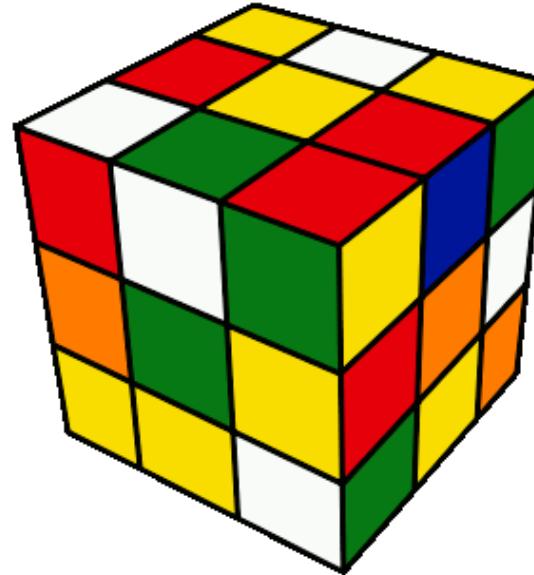


JS

AN ARRAY PRIMER



REACOLLEGE

Versie 1.0 – Maart 2024

Patrick Schevers / Wim Cuijten / Marco Bleekrode

INHOUDSOPGAVE

JS

GA NAAR EEN HOOFDSTUK DOOR OP DE TITEL TE KLIKKEN

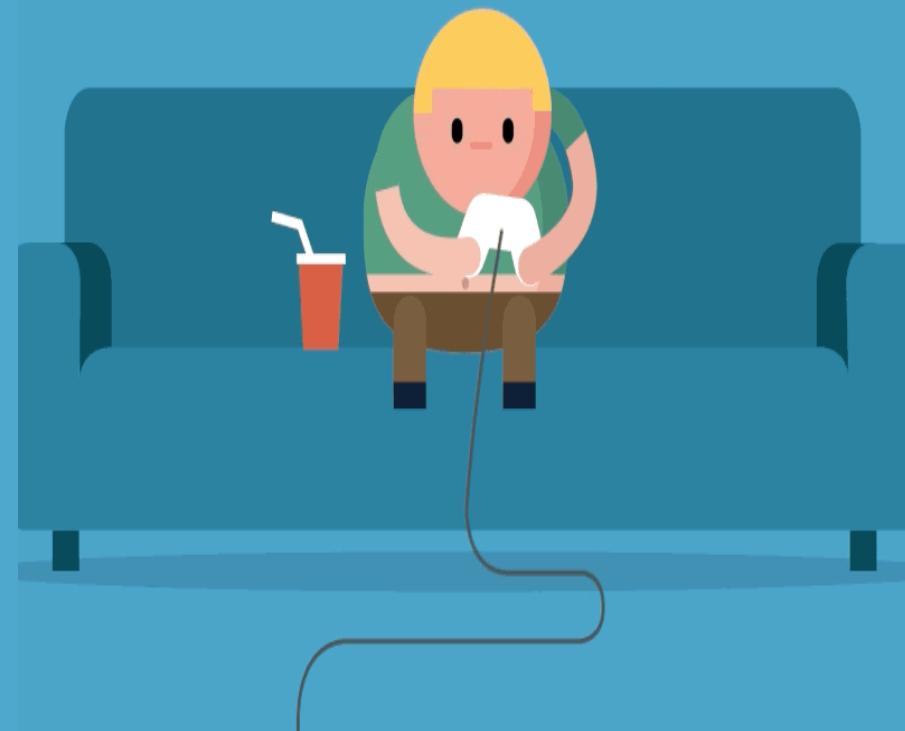
R
E
A
C
O
L
L
E
G
E

1. [Eerst even dit...](#)
2. [Wat is een array?](#)
3. [Hoe gebruiken we een array?](#)
4. [Raadsels in meer dimensies](#)
5. [Tips & tricks](#)
6. [Debuggen](#)
7. [En tenslotte nog dit...](#)
8. [Appendix - Een scenario](#)

REACOLLAGE

EERST EVEN DIT...

JS



EERST EVEN DIT...

JS

OVER DEZE PRESENTATIE

- Bedoeld voor studenten die graag JavaScript-“arrays” nog een keer uitgelegd willen zien.
- Is voor een aanzienlijk deel in een “verhalende” vorm om de lessen begrijpelijkér én onderhoudender te maken.
- Waar nodig worden onderwerpen behandeld die met programmeren in het algemeen en “arrays” in het bijzonder te maken hebben.
- De uitleg en het maken van opdrachten is klassikaal.
- Het is de bedoeling om maximaal 4 uur aan de hele presentatie te besteden.

EERST EVEN DIT...

JS

WAT LEVERT HET VOLGEN VAN DEZE PRESENTATIE OP?

- Beter begrip van arrays in JavaScript.
- Wat programmeer-“tips & tricks”.
- Gebruik van de JavaScript-debugger.

EERST EVEN DIT...

JS

WAT HEB JE WEL/NIET NODIG?

- Je telefoon heb je niet nodig.
- Een computer (PC, Mac of Linux-box) met een moderne browser én Visual Studio Code komt zeker van pas. 😊
- Voor de REA-PCs heb je een netwerkverbinding nodig. Je moet wel kunnen inloggen.
- Het lesmateriaal is onmisbaar. Pak het aangeleverde ZIP-bestand uit. Waar je de bestanden neerzet mag je zelf weten.

EERST EVEN DIT...

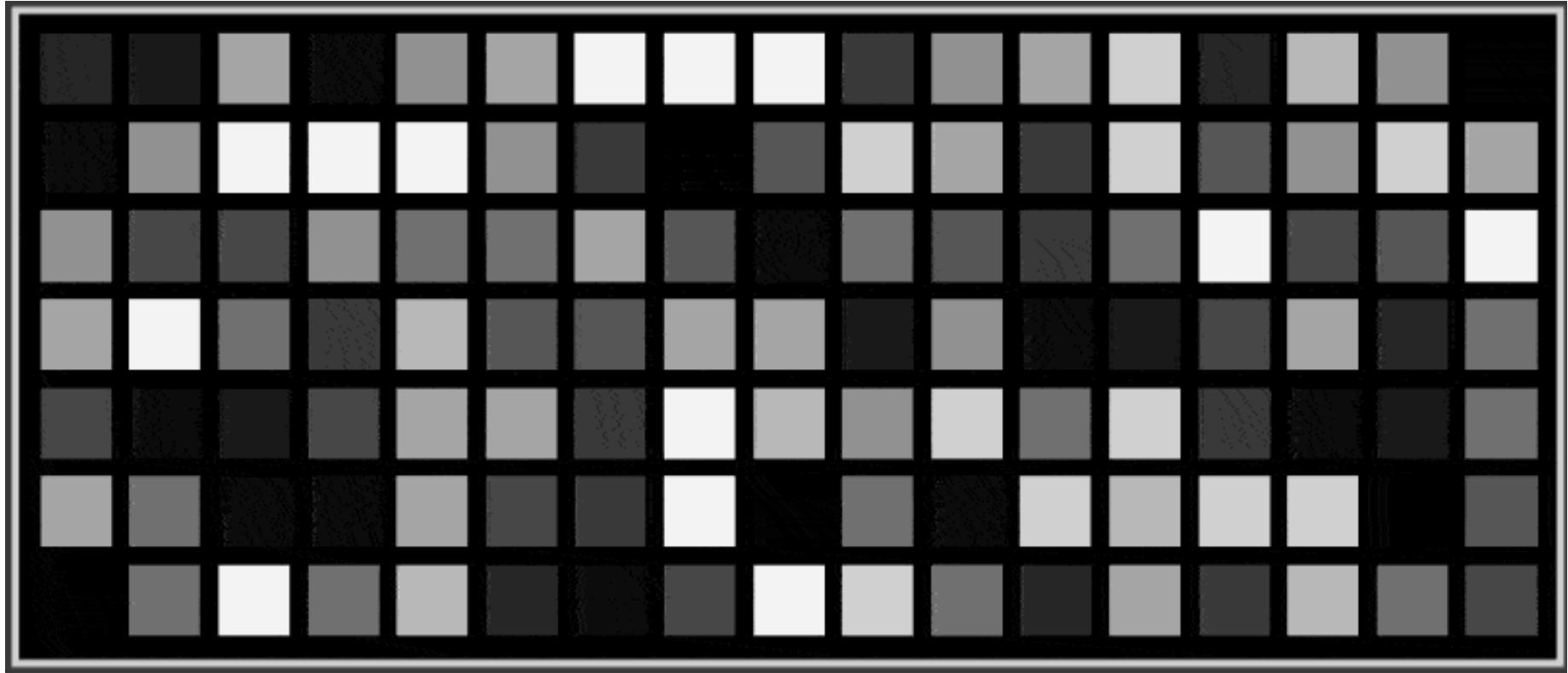
JS

OVER HET LESMATERIAAL

- In de “an-array-primer”-directory staat een PDF-versie van deze presentatie en een aantal HTML-bestanden die in de appendix gebruikt worden.
- In de “javascript-testomgeving”-directory staat de webapp waarmee je eenvoudige JavaScript-programma's kunt maken en testen.
- In de “an-array-primer/code”-directory staan alle voorbeelden en oplossingen t/m opdracht 12.
- De “an-array-primer/data”-directory bevat een JavaScript-bestand dat gebruikt wordt in de scenario-appendix.

WAT IS EEN ARRAY?

JS



WAT IS EEN ARRAY?

JS

WAT KOMT ER AAN BOD?

- Wat betekent het Engelse woord "array"?
- Wat doe je als je declareert en definieert?
- Je maakt een array.
- Samenvatting.
- Vragenuurtje.

WAT IS EEN ARRAY?

JS

ENGELS - NEDERLANDS

- Het woord "array" (hier een zelfstandig naamwoord) is afkomstig uit het Engels.
- Google Translate vertaalt "array" als "reeks", "rij", "rangschikking" of "slagorde".
- DeepL maakt er behalve "reeks" of "rij" ook nog "matrix" van.

WAT IS EEN ARRAY?

JS

DE DEFINITIE VAN EEN ARRAY

- Je weet wat een "variabele" is in JavaScript. Zo niet, dan heb je hier niets te zoeken. Ga de inleidende JavaScript-lessen nog maar eens een keer doen. 😊
- Een array is een al dan niet tastbaar ding dat is samengesteld uit één of meer andere dingen (de array-elementen). Als het om meer dingen gaat, dan hebben al die dingen gemeenschappelijke kenmerken.
- De inhoud van arrays (de array-elementen) kan variabel of constant zijn. Het gebruik bepaalt dat.
- Met concrete voorbeelden komen we hier later op terug.

WAT IS EEN ARRAY?

JS

DE EXPERT AAN HET WOORD...

- Wat zegt ChatGPT over arrays?
- De vraag: "Geef een beknopte definitie van een array in Jip-en-Janneketaal."
- Het antwoord: "Een array is een verzameling van dingen, zoals bijvoorbeeld een rij snoepjes waarbij je elk snoepje een volgnummer geeft, zodat je makkelijk het snoepje kunt vinden dat je wilt hebben."

