

Object georiënteerd programmeren

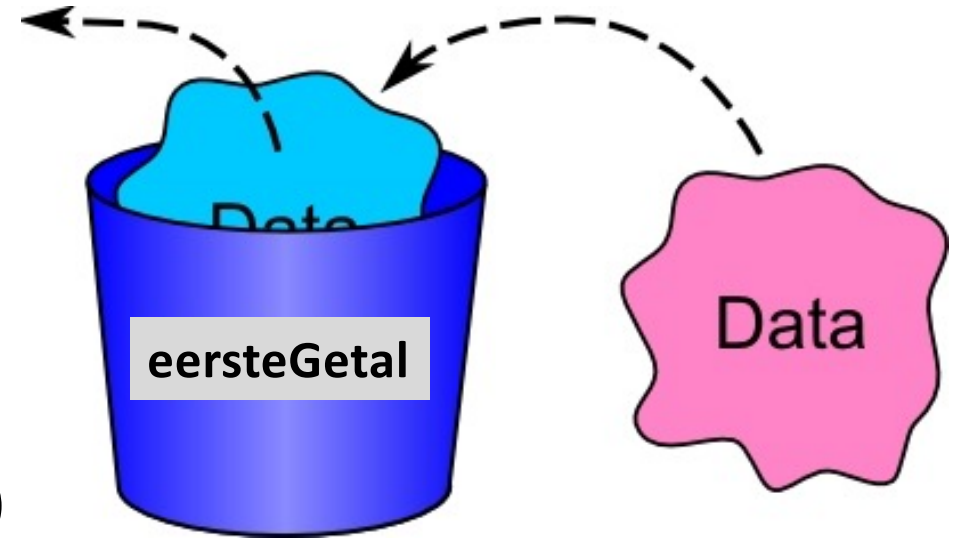
```
43 <body <?php body_class()></div>
44 <div id="fb-root"></div>
45 <script>(function(d, s, id) {
46     var js, fjs = d.getElementsByTagName(s)[0];
47     if (d.getElementById(id)) return;
48     js = d.createElement(s); js.id = id;
49     js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.6&appId=20064663628212";
50     fjs.parentNode.insertBefore(js, fjs);
51 }(document, 'script', 'facebook-jssdk'));</script>
52 <div id="page" class="site">
53     <a class="skip-link screen-reader-text" href="#content"><?php esc_html_e( 'Skip to content', 'urduube' )></a>
54     <div id="header" class="site-header">
55         <div class="site-branding">
56             <div class="navBtn pull-left">
57                 <?php if ( is_home() && $xpanel['homepage-style'] == 1 ) { ?>
58                     <a href="#" id="openMenu"><?php echo esc_html( 'Menu' )></a>
59                     <a href="#" id="openMenu2"><?php echo esc_html( 'Menu' )></a>
60                 <?php } ?>
61             </div>
62             <div class="logo pull-left">
63                 <a href="#"><?php echo esc_url( home_url() ) ?>>
64                     <?php echo $xpanel['logo']['url'] ?>
65                 </a>
66             </div>
67             <div class="search-box hidden-xs hidden-sm pull-left ml-10">
68                 <?php get_search_form(); ?>
69             </div>
70             <div class="submit-btn hidden-xs hidden-sm pull-left ml-10">
71                 <a href="#"><?php echo get_page_link($xpanel['submit-link']) ?> <?php esc_html_e( 'Submit', 'urduube' )></a>
72             </div>
73             <div class="user-info pull-right mr-10">
74                 <?php
75                 if ( is_user_logged_in() ) {
76                     <?php echo esc_html( 'Hello, ' . wp_get_current_user()->display_name . '!' )>
77                 }
```

Even opfrissen...

JavaScript

Wat zijn variabelen?

- Een variabele is een plek in het geheugen waar je een gegeven in kunt zetten. Deze plek geef je een naam in de programmeertaal die je gebruikt.
- Denk aan een bakje (=geheugenplek) met een etiket (=naam) erop, waar je een getal, tekst etc. (=gegeven) in kunt stoppen.



