Conversie van number naar string en omgekeerd
Het omgekeerde is (in principe) ook mogelijk: van een string een getal maken. Dit kan op verschillende manieren. Eén van die manieren is door een plusteken voor de string te zetten.

```
var s = "123";      // de string "123"
var g = +s;         // het getal 123
```

De omzetting van string naar getal kan natuurlijk niet als de string niet uit een getal bestaat.

```
var s = "elswout";  // de string "elswout"
var g = +s;         // een getal??
```

In dit geval geeft JavaScript aan de variabele g een speciale waarde met de naam *Not a Number* ofwel NaN. Zie ook w3schools.com onder het kopje The Unary + Operator.

Een andere manier om een string naar een number om te zetten, is met behulp van een van de functies `parseInt()` of `parseFloat()`. De functie `parseInt()` probeert (het eerste gedeelte van) de string naar een geheel getal te converteren, en `parseFloat()` probeert (het eerste deel van) de string te converteren naar een float, dat is een getal met een decimale punt. Deze functies werken niet helemaal hetzelfde als het plusteken. Een plusteken converteert bijvoorbeeld een string die enkel uit een spatie bestaat naar 0, en `parseInt()` converteert deze naar NaN.
En `parseInt()` converteert "123A" naar 123, maar het plusteken converteert dit naar NaN.

Zie w3schools.com - parseInt() voor voorbeelden van `parseInt()`, en w3schools.com - parseFloat() voor voorbeelden van `parseFloat()`.

Het converteren van het ene type naar het andere heet ook wel *typecasting*.

Indien u probeert een string (of een andere waarde) om te zetten naar een number, en dit lukt niet, dan levert dat de waarde NaN (Not a Number).

Invoer
Een prettige manier van invoeren voor de gebruiker is m.b.v. een HTML-tekstvak. Zoals u waarschijnlijk wee,t kunt u die maken met:

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

Door het vakje een id te geven, kunt u er vanuit JavaScript aan refereren.

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

Als de gebruiker iets in het vak invoert en op Enter drukt, treedt er een event op, een onchange-event. Deze event kunt u opvangen in een JavaScript-functie.

```
<input type="text" id="inputgetal" onchange="myFunction()">
```

Omdat de functie de event afhandelt, heet hij een event-handler. De event-handler moet gedefinieerd worden in een script:

```
<script>
function myFunction() {
  ...
}
</script>
```

In de functie kunt u het HTML-element (het tekstvak) via zijn id opvragen en in een variabele opbergen:

```
function myFunction() {
  var invoervak = document.getElementById("inputgetal");

  ...
}
```

De variabele met de naam invoervak is een object, beter gezegd een referentie naar een object. In dat object zit allerlei informatie over het HTML-tekstvak. Die informatie zijn de zogeheten *properties* van het object. Een van de properties van het invoervak heet `value`. In deze property zit de tekst die de gebruiker in het vak heeft ingevoerd. De inhoud van `value` kunt u opvragen met `invoervak.value`. Dus:

```
function myFunction() {
  var invoervak = document.getElementById("inputgetal");
  var getal = +invoervak.value;

  ...
}
```

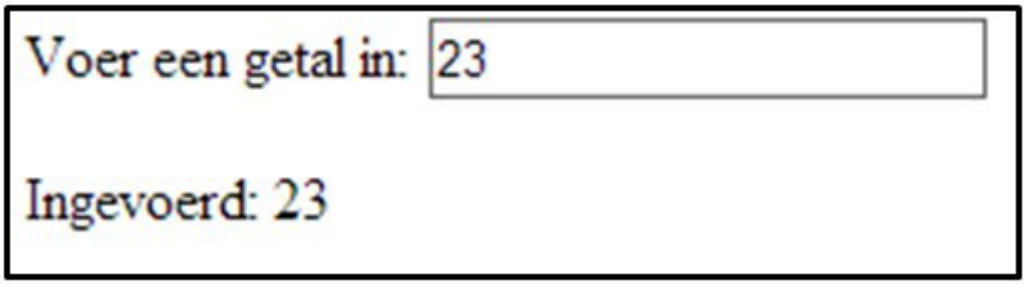
Let op de + voor `invoervak.value`. Een tekstvak levert een string als value en de plus zorgt voor de omzetting van een string naar een number.

