**Parallele und verteilte Systeme**

**Vorwort:**Grundlage für diese Aufgabe ist die Präsentation «Einführung JavaScript, NodeJS, Express, Git/Github»

**Dauer:** 90 Minuten  
**Optional:** 30 Minuten  
**Total:** 90 Minuten

Nachfolgend werden Sie sich mit den Grundlagen von JavaScript auseinandersetzen welches in diesem Modul als Programmiersprache dient. Weiter werden Sie sich auch vertraut machen mit Frameworks wie Express um auf eine einfache Art und Weise API’s zu bauen.

JavaScript Kompendium

<https://www.w3schools.com/js/>

**Aufgabe 1)**

Installieren Sie NodeJS auf Ihrem System damit Sie für die späteren Aufgaben in der Lage sein werden, JavaScript direkt auf der Konsole auszuführen, ohne dass ein Browser benötigt wird.

**Installieren Sie NodeJS mithilfe des Tools «nvm». Schauen Sie, dass Sie mindestens Version 18 und im besten Fall eine LTS-Version verwenden.**

<https://github.com/coreybutler/nvm-windows>

ℹ️ Tipp: «nvm install lts && nvm use …»

|  |
| --- |
| <https://github.com/coreybutler/nvm-windows/releases>  [nvm-setup.exe](https://github.com/coreybutler/nvm-windows/releases/download/1.1.11/nvm-setup.exe) |

**Worum handelt es sich bei NodeJS und was kann damit erreicht werden?**

|  |
| --- |
| Node.js ist eine Open-Source-Plattform, die es ermöglicht, serverseitige Anwendungen mit JavaScript zu entwickeln. Es ist eine Runtime |

**Welche Version von NodeJS haben Sie installiert und mit welchem Befehl kann man diese herausfinden?**

|  |
| --- |
| nvm -v  1.1.11  PS C:\Users\floer> nvm list  \* 18.14.2 (Currently using 64-bit executable) |

**Starten Sie NodeJS im interaktiven Modus. Was können Sie damit nun anstellen?**

|  |
| --- |
| JavaScript-Code ausführen und Ergebnisse anzeigen.  Hier sind einige Dinge, die Sie im interaktiven Modus von Node.js tun können:   * Ausführen von JavaScript-Befehlen und -Ausdrücken: Sie können jede gültige JavaScript-Anweisung oder jeden Ausdruck eingeben und Node.js führt ihn aus und gibt das Ergebnis in der Konsole aus. * Definieren von Variablen: Sie können Variablen definieren und Werte zuweisen, genau wie in einem regulären JavaScript-Programm. * Verwenden von Modulen: Sie können Node.js-Module laden und verwenden, genau wie in einem regulären JavaScript-Programm. Verwenden Sie dazu die require()-Funktion. * Schreiben von Funktionen: Sie können Funktionen definieren und aufrufen, genau wie in einem regulären JavaScript-Programm. * Verwenden von NPM-Paketen: Sie können NPM-Pakete im interaktiven Modus von Node.js installieren und verwenden, indem Sie npm install und require() verwenden. * Debuggen von Code: Der REPL-Modus bietet auch eine integrierte Debugging-Funktionalität, mit der Sie Schritt für Schritt durch Ihren Code navigieren und ihn debuggen können. |

**Aufgabe 2)**

Richten Sie sich eine Entwicklungsumgebung ein. Folgende Entwicklungsumgebung wird empfohlen:

Visual Studio Code: <https://code.visualstudio.com/download>

**Was glauben Sie, warum sich Visual Studio Code besonders für JavaScript eignet?**

|  |
| --- |
| 1. Integrierte Unterstützung für Node.js: Visual Studio Code bietet eine integrierte Unterstützung für Node.js, was bedeutet, dass Sie Node.js-Anwendungen direkt in der IDE ausführen und debuggen können. 2. Intelligente Code-Editoren: Visual Studio Code bietet intelligente Code-Editoren, die Ihnen helfen, schnell und effizient zu programmieren. Zum Beispiel kann die Autovervollständigungsfunktion Ihnen dabei helfen, Code schneller zu schreiben, indem sie automatisch Codevorschläge bereitstellt, während Sie tippen. 3. Debugging-Tools: Visual Studio Code bietet mächtige Debugging-Tools, mit denen Sie schnell und einfach Fehler in Ihrem Code finden und beheben können. 4. Erweiterbarkeit: Visual Studio Code ist sehr erweiterbar und bietet eine große Anzahl von Erweiterungen für JavaScript-Entwickler. Diese Erweiterungen können dazu beitragen, Ihre Produktivität zu verbessern, indem sie zusätzliche Funktionen und Tools bereitstellen, die speziell für JavaScript-Entwickler entwickelt wurden. 5. Plattformübergreifend: Visual Studio Code ist plattformübergreifend und läuft auf Windows, macOS und Linux. Das macht es zu einer guten Wahl für Entwickler, die auf verschiedenen Betriebssystemen arbeiten.   Insgesamt bietet Visual Studio Code eine reichhaltige Funktionalität für JavaScript-Entwickler und ist eine der besten IDEs für die Entwicklung von JavaScript-Anwendungen. |