Variabelen en datatypen

- Javascript bepaalt tijdens het uitvoeren welk soort informatie in een variabele wordt bewaart. Dit heet het datatype

```
var i = 0;
```

number

```
var gameOver = true;
```

boolean

```
var startTekst = "Welkom bij de game";
```

string

Variabelen en datatypen

- Javascript bepaalt tijdens het uitvoeren welk soort informatie in een variabele wordt bewaart. Dit heet het datatype

<code>let i = 0;</code>	je mag ook 'let' gebruiken in plaats van 'var'. Dat is niet verplicht maar kan je helpen bij het oplossen en voorkomen van fouten	number
<code>let gameOver = true;</code>		boolean
<code>let startTekst = "Welkom bij de game";</code>		string

If-statement

- met 'if' kun je een programma code wel of niet laten uitvoeren, afhankelijk van de situatie

```
if (mouseX === 0) {  
    println("muis zit helemaal links");  
}  
else {  
    println("muis zit niet helemaal links");  
}
```

Herhalingen

- met 'if' kun je een programma code wel of niet laten uitvoeren, afhankelijk van de situatie

```
for (let teller=0; teller < 10; teller++){  
    println("teller is: " + teller);  
}
```

Herhalingen

- met 'if' kun je een programma code wel of niet laten uitvoeren, afhankelijk van de situatie

```
for (let teller=0; teller < 10; teller++){  
    println("teller is: " + teller);  
}
```


Herhalingen

- met 'if' kun je een programma code wel of niet laten uitvoeren, afhankelijk van de situatie

```
for (let teller=0; teller < 10; teller++){  
    println("teller is: " + teller);  
}
```

Wat betekent dat ook alweer?

Herhalingen

- met 'if' kun je een programma code wel of niet laten uitvoeren, afhankelijk van de situatie

```
for (let teller=0; teller < 10; teller++) {  
    println("teller is: " + teller);  
}
```

- Wat verschijnt er op het scherm?

Herhalingen

- Naast 'for' is er in JavaScript nog een andere structuur waarmee je voor herhaling kunt zorgen. Welke?

Herhalingen

- Naast 'for' is er in JavaScript nog een andere structuur waarmee je voor herhaling kunt zorgen. Welke?

`while`

Herhalingen

- Naast 'for' is er in JavaScript nog een andere structuur waarmee je voor herhaling kunt zorgen. Welke?

```
let teller = 0;
while (teller < 10) {
    println("teller is: " + teller);
    teller++;
}
```

Arrays

Een **array** is een rij van variabelen in een programmeertaal.

- Het volgnummer in de rij (dit heet de index) wordt gebruikt om een losse variabele (dit heet een element van de array) aan te wijzen.

Arrays

Voorbeeld:

```
// maak array met strings en print elementen  
let namen = ["achmed", "bert", "carla"];  
print(namen[0]); // drukt de de naam achmed af  
print(namen[1]); // drukt de de naam bert af  
print(namen[2]); // drukt de de naam carla af
```


Arrays & herhaling is een krachtig duo

Voorbeeld:

```
// maak array met strings en print elementen  
let namen = ["achmed", "bert", "carla", "indy", "oxana"];  
for (let i=0; i<namen.length; i++) {  
    print(namen[i]); // drukt de de naam op index i af  
}
```

Wat is een functie?

- Een functie is een stuk programma met een door de programmeur gekozen naam.
- Een functie is een stuk programma dat je vaker kunt gebruiken.
- Een functie is een samenhangend stuk van een programma.
- Een functie is een stuk van een programma dat je later eenvoudig kunt aanpassen of uitbreiden.

Alle bovenstaande uitspraken zijn waar.

