INSTITUT
TEKNOLOGI
HARAPAN
BANGSA
School of Telematics

Desain Web IF-1P03

JavaScript : Function & Loop

Ken Ratri. MT

Function

- A function is a block of code that will be executed when "someone" calls it
- A function is written as a code block (inside curly { } braces), preceded by the **function** keyword:

```
function functionname()
{
some code to be executed
}
```

Calling a Function with Arguments

- When you call a function, you can pass along some values to it, these values are called *arguments* or *parameters*.
- These arguments can be used inside the function.
- You can send as many arguments as you like, separated by commas (,)

```
myFunction(argument1,argument2)
```

Declare the argument, as variables, when you declare the function:

```
function myFunction(var1,var2)
{
some code
}
```

Functions With a Return Value

- Sometimes you want your function to return a value back to where the call was made.
- This is possible by using the *return* statement.
- When using the *return* statement, the function will stop executing, and return the specified value.
- **Syntax**

```
function myFunction()
{
var x=5;
return x;
}
```

The function above will return the value 5.

Functions With a Return Value

- You can also use the returnvalue without storing it as a variable:

```
document.getElementById("demo").innerHTML=myFunction();
```

Example
Calculate the product of two numbers, and return the result:

```
function myFunction(a,b)
{
return a*b;
}

document.getElementById("demo").innerHTML=myFunction(4,3);
```

The + Operator Used on Strings

- The + operator can also be used to add string variables or text values together.
- Ex.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

What a verynice day

Adding Strings and Numbers

- Adding two numbers, will return the sum, but adding a number and a string will return a string:

```
x=5+5;
y="5"+5;
z="Hello"+5;
```

```
10
55
Hello5
```

Logical Operators

Logical operators are used to determine the logic between variables or values. Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

- JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.
- **Syntax**

```
variablename=(condition)?value1:value2
```

Ex

```
voteable=(age<18)?"Too young":"Old enough";
```

If...Else Statements

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
- In JavaScript we have the following conditional statements:
- **if statement -** use this statement to execute some code only if a specified condition is true
- **if...else statement -** use this statement to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement -** use this statement to select one of many blocks of code to be executed
- **switch statement -** use this statement to select one of many blocks of code to be executed

If Statement

```
if (condition)
  {
  code to be executed if condition is true
  }
```

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

If...else if...else Statement

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

Loop

- Loops are handy, if you want to run the same code over and over again, each time with a different value.

```
document.write(cars[0] + "<br>");
document.write(cars[1] + "<br>");
document.write(cars[2] + "<br>");
document.write(cars[3] + "<br>");
document.write(cars[4] + "<br>");
document.write(cars[5] + "<br>");
```

```
for (var i=0;i<cars.length;i++){
        document.write(cars[i] + "<br>");
{
```

Loop

- **Different Kinds of Loops**
- JavaScript supports different kinds of loops:
  - **for** - loops through a block of code a number of times
  - **for/in** - loops through the properties of an object
  - **while** - loops through a block of code while a specified condition is true
  - **do/while** - also loops through a block of code while a specified condition is true

For...In Statement

- The for loop is often the tool you will use when you want to create a loop.
- The for loop has the following syntax:

```
for (statement 1; statement 2; statement 3)
  {
  the code block to be executed
  }
```

Statement 1 is executed before the loop (the code block) starts.
Statement 2 defines the condition for running the loop (the code block).
Statement 3 is executed each time after the loop (the code block) has been executed.

```
for (var i=0; i<5; i++){
        x=x + "The number is " + i + "<br>";
  }
```

```
for (var i=0,len=cars.length; i<len; i++){
        document.write(cars[i] + "<br>");
{
```

```
var i=2,len=cars.length;
for (; i<len; i++){
        document.write(cars[i] + "<br>");
{
```

The For/In Loop

- The JavaScript for/in statement loops through the properties of an object:

```
var person={fname:"John",lname:"Doe",age:25};

for (x in person){
  txt=txt + person[x];
}
```

While Loop

- The while loop loops through a block of code as long as a specified condition is true.

```
while (condition){
  code block to be executed
  }
```

While Loop

```
while (i<5) {
  x=x + "The number is " + i + "<br>";
  i++;
  }
```

The Do/While Loop

- The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do
  {
        code block to be executed
  }
while (condition);
```

The Do/While Loop

```
do
  {
            x=x + "The number is " + i + "<br>";
  i++;
  }
while (i<5);
```

Break and Continue Statements

- the break statement "jumps out" of a loop.
- The continue statement "jumps over" one iteration in the loop.
- It was used to "jump out" of a switch() statement.

```
for (i=0;i<10;i++)
  {
  if (i==3)
    {
    break;
    }
  x=x + "The number is " + i + "<br>";
  }
```

Break and Continue Statements

- The **continue statement** breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.
- This example skips the value of 3:

```
for (i=0;i<=10;i++)
 {
            if (i==3) continue;
            x=x + "The number is " + i + "<br>";
  }
```