**Installieren Sie folgendes Plugin**

* Quokka

**Versuchen Sie mit diesem Plugin etwas herumzuspielen und beschreiben Sie die Funktionalität des Plugins kurz in eigenen Worten.**

|  |
| --- |
| Das Plugin "Quokka.js" ist eine interaktive JavaScript-Entwicklungserweiterung für Visual Studio Code, die Ihnen helfen soll, schneller und effizienter zu programmieren. Hier sind einige der Funktionen von Quokka.js:   1. Echtzeit-Code-Tests: Quokka.js führt Ihren Code in Echtzeit aus und zeigt Ihnen die Ergebnisse direkt in Ihrem Code-Editor an. Dadurch können Sie schneller Feedback erhalten und Bugs schneller beheben. 2. Inline-Ergebnisse: Quokka.js zeigt die Ergebnisse Ihrer Codezeilen direkt im Code-Editor an. Dadurch können Sie Ihre Ergebnisse schnell überprüfen und anpassen, ohne den Code-Editor verlassen zu müssen. 3. Live-Coding: Mit Quokka.js können Sie Ihren Code direkt im Editor schreiben und ausführen, ohne eine separate Entwicklungsumgebung zu öffnen. Dadurch können Sie schnell Prototypen erstellen und Ihre Ideen testen. 4. Code-Überprüfung: Quokka.js bietet eine statische Code-Analyse, um Syntaxfehler und potenzielle Probleme in Ihrem Code zu erkennen, bevor Sie ihn ausführen. Dadurch können Sie Bugs frühzeitig erkennen und beheben. 5. Erweiterbarkeit: Quokka.js ist sehr erweiterbar und bietet eine Reihe von Erweiterungen, die die Funktionalität der Erweiterung erweitern und anpassen können.   Insgesamt soll Quokka.js Ihnen helfen, schneller und effizienter zu programmieren, indem es Ihnen schnelles Feedback und interaktives Debugging bietet. Es ist eine nützliche Erweiterung für jeden JavaScript-Entwickler, der mit Visual Studio Code arbeitet. |

**Aufgabe 3)**

Beschäftigen Sie sich mit den verschiedenen primitiven Datentypen, welche in JavaScript existieren und wie diese Variablen zugeordnet werden können. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**primitive\_datatypes.js**».

**Mit welchem Befehl kann in JavaScript etwas auf die Konsole ausgegeben werden? Und welchen Sinn und Zweck hat die Konsole im Normalfall?**

|  |
| --- |
| In JavaScript kann man mit dem Befehl "console.log()" etwas auf die Konsole ausgeben. Der Befehl nimmt einen oder mehrere Argumente, die ausgegeben werden sollen, und gibt sie als Zeichenfolge auf der Konsole aus. |

**Definieren Sie in dieser Datei eine Variable vom Typ «String» und geben Sie diese auf der Konsole aus. Führen Sie danach Ihre Datei mit Quokka oder NodeJS aus. Wie kann eine JavaScript Datei mit Node ausgeführt werden?**

|  |
| --- |
| let myString = "Hallo Welt!";  console.log(myString); |

**Erweitern Sie das Script «primitive\_datatypes.js». Definieren Sie weitere Variablen mit den Datentypen «Number», «Boolean» und «undefined». Geben Sie auch diese jeweils in der Konsole aus. Führen Sie danach das Script aus und schauen Sie, ob die erwartete Ausgabe erscheint.**

|  |
| --- |
| let myString = "Hallo Welt!";  let myNumber = 42;  let myBoolean = true;  console.log(myString, myNumber, myBoolean);  Beachten Sie, dass die Variablenwerte durch Leerzeichen getrennt sind, da wir sie als Argumente an die console.log()-Funktion übergeben haben. |

Terminal:

ls (sichergehen um in Ordner sein)

Mode LastWriteTime Length Name

---- ------------- ------ ----

d---- 03.05.2023 20:01 Javascript

-a--- 03.05.2023 21:26 2968 Javascript.7z

-a--- 03.05.2023 21:35 607 primitive\_datatypes.js

PS D:\01 TEKO\Michel\FilesVisualStudioCode> node .\primitive\_datatypes.js

Hello, world!

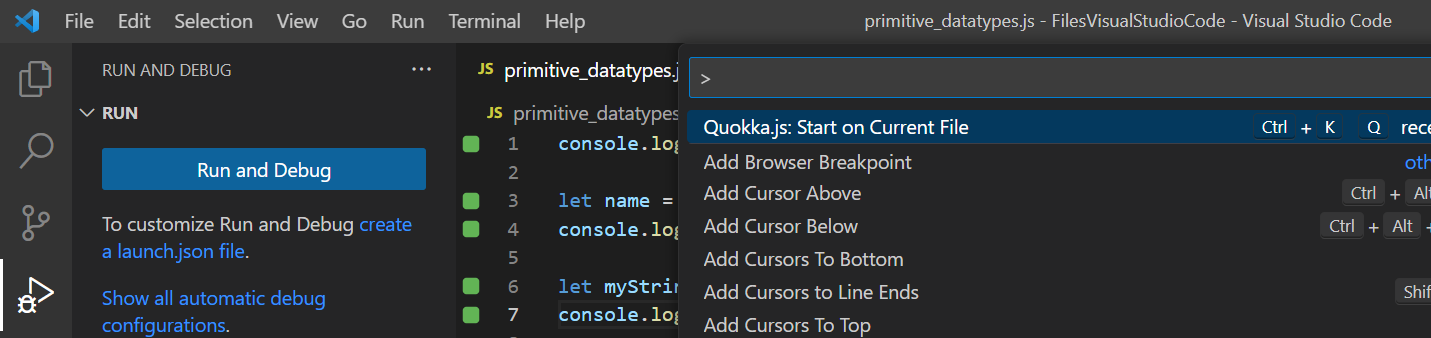
Hello, John!

Hallo Welt!

