Ex1.)

1. Das “window “wird durch alle Browser unterstützt. Alle globalen JavaScript Objekte, Funktionen und variablen werden automatisch teil des “window” Objekts.

window.document.getElementById("header");   
entspricht  
document.getElementById("header");

1. Scope: Bestimmt die Sichtbarkeit von Variabeln und anderen Ressourcen in Bereichen des Codes.   
     
   .global Scope  
   .local Scope
2. .var: deklarierert eine Variable unabhängig von ihrem  
    Scope (Gültigkeitsbereich).  
     
   .let: deklariert eine lokale Variable im BlockScope  
     
   .const: symbolische Konstante, deklariert eine unveränderliche  
    Variable  
     
   .kein Keyword: globale Deklarierung (wie bei .var).  
     
   “use strict”: liefert keine Fehlermeldung, auch wenn der ein Fehler in der Deklarierung ist.

Ex2.)

1. Function createCounter beinhaltet einen counter und als Rückgabewert liefert sie eine Funktion die diesen erhöht und ausgibt.  
   Var increment wird die createCounter – FUnktion zugewiesen und bei jeder Ausführung *increment()* wird die subfunction aufgerufen und der counter-Wert in die Variablen c1,2,3 gespeichert und dann ausgegeben  
     
   CURRYING:  
   greet ist eine function die ein gender und einen name erwartet. Am Ende wird eine Anrede mit *Dear Mr/Mrs (depends on gender) name* ausgegeben.  
   Es wird ein Array mit Frauennamen erzeugt und greetWoman wird greet(‘female’) zugewiesen -> fehlt nur noch der Name  
   in der for-Schleife wird das Array durchlaufen und mit den einzelnen Namen und die Anrede für erzeugt  
   .
2. Code:

|  |
| --- |
| let filter =  filterArray=>{  let ret = new Array();  for(entry in filterArray){  if(filterArray[entry]>0){  ret.push(filterArray[entry]);  }  }  return ret  }      let test = [1,2,3,-1,-2,4,5];  var filtered = filter(test)  console.log(filtered); |

1. Code:

|  |
| --- |
| let filter =  filterArray=>{  let ret = new Array();  for(entry in filterArray){  if(filterArray[entry]>0 && Number.isInteger(filterArray[entry])){  ret.push(filterArray[entry]\*filterArray[entry]);  }  }  return ret  }    let test = [1,2,3,-1,-2,4,5];  var filtered = filter(test)  console.log(filtered); |
|  |

1. A closure gives you access to an outer function’s scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.