Teken gezicht

```
1 var x = 50;  
2 ellipse(x, 100, 50, 50); //hoofd  
3 ellipse(x-10, 90, 10, 10); //oog links  
4 ellipse(x+10, 90, 10, 10); //oog rechts  
5 ellipse(x, 110, 30, 10); // mond  
6  
7  
8
```



Teken 2 gezichten

```

1  var x = 50;
2  ellipse(x, 100, 50, 50); //hoofd
3  ellipse(x-10, 90, 10, 10); //oog links
4  ellipse(x+10, 90, 10, 10); //oog rechts
5  ellipse(x, 110, 30, 10); // mond
6
7  x = 110;
8  ellipse(x, 100, 50, 50); //hoofd
9  ellipse(x-10, 90, 10, 10); //oog links
10 ellipse(x+10, 90, 10, 10); //oog rechts
11 ellipse(x, 110, 30, 10); // mond
12

```



Teken 2 gezichten, zonder dubbele code

```
1 var tekenSmiley = function(x) {  
2     ellipse(x, 100, 50, 50); //hoofd  
3     ellipse(x-10, 90, 10, 10); //oog links  
4     ellipse(x+10, 90, 10, 10); //oog rechts  
5     ellipse(x, 110, 30, 10); // mond  
6 };  
7  
8 tekenSmiley(50);  
9 tekenSmiley(110);  
10
```

Liever:

```
function tekenSmiley(x) {  
    ellipse(x, 100, 50, 50);  
    ellipse(x-10, 90, 10, 10);  
    ellipse(x+10, 90, 10, 10);  
    ellipse(x, 110, 30, 10);  
}
```



Objecten, klassen attributen en methoden

Variabelen en datatypen

- Javascript bepaalt tijdens het uitvoeren welk soort informatie in een variabele wordt bewaart. Dit heet het datatype

```
var i = 0;
```

number

```
var gameOver = true;
```

boolean

```
var startTekst = "Welkom bij de game";
```

string

Logische eenheden

- In je code heb je vaak stukjes **informatie** en / of **functionaliteit** die bij elkaar horen.
- bijvoorbeeld:
 - de **x- en y- waarde** van een bal, evt. met **horizontale en verticale snelheden**, **samen met de code om de bal een stukje te verplaatsen**.
 - evenzo van een kogel, auto, speler, enz enz
 - de **positie, titel en grootte** van een 'knop', met daarbij **de code die uitgevoerd wordt als je op de knop klikt**

Logische eenheden

- JavaScript kent daarvoor objecten.
- Verzameling van waarden met een label

```
var knopA = { x : 30,  
              y : 100,  
              breedte : 200,  
              hoogte : 50,  
              titel : "Niet klikken"  
            };  
rect(knopA.x, knopA.y, knopA.breedte, knopA.hoogte);  
text(knopA.titel, knopA.x + 10, knopA.y + 10, knopA.breedte-20, knopA.hoogte-20);
```

Logische eenheden

- Of bij een spelletje met een vallende appel

```
var appel = { x: 300,  
              y: 600,  
              speed: 3  
            }
```

- En dan verderop:

```
appel.y = appel.y + appel.speed;
```

Logische eenheden

- Of, in het geval van onze simulator:

```
var appel = { x: 300,  
              y: 600,  
              speed: 3  
            }
```

- En dan verderop:

```
appel.y = appel.y + appel.speed;
```

- Wat te doen bij meerdere appels? appelA, appelB, appelC?

Logische eenheden

- Nog beter: als naamloze objecten in een array

```
var appels = [ { x: 300,           for (var i=0; i < appels.length; i++) {
                  y: 600,           appels[i].y = appels[i].y + appels[i].speed;
                  speed: 3         }
                },
                { x: 800,
                  y: 300,
                  speed: 1
                } // etcetera
];
```

Logische eenheden

- De updatecode hoort eigenlijk ook bij het object. Dat doe je zo:

```
var appel = { x: 300,  
              y: 600,  
              speed: 3,  
              update() {  
                this.y = this.y + this.speed;  
              }  
            }  
appel.update();
```

- Waarom 'this'?
- -> De code in update kan niet 'weten' dat het object beschikbaar is onder het label 'appel';

Wat is wat?

```
var appel = { x: 300,  
              y: 600,  
              speed: 3,  
              update() {  
                this.y = this.y + this.speed;  
              }  
};  
appel.update();
```

