Priyanshu Dhyani

Java Script Features and Concepts

Introduction

- JavaScript was created by Brendan Eich at netscape and was first introduced in May 1995 .
- The original name of JavaScript was Mocha, in September 1995, name changed to LiveScript, again in Dec 1995 name changed to JavaScript .
- JavaScript is the programming language of HTML and the web. It makes web page dynamic.
- Javascript is a client side scripting language, that is a loosely typed programming language that is interpreted by the browser engine when the web page is loaded.
- JavaScript is used in web pages to add functionality, validate forms, communicating with server and read write html elements.

ECMA Script

- After its adoption outside of Netscape, a standard document was written to describe the way the JavaScript language should work so that the various pieces of software that claimed to support JavaScript were actually talking about the same language.
- This is called the ECMAScript (**European Computer Manufacturer Association Script**) standard, after the Ecma International organization that did the standardization. In practice, the terms ECMAScript and JavaScript can be used interchangeably—they are two names for the same language.
- 1995: javascript creation- netscape
- 1997: ES 1 (first version)
- 2009-2010: ES 5
- 2015: ES 6
- 2017: ES 7

Syntax

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

statements;

statements;

</script>

Or

<script>

statements;

statements;

</script>

Usage (Embed) in HTML:

<!doctype html>

<head>

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

document.write("Hello");

</script>

</head>

</html>

Variables

- JavaScript variables are containers for storing data values.
- JavaScript includes variables which hold the data value and it can be changed anytime.
- JavaScript uses reserved keyword **var** to declare a variable.
- A variable must have a unique name.
- A variable may include only the letters a-z, A-Z, 0-9, the $ symbol, and the underscore(_).
- No other characters such as spaces or punctuation are allowed in a variable name.
- The first character of a variable name can be only a-z, A-Z, $, or _(no numbers).
- Names are case sensitive Count, count, COUNT are all different variables.
- There is no set limit on variable name lengths.
- JavaScript variables declare with the var keyword.

Priyanshu Dhyani

Data Types

- In JavaScript DataType are divided into the following types.

1. Primitive DataType: There are five primitive data types

- **Numbers**: 5, 6.5, 7 etc.
- **String**: “Hello Graphic Era” etc.
- **Boolean**: Represent a logical entity and can have two values: true or false.
- **Null**: This type has only one value : *null.*
- **Undefined**: A variable that has not been assigned a value is *undefined.*

2. Non Primitive DataType: there is only one Non primitive data type.

- **Object**: It is the most important data-type and forms the building blocks for modern JavaScript.

Variable Declaration and Initialization

- var one = 1; // variable stores numeric value
- var two = 'two'; // variable stores string value
- var three; // declared a variable without assigning a value
- Var n= null;
- Multiple Variables in a Single Line
- var one = 1, two = 'two', three;

Example:

```
<!doctype html>
<head>
<script type="text/javascript" language="javascript">
var x=20;
document.write("the value is:"+x+"<br/>");
var x1=10+20
document.write("the value is:"+x1+"<br/>");
var y=10.5;
document.write("the value is:"+y+"<br/>");
var str="welcome to javascript";
document.write("the string is:"+str+"<br/>");
document.write("the boolean value is:"+(10>11)+"<br/>");
var z;
document.write("the value is:"+z+"<br/>");
var a=null;
document.write("the value is:"+a);
</script>
</head>
</html>
```

Priyanshu Dhyani

Variable scope in javascript

- Scope of a variable is the part of the program from where the variable may directly be accessible. In JavaScript, there are two types of scopes:
- **Global Scope** – Scope outside the function.
- **Local Scope** – Inside the function being executed.

Document Object Model (DOM)

- The way a document content is accessed and modified is called the **Document Object Model**, or **DOM**.
- DOM is a way to represent the webpage in the structured hierarchical way so that it will become easier for programmers and users to glide through the document.
- With DOM, we can easily access and manipulate tags, IDs, classes, Attributes or Elements using commands or methods provided by Document object.
- DOM is created by the browser when a web page is loaded.
- In graphical form, it looks like a tree of elements/nodes.
- Programatically, we can use JS read or change the DOM.
- Change/remove html elements in the DOM/on the page.
- Change and add CSS styles to elements.
- Read and change elements attributes(href, src, alt, custom).
- Attach event listeners to elements (click, keypress, submit).

Structure of DOM

- The term **structure model** is sometimes used to describe the tree-like representation of a document.
- **Why called as Object Model ?**
- model includes not only the structure of a document but also the behavior of a document and the objects of which it is composed of like tag elements with attributes in HTML.

Properties of DOM

- **Window Object:** Window Object is at always at top of hierarchy.
- **Document object:** When HTML document is loaded into a window, it becomes a document object.
- **Form Object:** It is represented by ***form*** tags.
- **Link Objects:** It is represented by ***link*** tags.
- **Anchor Objects:** It is represented by ***a href*** tags.
- **Form Control Elements:**: Form can have many control elements such as text fields, buttons, radio buttons, and checkboxes, etc.

Methods of DOM

- **getElementById():** returns the element having the given id value.
- **getElementsByName():** returns all the elements having the given name value.
- **getElementsByTagName():** returns all the elements having the given tag name.
- **getElementsByClassName():** returns all the elements having the given class name.

