JavaScript

1. Inline
2. Embedded
3. External Files
4. MIME Type
5. Strict Mode
6. Target Legacy Browser <!-- -->

JavaScript Reference Techniques

1. Refer by using BOM hierarchy

- Elements are arrange in a hierarchical order called DOM, which is a part of BOM.

BOM [Browser Object Model]

DOM [Document

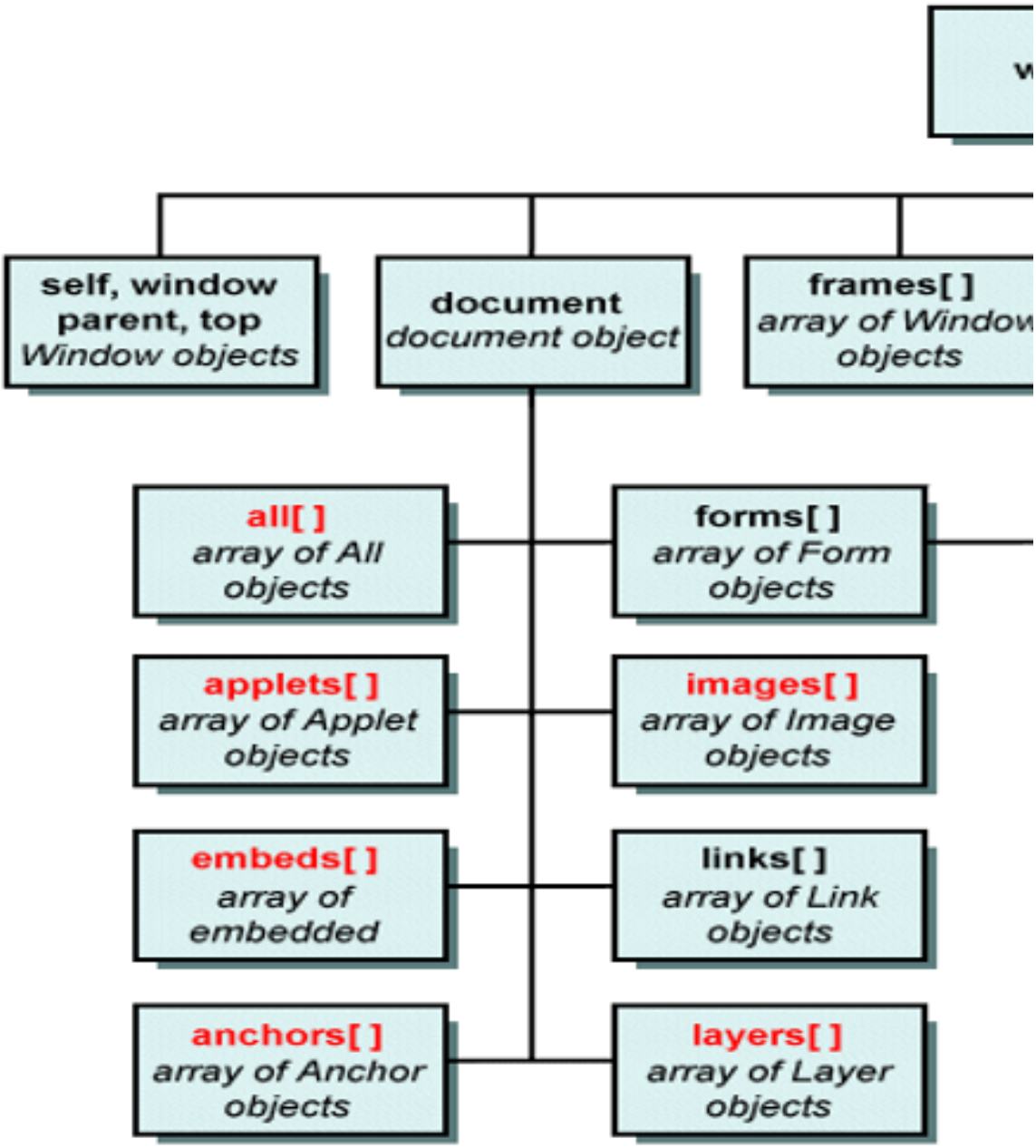
Object Model]

- JavaScript can refer elements using the hierarchical order.

- It is the native and fastest method for browser.

- It uses index reference for a collection of elements in page.

- It is not good for design that changes regularly, as you have to update the index number in logic whenever the position of element changes in design.



Syntax:

window.document.images[ ]

window.document.forms[ ].elements[ ]

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script>
        function bodyload(){

window.document.images[0].sr
c = "../public/images/women-
fashion.jpg";

window.document.forms[0].ele
ments[2].value = "Register";
```

```
window.document.forms[1].ele
ments[2].value = "Login";
        }
    </script>
</head>
<body onload="bodyload()">
    <div>
        <img width="100"
height="100" border="1">
    </div>
    <div>
        <form>
            <h2>Register</
h2>
            <dl>
                <dt>User
Name</dt>
                <dd><input
type="text"></dd>
                <dt>Age</dt>
                <dd><input
type="number"></dd>
            </dl>
```

```
                <input
type="button">
            </form>
        </div>
        <div>
            <form>
                <h2>User Login</
h2>
                <dl>
                    <dt>Email</
dt>
                    <dd><input
type="email"></dd>

<dt>Password</dt>
                    <dd><input
type="password"></dd>
                </dl>
                <input
type="button">
            </form>
        </div>
</body>
```

```
</html>
```

2. Refer by using name

- Every element in page can have a reference name.

```
<img name="pic">
<form name="frm">
    <input type="button" name="btn">
</form>
```

- You can access directly using name reference.
- If element is a generic child then it is mandatory to refer its parent.

