VARIABLES : HTML INTERFACE OF JAVASCRIPT

Defining variables

Variables are used in Javascript to store values in your scripts. Choosing a suitable name for a variable can help in understanding what a script does. Although not strictly necessary, it is good practice to declare variables before using them. You do this using the **var** statement.

Naming Variables

Javascript is **case-sensitive** so naming a variable **myVar** is different to the variable **myvar**.

Variable names can be of any length, but must follow certain rules:

- The first character must be a letter (either uppercase or lowercase), or an underscore (_), or a dollar sign ($).
- Subsequent characters can be letters, numbers, underscores, or dollar signs.
- The variable name cannot be a **reserved word**. I.e. a word such as function or return which form part of the Javascript language

Although variables can be defined at any point in your script, it is generally good practice to pre-define all variables using the**var** keyword at the beginning of a function; for ease of readability. No matter where the variable is defined, the JavaScript engine will create the variable at the start of the current **scope**. Variables defined inside a function with the var keyword are not accessible outside of that function. Any variables used in a function which are not explicitly defined with var will belong to the scope in which the function is defined. Usually the Global Object.
It is possible to define the variable, leaving its value undefined, or assigning a value immediately, or even to define multiple variables in a single command:

```
var area;                    // declaration
var width = 3, height = 4;   // declaration and initialisation.
```

Few restrictions are placed on variable names but one important rule is They must not conflict with Javascript keywords; which have a special meaning within the language. Try and choose variable names that are meaningful

within the context of the problem you are trying to solve. Variables which have been defined but not explicitly initialized are given the value**undefined**.

Once a variable is defined, **do not use the variable name again, as this can be a major source of confusion,** even if they are in different scopes.

Note that variables can be defined at any point in your script, including inside a control structure that causes that statement never to be executed. However, It will hold the value undefined until a statement is executed that gives it a value.

Javascript may forgive if you omit the **var** declaration when you assign a value to an undeclared variable, but on the other hand if you attempt to read the value of an undeclared variable, it will throw an error. Since Javascript is case sensitive the following code has created two variables not one called area (Area).

```
    var area;                    // declaration
    var width = 3, height = 4;  // declaration and (initial)
  assignment
    Area = width * height;
    alert ("area is " + area + " but Area = " + Area);
```

Variables are not given a **data type** in Javascript. A var may be a reference to an object or function, or it can be any one of the data types, **string, number, boolean, null or undefined**.

Defining functions (or methods)

Javascript functions combine several program statements together into a **block** which usually perform some recognisable task within the context of the whole program. It may be that the same task may need to be performed in several different places within the program. once written, a (global) function can then be called from anywhere within the programw it is needed. The body of a function is delimited by curly braces { }. The syntax for a function is

```
function name(arg1, arg2 ... )
{
  program statements;
}
```

Every time a function is **called**, a new set of variables are defined. This enables functions to call themselves. Such functions are known as **recursive functions**. The following function can be used to calculate the area of a rectangle.

```
function area(width,height)
{
  msg = "area of a " + width + " x " + height + " rectangle is ";
  msg += width * height;
  alert(msg);
}
```

Defining objects

If you want to be able to create variables that have child properties and methods, you must define the variable as an **object**. To define an object, use either the **new** keyword, or define the object using the object lieteral notation, i.e. { } a pair of curly braces.

```
 var myVariable = new Object() or simply
 var myVariable = {};     //empty object
```

Using **prototyping** you can then assign properties or methods to that object dynamically; simply by specifying them.

Objects may be characterised by *name:value* pairs, which are separated by commas. For example,

```
var rectangle = new Object();
rectangle = {height:undefined,width:undefined,function area()};

function test()
{
  var square = new Object(); //declares a square object
  square.size; //add a property to specify its size
  //define a function to calculate its area
  square.area = function area() {return square.size * square.size;}
  //for example
  square.size= 3;
  alert("a square, size " + square.size + " has area "  +
square.area());

  var rectangle = {width:0,height:0};
  rectangle.area = function area() {return this.width *
this.height;}
```

```
  rectangle.width = 3; rectangle.height = 6;
  alert("Area of rectangle = " + rectangle.area());

}
```

Source : http://www.soslug.org/node/1713