must be the time complexity of indexOf?

16

What must be the time complexity of indexOf?

Worst case, O(n). JavaScript arrays do not have any kind of smart hashing to find inserted values, so the indexOf method must be traversing the array element by element. As the array grows, so must the time that indexOf takes — linearly.


17

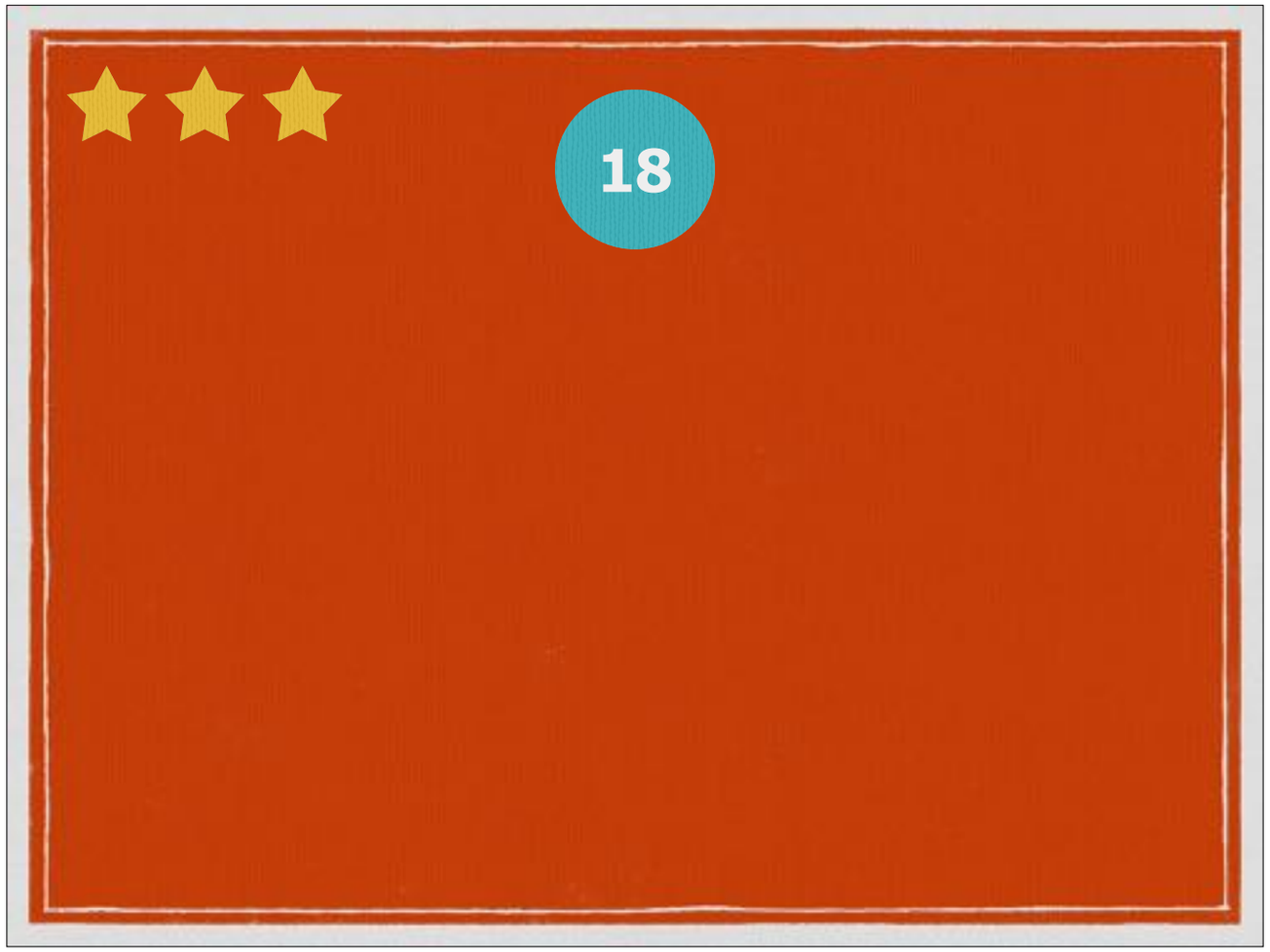
17

How do JavaScript arrays differ from "real" arrays?

17

How do JavaScript arrays differ from "real" arrays?

"Real" arrays are reserved, continuous space in memory; the index is actually an offset from the starting point. JavaScript arrays are actually hash maps to disparate locations in memory; the index maps dynamically to available blocks as needed.


