Operatoren

- De operator + voor optellen.
- De operator - voor aftrekken.
- De operator * voor vermenigvuldigen.
- De operator / voor de deling.
- De operator % voor de rest van de deling.

De eerste vier operatoren zijn nogal standaard. Alleen de operator % vraagt om wat toelichting. Een paar voorbeelden:

```
30 % 7    // levert 2, omdat na deling van 30 door 7 er 2 overblijft
125 % 60  // levert 5
10 % 8    // levert 2
```

De operator % kunt u gebruiken om te controleren of een bepaald getal deelbaar is door een ander getal. Immers, als twee getallen deelbaar zijn, is de rest gelijk aan 0.
Bijvoorbeeld: 15 % 3 == 0, dus 15 is deelbaar door 3.

Toekenningsoperatoren
JavaScript kent nog meer assignment operators. Een daarvan is +=.
Een voorbeeld:

```
var teller = 5;
teller += 1;
```

De laatste opdracht doet precies hetzelfde als:

```
teller = teller + 1;
```

Hier staat: de nieuwe waarde van `teller` wordt gelijk aan de huidige waarde plus 1.
Het voordeel van het gebruik van de operator += is dat u de naam van de variabele maar één keer hoeft te noemen. Op dezelfde manier kunt u andere waarden dan 1 optellen met behulp van deze operator:

```
var teller = 10;
teller += 3;        // teller wordt nu 13
```

of

```
var teller = 500;
var toename = 100;
teller += toename;  // teller wordt 600
```

De assignment-operator += staat niet op zichzelf. Ook voor de operatoren -, *, / en % zijn er overeenkomstige toekenningsoperatoren. Zo hebben bijvoorbeeld de statements in de linkerkolom dezelfde betekenis als die in de rechterkolom:

```
hoeveelheid=hoeveelheid - 10;  |  hoeveelheid -= 10;
aantal = aantal * 2;           |   aantal *= 2;
lengte = lengte / 3;           |   lengte /= 3;
rest = rest % 12               |   rest %= 12;
```

Volgorde van bewerking
Als er meer dan één operator in een bewerking staat, is het belangrijk de volgorde van de bewerkingen te kennen:

- Vermenigvuldigen en delen zijn gelijkwaardig.
- Optellen en aftrekken zijn gelijkwaardig.
- Gelijkwaardige bewerkingen worden in principe van links naar rechts uitgevoerd.
- Door middel van haakjes kunt u een andere volgorde afdwingen.

Voor de volgorde van bewerking worden in het Engels de termen *operator precedence* of *operator priority* gebruikt.

Een paar voorbeelden:

```
var a = 3 + 4 * 5;        // 23
var b = (3+4)*5;          // 35
var c = 10 / 2 * 5;       // 25
var d = 10 / (2 * 5);     // 1
```

Zie *Flanagan* paragraaf 4.7 (Operator Overview) voor een volledig overzicht.

De increment-operator ++ en de decrement-operator –
JavaScript kent de *increment-operator* ++ om de waarde 1 op te tellen bij een variabele.
Het statement

```
teller++;
```

heeft hetzelfde effect als

```
teller += 1;
```

en dit betekent weer hetzelfde als

```
teller = teller + 1;
```

De increment-operator ++ wordt veel gebruikt in *for-statements*, dat we in een volgend hoofdstuk zullen tegenkomen.

Als tegenhanger van ++ bestaat de *decrement-operator* --, die de waarde van de variabele met 1 vermindert.

Postfix en prefix
De operatoren ++ en -- kunt u achter of voor de variabele zetten. Het maakt verschil of u schrijft

```
bedrag = aantal++ * prijs;
```

of

```
bedrag = ++aantal * prijs;
```

Als de ++ voor de variabele in kwestie staat, noemen we dit een *prefix*-operator. Staat de ++ na de variabele, dan noemen we dit een *postfix*-operator. In de volgende twee voorbeelden komt het verschil tussen prefix- en postfix-gebruik van de operator ++ tot uitdrukking.

Eerst postfix:

```
var aantal = 10;
var prijs = 5.00, bedrag;
bedrag = aantal++ * prijs;      // postfix
```

Na afloop heeft bedrag de waarde 50 en aantal de waarde 11.

Vervolgens prefix:

```
var aantal = 10;
var prijs = 5.00, bedrag;

bedrag = ++aantal * prijs;      // prefix
```

Na afloop heeft bedrag de waarde 55 (in plaats van 50) en aantal de waarde 11.

Bij het gebruik van ++ als *prefix*-operator wordt eerst de variabele verhoogd, en daarna de waarde van de variabele gebruikt in de expressie. Dus in het laatste voorbeeld krijgt `aantal` eerst de waarde `11`, en die waarde wordt gebruikt in de berekening.

Als u de ++ als *postfix*-operator gebruikt, wordt eerst de waarde van de variabele gebruikt om de expressie te berekenen, en vervolgens wordt de variabele met `1` verhoogd. In het eerste voorbeeld is de waarde `10` van `aantal` gebruikt voor de berekening van `bedrag`. Als dat gebeurd is, wordt `aantal` verhoogd van `10` naar `11`.