Nu het getal dat de gebruiker heeft ingevoerd in de functie is aangekomen, kunt u er van alles mee doen. In dit eenvoudige voorbeeld zetten we de invoer op de pagina.
Hieronder staat de volledige code.

```
<!DOCTYPE html>
<html>
<body>
Voer een getal in:
<input type="text" id="inputgetal" onchange="myFunction()">
<p id="resultaat"></p>

<script>
function myFunction() {
    var invoervak = document.getElementById("inputgetal");
    var getal = +invoervak.value;
    document.getElementById("resultaat").innerHTML =
        "Ingevoerd: " + getal;
}
</script>
</body>
</html>
```

Uitvoer van de browser:



Zoals u weet kunt u als alternatief voor de omzetting met het plusteken van string naar number `parseInt()` of `parseFloat()` gebruiken. Zie w3schools.com - parseInt() voor voorbeelden van `parseInt()`, en w3schools.com - parseFloat() voor voorbeelden van `parseFloat()`.

Het type boolean
Het type *boolean* is nodig om condities te testen. Met behulp van een conditie kan een gedeelte van een script onder een bepaalde voorwaarde worden uitgevoerd of niet. Dit is wezenlijk voor veel toepassingen.

Een vergelijkingsoperator (comparison operator of relational operator) vergelijkt twee uitdrukkingen met elkaar, zoals de operator `>` (groter dan), die kijkt of het een groter is dan het ander. Zo kunt u bijvoorbeeld de vraag stellen of `x` een waarde heeft die groter is dan `3`. Met behulp van de operator `>` wordt dat: is `x > 3`? Het antwoord op deze vraag hangt natuurlijk van de waarde van `x` af. Als `x` de waarde `10` heeft, dan is het antwoord: ja, dat is waar. Als `x` daarentegen de waarde `1` heeft, dan is het antwoord: nee, dat is niet waar. Het antwoord op een vraag met een vergelijkingsoperator kent altijd maar twee mogelijkheden:

ja, dat is waar
nee, dat is niet waar

Het Engels voor *waar* en *niet waar* is `true` respectievelijk `false`. Ook JavaScript kent deze woorden, en ze vormen met zijn tweeën een apart primitief type dat *boolean* heet. Het woord boolean is afgeleid van de naam van de Engelse wiskundige George Boole, die rond 1850 een studie heeft gemaakt van de logica van true en false. De waarden `true` en `false` heten ook wel *logische waarden* (logical values) of *waarheidswaarden* (truth values).

In de tabel hierna staan de vergelijkingsoperatoren voor het vergelijken van waarden in JavaScript.

`<`	kleiner dan
`>`	groter dan
`<=`	kleiner dan of gelijk aan
`>=`	groter dan of gelijk aan
`==`	is gelijk aan
`===`	is gelijk aan en van hetzelfde type
`!=`	is ongelijk aan
`!==`	is ongelijk aan of niet van hetzelfde type

Merk op dat de operator *is gelijk aan* uit **twee** isgelijktekens bestaat. Een fout die veel beginnende JavaScript-programmeurs maken, is dat ze één enkel isgelijkteken (*krijgt de waarde*) gebruiken in plaats van het dubbele isgelijkteken (*is gelijk aan?*).

Als de variabele `jaartal` de waarde `2000` heeft, dan geldt:

`jaartal < 1900` is niet waar, dus levert de waarde `false.`
`jaartal == 2000` is waar, dus levert de waarde `true.`
`jaartal != 2000` is niet waar, dus levert de waarde `false.`
`jaartal >= 1900` is waar, dus levert de waarde `true.`
`jaartal <= 2000` is waar, dus levert de waarde `true.`

Voor de laatste uitdrukking geldt dat er `true` uitkomt als `jaartal` kleiner is dan `2000`, of als jaartal gelijk is aan `2000`. De betekenis van `<=` is immers: kleiner of gelijk. Voor de operator `>=` geldt iets dergelijks: er komt `true` uit als de waarde links groter is of gelijk is aan die aan de rechterkant.