18

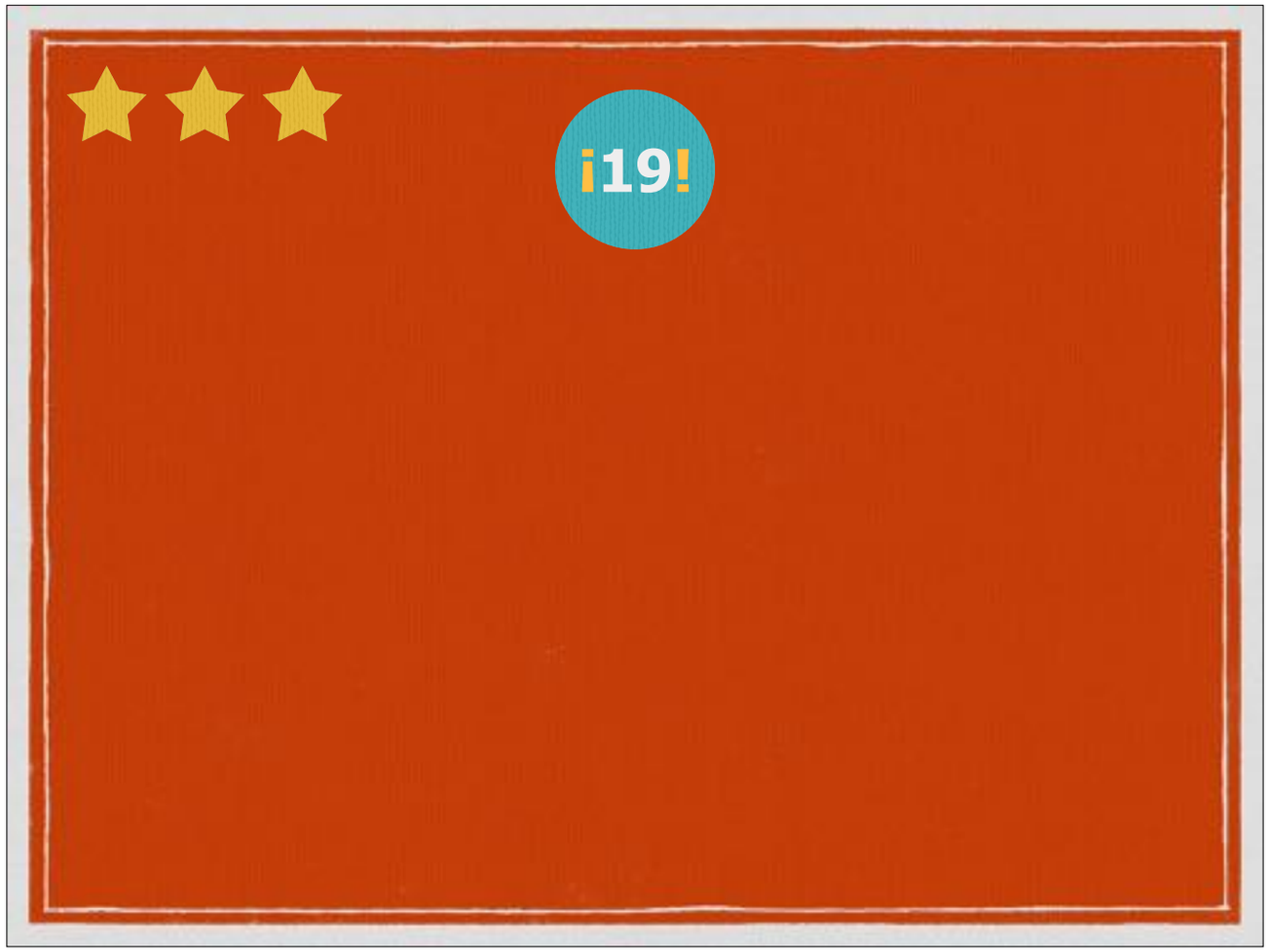
18

Describe insertion performance for linked lists vs arrays.

18

Describe insertion performance for linked lists vs arrays.

If you have a reference to a middle node, linked lists can inert a new node in constant time O(1), whereas splicing in a value to an array requires O(n) time (because all the following values need to be modified). Reaching a node in a linked list requires O(n) time, however.


¡19!

¡19!

What is the space complexity of this function?

```
function duplicate (stuff) {
    var stuffCopy1 = stuff.slice();
    var stuffCopy2 = stuff.slice();
    return [stuffCopy1, stuffCopy2];
}
```

¡19!

What is the space complexity of this function?

```
function duplicate (stuff) {
    var stuffCopy1 = stuff.slice();
    var stuffCopy2 = stuff.slice();
    return [stuffCopy1, stuffCopy2];
}
```

O(n). Space complexity = "how much *extra space* does the function need." Args don't count. Also, constants don't matter! 2n, n... irrelevant next to n^2.

Rubric:
Comparative classification for algorithms
Shape of growth curve
Time *or* space complexity
Based on *size* of input…
…as input gets *very large* (small isn't important)
Ignoring constants (except for O(1))
Upper bound (worst case)

20

What is big O notation?

☆☆☆☆☆

Rubric:
Comparative classification for algorithms
Shape of growth curve
Time *or* space complexity
Based on *size* of input…
…as input gets *very large* (small isn't important)
Ignoring constants (except for O(1))
Upper bound (worst case)

20

What is big O notation?

Rubric:

Comparative classification for algorithms

Shape of growth curve

Time *or* space complexity

Based on *size* of input…

…as input gets *very large* (small isn't important)

Ignoring constants

Upper bound (worst case)

Rubric:

Comparative classification for algorithms

Shape of growth curve

Time *or* space complexity

Based on *size* of input…

…as input gets *very large* (small isn't important)

Ignoring constants (except for O(1))

Upper bound (worst case)