WAT IS EEN ARRAY?

JS

DE EXPERT AAN HET WOORD...

- Wat zegt ChatGPT, iets professioneler, over arrays?
- De vraag: "Geef een beknopte definitie van een array".
- Het antwoord: "Een array is een gegevensstructuur die een geordende verzameling gegevens bevat, waarbij elk gegeven toegankelijk is via een *unieke* index of sleutel. Een array maakt het mogelijk om meerdere gegevens van hetzelfde type op te slaan en efficiënt te beheren."

WAT IS EEN ARRAY?

JS

INDEX? SLEUTEL?

- Als we het over een "index" hebben, dan bedoelen we een numerieke waarde.
- Een "sleutel" is doorgaans een tekstuele waarde.
- In de praktijk worden beide termen nogal eens verwisseld.

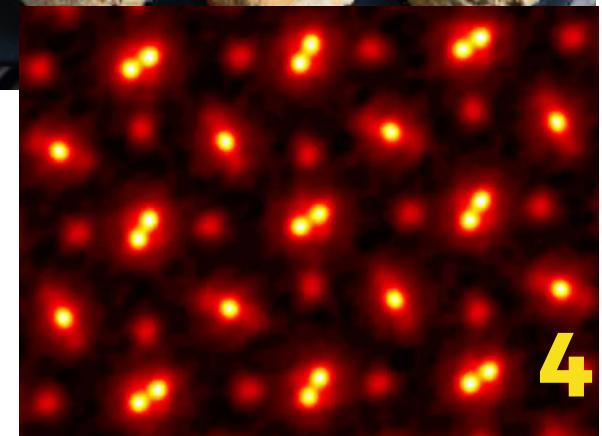
WAT IS EEN ARRAY?

JS

ARRAYS KOMEN OVERAL VOOR

- Een paar voorbeelden van tastbare arrays...

REACOLLEGE



1. Geparkeerde auto's, een 1-dimensionaal array.
2. Een rij bomen, nog een 1-dimensionaal array.
3. Muffins in een bakblik. Dit is een 2-dimensionaal array.
4. Een praseodymium-kristal. Dit lijkt een 2-dimensionaal array maar is in 't echt 3-dimensionaal. 't Is een kristalrooster.

WAT IS EEN ARRAY?

JS

EEN VERGELIJK

- Een array in JavaScript is niet tastbaar. Het is software.
- Van de tastbare voorbeelden komt een ladekast het best overeen met het JavaScript array.
- Wat er in de laden zit is nog even niet belangrijk.
- Bekijk de ladekast even goed.



WAT IS EEN ARRAY?

JS

NUMMERING

- Dat de laden genummerd zijn, is belangrijk.
- Met behulp van nummering kun je iets uit een bepaalde lade halen of er iets in stoppen.
- Dit is het indiceren, benaderen of adresseren.
- Wat er precies in de laden zit is ook nu nog even niet belangrijk.



WAT IS EEN ARRAY?

JS

NUMMERING

- De manier waarop laden genummerd worden, is belangrijk.
- Die begint in JavaScript altijd met **nul (0)**.
- En wat er in de laden zit is nog steeds niet interessant.



WAT IS EEN ARRAY?

JS

EEN BEGRIPPEN-UITSTAPJE - DECLARATIE

- Een variabele declareren (dus ook een array) is JavaScript vertellen dat je een variabele gaat gebruiken.
- Echter, je kent nog geen waarde en type toe aan de variabele.
- Door middel van een goed gekozen naam geef je aan waar de variabele voor gebruikt zal worden.
- De naam moet uniek zijn en is hoofd- en kleine lettergevoelig.

WAT IS EEN ARRAY?

JS

EEN BEGRIPPEN-UITSTAPJE - DEFINITIE

- Je definieert een variabele (dus ook een array) wanneer je JavaScript vertelt wat er in de variabele moet worden opgeslagen.
- Dan weet je ook meteen het type.
- Met andere woorden, je kent een waarde toe aan de variabele.
- Dit noemt men ook wel een “assignment”-statement of “toewijzings”-statement.
- Overigens, “statement” betekent “uitspraak”.

WAT IS EEN ARRAY?

JS

EEN BEGRIPPEN-UITSTAPJE - SAMENVATTING

- Declareren is het reserveren van werkgeheugen in je computer voor een variabele.
- Definiëren is het toekennen van een waarde en een type aan een variabele.

WAT IS EEN ARRAY?

JS

DECLARATIE EN DEFINITIE

- Een variabele met de naam "ladekast" declareren in JavaScript ziet er zo uit:

```
let ladekast;
```

Vb. 1

- De definitie van de variabele "ladekast" als een array met **3** "laden" gaat als volgt*:

```
ladekast = new Array(3);
```

Vb. 2

- "**3**" is de grootte van het array.



* Het kan ook anders, maar deze manier is nu even goed genoeg.

WAT IS EEN ARRAY?

JS

DECLARATIE EN DEFINITIE GECOMBINEERD

- Programmeurs zijn van nature lui. Programmeurs zouden geen programmeurs zijn als ze niet een manier bedachten om zo min mogelijk te moeten doen.
- Dus combineren ze de declaratie en de definitie:

```
let ladekast = new Array(3);
```

Vb. 3

- Maar wat gebeurt er hieronder?

```
let a = new Array(5, 6, 7, 8);
```

Vb. 4

- Niet wat je zou verwachten. Pas dus op bij je array-declaraties en -definities.



WAT IS EEN ARRAY?

JS

VOOR- EN NADELEN

- De voordelen:
 - a. Een array verenigt data van hetzelfde type onder één naam.
 - b. Een index vereenvoudigt het benaderen van of het zoeken naar data in een array.
 - c. Een array is eenvoudig te doorlopen door de index telkens met 1 te verhogen of te verlagen.
 - d. Snellere verwerking van data dan losse variabelen of andere datastructuren omdat een array bestaat uit aaneengesloten geheugenlocaties in het werkgeheugen.

WAT IS EEN ARRAY?

JS

VOOR- EN NADELEN

- De nadelen:
 - a. Arrays zijn niet erg flexibel. De grootte wijzigen is een inefficiënte bewerking.
 - b. Omdat array-elementen opeenvolgend worden geladen is het invoegen of verwijderen ook een inefficiënte bewerking.
 - c. Grote, gedeeltelijk gevulde arrays leiden tot verspilling van ongebruikt werkgeheugen.
 - d. De grootte van de array moet van tevoren bekend zijn.

WAT IS EEN ARRAY?

JS

WE HOUDEN HET SIMPEL

- Om de uitleg eenvoudig te houden beperken we ons tot arrays waarin numerieke gegevens worden opgeslagen.
- Echter, in de praktijk zul je zien dat er behalve getallen ook strings, logische variabelen, functies en objecten kunnen worden opgeslagen.

WAT IS EEN ARRAY?

JS

OPDRACHT 1

- Open de JavaScript Testomgeving in je browser.
- Maak een array genaamd "mijnInkomsten" van 12 (twaalf) elementen lang.
- In dit array sla je straks je maandelijkse inkomsten op.
- Voer je code uit om op fouten te controleren.
- Bewaar je code als "inkomsten-1".

WAT IS EEN ARRAY?

JS

OPLOSSING OPDRACHT 1

```
let mijnInkomsten = new Array(12);
```

Opdr. 1

WAT IS EEN ARRAY?

JS

LET, CONST OF VAR?

- Wat betekent “let” in JavaScript?
- Je kunt het Engelse JavaScript-sleutelwoord “let” vertalen in het Nederlands als “laat”.
- Dit assignment-statement kun je als volgt vertalen:

```
let mijnInkomsten = new Array(12);
```

Opdr. 1

- Maak de variabele “mijnInkomsten” aan **laat** deze dan de waarde krijgen van “new Array(12)”.
- “Let” beïnvloedt de manier waarop JavaScript een variabele behandelt. Je hoeft je daar tijdens deze presentatie geen zorgen over te maken.

WAT IS EEN ARRAY?

JS

LET, CONST OF VAR?

- Wat betekent “const” in JavaScript?
- Heel eenvoudig. “Const” is de afkorting van “constante”.
- Met andere woorden, de toegekende waarde is onveranderlijk geworden.

WAT IS EEN ARRAY?

JS

LET, CONST OF VAR?

- Wat betekent “var” in JavaScript?
- Ook heel eenvoudig. “Var” is de afkorting van “variabele”.
- “Var” is vervangen door “let” en “const” omdat je daarmee code betrouwbaarder maakt.
- Niet gebruiken dus!

WAT IS EEN ARRAY?

JS

LET, CONST OF VAR?

- Het komt er in het kort op neer dat je door “let” te gebruiken een variabele declareert waarvan je de inhoud mag wijzigen.
- Gebruik je daarentegen “const” om een variabele te declareren dan mag je de inhoud ná de definitie niet meer veranderen.
- Doe je dat toch, dan begint JavaScript te mopperen en stopt de uitvoer van je script.
- Gebruik bij twijfel voorlopig maar even “let”.
- Het juiste gebruik van “let” en “const” wordt in de reguliere JavaScript-lessen behandeld.

WAT IS EEN ARRAY?

JS

SAMENVATTING

- Het begrip "array" is verduidelijkt.
- Je weet nu wat "declaratie" en "definitie" betekenen.
- Je weet wat de voor- en nadelen van arrays zijn.
- Je hebt een array gemaakt.
- Je hoeft je niet al te druk te maken over het gebruik van "let" of "const".

WAT IS EEN ARRAY?

