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

De === operator vergelijkt, net zoals de == operator, twee expressies aan beide kanten en geeft of 'true' of 'false'. Maar in plaats van automatisch een type conversie toe te passen, moeten de typen van de expressies ook gelijk zijn. === voert dus geen type conversies uit. De equivalent van != is !==.

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

Een closure wordt aangemaakt bij een functie declaratie. Deze closure bevat de waardes van de globale en locale variabelen die bestonden tijdens de declaratie. In het onderstaande voorbeeld wordt er een functie in een functie gedefineerd. De closure van *myFunction* bevat onder andere de variabele *'counter'* met waarde 0. Zonder deze closure zou *'counter'* bij het aanroepen van increment() *undefined* zijn, dus worden c1, c2 en c3 allemaal gelijk aan $undefined + 1 \approx 0 + 1 = 1$. Maar met de closure wordt counter onthouden bij de functie variabele en wordt $c1 = 1$, $c2 = 2$ en $c3 = 3$.

```
1: function createCounter() {
2:    let counter = 0
3:    const myFunction = function() {
4:        counter = counter + 1
5:        return counter
6:    }
7:    return myFunction
8: }
9: const increment = createCounter()
10: const c1 = increment()
11: const c2 = increment()
12: const c3 = increment()
```

Explain what higher order functions are.

Hogere-order functies zijn functies die zelf andere functies gebruiken, of door ze als argument te nemen, of door een functie te returnen. (zoals in bovenstaande createCounter())

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

querySelectorAll(selector) en *querySelector(selector)* zijn html document methodes die elementen vinden die aan een bepaalde *selector* eigenschap voldoen. *querySelectorAll(selector)* vindt alle elementen die daar aan voldoen en vormt een array-achtige NodeList van die elementen. *querySelector(selector)* geeft echter alleen het eerste element die aan *selector* voldoet.

Een voorbeeld uit *Eloquent JavaScript*:
< *p* >And if you go chasing
< *spanclass* = *änimal*" >rabbits< */span* >< */p* >
< *p* >And you know you're going to fall< */p* >
< *p* >Tell 'em a < *spanclass* = "*character*" >hookah smoking
< *spanclass* = *änimal*" >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* >