

# JavaScript language fundamentals



JavaScript is Java code written in browsers!

JavaScript is programming language



# Myths or Facts → Let us discover



JavaScript can be used to do server side code as well



JavaScript are very slow and unpredictable!

JavaScript make webpage look more beautiful!




**HCL**

# JavaScript

- Is a scripting language developed by Netscape, and later standardized by W3C.
- It was originally called LiveScript.
- Used not only for web client and server side web programs, but also for iPhones, Adobe Photoshop, Adobe Flash action script to name a few.
- JavaScript on web page works with HTML and CSS to create a DHTML page.

# Scripting and Programming language


- Script are line of code that does not execute stand-alone. They run on browser (client-side) or application server (server-side) or on top of some other application.
- They are interpreted at runtime and are not compiled.
- Usually are loosely-typed language.
- Examples: JavaScript, JScript, VBScript, PHP



Is JScript not  
same as  
JavaScript?

JScript is  
Microsoft version  
scripting  
language which is  
very similar to  
JavaScript

Microsoft produced 2  
scripting languages-  
VBScript –(syntax similar  
to VB)and JScript

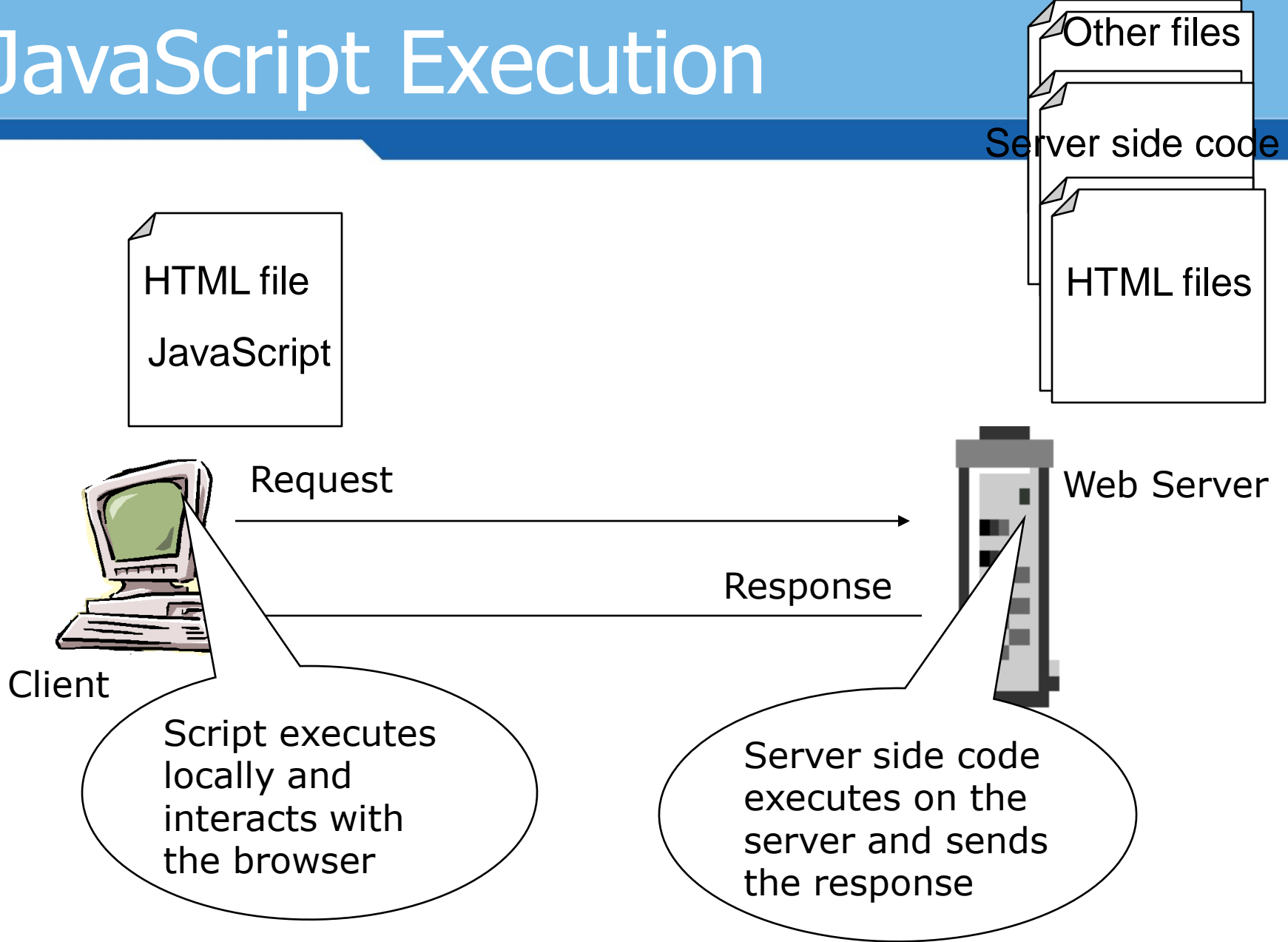


Oh! I  
thought it  
was  
VBScript

# JavaScript in a Web Page

- To create more interactive pages- client side validations etc.
- To generate html dynamically.
- Event handling
- To enhance browser capabilities by giving it a better look – printing on status bar etc.
- Interaction with embedded components like applets and active x controls.

# JavaScript Execution



So JavaScripts on client side executes faster than server-side code.