42 true undefined

PS D:\01 TEKO\Michel\FilesVisualStudioCode>

Ctrl+Shift+P, Quokka, Start on current file



**Deklarieren Sie eine Variable mit dem «var» Schlüsselwort und weisen Sie dieser einen Wert vom Typ «String» zu. Weisen Sie danach einen anderen Wert der gleichen Variable zu, welcher den Datentype «boolean» hat. Was stellen Sie fest?**

|  |
| --- |
| var myVariable = "Hallo Welt!";  console.log(myVariable); // gibt "Hallo Welt!" aus  myVariable = true;  console.log(myVariable); // gibt "true" aus  In diesem Beispiel wird die Variable "myVariable" mit dem "var" Schlüsselwort deklariert und ihr der Wert "Hallo Welt!" zugewiesen. Wenn Sie den Wert der Variable auf der Konsole ausgeben, sehen Sie, dass der Wert "Hallo Welt!" ist. Dann wird der Wert der Variable durch eine Zuweisung auf "true" geändert, welches vom Typ Boolean ist. Wenn Sie den Wert der Variable nun wieder auf der Konsole ausgeben, sehen Sie, dass der Wert "true" ist.  Sie stellen also fest, dass JavaScript eine schwache Typisierung hat, was bedeutet, dass der Datentyp einer Variable dynamisch zur Laufzeit bestimmt wird und sich ändern kann, wenn der Wert der Variable geändert wird. Dies kann sowohl nützlich als auch fehleranfällig sein, da es ermöglicht, schnell und flexibel mit Variablen umzugehen, aber auch zu unerwarteten Verhaltensweisen führen kann, wenn der Typ einer Variable nicht wie erwartet ist. Es wird empfohlen, Variablen in JavaScript immer explizit mit dem passenden Datentyp zu initialisieren und darauf zu achten, dass der Typ während der Laufzeit nicht unerwartet geändert wird.  Formularbeginn  Formularende |

**Deklarieren Sie 2-mal eine Variable mit dem «var» Schlüsselwort und der gleichen Bezeichnung. Was stellen Sie fest?**

|  |
| --- |
| Wenn Sie eine Variable in JavaScript mit dem Schlüsselwort "var" deklarieren und dabei die gleiche Bezeichnung zweimal verwenden, wird die Variable im zweiten Fall einfach überschrieben. |

**Deklarieren Sie 2-mal eine Variable mit dem «let» Schlüsselwort und der gleichen Bezeichnung. Was stellen Sie fest?**

|  |
| --- |
| Wenn Sie eine Variable in JavaScript mit dem Schlüsselwort "let" deklarieren und dabei die gleiche Bezeichnung zweimal verwenden, wird ein Fehler ausgegeben. |

**Welches Schlüsselwort zur Variablendeklaration würden Sie generell vorziehen «var» oder «let»? Und aus welchem Grund?**

|  |
| --- |
| Ich würde generell das Schlüsselwort "let" zur Variablendeklaration vorziehen. Der Grund dafür ist, dass Variablen, die mit "let" deklariert wurden, auf Blockebene begrenzt sind und somit in einem bestimmten Block, wie einer Schleife oder einer bedingten Anweisung, deklariert und verwendet werden können, ohne dass sie die globale oder funktionale Umgebung beeinflussen. Im Gegensatz dazu sind Variablen, die mit "var" deklariert wurden, funktionsweite oder globale Variablen und können überall im Code verwendet werden, was möglicherweise zu unerwarteten Ergebnissen führen kann. Die Verwendung von "let" trägt daher dazu bei, die Sichtbarkeit und Kontrolle von Variablen im Code zu verbessern und unerwartete Verhaltensweisen zu vermeiden. |

**Deklarieren Sie eine Variable mit dem «const» Schlüsselwort und weisen Sie dieser einen Wert zu. Versuchen Sie danach, den Wert anzupassen. Was stellen Sie fest?**

|  |
| --- |
| Wenn eine Variable mit dem Schlüsselwort const deklariert wird, kann ihr Wert nicht geändert werden. Wenn versucht wird, den Wert der Variable zu ändern, wird eine Fehlermeldung angezeigt. |

**Deklarieren Sie eine Variable mit dem «const» Schlüsselwort und weisen Sie dieser den Wert «undefined» zu. Deklarieren Sie eine weitere Variable mit dem «const» Schlüsselwort und weisen Sie dieser den Wert «null» zu. Vergleichen Sie danach diese 2 Variablen mit dem Vergleichsoperator «==» oder «===». Was stellen Sie fest und können Sie dieses Resultat erklären?**

|  |
| --- |
| Wenn wir die beiden Variablen mit dem Vergleichsoperator == vergleichen, erhalten wir das Ergebnis false, da undefined und null verschiedene Typen sind.  Wenn wir jedoch den strikten Vergleichsoperator === verwenden, erhalten wir auch das Ergebnis false, obwohl beide Variablen unterschiedliche Werte und unterschiedliche Typen haben. Das liegt daran, dass der strikte Vergleichsoperator auch die Typen der Variablen berücksichtigt und undefined und null unterschiedliche Typen haben.  In JavaScript repräsentiert undefined den Wert einer Variable, die deklariert, aber nicht initialisiert wurde, während null den Wert einer Variablen darstellt, die bewusst auf den Wert null gesetzt wurde, um auszudrücken, dass keine Wertzuweisung erfolgt ist. |

**Welche Namenskonvention für das Benennen von Variablen haben Sie für die Übungen verwendet und welche ist in der Welt von JavaScript am beliebtesten?**

|  |
| --- |
|  |

**Aufgabe 4)**

Beschäftigen Sie sich mit dem Definieren von Strukturen und Objekten, um mehrere Variablen in grösseren Formationen zusammenfassen zu können. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**objects.js**».

