

Zuerst wird die Funktion definiert, die über das Schlüsselwort `return` einen Wert zurückliefert. Zwischen Anweisung2 und Anweisung3 wird die Funktion aufgerufen, und es werden die Werte `Parameter1` und `Parameter2` übergeben. Der Interpreter verzweigt somit in die vorher definierte Funktion, führt die verschiedenen JavaScript-Anweisungen aus und liefert einen Wert zurück. Dieser Rückgabewert wird im oberen Beispiel in der Variable `x` für weitere Bearbeitungsschritte gespeichert.

## Funktionsdefinition von Bedingungen abhängig machen

Eine Funktionsdefinition kann auch von einer Bedingung abhängig gemacht werden. Auf diese Weise können Sie z. B. Funktionen mit gleichen Namen, aber unterschiedlichen Funktionalitäten definieren.

### Beispiel: *if\_funktion.htm*

Die Wirkungsweise der Funktion `operation()` wird vom Inhalt der Variable `name` abhängig gemacht. Auf diese Weise können Sie die Funktionalität eines Programms auf andere Weise vom Wert einer Variable abhängig machen.

```
name = "Addition";
if(name == "Addition") {
    function operation(zahl1, zahl2) { return zahl1 + zahl2; }
}
else {
    function operation(zahl1, zahl2) { return zahl1 - zahl2; }
}
document.write(operation(10, 11));
```

Die Möglichkeit, die Funktionsdefinition von Bedingungen abhängig zu machen, ist erst ab JavaScript 1.5 verfügbar. Zurzeit interpretiert nur der Netscape Navigator diese Anweisungen korrekt.

## Funktionen in Ausdrücken definieren

Durch die Definition von Funktionen in Ausdrücken haben Sie eine weitere Möglichkeit, die Programmausführung dynamisch zu ändern. Statt eines Funktionsnamens wird die gesamte Funktionsdefinition im Ausdruck angegeben. Der einzige Unterschied zur üblichen Funktionsdefinition liegt im Weglassen des Funktionsnamens. Deshalb werden diese Funktionen auch als anonyme Funktionen bezeichnet.

### Beispiel: *ausdr\_funktion.htm*

In der ersten Anweisung wird die Funktionsdefinition einer Variable zugewiesen. Der Variablenname steht somit für den Funktionsnamen und kann wiederum in Ausdrücken verwendet werden. Im zweiten Anwendungsfall wird der Funktion `TueEtwas()` als Parameter eine Funktion übergeben. Der übergebenen Funktion wird in der `return`-Anweisung der Wert 10 als Parameter übergeben. Der Funktionsaufruf der Funktion `TueEtwas()` beinhaltet im ersten Fall die vollständige anonyme Funktionsdefinition. Im zweiten Fall wird die Variable `quadrat` übergeben, die wiederum auf eine Funktionsdefinition verweist.

```
var quadrat = function (zahl) { return zahl * zahl; }
a = quadrat(10);

function TueEtwas(f) {
    return f(10);
}
document.write(TueEtwas(function(value) { return value * value; })); // oder
document.write(TueEtwas(quadrat));
```

## Unbestimmte Anzahl von Parametern

Über das Feld `arguments` können Sie ebenfalls auf alle Parameter einer Funktion zugreifen. Der Zugriff erfolgt dabei allerdings über den Index des Parameters und nicht über den Parameternamen. Die Anzahl der übergebenen Parameter erhalten Sie über `arguments.length`. Auf diese Weise können Sie einer Funktion eine variable Parameterliste übergeben.