LINGO

We gaan het spelletje Lingo maken. In het spel moet je een woord van vijf letters raden. We beginnen eenvoudig, maar je kunt het natuurlijk zelf moeilijker maken, door meer woorden of langere woorden te gebruiken.

Voor het spel heb je drie bestanden nodig:

- Lingo.css
- Lingo.html
- Lingolib.js

Je vindt deze bestanden in de map **Javascript\lingo**.

> Start eerst Visual Studio Code als je dit programma nog niet geopend hebt.

1. In Visual Studio Code kies je voor **File > Open Folder…**
2. Ga dan naar de map **Javascript\lingo** en kies **“Select map”**.
3. Kies nu **File > New File** om een nieuw bestand te maken.
4. Sla het lege bestand direct op via **File > Save** en geef het de naam **lingo.js**.

var

Met het sleutelwoord **var** geef je in Javascript aan dat er een variabele wordt gedefinieerd. Een variabele verwijst naar een stukje geheugen in de computer waar je gegevens in kunt opslaan. Je kan zo’n variabele zelf een naam geven. Zoals:

```
var naam;

var getal;

var heleLangeTekst;
```

Valt je nog iets op? We zetten aan het einde van de regel altijd een puntkomma **;** . Hiermee vertel je de computer dat het einde van de regel is bereikt en dat er een nieuwe instructie volgt. Je kunt alles op één regel zetten, met puntkomma’s er tussen, maar dat is voor ons mensen lastiger lezen. De computer maakt het niet uit. Die wil wel **;** zien.

We willen bijhouden hoeveel poging er nodig zijn om het woord te raden. Daarom maken we een variabele.

```
var poging = 0;
```

We kunnen bij het definiëren van een variabele gelijk opgeven wat de beginwaarde moet zijn. We beginnen dus bij 0.

array

Een array is een lijst van waarden. Dat kunnen woorden zijn, of getallen, of een combinatie ervan.

Ook een array kan je in een variabele stoppen. De volgende regel zet een lijst van woorden in een variabele.

```
var woorden = ["fiets","meten","patat","zwaan"];
```

function

Een functie is een verzameling coderegels die logisch bij elkaar horen. Dat heeft een paar voordelen. Je code wordt er leesbaarder van. En als je de coderegels op verschillende plekken opnieuw wil gebruiken hoef je die maar op één plaats bij te houden.

Uit de lijst van woorden kiezen we willekeurig één woord. Dat doen we met een functie. Het sleutelwoord voor een functie in Javascript is **function**. Net als bij variabelen mag je zelf kiezen welke naam de functie moet krijgen.

Onze functie ziet er zo uit.

```
function kiesWoord()
{
    var random = Math.random() * woorden.length;
    var woord = woorden[Math.floor(random)];
    return woord;
}
```

Je ziet, er zijn enkele regels waar we ons aan moeten houden bij het maken van een functie.

1. Vóór de naam van de functie schrijven we **function** zodat de computer weet dat er een functie gedefinieerd wordt.
2. Achter de functie staan haakjes **()**. Dat is ook verplicht. [1]
3. De coderegels die horen bij de functie moeten tussen accolades staan: **{** en **}**

Wat gebeurt er nu in deze functie? Lees de regels zelf eerst eens goed om te kijken of je het al snapt.

De functie doet het volgende:

1. We maken een variabele **random** en vullen die met een willekeurig getal die ten hoogste de waarde van het aantal woorden in de lijst kan hebben.

[1] Als je meer met functies werkt zal je leren dat je ook variabelen kunt doorgeven aan een functie. We noemen dat parameters.

2. Dan maken we een variabele **woord** en die bevat het woord uit de lijst van woorden op de positie **random**. In Javascript begint een lijst trouwens altijd bij 0. Dus het eerste woord staat op plek 0, het tweede op 1 enzovoort.
3. Tot slot stuurt de functie het gekozen woord terug naar de regel waar de functie wordt aangeroepen.