JS

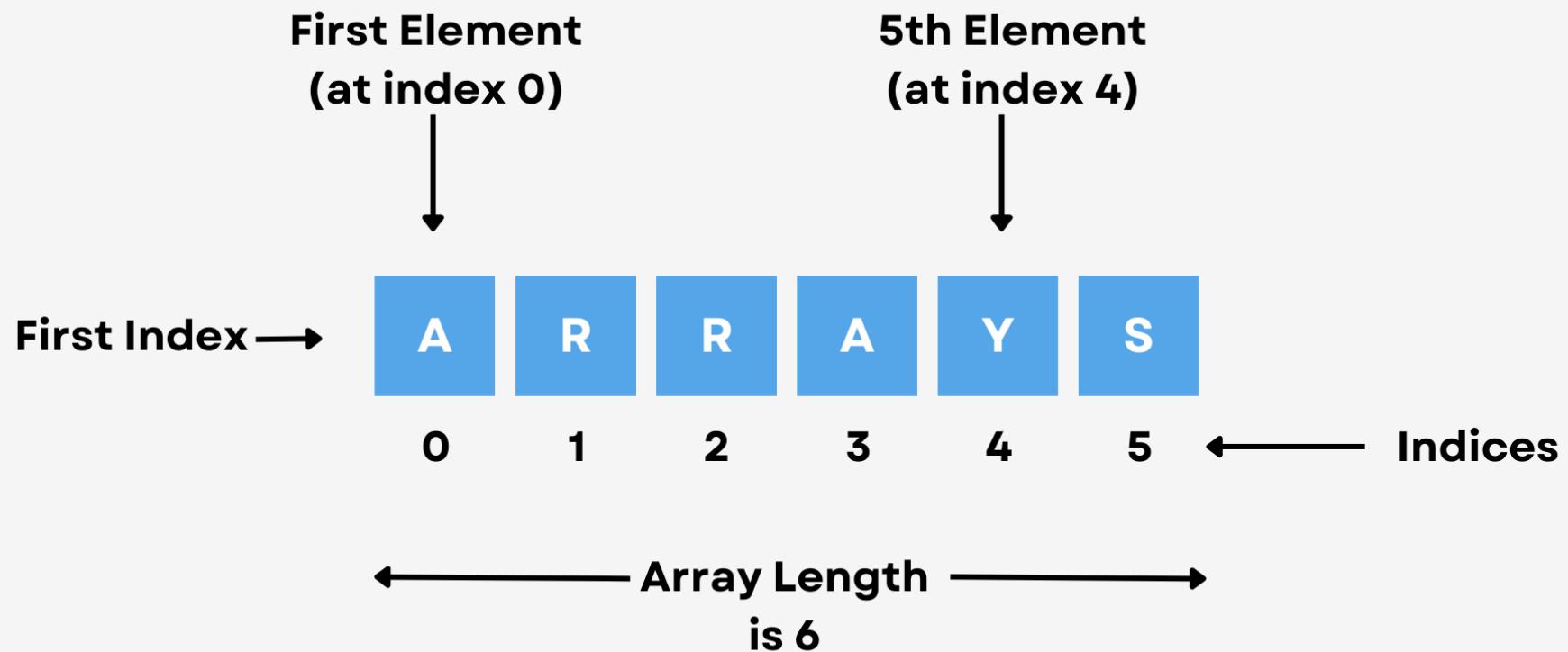
VRAGENUURTJE

- Zijn er onderwerpen aan bod gekomen die je niet snapt?
- Wat precies is je niet duidelijk?
- Hoe kunnen we je dat beter uitleggen?
- Heb je meer suggesties voor het verbeteren van deze onderwerpen?
- Heb je suggesties voor het verbeteren van de presentatie?

HOE GEBRUIKEN WE EEN ARRAY?

JS

R
E
A
C
O
L
L
E
G
E



HOE GEBRUIKEN WE EEN ARRAY?

JS

WAT KOMT ER AAN BOD?

- Het adresseren van array-elementen.
- Lets in een array-element opslaan.
- De inhoud van een array-element ophalen.
- Variabele indices.
- Ophalen, bewerken en opslaan herhalen.
- Samenvatting.
- Vragenuurtje.

HOE GEBRUIKEN WE EEN ARRAY?

JS

ARRAY-ELEMENTEN ADRESSEREN.

- Zoals eerder werd verteld, wordt de waarde waarmee je een array-element adresseert een index of een sleutel genoemd.
- We gebruiken de term index omdat we numerieke waarden gebruiken om te indiceren.
- Een index staat altijd tussen een teksthaak-openen (`[`) en een teksthaak-sluiten (`]`).
- De notatie is als volgt:

```
ladekast [ <<<HIER STAAT DE INDEX>>> ] ;
```

Vb. 5

HOE GEBRUIKEN WE EEN ARRAY?

JS

ARRAY-ELEMENTEN ADRESSEREN.

- Je zet 0 (nul) tussen de teksthaken om het eerste element van het ladekast-array te benaderen:

```
ladekast[ 0 ];
```

Vb. 6

- Je gebruikt een 1 als index (tussen de teksthaken) voor het tweede array-element en een 2 op dezelfde manier voor het derde element.

HOE GEBRUIKEN WE EEN ARRAY?

JS

ARRAY-ELEMENTEN ADRESSEREN.

- Verwarrend hè, die indices?
- Indices is het meervoud van index.
- Werken met indices is echter héél erg simpel.
- De waarde van een index is het volgnummer van het array-element vermindert met 1 (één).

```
ladekast[ 0 ]; // De eerste lade (element) .
```

Vb. 7

```
ladekast[ 1 ]; // De tweede lade (element) .
```

```
ladekast[ 2 ]; // De derde lade (element) .
```

HOE GEBRUIKEN WE EEN ARRAY?

JS

IETS OPSLAAN IN EEN ARRAY-ELEMENT

- Om iets op te halen uit array-elementen moeten we er eerst iets in stoppen.
- Array-elementen zijn variabelen!
- We houden het eenvoudig en stoppen getallen in de array-elementen.
- Dat gaat als volgt:

```
ladekast[ 0 ] = 2024;
```

Vb. 8

```
ladekast[ 1 ] = 11;
```

```
ladekast[ 2 ] = 6;
```

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 2

- Open de JavaScript Testomgeving in je browser.
- Laad "inkomsten-1".
- Ken (willekeurige) bedragen toe aan het derde, vierde, zevende en tiende array-element.
- Negatieve bedragen mogen ook.
- Bewaar je code als "inkomsten-2".

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPLOSSING OPDRACHT 2

```
let mijnInkomsten = new Array(12);
```

Opdr. 2

```
mijnInkomsten[2] = -64;
```

```
mijnInkomsten[3] = 128;
```

```
mijnInkomsten[6] = -1024;
```

```
mijnInkomsten[9] = 8192;
```

HOE GEBRUIKEN WE EEN ARRAY?

JS

DE INHOUD VAN EEN ARRAY-ELEMENT OPHALEN

- Bij het opslaan van een waarde in een variabele staat een array-element aan de linkerkant van het =-teken.
- Bij het ophalen verhuist het array-element naar de rechterkant van het =-teken.
- Dat ziet er dan als volgt uit:

R
E
A
C
O
L
L
E
G
E

```
let ladeEen = ladekast[ 0 ];
```

Vb. 9

```
let ladeTwee = ladekast[ 1 ];
```

```
let ladeDrie = ladekast[ 2 ];
```

HOE GEBRUIKEN WE EEN ARRAY?

JS

VARIABELE INDICES

- Tot nu toe hebben we constante indices (0, 1 en 2) gebruikt om array-elementen te benaderen.
- Je maakt de toegang tot array-elementen een stuk flexibeler door gebruik te maken van variabele indices.
- Discussie: Waarom zou je dat willen? Geef drie redenen.

HOE GEBRUIKEN WE EEN ARRAY?

JS

VARIABELE INDICES

- Drie ChatGPT-redenen waarom je variabele indices wilt gebruiken:
 1. Dynamische toegang tot array-elementen op basis van de huidige staat van je programma is handig als je:
 - a. Individuele array-elementen opeenvolgend wilt benaderen.
 - b. Specifieke array-elementen wilt selecteren op basis van gebruikersinvoer.

HOE GEBRUIKEN WE EEN ARRAY?

JS

VARIABELE INDICES

- Drie ChatGPT-argumenten waarom je variabele indices zou willen gebruiken:
 2. Gebruik dezelfde code om verschillende array-elementen te benaderen:
 - a. Je vermindert codeduplicatie en daardoor is je code eenvoudiger te onderhouden.
 - b. Je hoeft je code niet aan te passen als de grootte van je array verandert.

HOE GEBRUIKEN WE EEN ARRAY?

JS

VARIABELE INDICES

- Drie ChatGPT-argumenten waarom je variabele indices zou willen gebruiken:
 3. Door variabele indices in plaats van constante indices te gebruiken blijft de toegang tot de gewenste array-elementen bestaan als de grootte van je array verandert tijdens de uitvoer van je programma.
- Deze drie zaken komen in het verloop van de presentatie allemaal aan bod.

HOE GEBRUIKEN WE EEN ARRAY?

JS

VARIABELE INDICES

- Hoe zien variabele indices er uit? Zo:

```
let index = 0;  
let ladeEen = ladekast[ index ];
```

Vb.10

```
index = 1;  
let ladeTwee = ladekast[ index ];
```

```
index = 2;  
let ladeDrie = ladekast[ index ];
```

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 3

- Open de JavaScript Testomgeving in je browser.
- Laad "inkomsten-2".
- Ken willekeurige (negatieve) bedragen toe aan alle array-elementen die nog niet gedefinieerd zijn.
- Maak gebruik van variabele indices.
- Voer je code uit om op fouten te controleren.

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 3

- Voeg onderstaande regel toe aan het einde van je code om de inhoud van het mijnInkomsten-array te tonen.

```
printnl(mijnInkomsten);
```

Vb. 11

- Bewaar je code als "inkomsten-3".

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPLOSSING OPDRACHT 3

```
let mijnInkomsten = new Array(12);  
let index;
```

Opdr. 3

```
index = 0; mijnInkomsten[index] = 16;  
index = 1; mijnInkomsten[index] = 32;  
        mijnInkomsten[2] = -64;  
        mijnInkomsten[3] = 128;  
index = 4; mijnInkomsten[index] = 256;  
index = 5; mijnInkomsten[index] = 512;  
        mijnInkomsten[6] = -1024;  
index = 7; mijnInkomsten[index] = 2048;  
index = 8; mijnInkomsten[index] = 4096;  
        mijnInkomsten[9] = 8192;  
index = 10; mijnInkomsten[index] = 16384;  
index = 11; mijnInkomsten[index] = 32768;  
  
println(mijnInkomsten);
```

Omdat de dia niet genoeg ruimte heeft,
staan er op één regel twee statements.

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 4

- Open de JavaScript Testomgeving in je browser.
- Laad "inkomsten-3".
- Vervang alle constante indices door de variabele indices.
- Voer je code uit om op fouten te controleren.
- Bewaar je code als "inkomsten-4".

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPLOSSING OPDRACHT 4

```
let mijnInkomsten = new Array(12);  
let index;
```

Opdr. 4

```
index = 0; mijnInkomsten[index] = 16;  
index = 1; mijnInkomsten[index] = 32;  
index = 2; mijnInkomsten[index] = -64;  
index = 3; mijnInkomsten[index] = 128;  
index = 4; mijnInkomsten[index] = 256;  
index = 5; mijnInkomsten[index] = 512;  
index = 6; mijnInkomsten[index] = -1024;  
index = 7; mijnInkomsten[index] = 2048;  
index = 8; mijnInkomsten[index] = 4096;  
index = 9; mijnInkomsten[index] = 8192;  
index = 10; mijnInkomsten[index] = 16384;  
index = 11; mijnInkomsten[index] = 32768;
```

```
println(mijnInkomsten);
```

Omdat de dia niet genoeg ruimte heeft,
staan er op één regel twee statements.