**Deklarieren Sie eine Variable welche als Wert ein Objekt hat, dass mindestens 3 Eigenschaften und verschiedene Datentypen hat. Z.B. Person, Auto usw. Geben Sie nun die einzelnen Eigenschaften separat auf die Konsole aus.**

|  |
| --- |
| const person = {  name: "Max",  age: 30,  isStudent: true  };  console.log(person.name);  console.log(person.age);  console.log(person.isStudent); |

**Deklarieren Sie eine Variable welche als Wert ein Objekt hat, und dieses Objekt soll wiederum eine Eigenschaft besitzen, welche ein Objekt mit mehreren Eigenschaften hat. Ist das möglich und wie tief kann man Objekte verschachteln?**

|  |
| --- |
| Ja, es ist möglich, ein Objekt in einer Eigenschaft eines anderen Objekts zu verschachteln. In JavaScript gibt es keine direkte Begrenzung, wie tief man Objekte verschachteln kann. Theoretisch könnte man unendlich tief verschachtelte Objekte erstellen. Allerdings sollte man bedenken, dass zu tief verschachtelte Objekte möglicherweise schwer zu lesen und zu verstehen sind und möglicherweise zu Leistungsproblemen führen können. In der Praxis ist es oft sinnvoll, die Verschachtelung auf ein vernünftiges Maß zu begrenzen, um die Lesbarkeit und Leistung des Codes zu erhalten. |

**Aufgabe 5)**

Beschäftigen Sie sich mit Listen und den verschiedenen Operationen, welche auf diesen vorgenommen werden können. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**arrays.js**».

**Erstellen Sie eine Liste welche 3 Elemente vom Typ String enthält. Geben Sie diese in der Konsole aus.**

|  |
| --- |
| let myList = ["Element 1", "Element 2", "Element 3"];  console.log(myList); |

**Geben Sie nun das zweite Element aus. Wie kann direkt auf ein spezifisches Element zugegriffen werden?**

|  |
| --- |
| Um direkt auf ein spezifisches Element in einer Liste zuzugreifen, kann man den Index des Elements in eckigen Klammern nach dem Variablennamen angeben. Der Index beginnt bei 0, das heißt das erste Element hat den Index 0, das zweite den Index 1 usw. Um also das zweite Element auszugeben, könnte man folgenden Code verwenden:  let liste = ['Element 1', 'Element 2', 'Element 3'];  console.log(liste[1]);  Dies würde das zweite Element ("Element 2") auf der Konsole ausgeben. |

**Fügen Sie der zuvor erstellten Liste ein weiteres Element des Typs «Objekt» hinzu. Ist das möglich?**

|  |
| --- |
| Ja, es ist möglich, ein Objekt als Element einer JavaScript-Liste (Array) hinzuzufügen. Hier ist ein Beispiel:  let myList = ["Element 1", "Element 2", {name: "Objekt"}];  console.log(myList);  Dieses Beispiel erstellt eine Liste mit drei Elementen: den beiden Strings "Element 1" und "Element 2" und einem Objekt mit der Eigenschaft "name". Der Inhalt der Liste wird auf der Konsole ausgegeben. |

**Entfernen Sie das Element am Anfang der Liste und am Ende der Liste.**

|  |
| --- |
| Um das erste Element einer Liste zu entfernen, kann die shift()-Methode verwendet werden. Um das letzte Element einer Liste zu entfernen, kann die pop()-Methode verwendet werden.  Beispiel:  let liste = ["Element 1", "Element 2", "Element 3", {name: "Objekt"}];  liste.shift(); // entfernt "Element 1"  liste.pop(); // entfernt {name: "Objekt"}  console.log(liste); // gibt ["Element 2", "Element 3"] aus |

**Aufgabe 6)**

Beschäftigen Sie sich mit Kontrollstrukturen und wie mit diesen der Ablauf von Code gesteuert werden kann. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**control\_structures.js**».

**Erstellen Sie eine Variable welche mithilfe von Math.random() eine zufällige Zahl zwischen 0 und 1 als Wert erhält. Erstellen Sie danach eine Kontrollstruktur, die folgendermassen funktionieren soll.**

* **Wenn > 0.5 = Konsolenausgabe «Grösser als 0.5»**
* **Wenn erste nicht zutrifft aber > 0.25 = Konsolenausgabe «Grösser als 0.25»**
* **Wenn keines zutrifft = Konsolenausgabe «Grösser kleiner oder gleich 0.25»**

**Welche Kontrollstruktur verwenden Sie?**

|  |
| --- |
| let randomNumber = Math.random(); // generiere eine zufällige Zahl zwischen 0 und 1  console.log(randomNumber); // gib die zufällige Zahl auf der Konsole aus  Die Math.random()-Funktion gibt eine Pseudozufallszahl zwischen 0 (inklusive) und 1 (exklusive) zurück. Durch Speichern des Rückgabewerts in einer Variablen kann darauf zugegriffen werden, um später damit weiterzuarbeiten. |

**Erstellen Sie eine Variable welche mithilfe von «Math.floor(Math.random() \* 5)» eine zufällige ganzzahlige Zahl zwischen 0 und 5 als Wert erhält. Erstellen Sie danach eine Kontrollstruktur, die folgendermassen funktionieren soll.**

* **Wenn Zahl = 1 -> Konsolenausgabe «Der Wert der Zahl ist exakt 1»**
* **Wenn Zahl = 3 -> Konsolenausgabe «Der Wert der Zahl ist exakt 3»**
* **Wenn eine andere Zahl kommt -> Konsolenausgabe «Der Wert der Zahl ist weder 1 noch 3»**

