Een functie definiëren
In JavaScript kunt u de datum en tijd opvragen met behulp van `Date()`. Zie w3schools.com - dates en w3schools.com – date_methods.

Een voorbeeld:

```
<!DOCTYPE html>
<html>
<body>
<p id="resultaat"></p>
<script>
  var d = new Date();
  document.getElementById("resultaat").innerHTML = d;
</script>
</body>
</html>
```

Het resultaat ziet er ongeveer zo uit:

Sun Oct 12 2014 12:42:38 GMT+0200

Stel dat u een website maakt met verschillende pagina's waarop de gebruiker het huidige tijdstip moet kunnen opvragen, waarbij het juiste tijdstip verschijnt in een pop-up. Dat zou zo kunnen:

```
<script>
var d = new Date();
var uur = d.getHours();
var minuten = d.getMinutes();
alert( "Het is nu " + uur + ":" + minuten + " uur" );
</script>
```

Het pop-upvenster kan er zo uitzien (de vormgeving verschilt per browser):

Van deze vier regels code kunnen we een functie maken. Met bijvoorbeeld `tijdstip()` als naam. Dat gaat als volgt:

```
<script>
function tijdstip() {
  var d = new Date();
  var uur = d.getHours();
  var minuten = d.getMinutes();
  alert( "Het is nu " + uur + ":" + minuten + " uur" );
}
</script>
```

De definitie van een functie begint met het woord *function*, gevolgd door een (zelfverzonnen) naam. Namen van functies moeten voldoen aan dezelfde regels als die van variabelen. Het ronde openings- en sluithaakje achter de naam geeft ook aan dat het om een functie gaat. Verderop zullen we zien dat er soms ook iets tussen de haakjes kan staan. In dit geval staat er niets tussen de haakjes, maar ze zijn wel nodig.
Vervolgens komt er een gedeelte tussen accolades. Een gedeelte tussen accolades heet in JavaScript een *block*, een stuk code dat bij elkaar hoort, en in dit geval is het blok de *body* van de functie.

Als de browser een pagina leest waarop een functie is gedefinieerd, komt deze functie in het geheugen terecht, en **gebeurt er verder nog niets mee**. De code in de body van een functie wordt pas uitgevoerd als de naam van de functie genoemd wordt, met andere woorden: als de functie wordt aangeroepen.

Het aanroepen van de functie kan op dezelfde pagina in een ander script worden gedaan, bijvoorbeeld zo:

```
<script>
 tijdstip();
</script>
```

Het aanroepen van de functie kan ook in hetzelfde script als de functie:

```
<script>
// Definitie van de functie:
function tijdstip() {
  var d = new Date();
  var uur = d.getHours();
  var minuten = d.getMinutes();
  alert( "Het is nu " + uur + ":" + minuten + " uur" );
}

// Functieaanroep:
tijdstip();
</script>
```

Onthoud: een functie die niet wordt aangeroepen, wordt niet uitgevoerd.

Een functie met een parameter
In het volgende voorbeeld staat een functie die van de prijs van een artikel de btw en de totaalprijs, en deze toont in een pop-up:

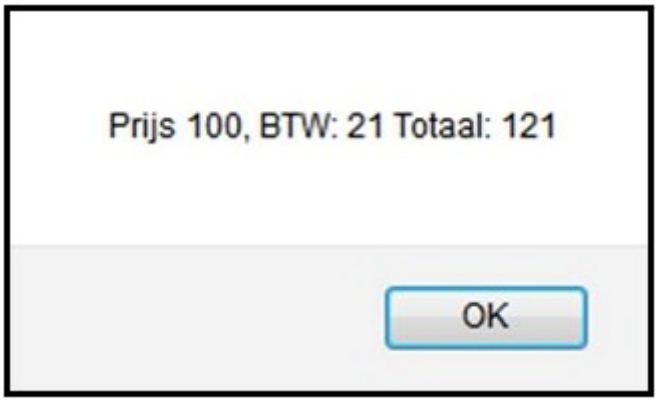
```
<script>
// Definitie van de functie:
function bereken_btw() {
  var prijs = 100;
```

```
  var btw = 0.21 * prijs;
  var totaalprijs = prijs + btw;

  alert( "Prijs " + prijs +
         ", BTW: " + btw +
         " Totaal: " + totaalprijs );
}

// Functieaanroep:
bereken_btw();
</script>
```

