Het do-while-statement, het switch-statement en de statements break en continue

Het do-while-statement

Naast een `for`-statement en een `while`-statement kent JavaScript nog een derde herhalingsopdracht: een `do-while`-statement. Een `do-while`-statement is eigenlijk hetzelfde als een `while`-statement met als enige verschil dat bij een `do-while`-statement de conditie wordt getest na afloop van de body in plaats van ervoor.

Dit is de algemene vorm:

```
do {
  body
} while( conditie );
```

Anders dan bij het `while`-statement uit de vorige paragraaf wordt de body van dit statement altijd ten minste één keer uitgevoerd. Er zijn weinig situaties waarin het handig is een `do-while`-statement te gebruiken. In verreweg de meeste gevallen waarin u een herhalingsopdracht nodig hebt, is het beter een `while`-statement of een `for`-statement gebruiken.

Het switch-statement

Het komt voor dat u tussen drie, tien of nog meer gevallen onderscheid wilt maken. U kunt dan een hele rij `if-else`-statements achter elkaar zetten, bijvoorbeeld:

```
var cijfer == ...;
var s;

if( cijfer == 10 )
  s = "Uitmuntend";
else
if( cijfer == 9 )
  s = "Zeer goed";
else
if( cijfer == 8 )
  s = "Goed";
else
if
  .
  .
else
if( cijfer == 1 )
  s = "Zeer slecht";
else
  s = "Geen geldig cijfer";
alert( s );
```

Hoewel dit voorbeeld in al zijn eenvoud misschien wel duidelijk is, komt zo'n lange reeks `if-else`-statements de leesbaarheid van een programma niet ten goede. U kunt dit overzichtelijker oplossen met een `switch`-statement:

```
var cijfer = ...;
vas s;
switch( cijfer ) {
  case 10:  s = "Uitmuntend";
            break;
  case  9:  s = "Zeer goed";
            break;
  case  8:  s = "Goed";
            break;
  //    .
  //    .    et cetera
  //    .
  case  1:  s = "Zeer slecht";
            break;
  default:  s = "Geen geldig cijfer";
}
alert(s);
```

In dit `switch`-statement wordt de waarde van `cijfer` vergeleken met de waarden die achter de woorden `case` staan. Deze waarden heten de *case labels*. Als de waarde gelijk is aan een van de case labels, worden de statements achter de dubbele punt uitgevoerd. Bij het `break`-statement aangekomen, springt het programma verder naar de sluitaccolade van het `switch`-statement, dat daarmee klaar is.

Als de waarde van `cijfer` niet gelijk is aan een van de case labels, worden de statements achter `default` uitgevoerd. Het is overigens niet verplicht zo'n default-label in elk `switch`-statement op te nemen.

Een `break`-statement sluit elk van de `case`-statements af. Als u een `break` weglaat, dan zal ook het `case`-statement daaronder worden uitgevoerd, bijvoorbeeld:

```
var s;
switch( cijfer ) {
  case 10:  s = "Uitmuntend";
            break;
  case  9:  s = "Zeer goed";
  case  8:  s = "Goed";
            break;
                                  //  et cetera
}
```

Als de waarde van `cijfer` gelijk is aan `9`, zal s eerst de waarde `"Zeer goed"` krijgen en daarna meteen worden overschreven door `"Goed"`.
Dit komt omdat na `case 9` *alle* statements worden uitgevoerd *tot aan de eerstvolgende* `break`. Dus in dit geval ook het statement achter `case 8`.

Zie w3schools.com - switch.

De statements break en continue

Het `break`-statement wordt gebruikt om naar het einde van een `switch`-statement te springen. Het kan ook dienen om naar het einde van een loop te springen. Een weinig zinvol voorbeeld staat hieronder:

```
var s = "";
for (i = 0; i < 10; i++) {
    if (i == 3) break;

    s += "De waarde van i is " + i + "<br>";
}
```

De body van het `for`-statement zal niet tien keer worden uitgevoerd, maar slechts drie keer voor i = 0, 1 en 2. Als i de waarde 3 heeft, zorgt het `break`-statement ervoor dat de loop meteen wordt afgebroken. Het was in dit geval veel duidelijker geweest als het `for` -statement er zo uit zou zien:

```
var s = "";
for (i = 0; i < 3; i++) {
     s += " De waarde van i is " + i + "<br>";
 }
```

Behalve in een `for`-statement kunt u een `break`-statement ook in een `while`- of in een `do-while`-statement gebruiken. In het algemeen wordt code er niet leesbaarder op als er veel `break`-statements in staan. Vaak kan door het iets anders formuleren van de voorwaarde van de loop hetzelfde resultaat worden bereikt zonder break.

Hetzelfde kan gezegd worden van een `continue`-statement. Dit statement zorgt ervoor dat er gesprongen wordt naar het einde van de body van de loop, waarna de conditie opnieuw getest wordt en de loop (afhankelijk van de conditie) eventueel verdergaat. Het `continue`-statement zorgt er dus voor dat er over een stukje van de body heen gesprongen wordt. Ook het `continue`-statement komt de leesbaarheid niet erg ten goede en kan meestal beter worden opgelost met een `if`- of `if-else`-statement.

Zie voor `break` en `continue`: w3schools.com - break.