} attributen

} methode

Probleempje...

```
var appels = [ { x: 300,  
                y: 600,  
                speedX: 2,  
                speedY: -3,  
                update() {  
                    this.y = this.y + this.speed;  
                }  
            },  
            { x: 800,  
              y: 300,  
              speedX: -4,  
              speedY: 1,  
              update() {  
                  this.y = this.y + this.speed;  
              }  
            } // etcetera  
];
```

```
for (var i=0; i<appels.length; i++) {  
    appels[i].update()  
}
```

Dubbele methoden

- Voor ieder object opnieuw de methodes schrijven is zonde van de tijd en opslagruimte.
- Waarom kunnen we geen objecten maken van eerder gemaakt ontwerp?
- Dat kan met de beschrijving van een klasse:

Beschrijf de class Appel

```
class Appel {  
    x;  
    y;  
    speed;  
  
    constructor(x, y, speed) {  
        this.x = x;  
        this.y = y;  
        this.speed = speed  
    }  
}
```


Beschrijf de class Appel (met update-methode)

```
class Appel {  
    x;  
    y;  
    speed;  
  
    constructor(x, y, speed) {  
        this.x = x;  
        this.y = y;  
        this.speed = speed;  
    }  
  
    update() {  
        this.y = this.y + this.speed;  
    }  
}
```

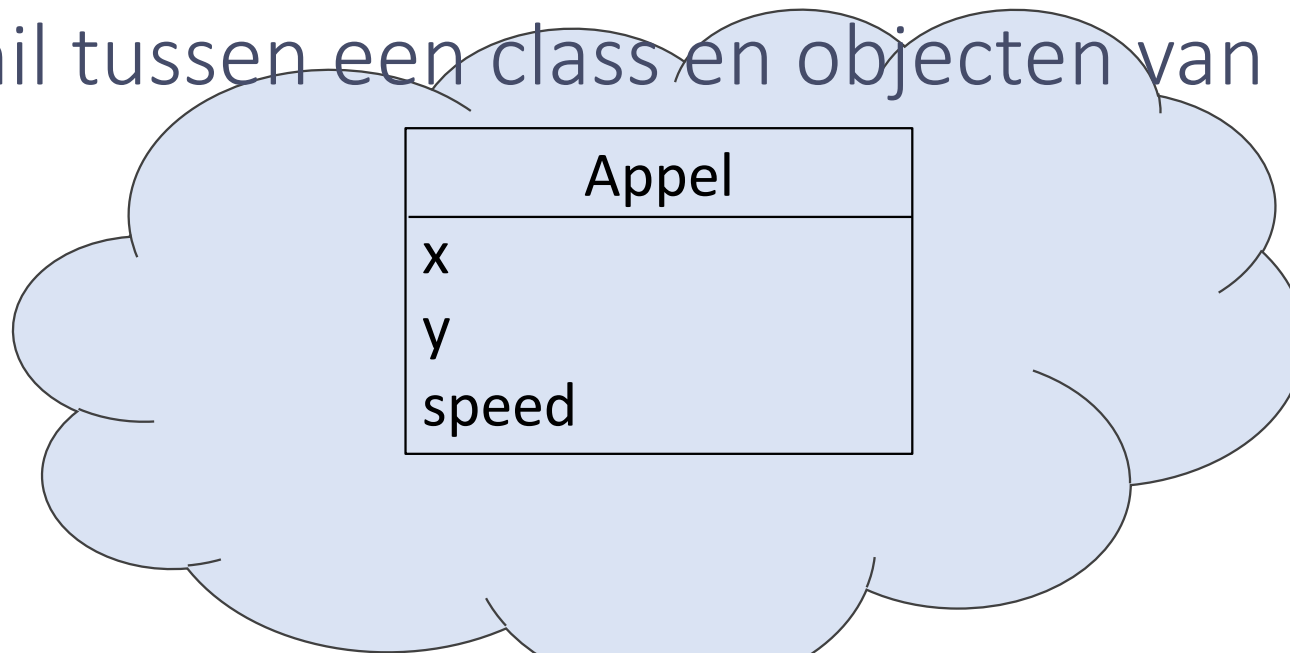
Beschrijf de class Mens (nu met deel van update)

```
class Appel {  
    x;  
    y;  
    speed;  
  
    constructor(x, y, speed) {  
        this.x = x;  
        this.y = y;  
        this.speed = speed;  
    }  
  
    update() {  
        this.y = this.y + this.speed;  
    }  
}
```

constructor wordt aangeroepen met 'new', zoals:
`var appel = new Appel(50, 50, -7);`



