JavaScript Syntax

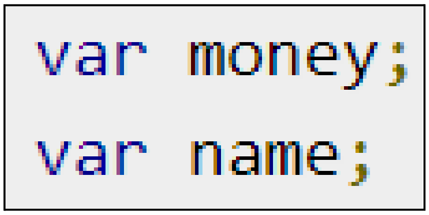
Topics Covered:

- JavaScript variables
- Types of variables
- Literals
- Features of Literals
- Types of Operators
- Comment statement
- Rules of writing the identifiers
- JavaScript data types

Topics in Detail:

JavaScript Variables

- **Variables** are the name of the data storing containers.
- The variable values may vary.
- JavaScript variables must have **unique names**. Hence, it is also called **identifiers**.
- Variables can be initialized at any point in the code.
- **var** keyword should only be used at the time of declaration or initialization.
- The keywords **var**, **let** and **const** are used to declare the variables.
- The same variable should not be redeclared twice.
- JavaScript is an untyped language, i.e a variable can hold any data type values.

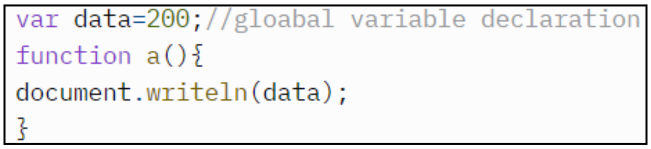


```
var money;
var name;
```

Types of Variables

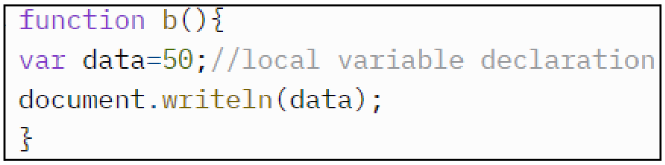
JavaScript variables are of two types based on the scope of the variable where it is defined.

- Global variables - It can be defined anywhere in the JS code.



```
var data=200;//gloabal variable declaration
function a(){
document.writeln(data);
}
```

- Local variables - It will be available only for a particular function where it is defined.

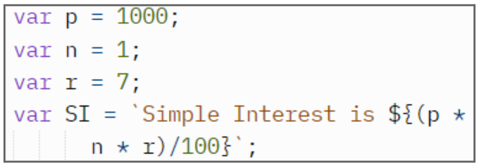


```
function b(){
var data=50;//local variable declaration
document.writeln(data);
}
```

Literals

- Literals are fixed values.
- Before ES6, the strings were created using single quotes (‘) or double quotes (“) and had very limited functionality.
- In ES6, the strings are created using backtick(`) and gives more control over dynamic strings.
- Syntax:

  var a = `some string`;



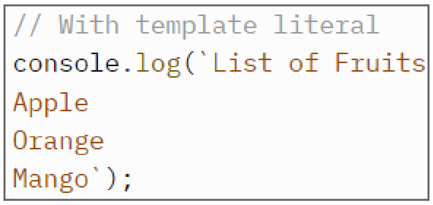
```
var p = 1000;
var n = 1;
var r = 7;
var SI = `Simple Interest is ${(p *
      n * r)/100}`;
```

Features of Literals

- **Multiline string** is the ability to span **multiple lines**.

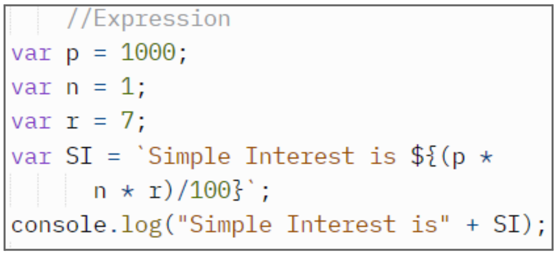
  Before Template Literals, an escape sequence \n was used to give new line character

  In Template Literals, no need to add \n.



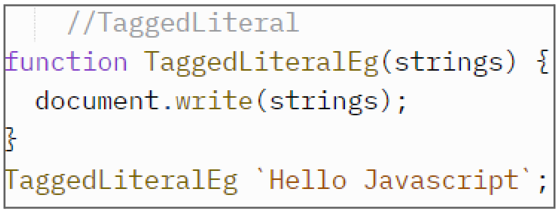
```
// With template literal
console.log(`List of Fruits
Apple
Orange
Mango`);
```

- **String formatting** is the ability to substitute a part of a string for **expression** or **variable values**. **${}** syntax expression produces the value. This value can be any computation operations or even a string.



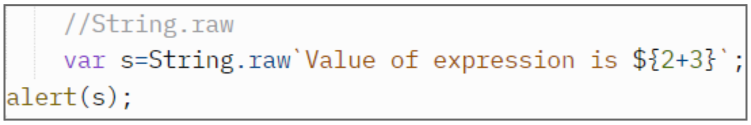
```
    //Expression
var p = 1000;
var n = 1;
var r = 7;
var SI = `Simple Interest is ${(p *
      n * r)/100}`;
console.log("Simple Interest is" + SI);
```

- **Tagged Template** is like **function** definition, but the only difference is at the time of call there will be **no parenthesis ()** and simply **an array of string** will be passed as **parameter**.