In JavaScript kunt u getallen met strings vergelijken:

```
5 == "5";  // levert true, omdat 5 gelijk is aan 5
11 < "10"; // levert false, omdat 11 niet kleiner is dan 10
```

De string wordt automatisch geconverteerd naar een getal om de vergelijking te kunnen uitvoeren. Als de string geen geldig getal is, levert de conversie NaN op, zie ook de vorige paragraaf.

JavaScript heeft nog een manier om waarden met elkaar te vergelijken in de vorm van drie isgelijktekens: `===`.
Met deze operator wordt niet alleen gekeken of de waarden links en rechts overeenkomen, maar ook of de typen hetzelfde zijn.

```
5 == "5";  // levert true, omdat 5 gelijk is aan "5" (na conversie)
5 === "5"; // levert false, omdat de typen niet overeenkomen
```

Ook voor ongelijk heeft JavaScript twee operatoren: `!=` en `!==`.

Waarden van het type boolean worden onder andere gebruikt in combinatie met zogeheten conditionele statements, zie een van de volgende paragrafen.

Het type vaststellen met typeof
JavaScript heeft dynamische typen, dat wil zeggen dat het type van dezelfde variabele x gedurende de uitvoering van het script kan veranderen:

```
var x;           // x is nu undefined
x = 3;           // en nu is x van het type number
x = "Johan";     // en nu is x van het type string
```

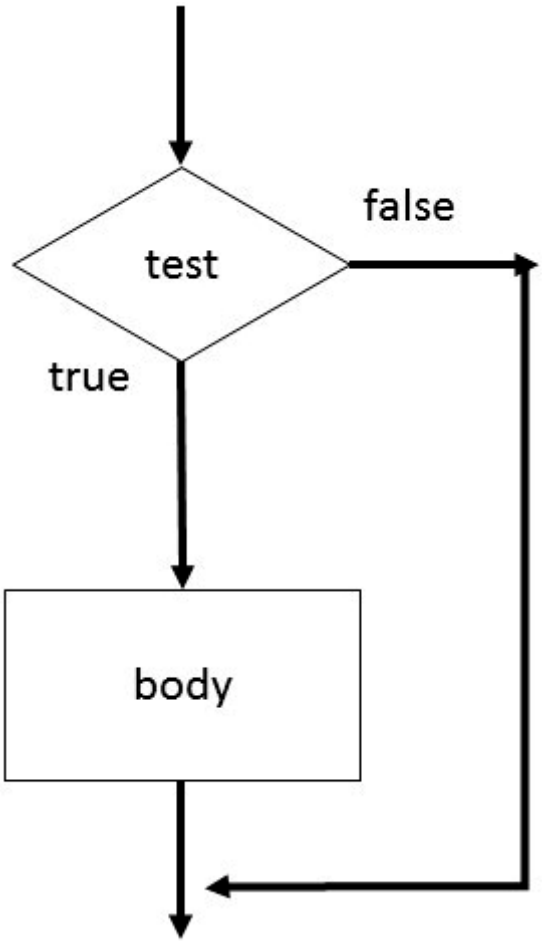
Met de operator *typeof* kunt u desgewenst het type van een variabele of waarde vaststellen, zie w3schools.com – datatypes onder het kopje The typeof Operator.

Conditionele statements
Het `if`-statement is een zogeheten conditioneel (voorwaardelijk) statement. Het statement bestaat uit het woord `if`, met daarachter tussen haakjes een *conditie* (test of voorwaarde) en een body:

```
if( conditie ) {
  body
}
```

Op de plaats van de body kunnen één of meer statements staan. Als de body uit slechts één statement bestaat, mag u de accolades weglaten, maar het is meestal verstandiger dat niet te doen, zie de opmerking aan het eind van deze paragraaf.

In een `if`-statement wordt getest of de conditie waar is of niet. Als de conditie `true` oplevert, wordt de body van het `if`-statement uitgevoerd en anders overgeslagen. Je kunt een `if`-statement weergeven met de volgende figuur.



Het if … else-statement
In veel gevallen moet er iets gebeuren als een conditie `true` oplevert, en moet er iets anders gebeuren als de conditie `false` oplevert. Als u de keuze hebt uit hetzij het een, hetzij het ander, gebruikt u een `if...else`-statement.