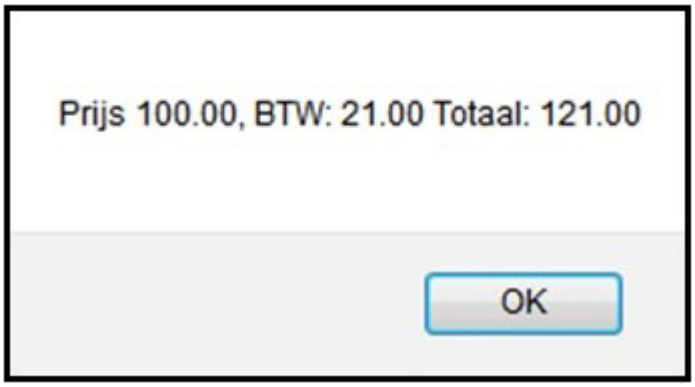
De uitvoer ziet er zo uit:



Dit is niet erg fraai, de nullen achter de bedragen zijn weggelaten. Met een van de ingebouwde functies (methoden) van JavaScript, `toFixed()`, kunnen we dit oplossen. De code wordt dan:

```
  alert( "Prijs " + prijs.toFixed(2) +
         ", BTW: " + btw.toFixed(2) +
         " Totaal: " + totaalprijs.toFixed(2) );
```

Het resultaat ziet er beter uit:



Een nadeel van deze functie is natuurlijk dat hij voor maar één waarde (100) deze berekening kan doen. Het zou mooi zijn als de functie voor elk willekeurig bedrag dat de gebruiker intikt de btw kan berekenen. Met een eenvoudige ingreep is dat mogelijk. Via een zogeheten *parameter* kan de 'buitenwereld' gegevens doorgeven aan een functie. De functie komt er, met de parameter `prijs`, zo uit te zien:

```
function bereken_btw( prijs ) {
  var btw = 0.21 * prijs;
  var totaalprijs = prijs + btw;
  alert( "Prijs " + prijs.toFixed(2) +
         ", BTW: " + btw.toFixed(2) +
         " Totaal: " + totaalprijs.toFixed(2) );
}
```

In de aanroep van de functie moet u nu het bedrag aangeven waarvan de btw berekend moet worden, bijvoorbeeld:

```
// Functieaanroep:
bereken_btw( 150 );
```

De waarde 150 komt in de parameter `prijs` terecht. Deze waarde heet ook wel het *argument* waarmee de functie wordt aangeroepen.

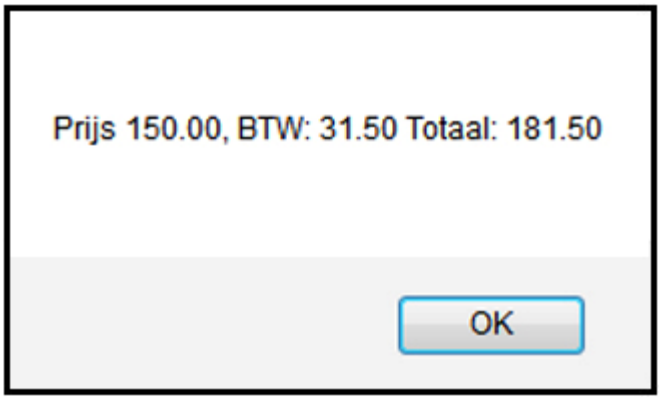
De volledige code van een werkende pagina staat hieronder.

```
<!DOCTYPE html>
<html>
<body>
Voer een prijs in:
<input type="text" id="inputprijs" onchange="myFunction()">

<script>
// Event-handler
function myFunction() {
  var invoervak = document.getElementById("inputprijs");
  var p= +invoervak.value;
  bereken_btw( p );       // aanroep van de functie
}
// Functie:
function bereken_btw( prijs ) {
  var btw = 0.21 * prijs;
  var totaalprijs = prijs + btw;

  alert( "Prijs " + prijs.toFixed(2) +
         ", BTW: " + btw.toFixed(2) +
         " Totaal: " + totaalprijs.toFixed(2) );
}
</script>
</body>
</html>
```

Mogelijke uitvoer:



Merk op dat het script nu twee functies bevat: de event-handler `myFunction()` en de functie `bereken_btw()`. De event-handler wordt door de browser aangeroepen op het moment dat de gebruiker klaar is met de invoer en op Enter drukt. De event-handler haalt de ingevoerde prijs uit het invoervak, bergt deze waarde op in de variabele met de naam `p`, en roept `bereken_btw()` aan met deze variabele als argument: `bereken_btw(p)`. Hierdoor komt de waarde van `p` in de parameter `prijs` terecht.

Een functie kan meerdere parameters hebben, zie voor een voorbeeld de volgende paragraaf.