**Welche Kontrollstruktur verwenden Sie?**

|  |
| --- |
| Hier ist eine Möglichkeit, diese Kontrollstruktur in JavaScript umzusetzen:  let randomNum = Math.random();  if (randomNum > 0.5) {  console.log("Grösser als 0.5");  } else if (randomNum > 0.25) {  console.log("Grösser als 0.25");  } else {  console.log("Grösser kleiner oder gleich 0.25");  }  Zuerst wird eine Zufallszahl zwischen 0 und 1 generiert und in der Variablen randomNum gespeichert. Dann wird eine if-else if-else-Struktur verwendet, um zu überprüfen, welche Bedingung zutrifft. Wenn randomNum größer als 0.5 ist, wird "Grösser als 0.5" in der Konsole ausgegeben. Wenn nicht, aber randomNum größer als 0.25 ist, wird "Grösser als 0.25" ausgegeben. Andernfalls wird "Grösser kleiner oder gleich 0.25" ausgegeben. |

**Aufgabe 7)**

Beschäftigen Sie sich mit Schleifen und wie mit diesen Listen oder auch Objekte verarbeitet werden können. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**loops.js**».

**Erstellen Sie eine Schleife, welche auf der Konsole von 0-10 hochzählt.**

**Welchen Schleifentyp verwenden Sie?**

|  |
| --- |
| Für diese Aufgabe kann man am besten eine for-Schleife verwenden:  for (let i = 0; i <= 10; i++) {  console.log(i);  }  Dies wird von 0 bis 10 durchlaufen und gibt jede Zahl auf der Konsole aus. |

**Erstellen Sie eine Schleife. Diese soll in jedem Durchlauf eine neue Zufallszahl zwischen 1-10 generieren diese ausgeben und wenn die Zahl grösser als 8 ist, soll die Schleife abgebrochen werden.**

**Welchen Schleifentyp verwenden Sie?**

|  |
| --- |
| ür diese Aufgabe eignet sich eine while-Schleife am besten, da die Anzahl der Durchläufe nicht vorher bekannt ist und die Schleife abgebrochen werden soll, wenn eine bestimmte Bedingung erfüllt ist.  Hier ist ein möglicher Lösungsansatz:  let randomNumber = 0;  while (randomNumber <= 8) {  randomNumber = Math.floor(Math.random() \* 10) + 1;  console.log(randomNumber);    if (randomNumber > 8) {  break;  }  }  Diese Schleife generiert in jedem Durchlauf eine neue Zufallszahl zwischen 1 und 10 und gibt sie auf der Konsole aus. Wenn die Zahl größer als 8 ist, wird die Schleife mit dem break-Schlüsselwort abgebrochen. |

**Erstellen Sie einen Array mit 5 beliebigen Elementen. Erstellen Sie danach eine Schleife, welche alle Elemente des Arrays durchläuft und jedes Element einzeln auf der Konsole ausgibt.**

**Welchen Schleifentyp verwenden Sie?**

|  |
| --- |
| Hier ist eine Lösung mit einer for-Schleife:  let myArray = ["apple", "banana", "cherry", "date", "elderberry"];  for (let i = 0; i < myArray.length; i++) {  console.log(myArray[i]);  }  Diese Schleife durchläuft das Array myArray mit einer inkrementellen Zählvariable i, beginnend bei 0 und endend bei der Länge des Arrays minus 1. Bei jedem Durchlauf wird das Element an der aktuellen Position i des Arrays mit myArray[i] auf der Konsole ausgegeben. |

**Aufgabe 8)**

Beschäftigen Sie sich mit Funktionen und wie diese dazu verwendet werden können JavaScript Befehle zu bündeln und wiederverwendbar zu machen. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**functions.js**».

**Definieren Sie eine Funktion, welche 2 Parameter entgegennimmt und aus diesen 2 Zahlen das Produkt bildet und auf der Konsole ausgibt. Rufen Sie dann die Funktion mit verschiedenen Parametern auf.**

|  |
| --- |
| function multiply(num1, num2) {  const result = num1 \* num2;  console.log(result);  }  multiply(2, 5); // Ausgabe: 10  multiply(3, 4); // Ausgabe: 12  multiply(7, 2); // Ausgabe: 14  In diesem Beispiel wird eine Funktion namens multiply definiert, die zwei Parameter num1 und num2 erwartet. Die Funktion multipliziert diese beiden Parameter und speichert das Ergebnis in der Variable result. Dann wird das Ergebnis auf der Konsole mit console.log ausgegeben.  Die Funktion wird dann drei Mal aufgerufen, jedes Mal mit verschiedenen Parameterwerten, um das Ergebnis der Multiplikation zu sehen. |

**Definieren Sie eine Funktion, welche 1 Parameter (Grad Celsius) entgegennimmt, diesen Wert in Fahrenheit umrechnet und auf der Konsole ausgibt. Rufen Sie dann die Funktion mit verschiedenen Parametern auf.**

ℹ️ Tipp: °F = °C \* 1,8 + 32

|  |
| --- |
| function celsiusToFahrenheit(celsius) {  const fahrenheit = (celsius \* 9/5) + 32;  console.log(`${celsius} Grad Celsius entsprechen ${fahrenheit} Grad Fahrenheit`);  }  // Beispielaufrufe  celsiusToFahrenheit(0);  celsiusToFahrenheit(20);  celsiusToFahrenheit(100);  In dieser Funktion wird der übergebene Wert in Fahrenheit umgerechnet und auf der Konsole ausgegeben. Die Formel zur Umrechnung von Celsius nach Fahrenheit lautet F = C \* 9/5 + 32. |