# Where will the code be written?

- Email format validation
- Password validation
- Changing the colour of the page based on user's input
- Displaying different page based on the user's role.
- Refreshing just part of the page

Client-side  
JavaScript code

Server-side code



# Language Features

- Syntax similar to C++ and Java
- Case sensitive
- Loosely typed
- Interpreted
- Platform independent
- Object-based language
- Semicolon, as separator for multiple statements in the same line.

# Object-based language

- Also called prototype-based object oriented language
- Object-based language
  - JavaScript objects are associative arrays
  - functions as object constructors and methods
  - prototypes instead of classes for inheritance.

More on this later

# Simple Scripts in HTML

```
<HTML><HEAD><TITLE>Hello</TITLE></HEAD>
```

```
<BODY>
```

First java script code<br>

```
<SCRIPT type="text/javascript" >
```

```
//Java script single line comment
```

```
alert("Hello java script");
```

```
/* java script script  
multi-line comment */
```

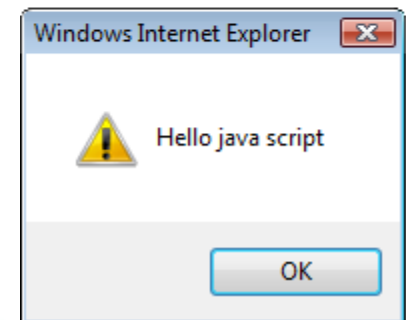
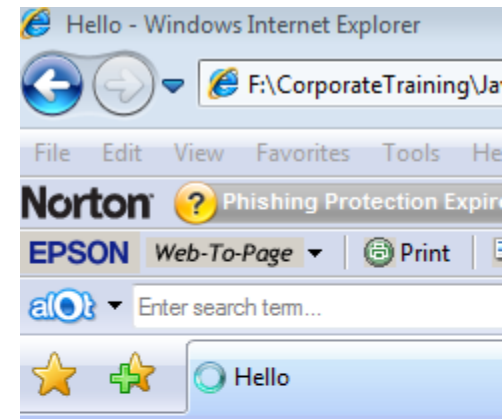
```
</SCRIPT>
```

```
</BODY>
```

```
</HTML>
```

What will happen if you run this on a browser that does not support JavaScript?

popup



# NOSCRIPT and Hiding scripts

- Because some browsers don't support JavaScript it is wise to put the JavaScript code inside HTML commented tags.

```
<SCRIPT type="text/javascript" >  
<!--  
    alert("Hello java script")  
-->  
</SCRIPT>  
<NOSCRIPT>  
Java script is not supported  
</NOSCRIPT>
```

- The text inside the `<NOSCRIPT>` tag will not be processed by the JavaScript aware browsers.

