* Node doet veel dingen asynchroon, waardoor het best handig is om te weten hoe asynchroon programmeren werkt.
* Node gebruikt intern een buffer, die minder geheugen opneemt. Dit werkt sneller dan platte tekst. Om dit naar tekst te converteren, kun je toString gebruiken, of utf-8 als extra parameter.
* Directory aangeven: Windows gebruikt \\ om een map te scheiden.
  + De path module zorgt ervoor dat alle path’s universeel werkt door te detecteren welke besturingsysteem de gebruiker gebruikt.
* De server blijft bezig met het afhandelen van requests. Bij synchroon kan de server maar 1 handeling tegelijk uitvoeren, waar dit bij asynchroon niet het geval is. Het zorgt er alleen voor dat servers requests af kan handelen.
* Je kunt geen return waarde teruggeven bij een asynchrone functie.
* Belangrijk om te weten hoe de geheugenmodel voor asynchroon werkt, maar niet tekenen.
* Een callback is een functie definitie. Geen haakjes gebruiken hierbij.
* De C++ deel in het geheugenmodel is niet onderdeel van de javascript engine. Dit gedeelte is door Node geleverd. Hierin wordt de functie van readFile daadwerkelijk uitgevoerd. Het wordt ook een web API genoemd. Zie voorbeeld geheugenmodel.
  + Asynchrone functies kunnen pas uitgevoerd worden als de stack leeg is.
* Een thread is vergelijkbaar met eens tack.
* Nadeel is dat je zelf moet beheren wat je op de stack zet. Alle werk wat je niet tegelijk wilt doen, laat je door Node doen.
* Er wordt een nieuwe functie in de heap aangemaakt bij een functie definitie.
* Volgorde van asynchrone functies in het geheugenmodel hangt af van welke eerder klaar is.
  + Er vanuit gaan dat asynchrone functies tegeleijk gebeuren.
* De asynchrone functie wordt pas uitgevoerd wanneer de stack leeg is, waardoor de variabelen die gedefinieerd zijn in het programma niet meer bestaan.
  + Op te lossen door een globale variabele aan te maken.
* Closure is als een functie data uit het buitenste scope gebruikt, dus een globale variabele die door een ander functie daarbinnen wordt gebruikt.
* Foreach wordt in de javascript/v8 omgeving gedraaid, tegenover de readFile functie die in de c++ omgeving wordt uitgevoerd.
* Je kan om een asynchrone functie geen try-catch eromheen bouwen, maar wel daarbinnen.
  + Daarvoor wordt de error variabele gebruikt binnen als eerste parameter.
* Return in een asynchrone functie wordt alleen gebruikt om de programma te stoppen met het uitvoeren van de programma. Dit is handig bij error handling.
* De delay bij setTimeout is relevant bij het bepalen van de volgorde van de output.
* Assignment uitleg
  + Check of een plek bestaat in de exit array
  + Check of een plek bestaat in de map array
    - Zo niet, haal deze van de map server m.b.v. een http request
    - Belangrijk om de map server te runnen voordat http requests gedaan kan worden.
    - Request en json.parse functie. Uitzoeken via de api.
    - De tweede parameter moet een asynchronische functie zijn.
    - Twee terminals nodig om te testen.