```
pic.src = "path";
```

frm.btn.value = “value”;

- Name can be same for multiple elements in page, which leads to issues in JavaScript reference.

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script>
        function bodyload(){
            pic.src = "../
public/images/men-
```

```
fashion.jpg";

frmRegister.btnRegister.valu
e = "Register";

frmLogin.btnLogin.value =
"Login";
        }
    </script>
</head>
<body onload="bodyload()">
    <div>
        <img width="100"
name="pic" height="100"
border="1">
    </div>
    <div>
        <form
name="frmRegister">
            <h2>Register</
h2>
            <dl>
                <dt>User
```

```
Name</dt>
                    <dd><input
type="text"></dd>

                </dl>
                <input
name="btnRegister"
type="button">
            </form>
        </div>
        <div>
            <form
name="frmLogin">
                <h2>User Login</
h2>
                <dl>
                    <dt>Email</
dt>
                    <dd><input
type="email"></dd>

<dt>Password</dt>
                    <dd><input
```

```
type="password"></dd>
                </dl>
                <input
name="btnLogin"
type="button">
            </form>
        </div>
</body>
</html>
```

3. Refer by using ID

- Every element can have a reference “ID”.
- JavaScript can access by using document method “**getElementById()**”.
- It allows to access element directly from any level of hierarchy.

<img id=“pic”>

<input type="button" id="btn">

document.getElementById("pic").src="path";

document.getElementById("btn").value = "value";

- ID is a reference for CSS in page, where same ID can be used for multiple elements.
- Same CSS ID for multiple elements can cause issues for JavaScript.

Ex:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script>
        function bodyload(){

document.getElementById("pic
").src = "../public/images/
kids-fashion.jpg";

document.getElementById("btn
Register").value =
"Register";

document.getElementById("btn
Login").value = "Login";
        }
    </script>

</head>
```

```
<body onload="bodyload()">
    <div>
        <img width="100"
id="pic" height="100"
border="1">

    </div>
    <div>
        <form
name="frmRegister">
            <h2>Register</
h2>
            <dl>
                <dt>User
Name</dt>
                <dd><input
type="text"></dd>

            </dl>
            <input
id="btnRegister"
type="button">
        </form>
```

```
    </div>
    <div>
        <form
name="frmLogin">
            <h2>User Login</
h2>
            <dl>
                <dt>Email</
dt>
                <dd><input
type="email"></dd>

<dt>Password</dt>
                <dd><input
type="password"></dd>
            </dl>
            <input
id="btnLogin" type="button">
        </form>
    </div>
</body>
</html>
```

4. Refer by using CSS selectors

- CSS provides various selectors for selecting HTML elements in page. So that it can apply various styles for elements.
- JavaScript can use the CSS selectors for accessing elements.
- It uses document method “**querySelector()**”.
- It can refer any element from any level of hierarchy.

```
<img>
<input type=“button” class=“btn-primary”
```

>

document.querySelector(“img”).src=“path”;

document.querySelector(“.btn-primary”).value = “value”;

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script>
        function bodyload(){
```

```
document.querySelector("img"
).src="../public/images/
women-fashion.jpg";

document.querySelector("#btn
Register").value="Register";

document.querySelector(".btn
-login").value = "Login";

        }
    </script>

</head>
<body onload="bodyload()">
    <div>
        <img width="100"
height="100" border="1">

    </div>
    <div>
        <form
name="frmRegister">
```

```
                <h2>Register</
h2>
                <dl>
                    <dt>User
Name</dt>
                    <dd><input
type="text"></dd>

                </dl>
                <input
id="btnRegister"
type="button">
            </form>
        </div>
        <div>
            <form
name="frmLogin">
                <h2>User Login</
h2>
                <dl>
                    <dt>Email</
dt>
                    <dd><input
```

```
type="email"></dd>

<dt>Password</dt>
                    <dd><input
type="password"></dd>
                </dl>
                <input
class="btn-login"
type="button">
            </form>
    </div>
</body>
</html>
```

Note: JavaScript can access multiple elements as collection by using various methods

getElementsByTagName()
getElementsByName()

getElementsByClassName() etc.

JavaScript Output Techniques:

- Output is the process of rendering results to user on UI.
- JavaScript has various output techniques

1. alert()
2. confirm()
3. document.write()
4. textContent
5. innerText
6. innerHTML
7. outerHTML
8. console methods etc.

alert()

- It pops up a message box in

browser window.

- It is RC type, can't display complex formats of text.
- It can present result of a value or expression.

Syntax:

```
alert("value l expression");
alert("Welcome");
alert(10);
alert(10+20);    // 30
alert("Addition=" + (10+20));
```

- You can add a line break using "\n". To display text in multiple lines.

Syntax:

alert(“line1 \n line2 \n line3”) ;

- It will not allow to cancel.

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script>
        function
DeleteClick(){
            alert("Record
Deleted..");
        }
    </script>
```

```
</head>
<body>
    <button
onclick="DeleteClick()">Dele
te</button>
</body>
</html>
```

confirm()

- It is similar to alert() but allows to cancel.
- It returns a boolean value. [true or false]

true	on OK
false	on Cancel

Syntax:

confirm("value | expression");

Ex:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width,
initial-scale=1.0">
    <title>Document</title>
    <script>

        function
DeleteClick(){
            result =
confirm("Delete Record\nAre
you sure?");
            if(result==true)
{

alert("Deleted..");
            } else {
```

```
alert("Canceled..");
                }

            }
        </script>
</head>
<body>
    <button
onclick="DeleteClick()">Dele
te</button>
</body>
</html>
```