**Deklarieren Sie eine Variable, weisen Sie dieser Variable danach ein Lambda/Arrow Funktion zu, welche aus 2 Parametern das Modulo bestimmt. Rufen Sie dann die Funktion mit verschiedenen Parametern auf. Welcher Unterschied besteht zwischen einer regulären Funktion und einer Lambda/Arrow Funktion? Verwenden Sie dazu Google.**

|  |
| --- |
|  |

**Was können Sie zum Schlüsselwort «this» in JavaScript sagen? Verwenden Sie dazu Google.**

|  |
| --- |
| In JavaScript bezieht sich das Schlüsselwort "this" auf das Objekt, auf das eine Funktion angewendet wird oder das den Kontext der aktuellen Funktion darstellt. Es wird verwendet, um auf Eigenschaften und Methoden des Objekts zuzugreifen, das die Funktion aufruft. Das Verhalten von "this" hängt davon ab, wie die Funktion aufgerufen wird.  Es gibt vier Möglichkeiten, wie "this" in JavaScript festgelegt wird:   1. Globales "this": Wenn "this" außerhalb einer Funktion oder eines Objekts aufgerufen wird, bezieht es sich auf das globale Objekt, das im Browserfenster das "window"-Objekt und im Node.js-Umfeld das "global"-Objekt ist. 2. Objekt "this": Wenn eine Methode von einem Objekt aufgerufen wird, bezieht sich "this" auf das Objekt selbst. 3. Explizites "this": Wenn "this" explizit durch die Verwendung von "call", "apply" oder "bind" festgelegt wird, wird es auf das angegebene Objekt festgelegt. 4. Arrow-Funktion "this": In Arrow-Funktionen wird "this" nicht neu gebunden und bezieht sich stattdessen auf das "this" des umgebenden Kontexts, in dem sie definiert wurden.   Es ist wichtig, "this" richtig zu verstehen und zu verwenden, um unerwartete Fehler in Ihrem Code zu vermeiden. |

**Aufgabe 9)**

Machen Sie sich mit NodeJS und mindestens einem Paket Manager vertraut. Danach erstellen Sie einen simplen Express Webserver, der einen GET-Endpunkt zur Verfügung stellt und vom Browser aus zugegriffen werden kann. Erstellen Sie für diese Aufgabe eine Datei mit dem Namen «**express\_webserver.js**»

**Initialisieren Sie ein neues NodeJS Projekt. Wozu dient die package.json Datei in einem NodeJS Projekt?**

ℹ️ Tipp: «npm init oder yarn init»

|  |
| --- |
| Die package.json-Datei ist eine Konfigurationsdatei für ein Node.js-Projekt. Sie wird im Stammverzeichnis des Projekts erstellt und enthält Metadaten über das Projekt wie z.B. den Projektnamen, die Version, Autor, Beschreibung und Abhängigkeiten.  Einige der wichtigsten Funktionen der package.json-Datei sind:   1. Abhängigkeiten verwalten: Die Datei listet die Abhängigkeiten des Projekts auf, einschließlich der Versionen, die benötigt werden, um das Projekt auszuführen. Diese Abhängigkeiten können entweder als direkte Abhängigkeiten oder als indirekte Abhängigkeiten durch andere Abhängigkeiten hinzugefügt werden. Die package.json-Datei wird von NPM (Node Package Manager) verwendet, um die Abhängigkeiten herunterzuladen und zu installieren. 2. Scripte definieren: Die package.json-Datei ermöglicht es, benutzerdefinierte Skripte zu definieren, die im Projekt ausgeführt werden können. Zum Beispiel können Skripte zum Starten, Testen oder Bereitstellen des Projekts definiert werden. 3. Metadaten bereitstellen: Die package.json-Datei enthält auch Metadaten über das Projekt wie z.B. den Projektnamen, die Version, den Autor und die Beschreibung. Diese Informationen können verwendet werden, um das Projekt zu beschreiben und anderen Entwicklern zu helfen, das Projekt besser zu verstehen.   Insgesamt dient die package.json-Datei als zentrales Konfigurations- und Informationsdokument für ein Node.js-Projekt. Es hilft bei der Verwaltung von Abhängigkeiten und Skripten, ermöglicht die Bereitstellung von Metadaten über das Projekt und erleichtert es anderen Entwicklern, das Projekt zu verstehen und zu verwenden. |

**Installieren Sie das Paket «express» mit einem beliebigen Paket Manager. Wozu dient ein Paket Manager in NodeJS oder auch andere Programmiersprachen?**

ℹ️ Tipp: «npm install <Paket> oder yarn add <Paket>»

|  |
| --- |
| Ein Paket-Manager ist eine Software, die verwendet wird, um die Installation, Verwaltung und Aktualisierung von Bibliotheken, Frameworks und anderen wiederverwendbaren Code-Paketen in einem Projekt zu vereinfachen. In Node.js wird der Paket-Manager NPM (Node Package Manager) verwendet, während andere Programmiersprachen eigene Paket-Manager haben, wie z.B. RubyGems für Ruby, Composer für PHP oder pip für Python.  Die Hauptvorteile eines Paket-Managers sind:   1. Einfache Verwaltung von Abhängigkeiten: Ein Paket-Manager ermöglicht es Entwicklern, Abhängigkeiten in einem Projekt einfach zu installieren, zu aktualisieren und zu entfernen. Statt jeden Code manuell herunterzuladen, kann der Paket-Manager die Abhängigkeiten automatisch herunterladen und installieren, basierend auf einer Konfigurationsdatei wie der package.json-Datei. 2. Wiederverwendbarkeit von Code: Ein Paket-Manager ermöglicht es Entwicklern, auf eine Vielzahl von wiederverwendbaren Code-Paketen zuzugreifen, die von anderen Entwicklern erstellt wurden. Dies reduziert den Zeitaufwand und die Kosten für die Entwicklung von Anwendungen, da Entwickler nicht jedes Mal von Grund auf neu beginnen müssen. 3. Versionskontrolle: Ein Paket-Manager ermöglicht es Entwicklern, die Versionen von Abhängigkeiten zu verwalten, um sicherzustellen, dass sie mit der aktuellen Version der Anwendung kompatibel sind. Entwickler können auch ältere Versionen von Abhängigkeiten verwenden, um Probleme mit neueren Versionen zu vermeiden.   Insgesamt macht der Einsatz eines Paket-Managers die Entwicklung von Anwendungen schneller, einfacher und zuverlässiger, indem er die Verwaltung von Abhängigkeiten automatisiert und den Zugriff auf wiederverwendbaren Code erleichtert. |

