**Skilaverkefni 2 - VEF2B3U**

1. ECMAScript er grunn forritunarmálið í javascript. Það er líka notað í grunnin á ýmsum öðrum forritunarmálum eins og t.d. Node.js.
2. Kóðinn reynir að keyra sama hvað. Jafnvel þó það séu villur í stafsetningu eða eitthvað þá keyrir kóðinn bara upp að þeirri villu.
3. Nei, en þú getur látið einfaldar gæsalappir inn í streng sem er með tvöfaldar gæsalappir, og öfugt.
4. Tómt value á breytu er null eða undefined.
5. “1” === 1  
   >false  
     
   “1” == 1  
   >true  
   == reynir að breyta týpunni á breytunni, en breytur með === þurfa að vera nákvæmlega eins.
6. result er gefið boolean expression, sem skilar alltaf true eða false. Expression-ið notar || sem er “or”. Fyrst það sem er vinstra megin við || er true þá lítur kóðinn ekkert á hægri hliðina og engin villa kemur.
7. Expression er bara einhver kóði sem skilar (return) einhverjum gögnum. Það sem er hægra megin við = merki er alltaf expression.  
   Dæmi:  
   2+3  
   Þetta er expression sem skilar 5. Það er hægt að gefa breytum gögnin úr expression með = merkinu.  
   var foo = 2 + 3;
8. Function expression:  
   function foo( bar ){  
    return bar \* 2;  
   }  
     
   Function declaration:  
   var foo = function(bar){  
    return bar \* 2;  
   };  
     
   Arrow function:  
   var foo = (bar) => { bar \* 2 };
9. let gerir block scope breytu sem hverfur þegar þú ferð úr blockinu.
10. Scope er þegar kóðinn er flokkaður niður í minni bita sem deila ekki breytum. Það er alltaf hægt að sækja breytu úr efsta scope-inu (global scope) en það er ekki hægt að ná í breytu úr neðra scope-i.
11. Þegar þú notar function expression skiptir engu máli hvar kóðinn er staðsettur því javascript færir hann alltaf efst.
12. “use strict”; kveikir á strict mode í javascript sem þýðir að það sem sem áður var talinn slæmur kóði, sem virkar samt, virkar ekki lengur. Þetta er nytsamlegt þegar nauðsin er á að kóðinn sé öruggur.
13. Fallið er “hoisted” upp fyrir ofan x(); skipunina.
14. function foo(){   
     function bar() {   
     return 3;   
     }  
     function bar() {   
     return 8;   
     }   
     return bar();  
    }   
    alert(foo());  
      
    Kóðinn ætti að skila 8, en ekki 3 þar sem föllin eru hoisted á undan öðrum kóða.
15. Interpreter-inn ákveður bara fyrir þig að age sé var, og kóðinn heldur bara áfram.
16. Þetta er anonymous function. Svigarnir utanum gera það að verkum að aðgerðin kallar í sig sjálfa.
17. Template literals eru gerð af strengjum sem leyfa þér að auðveldlega hafa margar línur og expression inni í strengnum. Hugmyndin er að það sé náttúrulegra að skrifa þetta svona, og þá sé það líka auðveldara. Dæmi:  
      
    Venjulegur strengur með tvöföldum eða einföldum gæsalöppum:  
    var age = 30;  
    console.log("The person is " + age + " years old today!\nHappy birthday!");  
      
    Template literal:  
    var age = 30;  
    console.log(`The person is ${age} years old today!  
    Happy birthday!`);  
      
    Bæði kóðadæmin skila sama texta, en template literal kóðinn er auðskiljanlegri.
18. Call stack er stakkinn sem heldur utan um kóðan sem er verið að vinna með. Þegar þú kallar í aðgerðir eru þær settar á toppinn á stakkanum og svo vinnur hann sig niður. Ef of mikið af gögnum eru í stakkanum, til dæmis þegar aðgerð vísar of oft í sjálfa sig, kemur hin alræmda stack overflow villa.  
       
    Heap er leið til að geyma gögn eins og tré. Gögnin eru röðuð á eina af tveim vegum. Annaðhvort er hæðsta value-ið á toppinum, eða það lægsta. Gögnin fyrir neðan toppinn eru þá alltaf lægri eða hærri en toppurinn og svona gengur það niður lögin.  
    Mér finnst besta leiðin til að sjá fyrir mér heap vera með mynd (max-heap): 