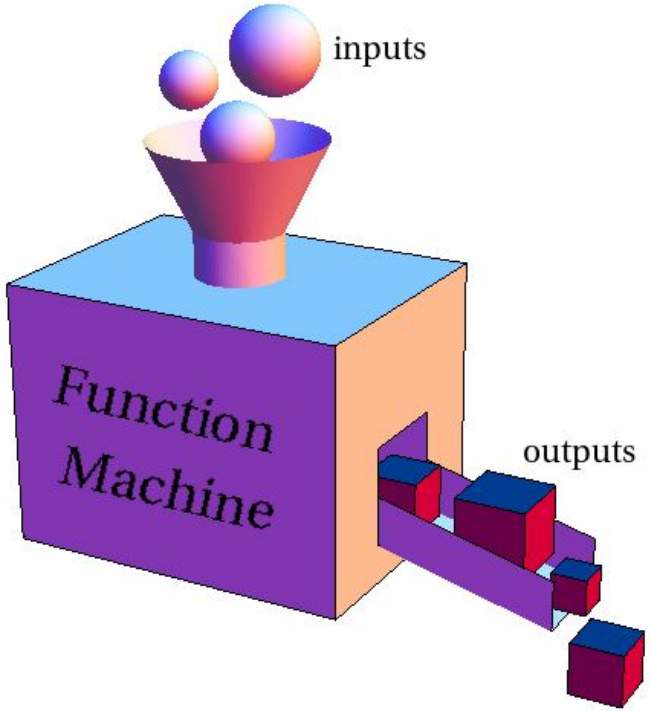
JavaScript 1

Funktioner del 1

David Andersson
ZoCom 2018

Funktioner

Vad är en funktion?



Funktioner

En funktion är något som har noll eller flera parametrar och returnerar (skickar tillbaka) ett värde.

let returvärde = `funktion( argument1, argument2, .. );`

Funktion

- Måste *definieras* innan den kan användas
- Namnges på samma sätt som variabler
- *Anropas* i koden, med värden (*argument*) för varje parameter
- Har noll eller flera parametrar
- Utför instruktionerna i funktionsblocket { }
- Returnerar ett värde
- Funktioner är värden av typen *object*
- Används för att undvika upprepningar, strukturera kod, lösa svårare problem
- Man kan ha funktioner i funktioner

Exempel

```
function printString(str) {
    console.log(str);
}

function returnToMe() {
    return 2;
}

let oneHigher = function( x ) {
    return x + 1;
}
```

```
function average3(x, y, z) {
    let sum = x + y + z;
    return sum / 3;
}

printString('hello');

let value1 = returnToMe();
let value2 = oneHigher(2);

let avg = average3(1, 2, 3);
printString('Average: ' + avg);
```

Fördefinierade funktioner

JavaScript har flera fördefinierade funktioner. Exempel:

```
console.log(string)

prompt(string, default)

alert(string)

confirm(string)
```

```
isNaN(value)

Math.random()

Math.min(x, y), Math.max(x, y)

string.charAt(number)

document.write(string)
```

Vilka av funktionerna returnerar ett värde som inte är *undefined*?

Definiera funktioner

```
let variabel = function( parametrar ) {
    // funktionens innehåll
    return värde;  // används om funktionen ska returnera något

}
```

scope

Eller:

```
function funktionsnamn( parametrar ) {
    // funktionens innehåll
    return värde;  // return är valfritt
}
```

Parametrar

JavaScript bryr sig inte om hur många parametrar man skickar till en funktion. Extra parametrar ignoreras och parametrar man glömmer får värdet *undefined*.

```
function param1() {
    console.log("function param1: too many are not a problem");
}
function param2(a, b, c, d=5) {
    console.log('param2: ' + a + ', ' + b + ', ' + c + ', ' + d);
}
param1('I am irrelevant', 'The function ignores me');
param2(1, 2);
```

Default-parametrar

Man kan ge ett defaultvärde som parametrarna till en funktion får om man inte skickar med dem.

```
function param3(a='Remember me') {
    console.log(a);
}
param3('standard case');
param3();  // ok att glömma, parametern har default
```

Funktioner och variabler

Vad kommer koden nedan att skriva ut? Varför?

```
console.log("before function definition");
var mySuperAwesomeFunction = function() {
    console.log("inside function");
};
console.log("after function definition");
mySuperAwesomeFunction();
console.log("after calling function");
```

Scope

Vad kommer koden nedan att skriva ut? Varför?

```
var x = "outside";

var f1 = function() {
    var x;
    x = "inside f1";
};
f1();

console.log(x);
```

Scope

Svar: koden skriver ut "outside".

Variabeln **x** skapas som en **global variabel**, eftersom den inte är inuti någon funktion. Det kallas att x ligger i *global scope*.

Variabeln **x** inuti funktionen är en ny, **lokal variabel** som råkar ha samma namn som den globala. Den existerar bara inuti funktionen. Det kallas för att x ligger i *local scope*.

Varning! Många programmeringsspråk har *block scope*, men JavaScript har *function scope* - om man inte använder *let*.

Exempel scope

```
var x = 5;
var y = 10;

function magic() {
    var x = 10;  // lokala variabeln x skymmer globala variabeln x
    y = 20;     //  skriver över den globala variabeln
}

magic();

console.log( 'x == ' + x + ', y == ' + y );
```

Vad skrivs ut på konsolen?

Övningar

A. Läs på w3Schools på kapitlena Function och Random
B. Läs och gör “Guess the number game”

Fler övningar:

- Gör övningarna från denna presentation i konsolen
- Det som ni inte hunnit med tidigare
- Bygg vidare på miniräknaren från förra veckan
  *Gör knappar 0-9 och knappar + - * / och en =*