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPHALEN, BEWERKEN EN OPSLAAN HERHALEN

- Je moet de getallen in de array-elementen van het ladekast-array door twee delen en het resultaat van de deling in het betreffende array-element opslaan.
- Het ladekast-array ziet er nog steeds hetzelfde uit en de inhoud van de array-elementen is ook ongewijzigd:

R
E
A
C
O
L
L
E
G
E

```
let ladekast = new Array(3);  
  
ladekast[ 0 ] = 2024;  
  
ladekast[ 1 ] = 11;  
  
ladekast[ 2 ] = 6;
```

Vb. 12

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPHALEN, BEWERKEN EN OPSLAAN HERHALEN

- Dat kan als volgt, waarbij "index" is afgekort tot "i":

```
let i = 0;  
ladekast[ i ] = ladekast[ i ] / 2;  
  
i = 1;  
ladekast[ i ] = ladekast[ i ] / 2;  
  
i = 2;  
ladekast[ i ] = ladekast[ i ] / 2;
```

Vb. 13

HOE GEBRUIKEN WE EEN ARRAY?

JS

IETS OVER DE FOR-LOOP

- Met behulp van de array-property "length" en de JavaScript "for-loop" los je de problemen als gevolg van veel array-elementen en wisselende array-grootten op. We gaan herhalen!
- Tussen twee haakjes... "loop" is het Engelse woord voor "lus". Het zal wel duidelijk zijn wat "length" is.
- De code ziet er als volgt uit:

```
for ( let i = 0; i < ladekast.length; i++ ) {  
    ladekast[ i ] = ladekast[ i ] / 2;  
}
```

Vb. 14

- Meer over de "for-loop" op de volgende dia's.

HOE GEBRUIKEN WE EEN ARRAY?

JS

IETS OVER DE FOR-LOOP

- Anatomie van een for-loop...

```
// De loop begint hier.  
for ( begin ; controle ; update ) {  
  
    // Statements...  
  
}  
// En eindigt hier.
```

R
E
A
C
O
L
L
E
G
E

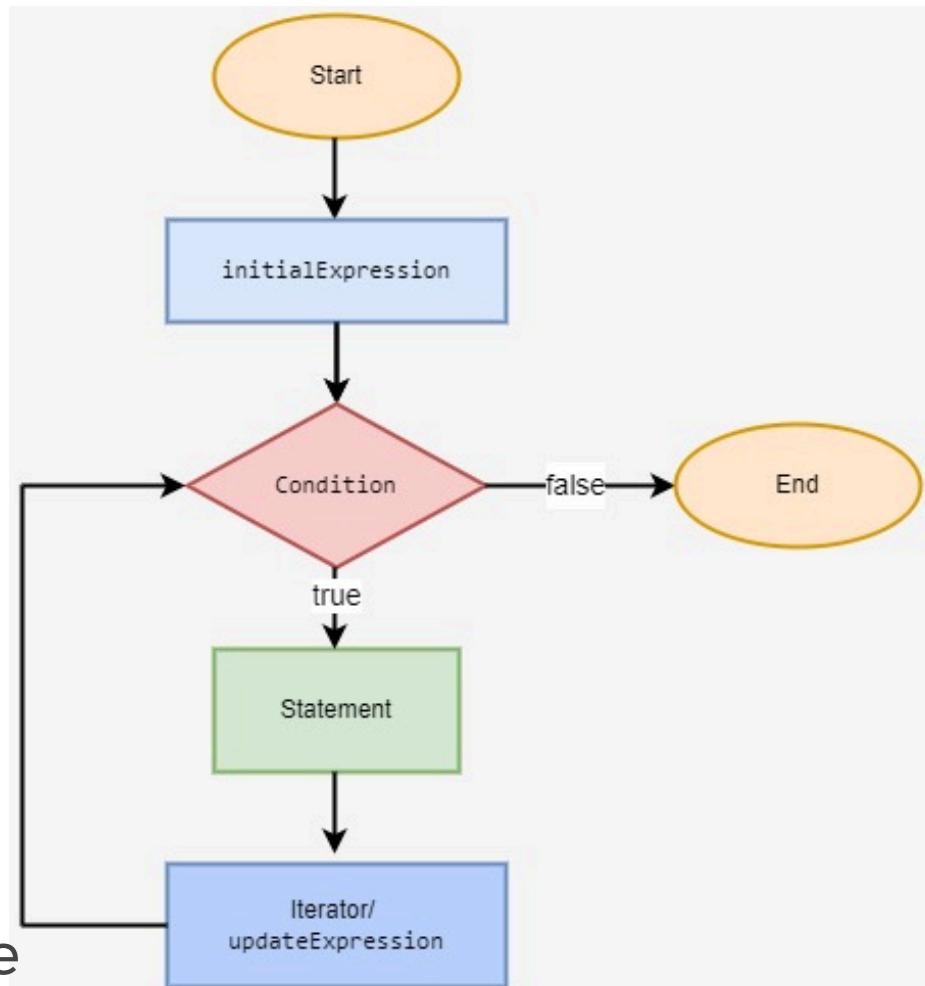
for	- Het Javascript taalelement.
(en)	- Bevatten altijd begin , controle en update statements, gescheiden door puntkomma's (;).
begin	- Initialisatie van de loop-variabele(n).
controle	- Moet de loop beëindigd worden?
update	- Wijzig de loop-variable(n) zodat de loop kan eindigen.
{ en }	- Bevatten het code-blok (bestaande uit één of meerdere statements) dat herhaald uitgevoerd moet worden.

HOE GEBRUIKEN WE EEN ARRAY?

JS

IETS OVER DE FOR-LOOP

- Deze flow-chart toont de werking van de for-loop.
- De animatie op de volgende dia laat een for-loop in actie zien.
- Merk op dat de het hier om Java-code gaat.
- JavaScript is **géén** Java!
- De taal is anders, maar de principes blijven hetzelfde.



HOE GEBRUIKEN WE EEN ARRAY?

JS

IETS OVER DE FOR-LOOP

```
for (int i = 0; i < 3; i++) {  
    System.out.println("i is now " + i);  
}
```

HOE GEBRUIKEN WE EEN ARRAY?

JS

DE GESCHIEDENIS VAN DE LOOP-VARIABELE

- Waar komt het gebruik van "i" t/m "n" als loop-variabelen vandaan?
- In een héél, héél ver verleden (de 60-er jaren van de vorige eeuw), toen computers van hout en steen 😊 werden gemaakt, was er een programmeertaal genaamd FORTRAN. (Wordt nog altijd veel gebruikt.)
- De eerste versies van FORTRAN kenden alleen maar hoofdletters. Variabelen waarvan de naam begint met een "I", "J", "K", "L", "M" of "N", zijn bij default van het type "INTEGER".
- Loops maken in veel gevallen gebruik van een loop-variabele van het type integer.

HOE GEBRUIKEN WE EEN ARRAY?

JS

DE GESCHIEDENIS VAN DE LOOP-VARIABELE

- Om je een idee te geven... Zo ziet een "for-loop" er in FORTRAN-IV uit om 5 keer "HELLO, WORLD!" te tonen:

```
C      VOORBEELD VAN EEN DO-LOOP IN FORTRAN-IV
      DO 100 I = 1, 5
      WRITE(6, 99) I
99      FORMAT(13HHELLO, WORLD!, I3)
100     CONTINUE
      STOP
      END
```

- Als je dat zo ziet dan ben je toch blij dat je in 2024 leeft en niet in 1961.
- Heb je vragen over deze code?
- ChatGPT.

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 5

- Open de JavaScript Testomgeving in je browser.
- Laad "inkomsten-4".
- Gebruik de for-loop om aan alle array-elementen van het "mijnInkomsten"-array een willekeurig bedrag tussen -2500 en 2500 toe te kennen met behulp van onderstaande expressie.

```
Math.floor(Math.random() * 5001) - 2500;
```

Vb. 15

- Voer je code uit om op fouten te controleren.
- Bewaar je code als "inkomsten-5".

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPLOSSING OPDRACHT 5

```
let mijnInkomsten = new Array(12);  
  
for(let i = 0; i < mijnInkomsten.length; i++) {  
    mijnInkomsten[i] =  
        Math.floor(Math.random() * 5001) - 2500;  
}  
  
printnl(mijnInkomsten);
```

Opdr. 5

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 6

- Open de JavaScript Testomgeving in je browser.
- Laad "inkomsten-5".
- Voeg code toe om de som van alle bedragen in het mijnInkomsten-array te berekenen.
- Voer je code uit om op fouten te controleren.
- Bewaar je code als "inkomsten-6".

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPLOSSING OPDRACHT 6

```
let mijnInkomsten = new Array(12);  
let totaalInkomsten = 0;  
  
for(let i = 0; i < mijnInkomsten.length; i++) {  
    let n = Math.floor(Math.random() * 5001) - 2500;  
  
    mijnInkomsten[i] = n;  
  
    totaalInkomsten = totaalInkomsten + n;  
}  
  
println(mijnInkomsten);  
println(totaalInkomsten);
```

Opdr. 6

R
E
A
C
O
L
L
E
G
E

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPDRACHT 7

- Open de JavaScript Testomgeving in je browser.
- Laad "inkomsten-6".
- Voeg code toe om de BTW (21%) van alle bedragen in het mijnInkomsten-array te berekenen en in een nieuw array op te slaan.
- Bereken ook de totale BTW-afdracht.
- Voer je code uit om op fouten te controleren.
- Bewaar je code als "inkomsten-7".

HOE GEBRUIKEN WE EEN ARRAY?

JS

OPLOSSING OPDRACHT 7

```
let mijnInkomsten = new Array(12);
let totaalInkomsten = 0;
for(let i = 0; i < mijnInkomsten.length; i++) {
    let n = Math.floor(Math.random() * 5001) - 2500;
    mijnInkomsten[i] = n;
    totaalInkomsten = totaalInkomsten + n;
}
print('Mijn inkomsten: '); println(mijnInkomsten);
print('Som inkomsten: '); println(totaalInkomsten);

let mijnBTW = new Array(12);
let totaalBTW = 0;
for(let i = 0; i < mijnBTW.length; i++) {
    let b = mijnInkomsten[i] * 0.21;
    mijnBTW[i] = b;
    totaalBTW = totaalBTW + b;
}
print('Mijn BTW: '); println(mijnBTW);
print('Som BTW: '); println(totaalBTW);
```

Opdr. 7

Omdat de dia niet genoeg ruimte heeft,
staan er op één regel twee statements.

HOE GEBRUIKEN WE EEN ARRAY?

JS

SAMENVATTING

- Je hebt array-elementen geadresseerd.
- Je hebt iets opgeslagen in array-elementen en de inhoud ook weer opgehaald en verwerkt.
- Dat heb je gedaan met constante indices en variabele indices.
- Je hebt met behulp van de "for-loop" gegevens bewerkt.

HOE GEBRUIKEN WE EEN ARRAY?