Verskil tussen een class en objecten van die class:



appelA : Appel		
x	=	50
y	=	50
speed	=	2

appelB : Appel		
x	=	74
y	=	24
speed	=	4

appelC : Appel		
x	=	150
y	=	91
speed	=	5

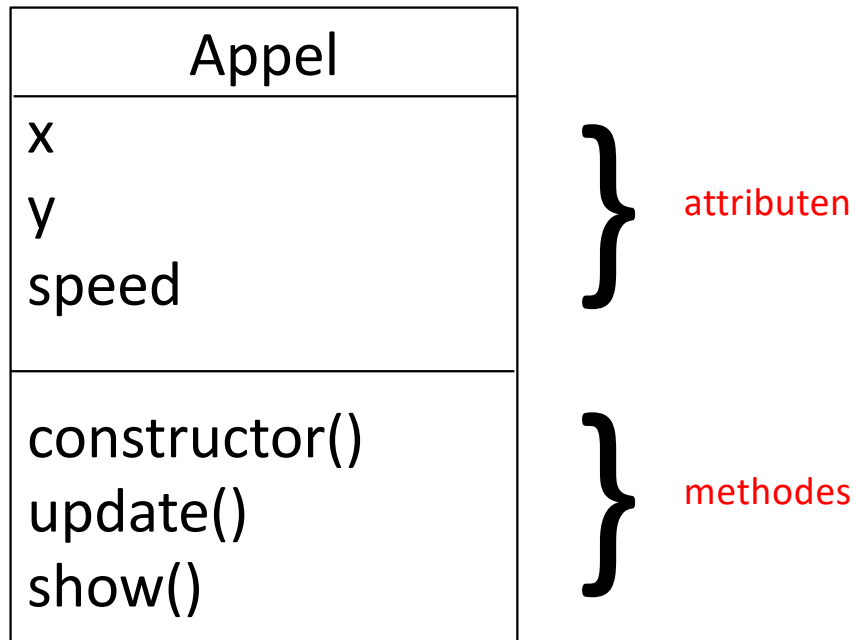
Hoe definieer ik een class?

```
class <Naam van de class> {  
    attribuut1;  
    attribuut2;  
  
    constructor(parameter1, parameter2) {  
        this.attribuut1 = parameter waarvan je de waarde wilt gebruiken;  
        this.attribuut2 = parameter waarvan je de waarde wilt gebruiken;  
    }  
  
    methodenaam() {  
        // code die uitgevoerd moet worden  
  
        return <waarde>; // alleen als er een waarde teruggegeven moet worden  
    }  
}
```