# External Script

- Scripts can also be written in a separate file and can be referenced in a HTML file.

```
<HTML><HEAD><BODY>
```

```
<SCRIPT type="text/javascript"
```

```
SRC="jsfile.js">
```

```
</SCRIPT>
```

```
</BODY>
```

```
</HTML>
```

jsfile.js

```
alert("Hello");
```



Why would you want to write JavaScript in another file?

Path can be

1. Absolute path

```
http://server1/get.jsp
```

2. Root relative:

```
/scripts/jsfile.js
```

3. Document relative

```
../scripts/jsfile.js
```

# Placeholders for scripts within HTML

- Inside head
  - only declarations. Declarations could for variables or functions.
  - No standalone executable statements must appear
- Inside the body
  - any statements can appear
  - standalone executable statements inside the body are interpreted in place where it appears.
- Along with the event handler
  - Script expression can be written as a value when an event like button click happens.

# Data types

- Data types supported by Java Script are
  - a) Numeric – integer and floating point numbers (64 bit, IEE754 floating point)

```
alert(5+154e-2) ;
```

- b) String-

```
alert("Hello") ;or  
alert('Hello') ;
```

- c) Boolean- **true, false**



```
alert('God gives every bird  
it's food but he doesn't throw  
it into it's nest'); gives me
```

Hey! use escape sequences

```
alert('God gives every bird  
it\'s food but he doesn\'t  
throw it into it\'s nest');
```



Yes ! escape sequences can be used or you could also use double quote for string literal.

```
alert("God gives every bird it's  
food but he doesn't throw it  
into it's nest");
```





# Variables and Data types

- Variable names must begin with a letter, under-score or \$, subsequent characters can be a letter, under-score or \$ or number.
- They can be assigned with proper value and used where ever appropriate . They are called data stores.
- To declare a variable:
  - **var x;**
  - **var x=1;** → declare and initialize
- Variables declaration is not compulsory in JavaScript. (If you don't declare a variable explicitly, JavaScript will declare it implicitly for you.)

# Example

```
<HTML><HEAD>  
<SCRIPT type="text/javascript">  
$x=false;  
</SCRIPT>  
</HEAD>  
<BODY>  
<SCRIPT type="text/javascript">  
alert($x==0) ;  
</SCRIPT>  
</BODY>  
</HTML>
```

What do you think the alert box will display?



Find what alert box will display if \$x is uninitialized?

# Operators

- Arithmetic:

`+` `-` `*` `/` `%` `+=` `-=` `*=` `/=` `%=` `++` `--`

- Logical:

`&` `|` `!` `&&` `||`

- Relational:

`>` `>=` `<` `<=` `==` `!=` `===` `!==`

- String concatenation: `+`

- Bit wise:

`>>` `<<` `>>>` `>>=` `<<=` `>>>=`

- Ternary: `?:`

- Conversion functions:

`parseInt()` and `parseFloat()`

To convert string to `int` and `float` respectively

# Examples

Mixing up data types:

```
S="abc";
```

```
I=123;
```

```
alert(S+I) ; → abc123
```

```
B=true;
```

```
alert(S+B) ; → abctrue
```

```
alert(B+1) ; → 124
```

```
alert(5+154e-2 +" kgs") ; → 6.54 kgs
```

```
alert("Rs." +5+154e-2 ) ; → Rs .51.54
```

```
alert("34"==34) ;  
alert(1<".23") ; } → true
```

```
alert(10*"55") ; → 550
```