JS

VRAGENUURTJE

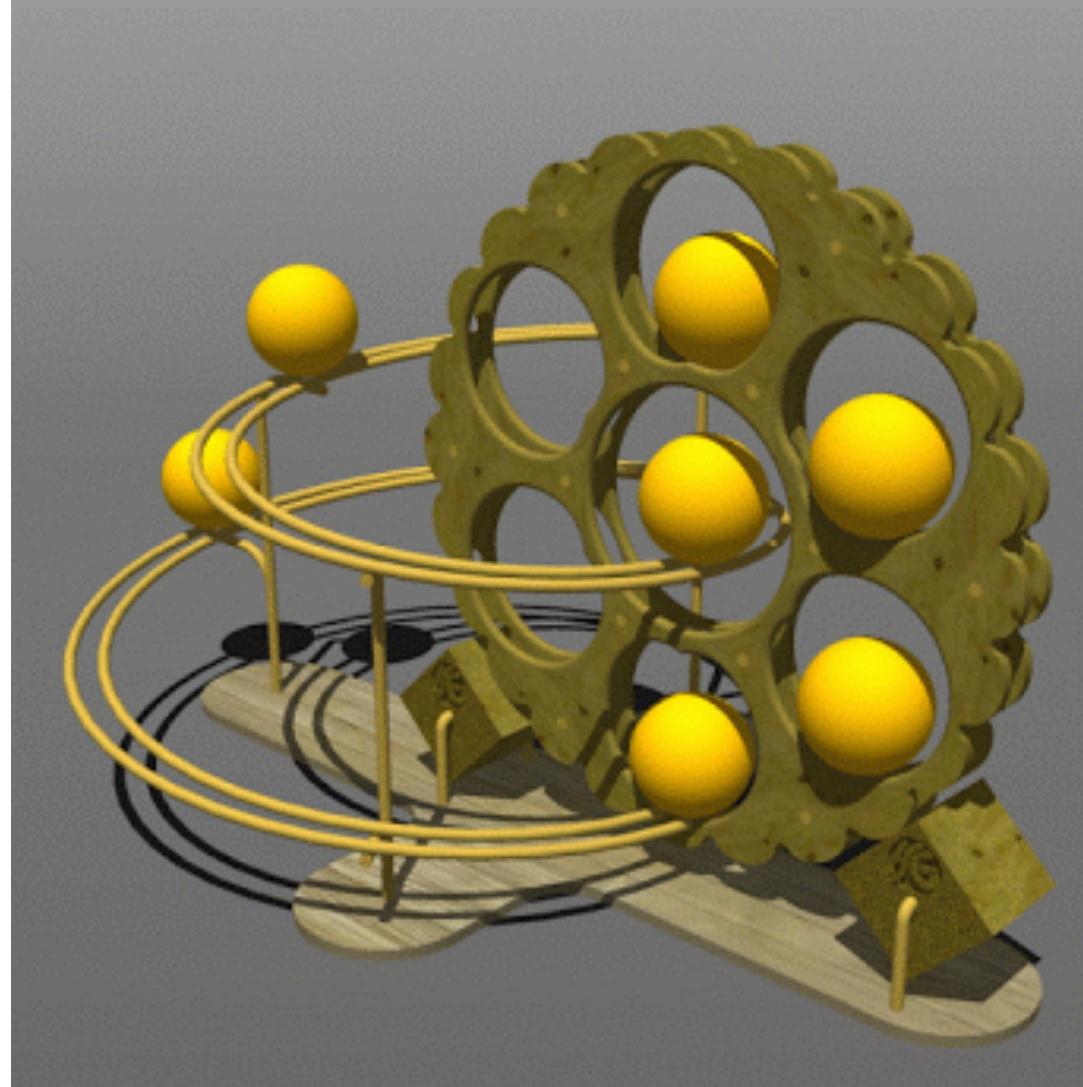
REACOLLEGE

- Zijn er onderwerpen aan bod gekomen die je niet snapt?
- Wat precies is je niet duidelijk?
- Hoe kunnen we je dat beter uitleggen?
- Heb je meer suggesties voor het verbeteren van deze onderwerpen?
- Heb je suggesties voor het verbeteren van de presentatie?

RAADSELS IN MEER DIMENSIES

JS

R
E
A
C
O
L
L
E
G
E



RAADSELS IN MEER DIMENSIES

JS

WAT KOMT ER AAN BOD?

- Over dimensies en afmetingen.
- Waarom meer-dimensionale arrays?
- Nieuwe ladekasten.
- Samenvatting.
- Vragenuurtje.

RAADSELS IN MEER DIMENSIES

JS

OVER DIMENSIES EN AFMETINGEN

- Tot nu toe heb je alleen nog maar 1-dimensionale arrays gezien.
- Dat was al tamelijk spannend. Toch?
- We maken het nog wat spannender met 2-, 3- en 3+-dimensionale arrays.
- Voor het goede begrip:
 - a. Dimensie bepaalt de vorm van een array.
 - b. Afmeting bepaalt de grootte van een array.
- Bijv.: Een 3-dimensionaal array heeft voor elke dimensie een afmeting (lengte x breedte x hoogte).

RAADSELS IN MEER DIMENSIES

JS

ALTIJD MAAR WEER DIE WISKUNDE!

- We verwaarlozen de link naar de wetenschap niet en daarom - er valt helaas niet aan te ontkomen - een heel klein beetje wiskunde (meetkunde om precies te zijn):
 1. Een punt heeft geen dimensies.
 2. Een lijn heeft één dimensie en bestaat uit oneindig veel punten naast elkaar.
 3. Een vlak heeft twee dimensies en bestaat uit oneindig veel lijnen naast elkaar.
 4. Een volume heeft drie dimensies en bestaat uit oneindig veel gestapelde vlakken.
 5. De tijd...?

EEN INNIGE RELATIE

- Hoe verhoudt zich de meetkunde tot arrays?
 1. Een eenvoudige variabele (vergelijkbaar met een punt) heeft geen index.
 2. Een één-dimensionaal array (vergelijkbaar met een lijn) heeft één index voor alle array-elementen.
 3. Een twee-dimensionaal array (vergelijkbaar met een vlak) heeft een index voor alle lijnen én een index voor alle array-elementen.
 4. Een drie-dimensionaal array (vergelijkbaar met een volume) heeft een index voor alle vlakken én een index voor alle lijnen én een index voor alle array-elementen.
 5. Een vier-dimensionaal array...?

RAADSELS IN MEER DIMENSIES

JS

HOE STEL IK ME DIE DIMENSIES VOOR?

RECOMMEND

1D array

7	2	9	10
---	---	---	----

axis 0 →

2D array

5.2	3.0	4.5
9.1	0.1	0.3

axis 0 →

axis 1 →

3D array

2	3	4	4
1	4	1	4
1	4	7	7
2	9	7	5
1	3	7	2
9	6	0	8
9	9	9	9

axis 0 ↓

axis 1 ↓

axis 2 →

RAADSELS IN MEER DIMENSIES

JS

WAAROM MEER-DIMENSIONALE ARRAYS?

- 1-dimensionale arrays voor de opslag van o.a.:
 1. Strings, een speciale ("read-only") array-vorm.

```
let text = 'That array ate all data!';
for(let i = 0; i < text.length; i++) {
  print(text[i] === 'a' ? 'o' : text[i]);
}
```

Vb. 16

2. Tijd-gerelateerde (gem. temperatuur per maand).

```
let gemTemp = [
  -4, -2, 4, 8, 13, 22, 25, 21, 19, 11, 7, 0
];
```

Vb. 17

3. Inventarissen (arrays van strings).

```
let uitgeleend = [
  'Laptop', 'Telefoon', 'Tablet', 'Stylus',
  'Tas', 'USB-stick', 'Muis', 'Headset'
];
```

Vb. 18

RAADSELS IN MEER DIMENSIES

JS

WAAROM MEER-DIMENSIONALE ARRAYS?

- 2-dimensionale arrays voor de opslag van o.a.:
 1. Spelborden (schaken, dammen, etc.) en puzzles.
 2. Pixels (beeldpunten).
 3. Spreadsheets.
 4. Matrices (worden gebruikt in hogere wiskunde, voor computer graphics en beeldverwerking, machine-learning, data-analyse, etc.).
 5. Tabellen.
- Bedenk voorbeelden van arrays voor deze vijf toepassingsgebieden.

RAADSELS IN MEER DIMENSIES

JS

WAAROM MEER-DIMENSIONALE ARRAYS?

1.

```
let schaakbord = [
    ['T', 'R', 'L', 'D', 'K', 'L', 'R', 'T'],
    ['P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'],
    // ... (De rest van het schaakbord.)
];
```

Vb. 19

2.

```
let afbeelding = [
    [127, 0, 0], // Een donkerrode pixel.
    [0, 127, 0], // Een donkergroene pixel.
    // ... (De rest van de afbeelding.)
];
```

Vb. 20

3.

```
let spreadsheet = [
    ['Naam', 'Leeftijd', 'Woonplaats', 'Inkomen'],
    ['Dirksie', 25, 'Potsverjansdorp', 1250],
    ['Zappolientje', 30, 'Goocheloord', 3100],
    // ... (De rest van de spreadsheet.)
];
```

Vb. 21

RAADSELS IN MEER DIMENSIES

JS

WAAROM MEER-DIMENSIONALE ARRAYS?

4. `let matrixA = [
 [2, 4],
 [1, 3]
];`

Vb. 22

`let matrixB = [
 [5, 2],
 [6, 1]
];`

R
E
A
C
O
L
L
E
G
E

5. `let tabel = [
 [1, 25, 150, 'Azijn'],
 [2, 30, 200, 'Olijfolie'],
 // ... (De rest van de tabel.)
];`

Vb. 23

RAADSELS IN MEER DIMENSIES

JS

WAAROM MEER-DIMENSIONALE ARRAYS?

- 3-Dimensionale arrays worden gebruikt voor de opslag van ruimtelijke modellen.
- De tijd wordt vaak als de vierde dimensie gezien.
- Denk bij 3- en 4-dimensionale arrays aan:
 1. Simulaties.
 2. Klimaatmodellen.
 3. Video.
 4. Medische toepassingen.
- Ingewikkelde berekeningen maken vaak gebruik van arrays met hogere dimensies dan 3.

RAADSELS IN MEER DIMENSIES

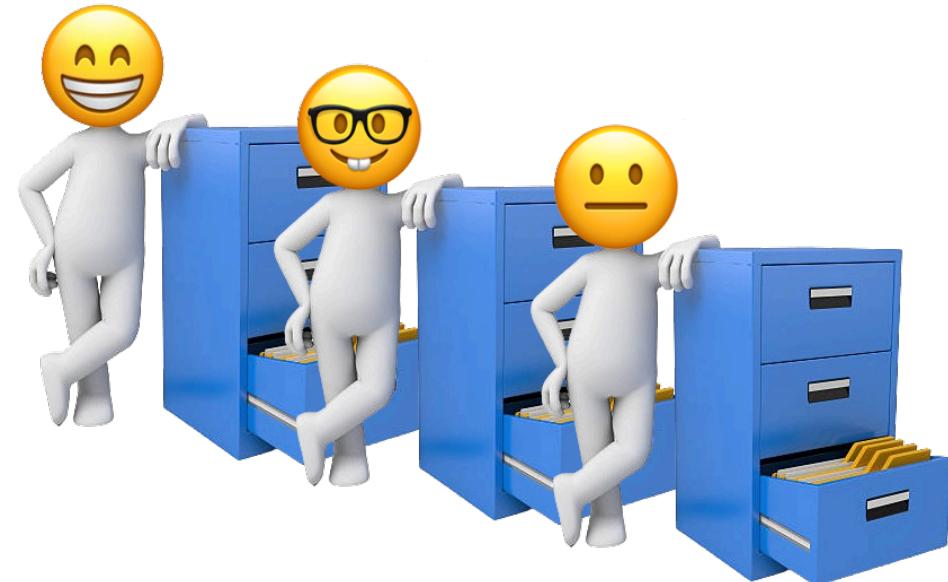
JS

VAN 1 NAAR 4 DIMENSIES IN SLOW-MOTION

Point (0)

NIEUWE LADEKASTEN

- Je hebt een tweede ladekast nodig. De eerste zit propvol. Daarom maar even naar de woonboulevard.
- Bij de Zweedse meubelgigant Ykkéjà koop je meteen maar twee Arkivlådor ladekasten van de populaire Værdeløs-serie. Ze zijn flink afgeprijsd en je denkt aan de toekomst.
- Je broers helpen je bij het in elkaar zetten.



