Functions

Simple Calculator

Addition und Subtraktion

Erstelle ein neues WebStorm-Projekt namens `Calculator` und lege darin die JavaScript-Datei `simpleApp.js` an.

Erstelle in der `simpleApp.js` eine Funktion `printCalculation`. Dieser werden zwei Parameter übergeben: Ein Rechenzeichen (`operator`) und eine entsprechende Funktion zur Berechnung des Ergebnisses (`calculate`). `printCalculation` gibt die vollständige Rechnung mit den Zahlen *20* und *3* aus.

```
function printCalculation(operator, calculate){
    console.log("20 " + operator + " 3 = " + calculate(20, 3));
}
```

Erstelle als nächstes zwei Funktionen zur Berechnung der Summe bzw. der Differenz zweier Zahlen.

```
function sum(num1, num2){
    return num1 + num2;
}

function difference(num1, num2){
    return num1 - num2;
}
```

Gib am Ende der `simpleApp.js` die Summe und die Differenz von *20* und *3* folgendermaßen aus:

```
printCalculation("+", sum);
printCalculation("-", difference);
```

Teste deine Applikation indem du die `simpleApp.js` ausführst - die Konsolenausgabe in WebStorm sollte wie folgt aussehen:

```
20 + 3 = 23
20 - 3 = 17
```

Multiplikation und Division

Erstelle jetzt (ohne Musterlösung) die Funktionen `product` und `quotient`, um auch eine Multiplikation und eine Division durchführen zu können. Vergiss nicht die entsprechenden Aufrufe von `printCalculation` am Ende der `simpleApp.js` hinzuzufügen. In der Konsole sollte dann folgendes ausgegeben werden:

```
20 + 3 = 23
20 - 3 = 17
20 * 3 = 60
20 / 3 = 6.666666666666667
```

Advanced Calculator

Function Expressions

Erstelle im `Calculator` Projekt eine neue JavaScript-Datei `advancedApp.js`. Füge darin den gesamten Inhalt der `simpleApp.js` mithilfe von *Copy & Paste* ein. Ändere anschließend in der `advancedApp.js` **alle** Funktionsdeklarationen, sodass sie Variablen zugewiesen werden. Die Funktion `printCalculation` sollte dann beispielsweise so aussehen:

```
let printCalculation = function(operator, calculate){
    console.log("20 " + operator + " 3 = " + calculate(20, 3));
}
```

Führe, nachdem du **alle** Funktionen angepasst hast, die `advancedApp.js` erneut aus. Die Konsolenausgabe sollte sich nicht verändert haben, da die Schreibweise in der `simpleApp.js` (sogenannte *function declarations*) und jene in der `advancedApp.js` (sogenannte *function expressions*) gleichbedeutend sind.

Anonymous Functions

Als Nächstes soll das Ergebnis von 20^3 (in JavaScript `20 ** 3`) ausgeben werden. Hierfür deklarieren wir keine neue Berechnungsmethode, sondern verwenden beim Aufruf von `printCalculation` eine anonmye Funktion. Diese wird also im Gegensatz zu den bisherigen Funktionen keiner Variable zugewiesen (und ist daher auch nur einmalig verwendbar).

```
printCalculation("^", function(num1, num2){
    return num1 ** num2;
});
```

Rufe `printCalculation` nochmal mit einer anonmyen Funktion auf, um auch das Ergebnis der Modulo-Operation `20 % 3` darzustellen. Zur Erinnerung: Modulo ermittelt den Divisionsrest (in diesem Fall also *2*). Die Konsoleausgabe sollte somit folgendermaßen aussehen:

```
20 + 3 = 23
20 - 3 = 17
20 * 3 = 60
20 / 3 = 6.666666666666667
20 ^ 3 = 8000
20 % 3 = 2
```

Arrow Function Expressions

Mithilfe einer *arrow function expression* kann zum Beispiel die anonyme Funktion zur Berechnung der Potenz noch kompakter programmiert werden. Füge hierfür folgenden Funktionsaufruf am Ende der `advancedApp.js` hinzu:

```
printCalculation(("^"), (num1, num2) => num1 ** num2);
```

Die darin vorkommende *arrow function expression* `(num1, num2) => num1 ** num2` ist gleichbedeutend mit der (zuvor verwendeten) anonymen Funktion:

```
function(num1, num2){
    return num1 ** num2;
}
```

Gib abschließend nochmal die Modulo-Operation aus. Verwende dabei ebenfalls eine (anonyme) *arrow function expression*. Unten ist zur Kontrolle dargestellt, wie die endgültige Konsolenaugabe aussehen sollte.

```
20 + 3 = 23
20 - 3 = 17
20 * 3 = 60
20 / 3 = 6.666666666666667
20 ^ 3 = 8000
20 % 3 = 2
20 ^ 3 = 8000
20 % 3 = 2
```