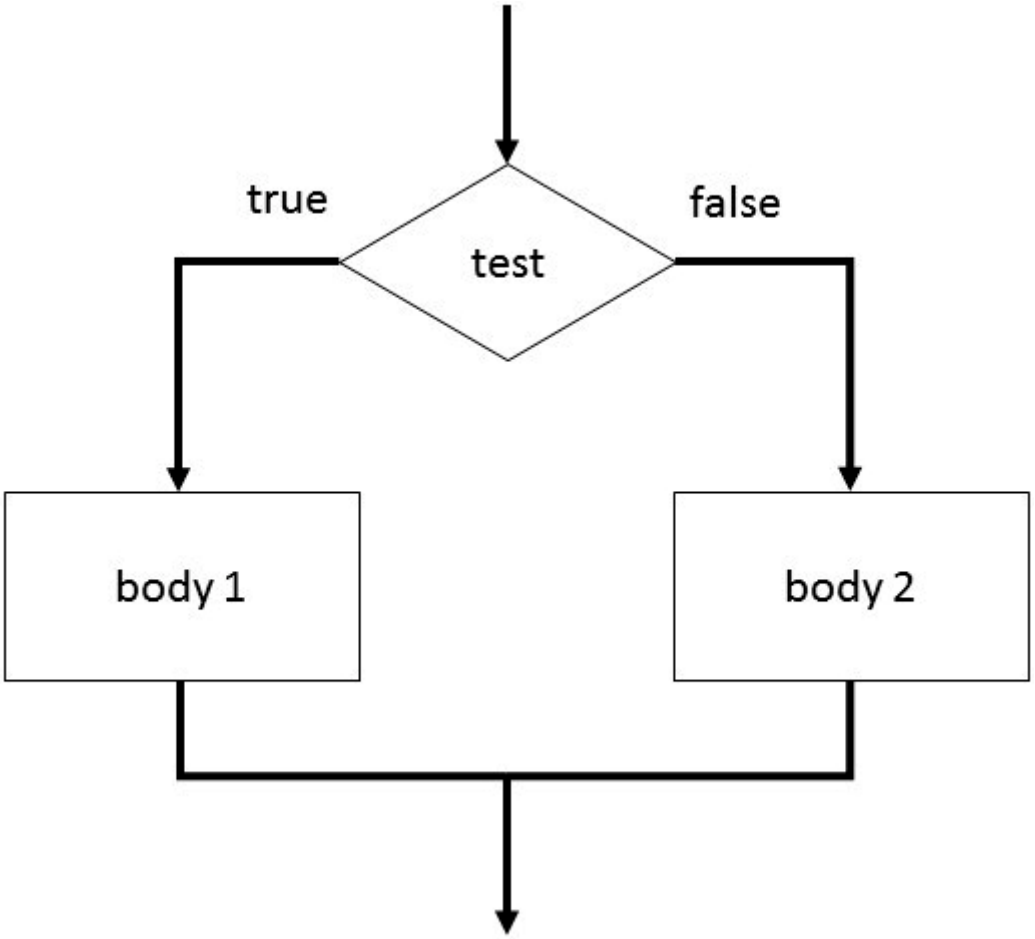
De algemene gedaante van een `if...else`-statement is:

```
if( conditie ) {
  body1
}
else {
  body2
}
```

Het `if...else`-statement heeft twee body's: de eerste wordt uitgevoerd als de voorwaarde `true` oplevert, de tweede als de voorwaarde `false` oplevert. Wat de conditie ook mag zijn, in elk geval zal één van de twee body's worden uitgevoerd.

Ook hier geldt dat als in de body maar één statement staat, u de accolades om de body weg kunt laten, maar vaak is het duidelijker ze te laten staan. Zodra er meer dan één statement in de body staat, moeten er accolades omheen.

De volgende figuur maakt nog eens duidelijk wat er precies gebeurt bij een `if...else`-statement.