NIEUWE LADEKASTEN IN JAVASCRIPT

- Je hebt nu dus drie ladekasten met elk drie laden.
- Hoe ziet dat er uit in JavaScript?
- Zo dus, op de dynamische manier:

```
let ladekasten = new Array(3); // Alle ladekasten.  
  
let ladekast = new Array(3); // De eerste kast  
ladekasten[0] = ladekast; // met laden.  
  
ladekast = new Array(3); // De tweede kast  
ladekasten[1] = ladekast; // met laden.  
  
ladekast = new Array(3); // De derde kast  
ladekasten[2] = ladekast; // met laden.
```

Vb. 24

- De waarden in de laden zijn hier niet gedefinieerd.

RAADSELS IN MEER DIMENSIES

JS

NIEUWE LADEKASTEN IN JAVASCRIPT

- Maar het kan ook anders. Dit is de statische manier:

```
let ladekasten = [ // Alle ladekasten.  
  [ 0, 0, 0 ], // Kast 1 met 3 laden.  
  [ 0, 0, 0 ], // Kast 2 met 3 laden.  
  [ 0, 0, 0 ] // Kast 3 met 3 laden.  
];
```

Vb. 25

- Waarbij alle laden hier de waarde 0 krijgen.

- En dit is weer dynamisch:

```
let ladekasten = new Array(3);  
  
for(let i = 0; i < ladekasten.length; i++) {  
  ladekasten[i] = new Array(3);  
}
```

Vb. 26

- Waarbij de waarden in de laden niet zijn gedefinieerd.

RAADSELS IN MEER DIMENSIES

JS

NIEUWE LADEKASTEN IN JAVASCRIPT NUMMEREN EN INDICEREN

- Je ziet dat de nummering van de ladekasten op dezelfde manier gaat als de nummering van de laden.
- Hoe benader je nu een array-element? Bijvoorbeeld het 3^e array-element in de 2^e ladekast. Wel, als volgt:

```
let ladeInhoud = ladekasten[ 1 ][ 2 ];
```

Vb. 27

NOG MEER NIEUWE LADEKASTEN

- Om aan alle chaos definitief een eind te maken gebruik je dossiermappen in elke lade van elke kast.
- Elke dossiermap van de maximaal 26 krijgt als waarde het volgnummer. Dat gaat dus van 0 t/m 25.
- Tevens zet je op vier andere locaties ook drie kasten neer. (Ykkéjà is er maar goed mee! 😊)
- Hoe zien declaraties/definities en gebruik van de locaties, kasten, laden en dossiermappen er uit?
- Zoals eerder gezegd gaat de volgorde van indices van "groot" naar "klein", m.a.w. van locatie naar dossiermap.
- Je wilt geen internationale activiteiten. Gelukkig! 😊

RAADSELS IN MEER DIMENSIES

JS

NOG MEER NIEUWE LADEKASTEN - VOORBEELDEN

```
// Declaraties en definities.

let locaties = new Array(5);
for(let loc = 0; loc < locaties.length; loc++) {
    let kasten = new Array(3);
    locaties[loc] = kasten;
    for(let kast = 0; kast < kasten.length; kast++) {
        let laden = new Array(3);
        kasten[kast] = laden;
        for(let lade = 0; lade < laden.length; lade++) {
            let mappen = new Array(26);
            laden[lade] = mappen;
            for(let map = 0; map < mappen.length; map++) {
                mappen[map] = map;
            }
        }
    }
}
```

Vb. 28

RAADSELS IN MEER DIMENSIES

JS

NOG MEER NIEUWE LADEKASTEN - VOORBEELDEN

```
// De inhoud van alle dossiermappen wijzigen.
```

Vb. 30

```
for(let loc = 0; loc < locaties.length; loc++) {  
    for(let kast = 0; kast < locaties[loc].length; kast++) {  
        for(let lade = 0; lade < locaties[loc][kast].length; lade++) {  
            for(let map = 0; map < locaties[loc][kast][lade].length; map++) {  
                locaties[loc][kast][lade][map] = map * 2 + 1;  
            }  
        }  
    }  
}
```

RAADSELS IN MEER DIMENSIES

JS

NOG MEER NIEUWE LADEKASTEN - VOORBEELDEN

// Alles laten zien.

Vb. 29

```
for(let loc = 0; loc < locaties.length; loc++) {
    for(let kast = 0; kast < locaties[loc].length; kast++) {
        for(let lade = 0; lade < locaties[loc][kast].length; lade++) {
            println('Locatie: ' + (loc + 1));
            println('Kast: ' + (kast + 1));
            println('Lade: ' + (lade + 1));
            print('Mappen:');
            for(let map=0; map<locaties[loc][kast][lade].length; map++) {
                let mapInhoud = locaties[loc][kast][lade][map];
                print(' ' + mapInhoud);
            }
            println('');
            println('');
        }
    }
}
```

RAADSELS IN MEER DIMENSIES

JS

VRAGEN OVER NOG MEER NIEUWE LADEKASTEN

1. Hoeveel dimensies heeft het "locaties"-array?
2. Wat zijn de afmetingen van het "locaties"-array?
3. Hoeveel array-elementen heeft het "locaties"-array?
4. Wat gebeurt er met de afmetingen als je er op de oneven locaties nog eens vier kasten bij plaatst?
5. Veranderen de dimensies? En waarom niet?
6. Hoeveel array-elementen zijn er nu?
7. Wat wordt de totale grootte in bytes van het gewijzigde array als de maximale inhoud van een map 1 MiB (= 1 048 576 bytes) is?

RAADSELS IN MEER DIMENSIES

JS

ANTWOORDEN OP VRAGEN OVER NOG MEER NIEUWE LADEKASTEN

1. Dat zijn er 4 (locaties x kasten x laden x dossiermappen).
2. Dat is $5 \times 3 \times 3 \times 26$.
3. Dat zijn 1170 array-elementen.
4. Nieuwe afmetingen: $2 \times 3 \times 3 \times 26 + 3 \times 7 \times 3 \times 26$.
5. Nee, je verandert alleen de afmeting.
6. Er zijn 2106 array-elementen.
7. Dat zijn maximaal 2 208 301 056 bytes. Dat is meer dan 2 GiB!

RAADSELS IN MEER DIMENSIES

JS

NOG MEER NIEUWE LADEKASTEN - NOG MEER VOORBEELDEN

- Haal de inhoud van de 15^e dossiermap op uit de 3^e lade van de 2^e ladekast van de 4^e locatie.

```
let dossiermap = locaties[3][1][2][14];
```

Vb. 31

- Sla die waarde op in de laatste dossiermap van de 1^e lade van de 1^e ladekast van de 2^e locatie.

```
locaties[1][0][0][25] = dossiermap;
```

Vb. 32

- Vul de **even** dossiermappen van de 2^e locatie met -1.

```
// Alle dossiermappen met een even volgnummer worden -1.
```

Vb. 33

```
for ( let kast = 0; kast < kasten.length; kast++ ) {  
    for ( let lade = 0; laden.length < 3; lade++ ) {  
        for ( let map = 1; map < mappen.length; map += 2 ) {  
            locaties[1][kast][lade][map] = -1;  
        }  
    }  
}
```

RAADSELS IN MEER DIMENSIES

JS

NOG MEER NIEUWE LADEKASTEN - NOG MEER VOORBEELDEN

- . Tel alle dossiermappen die 13 bevatten.

```
// Waar zijn de dossiermappen met volgnummer 13?
```

Vb. 34

```
const volgnr = 13;
let teller = 0;
for ( let loc = 0; loc < locaties.length; loc++ ) {
    for ( let kast = 0; kast < kasten.length; kast++ ) {
        for ( let lade = 0; lade < laden.length; lade++ ) {
            for ( let map = 0; map < mappen.length; map++ ) {
                if ( locaties[loc][kast][lade][map] === volgnr ) {
                    teller++;
                }
            }
        }
    }
}

print('Dossiermappen met het volgnummer 13 komen ');
print(teller);
println(' keer voor.');
```

RAADSELS IN MEER DIMENSIES

JS

NOG MEER NIEUWE LADEKASTEN - NOG MEER VOORBEELDEN

- Kopieer alle dossiermappen van de 3^e locatie naar de 2^e locatie en tel telkens 100 op bij de inhoud van elke dossiermap in de 2^e locatie.

```
// Kopieer alles van locatie 3 naar locatie 2 en tel 100      Vb. 35
// op bij de inhoud van elke dossiermap van locatie 2.
```

```
const bron = 2;
const doel = 1;
for ( let kast = 0; kast < kasten.length; kast++ ) {
    for ( let lade = 0; lade < laden.length; lade++ ) {
        for ( let map = 0; map < mappen.length; map++ ) {
            let m = locaties[bron][kast][lade][map] + 100;
            locaties[doel][kast][lade][map] = m;
        }
    }
}
```

- En met dit laatste voorbeeld (be)sluiten we definitief de ladekasten én de JavaScript testomgeving.

RAADSELS IN MEER DIMENSIES

JS

SAMENVATTING

- Je weet nu wat het verschil tussen dimensies en afmetingen is.
- Het nut van meer-dimensionale arrays is verduidelijkt.
- Je hebt array-elementen uit een multi-dimensionaal array herhaald opgehaald, bewerkt en weer opgeslagen.

VRAGENUURTJE

- Zijn er onderwerpen aan bod gekomen die je niet snapt?
- Wat precies is je niet duidelijk?
- Hoe kunnen we je dat beter uitleggen?
- Heb je meer suggesties voor het verbeteren van deze onderwerpen?
- Heb je suggesties voor het verbeteren van de presentatie?

TIPS & TRICKS

JS

TRIPS &
TICKS

- Maak het jezelf makkelijk* bij het programmeren in HTML, CSS en JavaScript. Maak er een gewoonte van om:
 1. Met **F12** de "Developer Tools" van je browser **altijd** aan te zetten op de "Console"-tab. Dan zie meteen of je code fouten heeft als je die uitvoert.
 2. Je JavaScript code **altijd** met `'use strict'` te beginnen om er voor te zorgen dat je variabelen eerst moet declareren voordat je ze kunt gebruiken.
 3. Extra typwerk te voorkomen door veelgebruikte code/statements af te korten. Bijv.: `console.log` kort je af met `const cl = console.log;`. Typ van nu af aan `cl(variabele);` om de inhoud van "variabele" te tonen.