**Versuchen Sie nun mithilfe der Folien und des Internets einen simplen Webserver zum Laufen zu bringen der, wenn Sie «http://localhost:3000/hello» in Ihrem Browser aufrufen, eine entsprechende Antwort wie zum Beispiel «Hallo Welt» zurückliefert.**

|  |
| --- |
|  |

**Erweitern Sie nun Ihren Webserver um einen weiteren GET API-Endpunkt, dieser soll in der Route einen Parameter akzeptieren «http://localhost:3000/hello/:myName» und diesen in der Antwort verwenden.**

[**https://masteringjs.io/tutorials/express/route-parameters**](https://masteringjs.io/tutorials/express/route-parameters)

|  |
| --- |
|  |

**Welche HTTP-Methoden kennen Sie und wozu Sie und wozu dienen diese jeweils?**

|  |
| --- |
| Es gibt mehrere HTTP-Methoden, die in Webanwendungen und APIs verwendet werden, um Aktionen auf Serverressourcen auszuführen. Die wichtigsten HTTP-Methoden sind:   1. GET: Die GET-Methode wird verwendet, um Daten von einem Server abzurufen. Wenn ein Client eine HTTP-GET-Anforderung an einen Server sendet, gibt der Server normalerweise Daten zurück, die vom Client angefordert wurden. 2. POST: Die POST-Methode wird verwendet, um Daten an einen Server zu senden, um sie zu verarbeiten oder zu speichern. Wenn ein Client eine HTTP-POST-Anforderung an einen Server sendet, sendet der Client normalerweise Daten im Body der Anforderung an den Server, der diese Daten dann verarbeitet oder speichert. 3. PUT: Die PUT-Methode wird verwendet, um Daten an einen Server zu senden, um sie zu aktualisieren. Wenn ein Client eine HTTP-PUT-Anforderung an einen Server sendet, sendet der Client normalerweise Daten im Body der Anforderung an den Server, um die vorhandenen Daten zu aktualisieren. 4. DELETE: Die DELETE-Methode wird verwendet, um Daten von einem Server zu löschen. Wenn ein Client eine HTTP-DELETE-Anforderung an einen Server sendet, löscht der Server normalerweise die entsprechenden Daten. 5. PATCH: Die PATCH-Methode wird verwendet, um Daten zu aktualisieren, ähnlich wie PUT. Der Unterschied besteht darin, dass bei der PATCH-Methode nur die spezifischen Daten aktualisiert werden, die im Body der Anforderung enthalten sind, während bei PUT alle Daten in der Ressource aktualisiert werden. 6. HEAD: Die HEAD-Methode ähnelt der GET-Methode, gibt jedoch nur die Header der Antwort zurück, ohne den Body der Antwort zu senden. Dies wird normalerweise verwendet, um zu überprüfen, ob eine Ressource verfügbar ist oder nicht, ohne die Daten tatsächlich abzurufen. 7. OPTIONS: Die OPTIONS-Methode wird verwendet, um zu überprüfen, welche Methoden auf einer bestimmten Ressource verfügbar sind. Ein Client sendet eine OPTIONS-Anforderung an den Server, und der Server gibt eine Liste der verfügbaren Methoden in der Antwort zurück.   Diese HTTP-Methoden sind standardisiert und werden von Webanwendungen und APIs verwendet, um Aktionen auf Serverressourcen auszuführen. |

**Wozu kann nun Express oder allgemein ein Webserver dieser Art verwendet werden?**

|  |
| --- |
| Express ist ein Framework für Node.js, das verwendet wird, um Webanwendungen und APIs zu erstellen. Ein Webserver wie Express kann für eine Vielzahl von Anwendungsfällen verwendet werden, darunter:   1. Entwicklung von Webanwendungen: Express bietet eine einfache Möglichkeit, Serverseiten-Webanwendungen zu erstellen, die dynamischen Inhalt generieren. Mit Express können Entwickler Routen definieren, die URLs mit bestimmten Server-Code-Operationen wie der Abfrage von Datenbanken, der Verarbeitung von Benutzerdaten und der Erstellung von HTML-Seiten verknüpfen. 2. Entwicklung von APIs: Express kann auch verwendet werden, um RESTful-APIs (Representational State Transfer) zu erstellen, die eine standardisierte Möglichkeit bieten, um Daten zwischen Servern und Clients auszutauschen. Mit Express können Entwickler Routen definieren, die bestimmte HTTP-Methoden wie GET, POST, PUT und DELETE unterstützen und so Daten im JSON-Format bereitstellen. 3. Verwaltung von Anforderungen und Antworten: Express kann verwendet werden, um den Umgang mit HTTP-Anforderungen und -Antworten auf einem Webserver zu vereinfachen. Es bietet Funktionen wie Middleware, um den Zugriff auf die HTTP-Anforderung und -Antwort zu erleichtern, sowie das Routing, um Anfragen an bestimmte Endpunkte weiterzuleiten und entsprechende Antworten zu generieren. 4. Auslieferung von statischen Ressourcen: Express kann auch verwendet werden, um statische Ressourcen wie HTML-Seiten, CSS-Stylesheets, Bilder und andere Dateien auf einem Webserver auszuliefern.   Insgesamt bietet Express eine flexible und einfach zu verwendende Möglichkeit, um Webanwendungen und APIs auf einem Node.js-Webserver zu entwickeln. Es erleichtert die Verwaltung von Anforderungen und Antworten, ermöglicht die einfache Erstellung von Routen und bietet eine Vielzahl von Funktionen, um die Entwicklung von Webanwendungen zu vereinfachen. |