Nu moeten we die functie nog gebruiken. Een functie doet op zichzelf namelijk niks. Met de volgende regel zetten we het gekozen woord in een variabele.

```
var woord = kiesWoord();
```

HTML

Een programma in Javascript werkt in een browser, zoals Google Chrome, Firefox en Internet Explorer. Zo'n browser leest HTML. Dat is eigenlijk tekst, maar er zijn een paar bijzondere tekstelementen afgesproken zodat de browser weet dat hij bepaalde tekst moet tonen als:

- Een keuzelijst (**<select>**)
- Een invoerveld (**<input type="text">**)
- Een knop (**<button>** maar ook **<input type="submit">**)
- Een tabel (**<table>**)
- Een paragraaf (**<p>**)

Dit is maar een kleine selectie van de elementen. In ons Lingo-spel gebruiken we HTML zodat de speler een woord kan invoeren en de uitkomst kan zien.

Bekijk maar eens de inhoud van het bestand **lingo.html**.

Je ziet hier onder meer staan

```
<input type="text" id="invoer">

<input type="button" id="controleer" value="controleer">

<input type="text" id="uitvoer">
```

We gebruiken deze elementen om een woord in te tikken, te laten controleren en te zien wat de uitkomst is.

In de HTML staat ook

```
<script src="lingolib.js" ></script>
```

Deze regel zorgt ervoor dat we in ons Javascript een paar hulpfuncties kunnen gebruiken. Deze hulpfuncties maken het ons in de volgende functie om het woord te controleren, wat makkelijker.

Invoer verwerken

We maken een functie om het woord te controleren. Weet je nog hoe je een functie begint?

Een functie begint altijd zo:

```
function EenNaamDieJeZelfKiest()
{
```

We zetten hieronder eerst de hele functie om een woord te controleren, en leggen dan uit wat deze functie doet.

```
function controleerWoord()
{
    var invoer = LeesWaarde("invoer");
    var resultaat = "";
    for(var i=0; i<5; i++)
    {
        if(invoer[i] == woord[i])
        {
            resultaat += woord[i];
        }
        else
        {
            resultaat += "*";
        }
    }
    SchrijfWaarde("uitvoer", resultaat);
    poging++;
}
```

Het eerst wat je in de functie ziet is

```
    var invoer = LeesWaarde("invoer");
```

We zetten in een variabele de waarde die de speler heeft ingevoerd. De functie **LeesWaarde(…)** is een hulpfunctie. Als je wilt weten wat deze functie doet, kijk dan in **lingolib.js**.

```
var resultaat = "";
```

Het resultaat van de controle stoppen we in de variabele **resultaat**. Deze is in het begin leeg.

```
for(var i=0; i<5; i++)
```

Deze instructie heb je waarschijnlijk nog niet eerder gezien. De instructie **for** is de start van een zogenaamde lus of loop. De hele for-regel vertelt de computer dat het stukje code dat erna komt 5 keer moet worden uitgevoerd. We controleren dus letter voor letter.

```
if(invoer[i] == woord[i])
  {
      resultaat += woord[i];
  }
  else
  {
      resultaat += "*";
  }
```

We controleren letter voor letter of het woord dat de speler heeft ingevoerd klopt met de letter van het willekeurig gekozen woord. Als de letter klopt (vandaar het sleutelwoord **if**) voegen we die letter toe aan het resultaat. Als de letter niet klopt (**else**) dan voegen een sterretje (*****) toe.

Daarna sturen we het resultaat naar het scherm. Ook hiervoor gebruiken we een hulpfunctie.

```
SchrijfWaarde("uitvoer", resultaat);
```

Tot slot verhogen we het aantal pogingen met 1.

```
poging++;
```

We kunnen dit ook schrijven als **poging = poging + 1**. Maar **poging++** is korter.

Gebeurtenissen

Nu moeten we er nog voor zorgen dat het ingevoerde woord ook echt wordt gecontroleerd. De computer moet daarvoor eerst weten dat je speler het woord heeft ingevoerd. Daarom staat er een knop op het scherm. Als de speler het woord heeft ingetikt, drukt deze op de knop **controleer**. Het drukken op een knop noemen we een gebeurtenis, ofwel **event** in het Engels. Met de volgende hulpfunctie koppelen we aan deze gebeurtenis de functie **controleerWoord**.

```
VerbindKlik("controleer", controleerWoord);
```

Zo, het spelletje is nu klaar.

Je kunt het natuurlijk zelf uitbreiden. Zo hebben we nog niets gedaan met het aantal pogingen. Wat je dus kunt doen is:

1) Het aantal pogingen bijhouden
2) Bij een maximaal aantal pogingen stoppen.
3) Meer woorden toevoegen.
4) Langere woorden toevoegen.