* Doe er je voordeel mee. Of niet. Jouw feestje, jouw keuze.

4. Zijn wijzigingen in je code niet goed?

a. **Undo:** Gebruik Ctrl-Z (Windows en Linux) of Cmd-Z (macOS) om wijzigingen (herhaald) ongedaan te maken.

b. **Redo:** Ctrl-Y (Windows), Shift-Ctrl-Z (Linux) of Shift-Cmd-Z (macOS) om wijzigingen (herhaald) opnieuw toe te passen.

5. Grote wijzigingen waarvan je niet zeker weet of die goed zijn? Sla op je code op en maak met de "file manager" door middel van een klik op de file en Ctrl-C/Ctrl-V (Cmd-C/Cmd-V) een reservekopie.

6. Je hebt een reusachtig scherm. Maak gebruik van die ruimte! Zet de vensters van je editor en je browser naast elkaar.
7. Met Alt-Tab schakel je snel tussen beide vensters (apps).
8. Gebruik de pijltjestoetsen en de verschillende andere toetscombinaties om je edit-cursor te positioneren als je zit te editeren. Je hebt je handen tenslotte al op het toetsenbord en het is daarom veel minder belastend voor je muishand en -arm dan telkens je muis te moeten pakken.

9. Verwijder code waar je je ziel en zaligheid ingestopt hebt niet als het niet (helemaal) werkt zoals gepland. Maak er commentaar van voordat je een nieuwe poging waagt. Dan heb je tenminste iets om op terug te vallen en vaak zijn delen ervan opnieuw te gebruiken.

10. In tegenstelling tot wat veel zogenoamde experts met grote stelligheid beweren, is code echt niet zelf-documenterend. Voorzie (ingewikkelde) code van commentaar waar nodig. Je helpt er je collega's mee. Bijkomend voordeel; na 1 maand ben je alles "kwijt". Met wat verklarende tekst is het hoe en het waarom van je code geen puzzle meer.

11. Maak gebruik van de JavaScript-debugger in je browser als je er om wat voor reden dan ook niet meer uitkomt.

Het volgende hoofdstuk is speciaal aan JavaScript-debugging gewijd omdat er in de reguliere lessen geen aandacht wordt besteed aan deze belangrijke programmeeractiviteit.

Een apart hoofdstuk is sowieso gerechtvaardigd omdat het een tamelijk uitgebreid onderwerp is.

SAMENVATTING

- Je leven als JavaScript-programmeur wordt er wat eenvoudiger door bij het consequent gebruik van deze tips & tricks.
- Echter, bedenk wel dat wat voor de één goed werkt precies het tegendeel betekent voor een ander.
- Dokter dus zelf uit wat wel en wat dus echt niet werkt.
- Blijf niet strak vasthouden aan bepaalde gewoonten.
- Zeker niet als je straks in een professionele omgeving een heel andere manier van werken moet gaan gebruiken.

VRAGENUURTJE

- Zijn er onderwerpen aan bod gekomen die je niet snapt?
- Wat precies is je niet duidelijk?
- Hoe kunnen we je dat beter uitleggen?
- Heb je meer suggesties voor het verbeteren van deze onderwerpen?
- Heb je suggesties voor het verbeteren van de presentatie?

DEBUGGEN

JS



WAT KOMT ER AAN BOD?

- Wat is debuggen?
- Waar komt de term vandaan?
- Waarom zou je debuggen?
- Hoe debug je JavaScript?
- Oefeningen.
- Samenvatting.
- Vragenuurtje.

WAT IS DEBUGGEN?

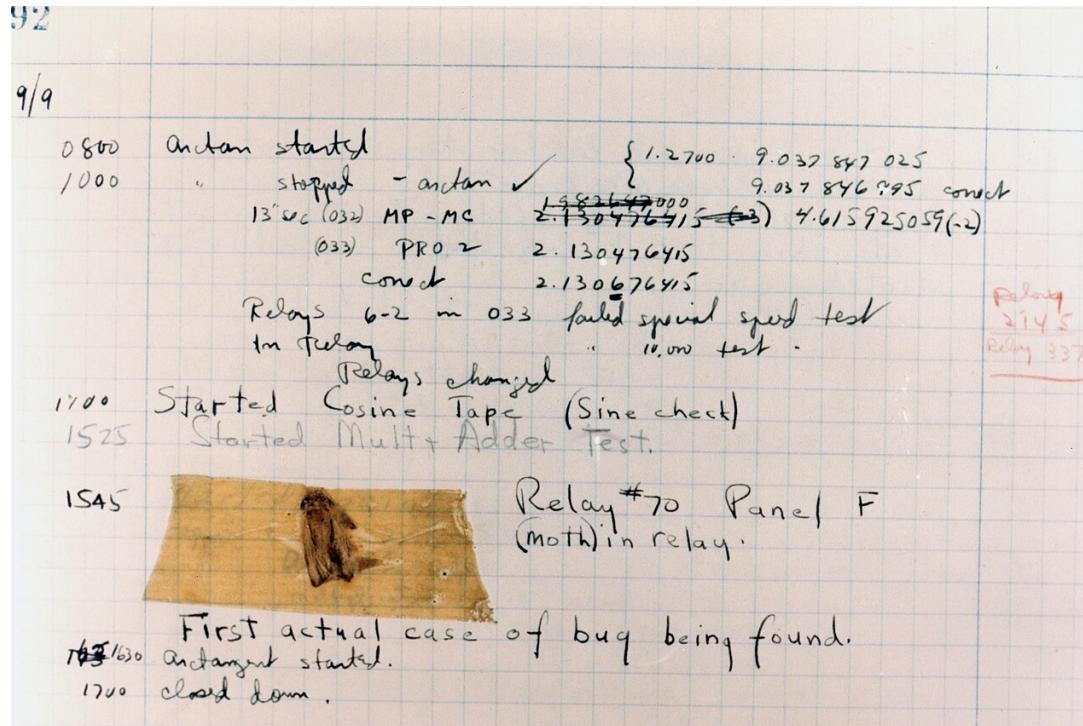
- Debuggen is eigenlijk een beetje als detective spelen voor programma's.
- Stel je een speurtocht voor om uit te zoeken waarom een programma niet werkt.
- Je gaat op zoek naar aanwijzingen en fouten.
- Vervolgens pas je je programma aan zodat alles naar behoren werkt.

DEBUGGEN

JS

WAAR KOMT DE TERM VANDAAN?

- Die vindt zijn oorsprong in de beginlagen van de computers. Grace Hopper, pionier op het gebied van de informatica én drijvende kracht achter de ontwikkeling van de programmeertaal COBOL, gebruikte de term "bug" als eerste.



WAAR KOMT DE TERM VANDAAN?

- Het verhaal gaat dat in 1947 een programma op de Harvard Mark II computer, waar Hopper aan werkte, het niet goed deed.
- Het probleem werd veroorzaakt doordat er een mot ("bug") in een relais van de computer was gekropen.
- Zij verwijderde de mot en noteerde dit in het logboek als "debuggen".
- Sindsdien heet het zoeken naar en verbeteren van fouten in programma's "debuggen".

WAAROM ZOU JE DEBUGGEN?

- Twee redenen:
 1. Om fouten in je programma's te vinden en te verhelpen.

Alleen deze reden is hier van interesse.

- 2. Om er zeker van te zijn dat je programma's werken zoals ze zijn ontworpen.

Dit is feitelijk het testen van programma's. Bij het testen van programma's stel je vast dat er geen bugs zijn. Je mag dat zien als een vorm van debuggen, maar verwarring echt debuggen en testen niet.

WAAROM ZOU JE DEBUGGEN?

- Als je programmeert kun je te maken krijgen met ten minste 3 soorten fouten:
 1. Syntax-fouten. Je script wordt niet eens uitgevoerd als je typfouten, etc. niet verbetert. Typ bijvoorbeeld eens "Arry" in plaats van "Array" of vergeet haakjes, accolades, etc. en JavaScript begint te klagen.
 2. Runtime-fouten. Dat zijn fouten die tijdens de uitvoer van je programma optreden. Deel bijvoorbeeld eens door 0 of probeer een berekening uit te voeren met een "undefined" variabele. Foutief ontwerp van je programma is ook vaak de oorzaak van runtime-fouten.

WAAROM ZOU JE DEBUGGEN?

3. Logische fouten. Het programma gedraagt zich heel anders dan bedoeld. Bijvoorbeeld, een loop die nooit eindigt, een verkeerde expressie in een if, foutieve waarden bij het definiëren van variabelen, onbedoelde neveneffecten, floating-point precisiefouten, etc.

- Met name logische fouten zijn vaak lastig te vinden.
- Als je je realiseert dat:
 - a. Een gemiddelde programmeur **ongewild** 70 bugs voor elke KLOC (**Kilo Lines Of Code** = 1000 regels code) produceert en...

WAAROM ZOU JE DEBUGGEN?

- c. Het oplossen van deze bugs zelfs 30 keer langer kan duren dan het schrijven van de code en...
- b. Dit waarschijnlijk de reden is waarom 15 bugs per KLOC uiteindelijk hun weg vinden naar de klant en...
- c. Ondanks dat goede testprocedures en -protocollen een hoop ellende voorkomen, dan...
- Zal het je duidelijk zijn dat debug-sessies onmisbaar zijn bij het opsporen van fouten.

HOE DEBUG JE?

- Debuggen kan op een aantal manieren:
 1. Door je code statement voor statement te lezen (dus niet op je computer uit te voeren) en vervolgens beredeneren of uitleggen aan collegaprogrammeurs wat er gebeurt. Dit noemt men ook wel "bench checking". Het is effectief, leerzaam en leuk, maar kost tijd.
 2. Je code op strategische plaatsen laten melden wat de toestand is (de plaats en indien nodig, variabelen). Dit is een primitieve vorm van debuggen en wordt veel gebruikt bij eenvoudige code. Effectief en kost relatief weinig tijd.

HOE DEBUG JE?

3. Gebruik maken van een debugger-programma voor een debug-sessie. Een debug-sessie bestaat er kort gezegd uit dat je de code waarvan je vermoedt dat die zich misdraagt stap voor stap uitvoert en controleert of het juiste statement is uitgevoerd én of de inhoud van variabelen klopt.

Debuggers stellen je ondermeer in staat de uitvoer van je programma te onderbreken ("Breakpoints"), variabelen te wijzigen en in ("Step Into") en uit ("Step Out") functies te springen.

Deze debugger-commando's en meer komen straks aan de orde.

HOE DEBUG JE JAVASCRIPT?