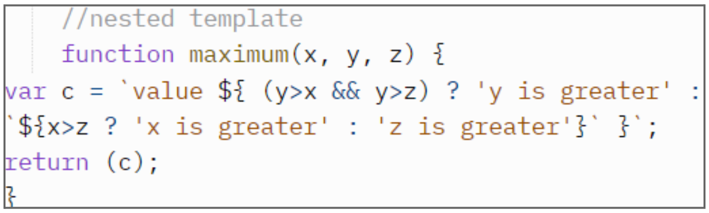
```
    //TaggedLiteral
function TaggedLiteralEg(strings) {
  document.write(strings);
}
TaggedLiteralEg `Hello Javascript`;
```

- **String.raw()** creates a raw string just like the default function and does string concatenation.



```
    //String.raw
    var s=String.raw`Value of expression is ${2+3}`;
alert(s);
```

- **Nested Template** allows checking multiple conditions



```
    //nested template
    function maximum(x, y, z) {
var c = `value ${ (y>x && y>z) ? 'y is greater' :
`${x>z ? 'x is greater' : 'z is greater'}` }`;
return (c);
}
```

JavaScript Operator

- An Operator is a symbol reserved for performing a special task.
- Operators perform operations on one or more operands. Operands can be variables, string or numeric literals.

Types of Operators

- Arithmetic Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Ternary Operators
- TypeOf Operator

Comment Statement

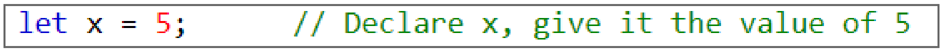
- JavaScript comment is used to **add information** about the code, warnings or suggestions etc. and helps in the easy **interpretation** of the code.
- These comments will be **ignored** by the JavaScript engine.

Advantages of adding comment statements

- It helps in easy **understanding** of the code.
- It is also used to **disable** a part of code from being executed.

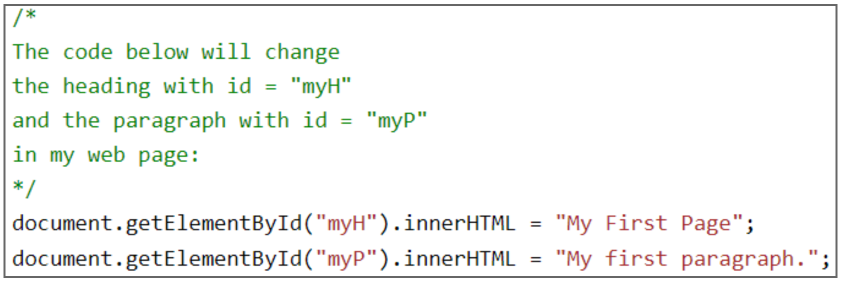
Types of JavaScript Comments

- **Single Line Comment - Double forward slashes (//)** is used before or after the statement



```
let x = 5;        // Declare x, give it the value of 5
```

- **Multi Line Comment -** It can be used as a single or multi line comment. It is represented as **/* comment */.**



```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

JavaScript Identifiers

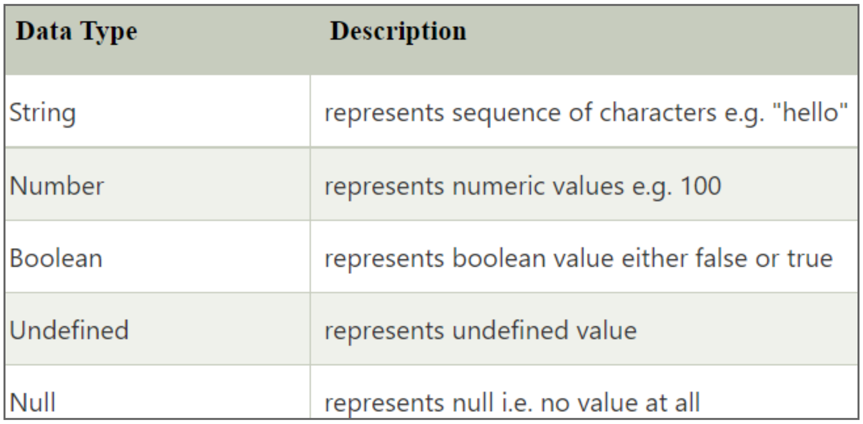
- The names given to variables, functions, parameters, classes, etc. are called Identifiers.
- Rules for declaring a JavaScript variable
  - Names can begin with a **letter** (a to z or A to Z).
  - Names can also begin with (_) or ($).
  - Variable names should not start with digits(0-9).
  - Variable names are case-sensitive (a and A are different variables).
  - Variable names can have letters, digits(0-9), underscore(_) or dollar($).
  - JavaScript keywords cannot be used as variable names.

JavaScript Data Types

- JavaScript is a **dynamic type language** because the JS engine chooses the data type itself dynamically.
- The variables can hold **different data types**.

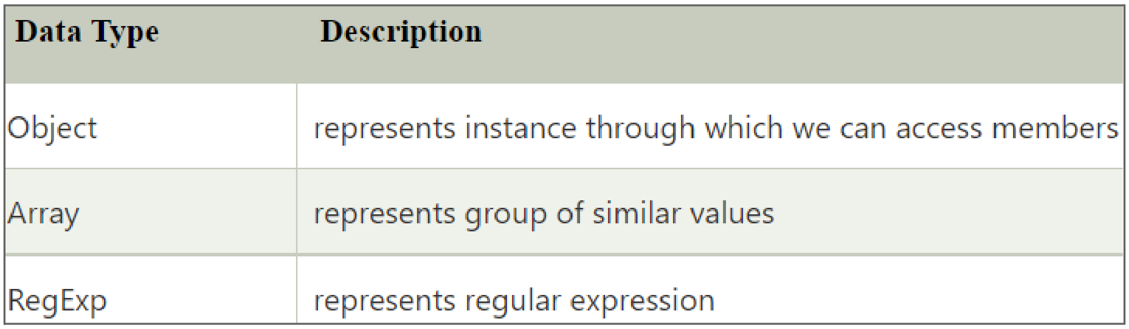
Types of Data Types

- Primitive Data Type



Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

- Non - Primitive Data Type



Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression