JavaScript (1)

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

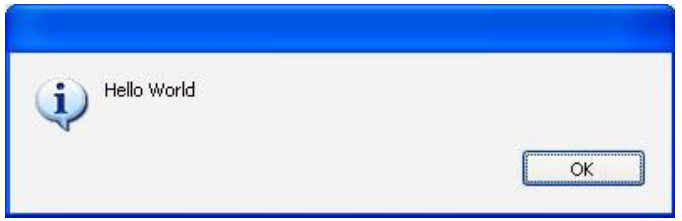
Authorized & published by Summitworks Technologies Inc

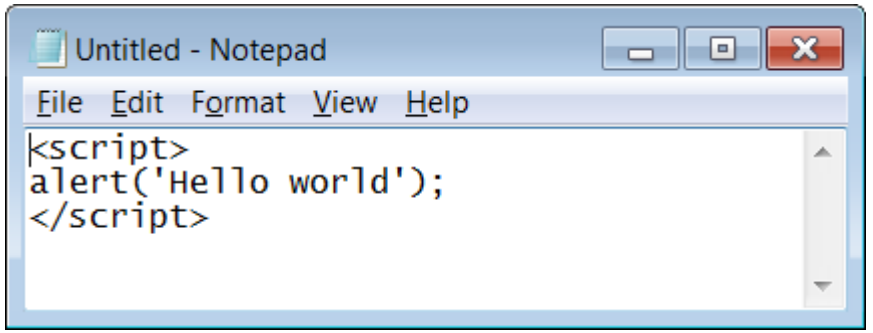
Agenda

- Introduction
- Data Types
- Variables
- Console.log
- Alerts, Prompts, Confirms
- Document.write
- Operators
- Conditional statements
- Functions
- Parameters
- Function Arguments
- Variable Scope
- Comments

Introduction

- JavaScript is the third of the three fundamental programming languages of the modern web (along with HTML, CSS).
- JavaScript allows developers to create dynamic web applications capable of taking in user inputs, changing what's displayed to users, animating elements, and much more.
- All browser has JavaScript engines (Firefox engine is Spider monkey and Chrome is v8) and previously JavaScript runs only in browser.
- Later in 2009, Ryan Dahl developed Node JS using C++ that includes Google's JavaScript V8 engine. Now we can run JavaScript outside browser.
- As every browser has JavaScript engine, we can easily run JavaScript code in browser without any additional tool.





JavaScript placement on your webpage

```
<!doctype html>
<html>
  <head>
    <title>A Web Page Title</title>
    <script type="text/javascript">
    // JavaScript Goes Here
    </script>
  </head>
  <body>
    <script type="text/javascript">
    </script>
  </body>
</html>
```

Data types

A datatype is a specialized form of information (number, string. etc.)

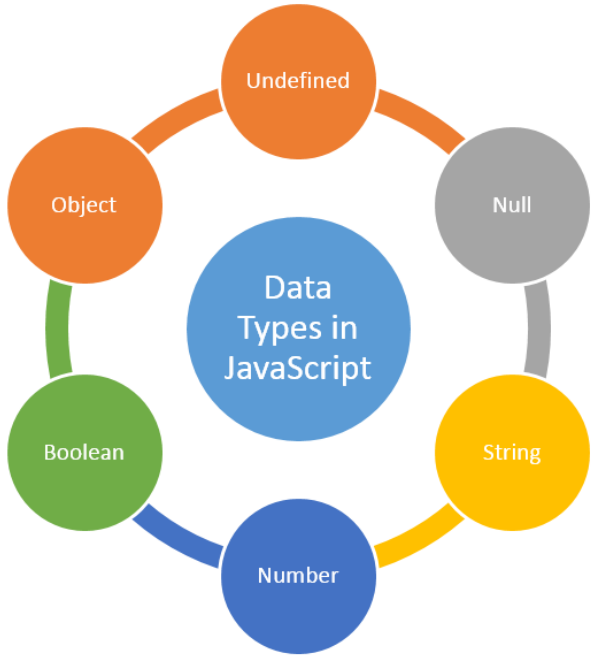
JavaScript supports the following data types:

- ***String:*** represents a list of characters. (ex: “abc”, ‘Ind*3j2n’)
- ***Number:*** represents an integer or decimal value. (ex: 100, 12.0383)
- ***Boolean:*** represents either true or false. (ex: true or false)
- ***Object:*** are special data types which has properties that are identified as key/value pairs.

```
ex: Cat = {
          name: ‘Mittens’,
          age: 2
        }
```

- ***Undefined:*** represents a value not yet defined
- ***Null:*** represents an empty value

Lets look into object in detail in coming slides.



Data types cont.…

Few built-in object based data types are:

- ***Date:*** represents date/time
- ***Date.now():*** represents current time

```
//Current Date and Time
var current = new Date();
console.log(curent);
```

- ***Array:*** represents a collection of data

```
//Creates an empty Array
var myArray = new Array();
//Creates an array of size 3
var myArray = new Array(3);
//Creates Array whith: apples, bananas and oranges
var myArray = ["apples","bananas","oranges",];
//Creates Array whith: blue, green, yellow and red
myArray = ['blue','green','yellow','red'];
//Creates Array with element 3;
var myArray = [3];
console.log(myArray);
```

Variables

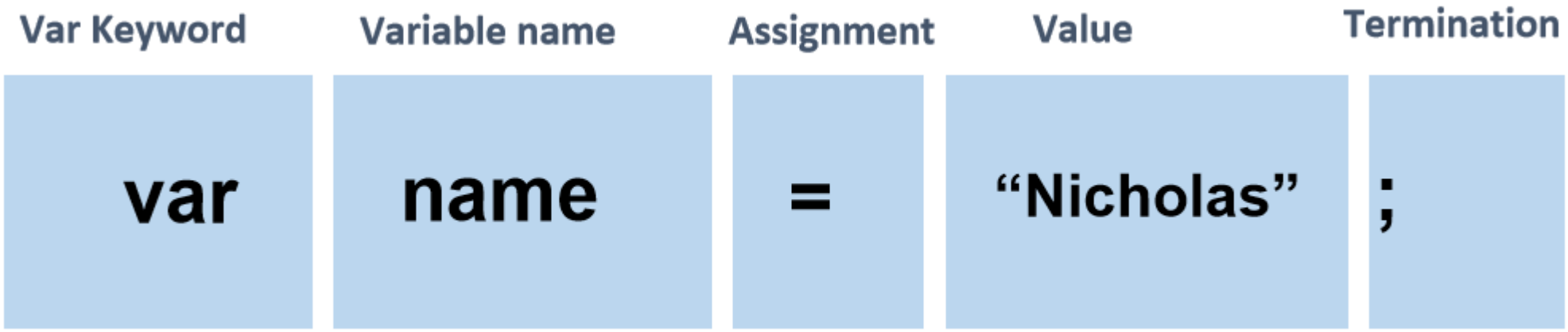
JavaScript *variables* are containers for storing data values.

```
var x = 5;
var y = 6;
var z = x + y;
```

In this example, x, y and z are variables

JavaScript variables can hold numbers like 100 and text values like *"John Doe".*

Variables are case-sensitive: *“name”* is not equal to *“Name”.*

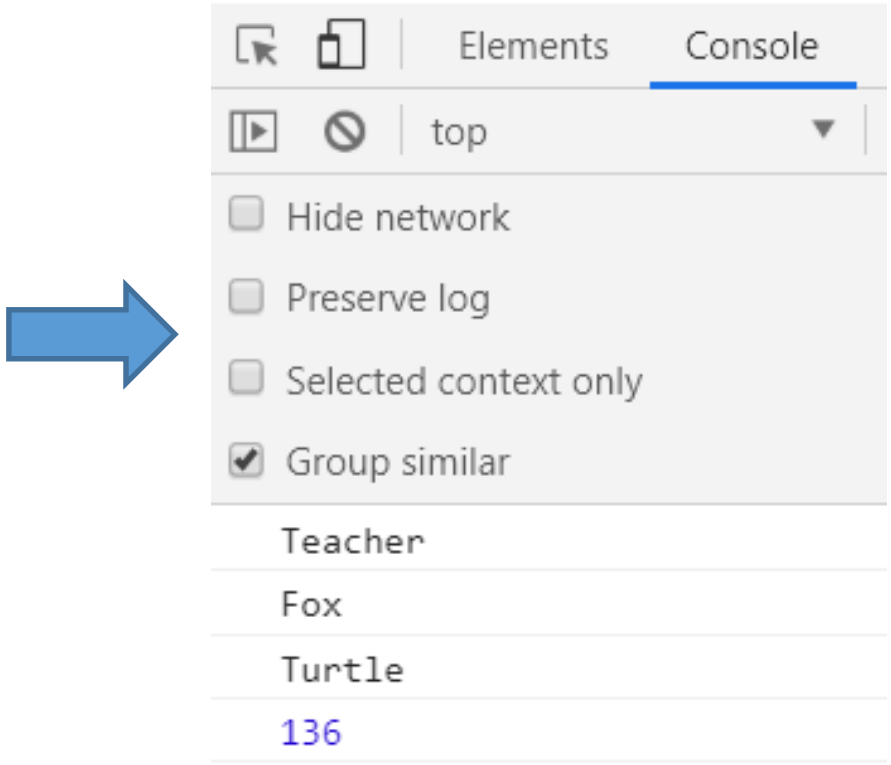


Console.log

console.log is a quick expression used to print content to the debugger.

It is a very useful tool to use during development and debugging.

```
var quick = "Fox";
var slow = "Turtle";
var numbers = 121;

//The console.log() method is used to display
//data in the browse's console
//We can log strings, variables and even equations

console.log("Teacher");
console.log(quick);
console.log(slow);
console.log(numbers + 15);
```

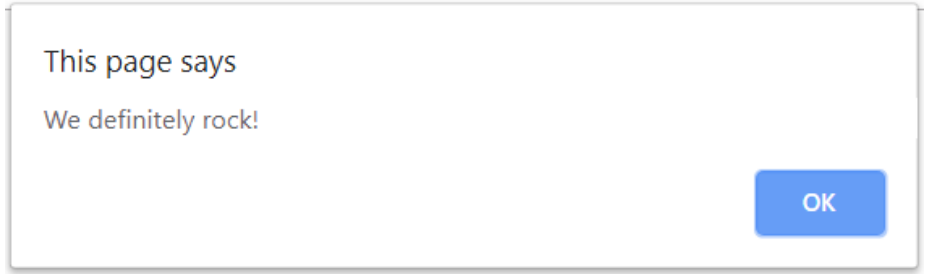


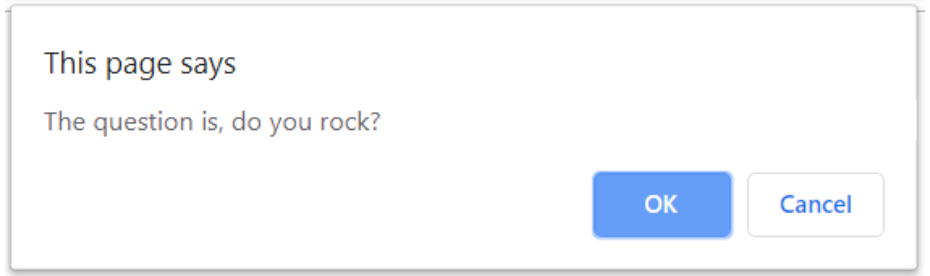
Alerts, Prompts, Confirms

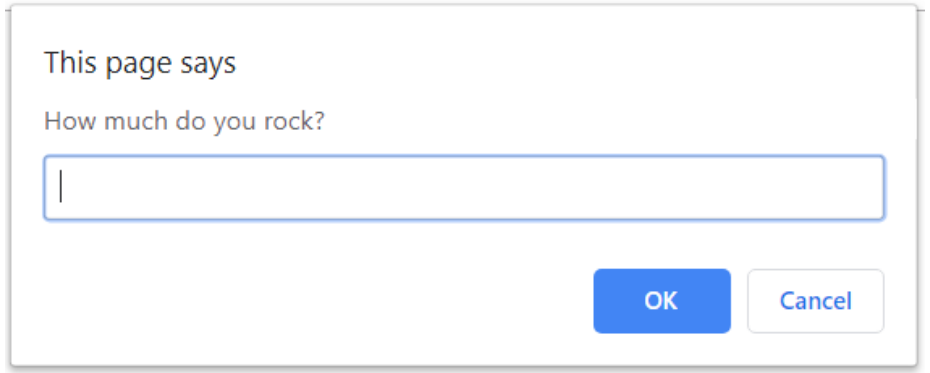
alerts, ***confirms***, and ***prompts*** will create a popup box in the browser when run.

These are also useful for development and debugging.

```
//Alert
alert("We definitely rock!");

//Confirm
var doYouRock =  confirm("The question is, do you rock?");

//Propt
var howMuchRock = propt("How much do you rock?");
```



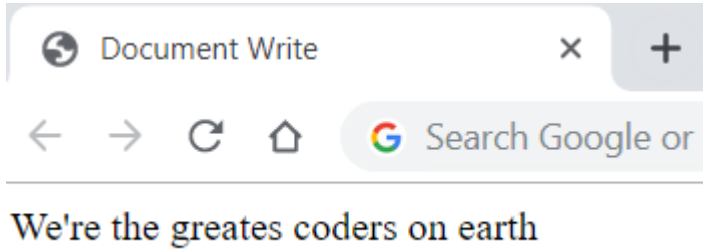




Document Write: Writing to HTML

document.write( ) is used in JavaScript to directly write to the HTML page

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Document Write</title>
  </head>
  <body>
      <script type="text/javascript">
        document.write("We're the greates coders on earth");
      </script>
  </body>
</html>
```



Operators

An ***Operator*** is a symbol used to perform a calculation or comparison.

- Addition (+) a + b
- Subtraction (-) a – b
- Multiplication (*) a * b
- Division (/) a / b
- Modulus (%) a % b
- Increment (++) a ++
- Decrement (--) a—

You can combine or concatenate 2 strings by using the plus operator (+)

ex: var a = “Lazy” + “alligator”;

a is now “Lazy alligator”

Conditional 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:
  - Use ***if*** to specify a block of code to be executed, if a specified condition is true
  - Use ***else*** to specify a block of code to be executed, if the same condition is false
  - Use ***else if*** to specify a new condition to test, if the first condition is false
  - Use ***switch*** to specify many alternative blocks of code to be executed

If/Else Statements

The ***if*** statement executes a statement if a specified condition is truthy. If the condition is falsy, another statement can be executed.

```
//If the user likes sushi,
//we run the following block of code
if(confirmSushi){
  alert("You like " + sushiType + "!");
}
//If user likes ginger tea,
//we run the following block of code
else if(confirmGingerTea){
  alert("You like ginger tea!");
}
//If neither of the previous condition were true,
//we run the following block of code
else{
  document.write("You dont like sushi or ginger tea");
}
```

Conditional Operators

Operator:

==	returns true if both sides of the equation are equal	a == b
!=	returns true if both sides of the equation are NOT equal	a != b
> (>=)	returns true if the left side is greater than (or equal to) the right side	a > b a >= b
< (<=)	returns true if the left side is less than (or equal to) the right side	a < b a <= b

Comparing different Types

When using triple equals **(===)** in JavaScript, we are testing for strict equality. This means both the type and the value we are comparing have to be the same.

When using double equals **(==)** in JavaScript we are testing for loose equality. This means only value has to be same.

```
console.log("5" == 5)   //true
console.log("5" === 5)  //false
```

Switch Statements

The ***switch*** statement evaluates an expression, matching the expression's value to a case clause, and executes statements associated with that case, as well as statements in cases that follow the matching case.

Expression: An expression whose result is matched against each case clause.

Case valueN Optional: A case clause used to match against expression. If the expression matches the specified valueN, the statements inside the case clause are executed until either the end of the switch statement or a break.

Break: Basically is used to execute the statements of a single case statement. If no break appears, the flow of control will fall through all the subsequent cases until a break is reached or the closing curly brace '***}***' is reached.

Default Optional: A default clause; if provided, this clause is executed if the value of expression doesn't match any of the case clauses.

```
var expr = 'Papayas';
switch (expr){
  case 'Oranges':
    console.log('Oranges are $0.59 a pound');
    break;
  case: 'Mangoes':
  case: 'Papayas':
    console.log('Mangoes and Papayas are $2.79 a pound');
    break;
  default:
  console.log('Sorry, we are out of '+ expr);
}
```

Loops

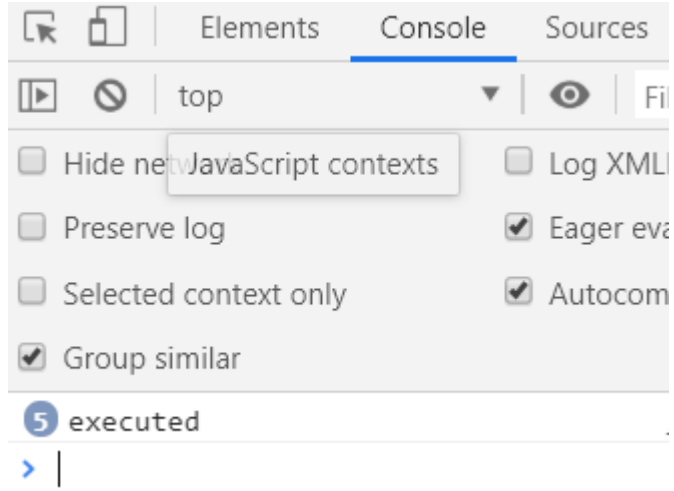
There are several ways to execute a statement or block of statements repeatedly. In general, repetitive execution is called looping. It is typically controlled by a test of some variable, the value of which is changed each time the loop is executed. JavaScript supports many types of loops: ***for loops, while loops, do...while loops***.

For Loop

For Loop: The **for statement** creates a loop that consists of three optional expressions, enclosed in parentheses and separated by semicolons, followed by a statement to be executed in the loop.

for ([initialization]; [condition]; [final-expression]) statement

```
<html>
  <head>
    <title>JavaScript Looping Statements</title>
    <script type="text/javascript">
      for(var i = 1; i <= 5; i++){
        console.log("executed");
      }
    </script>
  </head>
  <body>
    <p>Please open the developer's console to see the uutput of the script</p>
  </body>
</html>
```

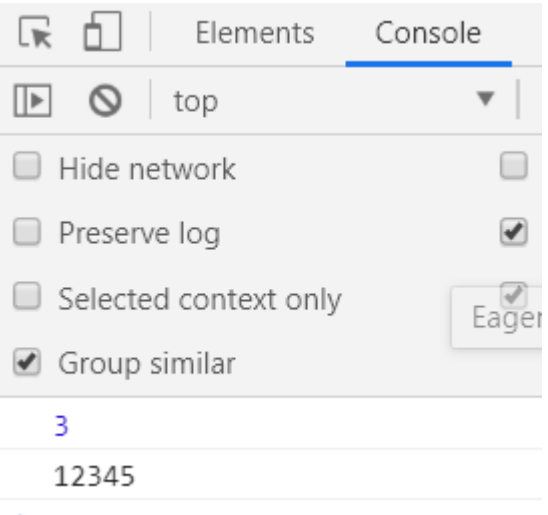


While Loop

While Loop: The while statement creates a loop that executes a specified statement as long as the test condition evaluates to true. The condition is evaluated before executing the statement.

Do While Loop: The do...while statement creates a loop that executes a specified statement until the test condition evaluates to false. The condition is evaluated after executing the statement, resulting in the specified statement executing at least once.