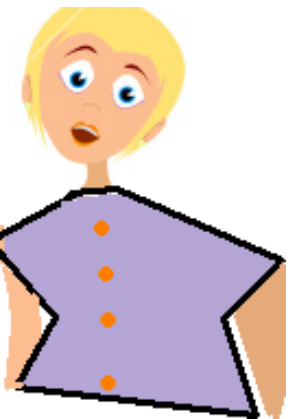
```
alert(10/"abc");    → NaN  
alert(1==true);    → true  
alert(1=="true");  → false  
alert(5/0);        → Infinity  
alert(true & "3"); → 1  
alert(true & "a"); → 0  
alert(10/"abc"==NaN); → false
```



Use `isNaN()`



Then how do we compare NaNs?



There is no char type in JavaScript. How do I increment chars. Can I do 'a'++?

No we cannot, because 'a' is a string. We will look at the methods of String and there we will check out if we can do the achieve the same.

# Shift operators

- The shift left operator looks at the integer to the left of the operator as a 32-bit binary number.
- `>>` the SHIFT RIGHT operator
- `<<` the SHIFT LEFT operator
- `>>>` the UNSIGNED SHIFT RIGHT operator

```
s=2;
```

```
t=-2;
```

```
alert((s<<2) +" "+(s>>2)+ " "+(s>>>2));
```

```
alert((t<<2)+" "+ (t>>2)+ " "+ (t>>>2));
```

**Result of popup:**

8      0      0

-8    -1 1073741823

Find what happens when you shift beyond 32 bits?



# === and !==

- `alert("34"==34) ;`
  - Returns true
- `alert("34"===34) ;`
  - Returns false
- `===` and `!==` are used for stricter comparisons based on types.
- Note that `alert(34.0===34)` returns true
- `alert(true===1) ;` returns false

# Control statements

- Same as in C++ or Java
  - `if else`
  - `for`
  - `for..in`
  - `while`
  - `do .. while`
  - `switch`



# Getting input from user

- `String prompt(question,defaultanswer)`
- **Example:** `prompt("what is your name","") ;`
- A prompt box pops up. The user can enter some text and click either "OK" or "Cancel" to proceed after entering text.
- If the user clicks "OK" the box returns the input value.
- If the user clicks "Cancel" the box returns **null**.

# Function

- Like other programming languages, in JavaScript also we can define a reusable code-block which will execute when ever it is called.
- A function can be defined inside **<head>** , **<body>** or in a external file and can be called from anywhere after it has been read.

- Syntax

```
function  
  functionname(var1, var2, . . . , varX)  
{  
  //some code  
}
```

# Example

```
<html><head>
<script type="text/javascript">
<!--
function display(x) {
alert(x) ;}
-->
</script>
</head>
<body>
<script>
<!--
display("hello") ;
-->
</script>
</body></html>
```

**display()** can be defined  
anywhere even after the  
function call

# Calling a function

- The above function can be called as
  - `display()` ;
  - `display("hello")` ;
  - Or `display` with any number of arguments

```
function display(x) {  
  if(x==null)  
    x="Greetings";  
  alert(x) ;  
}
```

- You can also pass values to a function that does not take any arguments!



```
<body>
```

```
<script type="text/javascript">
```

```
display("hello");
```

```
display();
```

```
function display(x)
```

```
{
```

```
if (x==null)
```

```
x="Greetings";
```

```
alert(x) ;
```

```
}
```

```
function display() {
```

```
alert("Greet") ;
```

```
}
```

```
</script>
```

```
</body>
```

No overloading possible. If overloaded functions are provided, only the last defined function is considered.

Prints **Greet** for both the calls

# Local and Global variables

- All the variables that are not explicitly declared are global.
- Local variables are created using **var** inside the function

```
<html><head>
<script>
total=0;
function sum() {
y=20;
var x=10;
total=x+y;
}
function display() {
sum();
alert(total);
alert(y);
alert(x);
}
```

Global variable

Local variable

Error.

```
</script></head><body>  
<script>  
display() ;  
</script>  
</body></html>
```