* Runtime omgeving = een specifieke omgeving
* Compiler = taal die converteert naar een taal die de computer kan lezen.
* Niet veel verschil tussen binding en variabele. Binding is technisch gezien goed.
* Puntkomma bij javascript niet verplicht
* Reference errors crashen je webprogramma
* ES6 Javascript
* Var die onder wordt gedefinieerd wordt niet gelezen
* Een variabele kan een functie toegewezen krijgen.
* Geen public of private bij javascript
* Als een functie klaar is met uitvoeren, worden alle lokale variabelen verwijderd
  + Javascript closure
* Geen var en let definiëren = een variabele is globaal beschikbaar
* Geen ingebouwde functies opnieuw herschrijven
* Module.export exporteert functies van een bestand, en om deze aan te roepen, maken we gebruik van require.

# Geheugenmodel

* Tekenen waar variabelen zit
* Variabele komt op de stacks, en de variabelen binnen de functie komen ook in de stack.
* Functie definities komen in de heap en er wordt in de stack een verwijzing gemaakt als dat nodig is
* Als een functie aangeroepen wordt, moeten alle variabelen als eerst opgenomen worden
  + nrTimes
  + func
  + i
* Verwijzingen zijn belangrijk
* Geheugenmodel kan alleen getekend worden wanneer het moment duidelijk is dat de programma is uitgevoerd
* Het kan handig zijn om een geheugenmodel te tekenen bij het debuggen
* Bij variabelen hebben functies in de heap geen functienamen, maar functies zelf wel.
* Draw io gebruiken om SVG bestanden aan te passen. Visual studio code plugin kan dit ook.

# Pure functies

* Niet puur is als het omgeving de resultaat van een functie bepaald, bijvoorbeeld console.log binnen de functie.
* Pure functies zijn vooral afhankelijk van zichzelf, en niet van het resultaat.
* Puur is stricter dan immutable
  + Immutable opzoeken