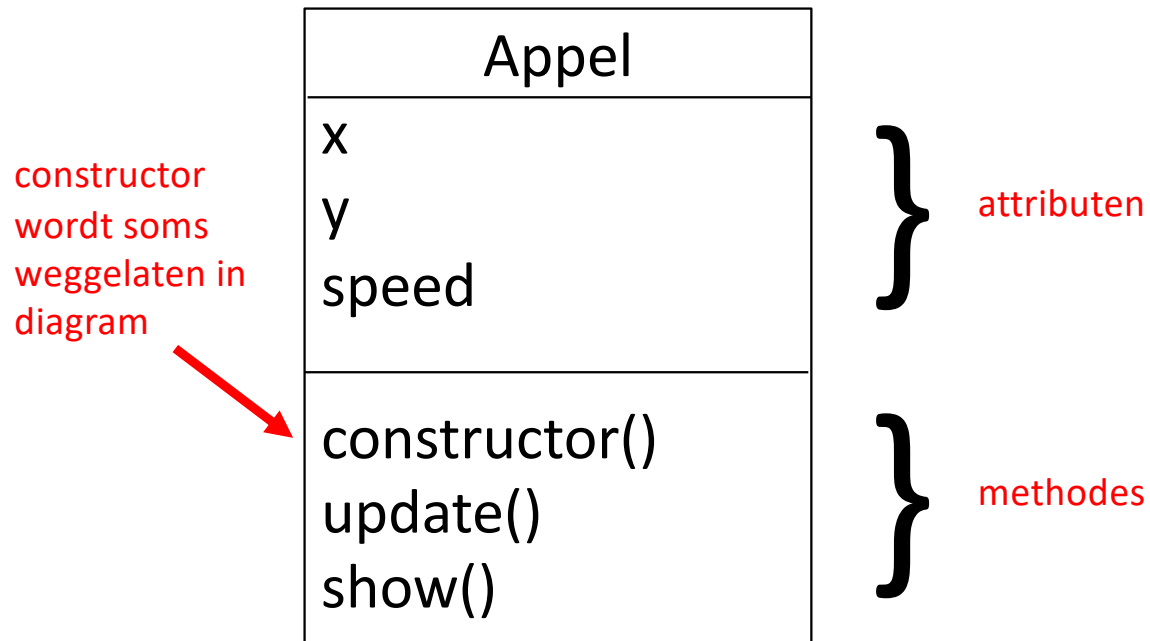


```
class Bal {  
    x;  
    y;  
    speedX;  
    speedY;  
  
    constructor(x, y, speedX, speedY) {  
        this.x = x;  
        this.y = y;  
        this.speedX = speedX;  
        this.speedY = speedY;  
    }  
  
    show() {  
        fill(255, 100, 255);  
        ellipse(this.x, this.y, 80, 80);  
    }  
  
    update() {  
        this.x = this.x + this.speedX;  
        this.y = this.y + this.speedY;  
  
        // hier moet ook de code voor het kaatsen komen  
        // . . .  
    }  
}
```

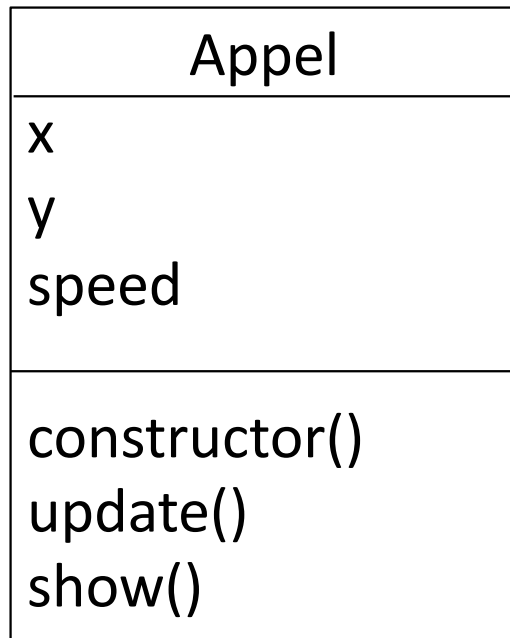
Een diagram praat vaak makkelijker



Een diagram praat vaak makkelijker



Een diagram praat vaak makkelijker



Dit is een klassendiagram

Objectdiagram

Soms wil je de toestand van objecten in een diagram zetten. Dat doe je zo:

appelA : Appel	
x	= 50
y	= 50
speed	= 2

appelB : Appel	
x	= 74
y	= 24
speed	= 4

appelC : Appel	
x	= 150
y	= 91
speed	= 5

Objectdiagram

Soms wil je de toestand van objecten in een diagram zetten. Dat doe je zo:

objectnaam : klassenaam

appelA : Appel	
x	= 50
y	= 50
speed	= 2

appelB : Appel	
x	= 74
y	= 24
speed	= 4

appelC : Appel	
x	= 150
y	= 91
speed	= 5

in een objectdiagram zet je
geen methodes