Andere waarden die false of true kunnen zijn
Behalve false en true zelf en de uitkomst van een uitdrukking met een vergelijkingsoperator, kunnen in JavaScript ook alle andere waarden als false of true fungeren.
De volgende waarden kunnen de rol van `false` spelen:
`false`
`undefined`
`null`
`0`
`NaN`
de lege string ("")

Alle andere waarden kunnen de rol van `true` spelen.

Er zijn situaties waarin u hiervan handig gebruik van kunt maken, maar voorlopig concentreren we ons op de 'normalere' gevallen.

Logische operatoren
De en-operator &&
Soms moet u twee of meer dingen met elkaar vergelijken:
Is `jaartal >= 2000` én is `jaartal <= 2010`?

Met andere woorden: ligt `jaartal` tussen `2000` en `2010`? In zo'n geval hebt u de *en*-operator `&&` nodig.

De gedeelten die links en rechts van een operator staan, heten de operanden. In de expressie:

```
jaartal >= 2000 && jaartal <= 2010
```

geldt het volgende:

`jaartal >= 2000` en `jaartal <= 2010` zijn de operanden van `&&`
`jaartal` en `2000` zijn de operanden van `>=`
`jaartal` en `2010` zijn de operanden van `<=`

De *en*-operator `&&` in JavaScript kan niet alleen met operanden van het type `boolean` werken, maar ook met andere typen. Om het niet te ingewikkeld te maken, beperken we ons nu tot operanden van het type `boolean`, dus twee operanden die `false` of `true` zijn. Een expressie met de *en*-operator `&&` heeft als resultaat `true` als beide operanden `true` leveren. Als één van beide operanden `false` levert of beide leveren `false`, dan levert de *en*- operator `&&` ook `false`.

De werking van de *en*-operator met `boolean`-operanden is samengevat in de volgende tabel.

1ste operand	2de operand	resultaat van `&&`
`true`	`true`	`true`
`true`	`false`	`false`
`false`	`true`	`false`
`false`	`false`	`false`

Het resultaat van de *en*-operator is dus altijd `false`, behalve als beide operanden `true` zijn.

Opmerking: houd er rekening mee dat de `&&`-operator in JavaScript ook met andere waarden dan van een boolean-type kan werken. Zie voor een precieze beschrijving: Logical_Operators.

De of-operator ||
Behalve de *en*-operator `&&` kent Java nog twee logische operatoren:

- De *of*-operator `||`, twee verticale streepjes (een verticaal streepje staat meestal als twee onderbroken verticale streepjes op het toetsenbord).
- De *niet*-operator `!`, een uitroepteken.

Als de *of*-operator met twee operanden van het type `boolean` werkt, gehoorzaamt deze aan de regels in de volgende tabel.

1ste operand	2de operand	resultaat van `||`
`true`	`true`	`true`
`true`	`false`	`true`
`false`	`true`	`true`
`false`	`false`	`false`

(Bron: wikipedia.org.)

Het resultaat van de *of*-operator is met `boolean`-operanden dus altijd `true`, behalve als beide operanden `false` zijn. Voorbeelden:

Als `jaartal` de waarde `1995` heeft, dan levert

```
jaartal == 1995 || jaartal == 2008
```

de waarde `true`, omdat de eerste operand `jaartal==1995` waar is en dus de waarde `true` levert.

Als `jaartal` de waarde `2008` heeft, dan levert

```
jaartal == 1995 || jaartal == 2008
```

ook de waarde `true`, omdat de tweede operand waar is.

Maar als `jaartal` de waarde `1999` heeft, dan levert deze uitdrukking de waarde `false`, omdat beide operanden nu de waarde `false` hebben.

De niet-operator !
De *niet*-operator `!` is de eenvoudigste logische operator. Hij werkt met maar één operand. Bijvoorbeeld:

```
! ( jaartal == 1995 || jaartal == 2008 )
```

De *niet*-operator keert de waarheidswaarde om: `true` wordt `false` en `false` wordt `true`. In tabel hieronder ziet u de regels voor de *niet*-operator.

operand (achter `!`)	resultaat van `!`
`true`	`false`
`false`	`true`

Als `jaartal` de waarde `1850` heeft, dan levert de uitdrukking

```
! ( jaartal == 1995 || jaartal == 2008 )
```

de waarde `true`.