Explain the difference between the == operator and the === operator.

A: === is == zonder type conversions (!= versie is !==)

Explain what a closure is. (Note that JavaScript programs use closures very often)

A: Closure
The ability to treat functions as values, combined with the fact that local bindings are re-created every time a function is called, brings up an interesting question. What happens to local bindings when the function call that created them is no longer active?

The following code shows an example of this. It defines a function, wrapValue, that creates a local binding. It then returns a function that accesses and returns this local binding.

```
function wrapValue(n)
let local = n;
return () =¿ local;

let wrap1 = wrapValue(1);
let wrap2 = wrapValue(2);
console.log(wrap1());
//  1
console.log(wrap2());
//  2
```

This is allowed and works as youd hopeboth instances of the binding can still be accessed. This situation is a good demonstration of the fact that local bindings are created anew for every call, and different calls cant trample on one anothers local bindings.

This featurebeing able to reference a specific instance of a local binding in an enclosing scopeis called closure. A function that references bindings from local scopes around it is called a closure. This behavior not only frees you from having to worry about lifetimes of bindings but also makes it possible to use function values in some creative ways.

Explain what higher order functions are.

Higher-order functions
Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions. Since we have already seen that functions are regular values, there is nothing particularly remarkable about the fact that such functions exist. The term comes from mathematics, where the distinction between functions and other values is taken more seriously.

Higher-order functions allow us to abstract over actions, not just values. They come in several forms. For example, we can have functions that create new functions.

Explain what a query selector is and give an example line of JavaScript that uses a query selector.

Query selectors We wont be using style sheets all that much in this book. Understanding them is helpful when programming in the browser, but they are complicated enough to warrant a separate book.

The main reason I introduced selector syntaxthe notation used in style sheets to determine which elements a set of styles apply tois that we can use this same mini-language as an effective way to find

DOM elements.
The querySelectorAll method, which is defined both on the document object and on element nodes, takes a selector string and returns a NodeList containing all the elements that it matches.

p $And if you go chasing$
$< span class = \ddot{a}nimal" > rabbits < /span >< /p >$
$< p > And you know you' re going to fall < /p >$
$< p > Tell' em a < span class = "character" > hookah smoking$
$< span class = \ddot{a}nimal" > caterpillar < /span >< /span >< /p >$
$< p > Has given you the call < /p >$

$< script >$
$function count(selector) return document.querySelectorAll(selector).length;$
$console.log(count("p")); //All < p > elements$
$//4$
$console.log(count(".animal")); //Class animal$
$//2$
$console.log(count("p.animal")); //Animal inside of < p >$
$//2$
$console.log(count("p > .animal")); //Direct child of < p >$
$//1$
$< /script >$

$Unlike methods such as getElementsByTagName, the object returned by querySelectorAll is not live. It won't change when you$

The querySelector method (without the All part) works in a similar way. This one is useful if you want a specific, single element. It will return only the first matching element or null when no element matches.