**Aufgabe 10)**

Erstellen Sie einen github Account. Sie werden in diesem Kurs alle Ihre Aufgaben welche Source Code beinhalten auf diese Plattform hochladen. Achten Sie dabei immer, dass Ihre Repositories öffentlich zugänglich sind.

**Erstellen Sie einen Account. Um was handelt es sich bei Github?**

<https://github.com/signup>

|  |
| --- |
| GitHub ist ein webbasierter Hosting-Service, der hauptsächlich für die Versionsverwaltung von Softwareprojekten verwendet wird. Es ist eines der bekanntesten und meistgenutzten Online-Repository-Hosting-Services, der Millionen von Nutzern und Projekten beherbergt.  GitHub bietet Entwicklern die Möglichkeit, ihre Quellcode-Repositories online zu speichern und zu verwalten. Es verwendet das Git-Versionierungssystem, um Änderungen am Quellcode zu verfolgen und zu verwalten. Git ist ein verteiltes Versionskontrollsystem, das es Entwicklern ermöglicht, Änderungen an Quellcodedateien auf ihren lokalen Systemen vorzunehmen und diese Änderungen dann auf den zentralen Server (GitHub) hochzuladen.  GitHub bietet eine Vielzahl von Funktionen, darunter Issue-Tracking, Pull-Requests, Wikis, Code-Review-Tools und viele andere Funktionen, die die Zusammenarbeit zwischen Entwicklern erleichtern. GitHub ist auch ein wichtiger Beitrag zur Open-Source-Community, da es Entwicklern ermöglicht, ihre Codeprojekte öffentlich zugänglich zu machen, wodurch andere Entwickler den Code lesen, verbessern und beitragen können. |

**Installieren Sie git auf Ihrem Hostsystem. Um was handelt es sich bei git?**

<https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>

ℹ️ Tipp: «git init»

|  |
| --- |
|  |

**Was ist der Unterschied zwischen git und github?**

|  |
| --- |
| Git und GitHub sind zwei unterschiedliche Dinge, die oft miteinander verwechselt werden. Git ist ein Versionskontrollsystem, während GitHub ein webbasierter Hosting-Service ist, der Git-Repositories hostet.  Git ist eine Software zur Versionskontrolle von Quellcode, die von Linus Torvalds entwickelt wurde. Git ermöglicht es Entwicklern, Änderungen an ihren Codeprojekten zu verfolgen, zu speichern und zu verwalten. Mit Git können Entwickler ihre Codeprojekte auf ihren lokalen Systemen verwalten und Änderungen an diesen Projekten verfolgen. Git ist ein verteiltes Versionskontrollsystem, was bedeutet, dass jeder Entwickler eine lokale Kopie des Code-Repositories auf seinem System hat.  GitHub hingegen ist ein webbasierter Hosting-Service für Git-Repositories. Es bietet Entwicklern eine Möglichkeit, ihre Git-Repositories online zu speichern und zu verwalten. GitHub bietet viele Funktionen wie Issue-Tracking, Pull-Requests, Wikis und Code-Review-Tools, die Entwicklern helfen, ihre Codeprojekte zusammenzuarbeiten und zu verwalten. Es ist auch ein wichtiger Beitrag zur Open-Source-Community, da es Entwicklern ermöglicht, ihre Codeprojekte öffentlich zugänglich zu machen, was anderen Entwicklern die Möglichkeit gibt, den Code zu lesen, zu verbessern und beizutragen.  Zusammenfassend lässt sich sagen, dass Git ein Versionskontrollsystem ist, das Entwicklern hilft, Änderungen an ihrem Codeprojekt zu verfolgen, während GitHub ein webbasierter Hosting-Service ist, der Git-Repositories hostet und viele Funktionen zur Zusammenarbeit zwischen Entwicklern bereitstellt. |

**Initialisieren Sie lokal ein neues git Repository und commiten Sie den Source Code, der in diesem Arbeitsblatt angefallen ist. Achten Sie darauf, dass Sie nur die Dateien commiten welche relevant sind. Alle anderen Dateien sollen in der Datei «.gitignore» aufgelistet werden damit diese nicht aus Versehen commited werden.**

ℹ️ Tipp: «git add . && git commit -m ‘Initial commit’»

|  |
| --- |
|  |

**Erstellen Sie unter Ihrem github Account ein neues passendes Repository mit sprechender Bezeichnung. Verknüpfen Sie dieses Repository und pushen Sie den Source Code auf github. Fügen Sie danach den Link zu Ihrem Repository in die untenstehende Textbox ein.**

|  |
| --- |
|  |

**Aufgabe 11) (Optional)**

Falls Sie mit dem Arbeitsblatt bereits fertig sind, können Sie sich den nachfolgenden Onlinekurs anschauen und selbstständig anfangen die Übungen zu erledigen. Das kann Ihnen zusätzlich helfen, in das ganze Thema hereinzukommen.

<https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/>

💡 Um die Übungen und den Fortschritt speichern zu können, müssen Sie einen gratis Account auf freecodecamp erstellen.