Priyanshu Dhyani

getElementById():

- The most common way to access an HTML element is to use the id of the element.
- The id attribute defines the HTML element. The innerHTML property defines the HTML content.

innerHTML Property

- The innerHTML property is useful for getting or replacing the content of HTML elements.

getElementsByTagName():

- **getElementsByTagName("Tag_Name"):** allows you to traverse the DOM looking for all the elements on your page with a specified tag name.
- It returns a live node list meaning that it updates itself with the DOM tree automatically, so modification of the DOM tree will be reflected in the returned collection.
- The returned node list or collection of nodes can be accessed by index numbers starting with index 0 like array.
- This method accepts a string indicating the type of elements that be retrieved, a special value "*" returns all elements in the documents.

getElementByClassName("Class_Name")

- This method accepts string indicating the class name of elements that be retrieved.
- We can also pass multiple class name. for matching, all the class name should match.
- You can use the length property of the NodeList object to determine the number of elements with the specified class name, then you can loop through all elements and extract the info you want.

Syntax

- getElementByClassName("ex")

Where 'ex' is the name of the class.

- getElementByClassName("ex color")

The parameter value " ex color" returns all elements in the document which has both the class name.

Example

 welcome MCA
document.getElementByClassName("p1")

getElementByName():

- The getElementsByName() method of the Document object returns a NodeList Collection of elements with a given name in the document.
- It group the components into an array with unique index.
- The nodes can be accessed by index numbers. The index starts at 0.
- Syntax
- var *element* = document.getElementsByName(*name*);

Priyanshu Dhyani

Conditional statements

- Perform different actions for different decisions. Conditional statements are used for this.
- In JavaScript we have the following conditional statements.
- If statements: use this statements to execute some code only if a specified condition is true.
- If..else statements: use this statement to execute some code if the condition is true and another code if the condition is false.
- If….else if….else statements: use this statement to select one of many blocks of code to be executed.
- Switch statements: use this statement to select one of many blocks of code to be executed.

If Statements

```
<!doctype html>
<head>
<script>
var x=prompt("enter any no:");
if(x>10)
{
alert("user entered no is:"+x);
}
</script>
</head>
</html>
```

If … else

```
<!doctype html>
<head>
<script>
var x=prompt("enter any no:");
if(x>10)
{
alert("user entered no is:"+x);
}
else
{
alert("user entered no is small:"+x);
}
</script>
</head>
</html>
```

If …else If …else

```
<!doctype html>
<head>
<script>
var x=prompt("enter any no:");
if(x>10)
{
```

```
alert("user entered no is:"+x);
}
else if(x<10)
{
alert("user entered no is small:"+x);
}
else if(x==10)
{
alert("user entered no is equal:"+x);
}
else
{
alert("invalid input");
}
</script>
</head>
</html>
```

Switch Statement

- Switch is a predefined structure. It is better than if else.

Syntax

```
Switch(n)
{
Case1:
Execute code block 1
Break;
Case2:
Execute code block 2
Break;
Default:
Code to be executed if n is different from case 1 and 2
}
```

```
<!doctype html>
<head>
<script>
var course=prompt("enter any course name:");
switch(course)
{
Case”MCA“:
document.write("selected course is:"+course);
break;
case"BCA":
document.write("selected course is:"+course);
break;
case"Bsc IT":
document.write("selected course is:"+course);
```

```
break;
default:
document.write("selected course not existed");
break;
}
</script>
</head>
```

Loops

- For loop allows to pass three parameters for each statement:
- An initialization expression
- A condition expression
- A modification expression
- These are separated by semicolon, like this

```
For(exp 1; exp2; exp3)
```

Syntax

```
For(initialization; test condition, iteration statement)
{
Statement(s) to be executed if test condition is true
Statement(s) to be executed if test condition is true
}
```

For loop

```
<!doctype html>
<head>
<script >
for(i=1;i<=10;i++)
{
document.write(”<h1> “the value is:“ <h1>”+i+"<br/>");
document.write(“<h1 font-size: 30px> mca</h1>”);
}
</script>
</head>
</html>
```

Break and continues

```
<!doctype html>
<head>
<script>
for(i=1;i<=10;i++)
{
if(i==5)
{
break;
}
document.write("the value is:" +i+ "<br>");
```

```
}
</script>
</head>
</html>
```

While loop

- Syntax

```
While(variable<=endvalue)
{
Code to be executed
Code to be executed
}
```

Example:

```
<!doctype html>
<head>
<script>
var i=1;
while(i<=10)
{
document.write("the value is:" +i+ "<br>");
i++;
}
</script>
</head>
</html>
```

Do while

Syntax

```
Do
{
Code to be executed
Code to be executed
}
While(variable<=endvalue);
```

Example

```
<!doctype html>
<head>
<script>
var i=1;
do
{
document.write("the value is:" +i+ "<br>");
i++;
}
while(i<=10);
```

```
</script>
</head>
</html>
```

Popup Boxes

- JavaScript has three kind of popup boxes.
- Alert box
- Confirm box
- Prompt box

Alert box:

- An alert box is often used if you want to make sure information comes through the user. When an alert box pops up, the user will have to click ‘OK’ to proceed.
- Syntax

alert(“message”);

Confirm box:

- A confirm box is often used if you want the user to verify or accept something. When a confirm box pops up, the user will have to click either “OK” or “cancel” to proceed. If the user clicks “ok”, the box returns true. If the user clicks “cancel”, the box returns false.
- Syntax
- Confirm(“message”);

Escape Sequences:

- Here are some commonly used escape sequences.
- \n: inserts a new line
- \t: inserts a tab
- \r: carriage return
- \b: backspace
- \f: form feed
- \’: single quote
- \”: double quote
- \\: backslash

Prompt box:

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- Syntax
- Prompt(“message”);