```
var n = 0;

while (n < 3){
  n++;
}

console.log(n);

var result = "";
var i = 0;

do{
  i = i + 1;
  result = result + i;
}while (i < 5);

console.log(result);
```



Function Declaration vs. Expressions

```
//Function Declaration
function add(num1, num2){
  return num1 + num2;
}

//Function Expressions
var add = function (num1, num2){
  return num1 + num2;
}
```

```
//Function Declaration are Hoisted
var result = add(5,5);
function add(num1, num2){
  return num1 + num2;
}

//error!
var result = add(5,5);
var add = function(num1, num2){
  return num1 + num2;
}
```

Function Declaration are hoisted to the top of the context when the code is executed. That means you can actually define a function after it is used in code without generating an error

Functions as Values

Function can be assigned to variables, added to objects, passed to another functions as arguments, and returned from functions.

```
function sayHi(){
  console.log("Hi!");
}

sayHi();    //outputs "Hi!";
var sayHi2 = sayHi;
sayHi2();   //outputs "Hi!";
```

Parameters

Parameters are values that you can pass to a function.

Parameters copy the value of what's provided.

To create a function with ***parameters***, declare parameter names inside of the parenthesis.

In JavaScript, any number of ***parameter*** can be passed to any function.

```
function funcName(param1, param2){
   ...
}
```

Function Arguments

To pass value to a function, you will call that function with it's name and with what's called arguments.

This arguments directly corresponds to the parameter of your method. (ex. arg 1 corresponds to param1, arg 2 corresponds to param2 etc.)

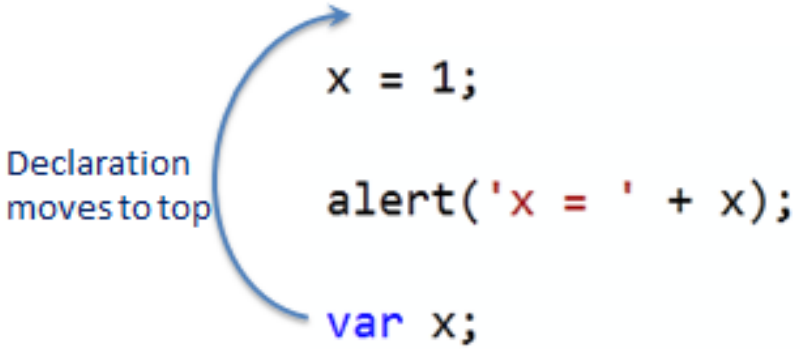
```
funcName(arg1, arg2){
```

Variable Scope

JavaScript Variables are said to have function scope. That means variables declared within a function using var keyword do **NOT** exist outside of the function. It is considered as local and will give undefined if we try to access it outside the function.

If you declare a variable without using the var keyword, then the variable will be available outside of the function.

This process is called ***Hoisting***. It is the process of assigning loose variables as global.



Comments

A ***comment*** is a statement that is NOT processed.

Use a comment to leave notes for yourself and other developers.

Single-Line Comments

- Only covers a span of one line of code.
- A single line comment is specified using two forward-slashes(//).

```
// this is a comment.
var a = 12;
var elem =
document.getElementById('btn');
...
```