- Elke moderne browser heeft een JavaScript debugger.
- In deze presentatie gebruiken we Firefox.
- De debugger in je favoriete browser werkt min of meer hetzelfde.
- De JavaScript debugger zit onder de "Debugger"-tab in de "Developer Tools". (Hoe kon 't ook anders!)
- De debugger ziet er als volgt uit... Zie de screenshot op de volgende dia.
- Merk op dat de "client area" van je browser niet getoond wordt. Die is hier niet van belang.

DEBUGGEN

JS

HOE DEBUG JE JAVASCRIPT?

R
E
A
C
O
L
L
E
G
E

The screenshot shows a browser developer tools window with the 'Debugger' tab selected. A yellow arrow points to the 'Debugger' tab at the top. The interface is divided into several sections:

- Sources:** Shows the file structure. A blue box highlights 'JS opdracht-12.js'. A large yellow number '1' is overlaid on this section.
- Console:** Displays three log statements:

```
516.04 is de gemiddelde waarde over de periode van 2023.  
798.05 is de hoogste waarde over de periode van 2023.  
694.88 is de laagste waarde over de periode van 2023.
```

A large yellow number '4' is overlaid on this section.
- Debugger Panel:** Shows the code being debugged. Line 238 is highlighted with a blue box. A large yellow number '2' is overlaid on this section.
- Breakpoints:** Shows the current state of the debugger. A blue box highlights a checked breakpoint at line 238. A large yellow number '3' is overlaid on this section.
- Call stack:** Shows the execution path. A blue box highlights the entry point 'toonGegevens'. A large yellow number '3+' is overlaid on this section.
- Scopes:** Shows the variable environment. A blue box highlights the variable 'maandNaam'. A large yellow number '3+' is overlaid on this section.
- XHR Breakpoints:** Shows network requests. A blue box highlights an entry. A large yellow number '3+' is overlaid on this section.
- Event Listener Breakpoints:** Shows event listeners. A blue box highlights an entry. A large yellow number '3+' is overlaid on this section.
- DOM Mutation Breakpoints:** Shows DOM mutations. A blue box highlights an entry. A large yellow number '3+' is overlaid on this section.

HOE DEBUG JE JAVASCRIPT?

- De functies van de 4 panelen in de Firefox JavaScript-debugger:
 1. De file-explorer.
 2. De broncode die gedebugd wordt.
 3. Controle- en inspectie.
 4. De JavaScript-console.

HOE DEBUG JE JAVASCRIPT?

- De JavaScript-debugger verwacht dat de code in een extern (apart) JavaScript-bestand staat. Dus zo iets:

```
<script src=".//javascript/voorbeeld.js"></script>
```

- Gebruik je interne JavaScript dan bestaat de kans dat de HTML-code het correct tonen van de JavaScript-code in het broncode-paneel beïnvloedt.
- Het is sowieso "best practice" om je JavaScript in een apart bestand te hebben staan.

HOE DEBUG JE JAVASCRIPT?

- Een voorbeeld van een debug-sessie.
- Open "./code/debug-voorbeeld-1.html" in je browser.
- In "./code/debug-voorbeeld-1.js" zitten een paar opzettelijke fouten.
 1. Het beëindigen van de for-loop in de "processData"-functie is fout.
 2. De property "val" in het "item"-object bestaat niet.
- Open je "Developer Tools".
- Klik op de "Debugger"-tab.
- Vouw de "file-tree" open in het "file-explorer"-paneel.

HOE DEBUG JE JAVASCRIPT?

- Open vervolgens "debug-voorbeeld-1.js" in het "broncode"-paneel door op de filenaam te klikken.
- Scroll (als dat nodig is) naar de regel waar de aanroep van de "processData"-functie staat.
- Klik op het regelnummer.
- Het regelnummer wordt blauw ge-"highlight" en in het "controle- en inspectiepaneel" worden regelnummer en statement getoond.
- Proficiat! Je hebt een "breakpoint" gezet.

HOE DEBUG JE JAVASCRIPT?

- Wat is een "breakpoint"?
 - a. Een "breakpoint" is een punt (een regelnummer) in de broncode van een programma waar een programmeur aangeeft dat de uitvoering van het programma moet pauzeren.
 - b. Nu kan de programmeur de toestand van het programma inspecteren, variabelen bekijken of wijzigen en statements stap voor stap uitvoeren.
 - c. "Breakpoints" helpen programmeurs bij het analyseren wat er in programma's gebeurt en daarmee bij het vinden en oplossen van fouten.

HOE DEBUG JE JAVASCRIPT?

- Ververs je browser-venster.
- De uitvoer van je JavaScript-code "breekt" op regel 31, vóór de aanroep van de "processData"-functie.
- Om de uitvoer in de "processData"-functie te vervolgen klik je op "Step into"-knop, boven aan het "controle- en inspectiepaneel".

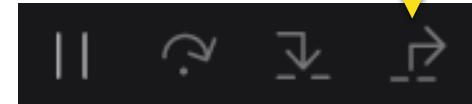
- Door herhaald op deze knop te klikken wordt er telkens een statement uitgevoerd.
- In het "controle- en inspectie"-paneel zie je welke variabelen er "leven" en wat ze bevatten.

HOE DEBUG JE JAVASCRIPT?

- Weet je zeker dat een functie naar behoren werkt, dan kun je die uitvoeren zonder er in te stappen met de "Step over"-knop.



- Stel je na de uitvoer van enkele statements vast dat een functie in orde is, dan kun je met de "Step out"-knop de uitvoer van je programma vervolgen tot het statement dat volgt op de functie-aanroep.
- Voer nu stap voor stap de code uit totdat je de eerste fout tegenkomt.



HOE DEBUG JE JAVASCRIPT?

- Dat zal de niet-bestaaande property zijn.
"transformedValue" krijgt de waarde
"undefined_transformed" omdat "item.val" niet bestaat.
- Corrigeer de code in je editor. Verander "val" in "value" in regel 16. Vergeet niet op te slaan. Je kunt je code niet wijzigen in het "broncode"-paneel.
- Herstart de debug-sessie door je browservenster te verversen.
- Herhaal de debug-sessie totdat je de volgende fout tegenkomt.

DEBUGGEN

JS

HOE DEBUG JE JAVASCRIPT?

- Dat is de beëindiging van de for-loop. Er zijn geen 4 array-elementen.
- Corrigeer de code in je editor. Verander "i <= data.length" in "i < data.length" in regel 21. Je kunt je code nog steeds niet wijzigen in het "broncode"-paneel.
- Herstart de debug-sessie door je browservenster te verversen.
- Herhaal de debug-sessie om vast te stellen dat de code nu probleemloos werkt.
- Verwijder daarna alle "breakpoints".

OEFENINGEN

- Vind met behulp van de JavaScript-debugger de fouten in de vijf volgende stukken code.

1.

```
'use strict';

const cl = console.log;

const multiArray = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

for (let i = 0; i < multiArray.length; i++) {
    for (let j = 0; j < multiArray[i].length; i++) {
        cl(multiArray[i][j]);
    }
}
```

OEFENINGEN

```
2. 'use strict';

const cl = console.log;

const multiArray = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

for (let i = 0; i < multiArray.length; i++) {
    for (let j = 0; j < multiArray[i].length; ) {
        cl(multiArray[i][j]);
    }
}
```

OEFENINGEN

```
3. 'use strict';

const cl = console.log;

const multiArray = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

for (let i = 0; i < multiArray[0].length; i++) {
    for (let j = 0; j < multiArray.length; j++) {
        cl(multiArray[j][i]);
    }
}
```

OEFENINGEN

```
4. 'use strict';

const cl = console.log;

const multiArray = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

for (let i = 0; i < multiArray.length; i++) {
    for (let j = 0; j <= multiArray[i].length; j++) {
        cl(multiArray[i][j]);
    }
}
```

OEFENINGEN

```
5. 'use strict';

const cl = console.log;

const multiArray = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

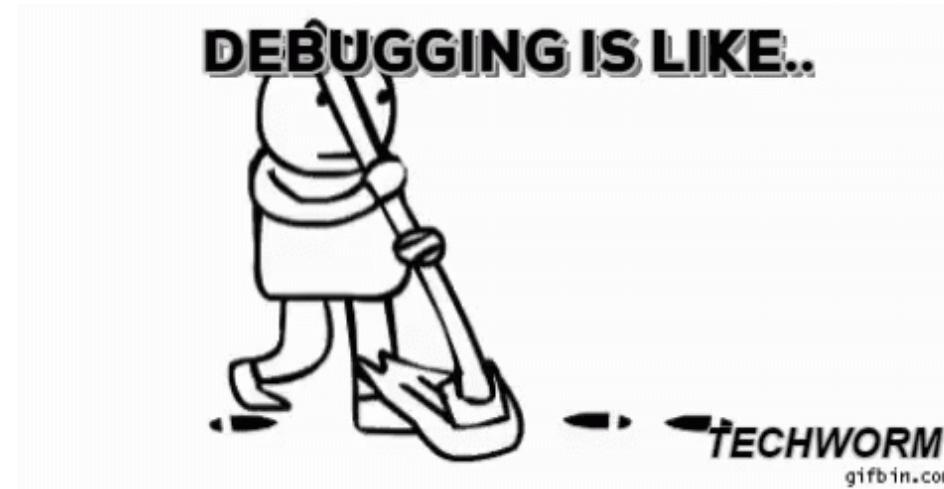
for (let j = 0; j < multiArray[0].length; j++) {
    for (let i = 0; i < multiArray.length; i++) {
        cl(multiArray[i][j]);
    }
}
```

DEBUGGEN

JS

WAARSCHUWING

- Let er wel op dat je debug-sessie niet ontaardt in dit:



SAMENVATTING

- Je weet nu wat debuggen is.
- En ook wat de oorsprong van de term is.
- Je kent een aantal goede redenen om te debuggen.
- Je hebt een voorbeeld JavaScript debug-sessie gedaan.
- Zelf kunnen oefenen.

VRAGENUURTJE

- Zijn er onderwerpen aan bod gekomen die je niet snapt?
- Wat precies is je niet duidelijk?
- Hoe kunnen we je dat beter uitleggen?
- Heb je meer suggesties voor het verbeteren van deze onderwerpen?
- Heb je suggesties voor het verbeteren van de presentatie?

EN TENSLOTTE NOG DIT...

JS

- Met het hoofdstuk over debuggen (ondanks het belang een sterk verwaarloosd onderwerp) besluiten we de presentatie.
- Dank je wel voor jullie deelname aan deze lessen, jullie aandacht voor de diverse onderwerpen, jullie al dan niet opbouwende kritiek én, niet te vergeten, jullie geduld met de presentator.

EN TENSLOTTE NOG DIT...

JS

- Het is al eerder gezegd dat JavaScript een hele reeks gelikte functies bevat om arrays te verwerken.
- Die functies zijn met opzet niet gebruikt om het karakter van arrays te benadrukken.
- Wij hopen dat je door deze presentatie een beter beeld hebt gekregen van wat JavaScript-arrays zijn en hoe je ze gebruikt.
- En heel veel succes met je verdere studie.

APPENDIX - EEN SCENARIO

JS

**WORDT AAN
GEWERKT!!!**