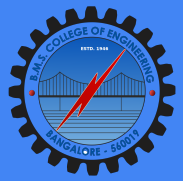



BMSCE
IEEE

Advance JavaScript - Part 1

Rahul Kumar
Webmaster BMSCE IEEE
Chairman BMSCE IEEE

Loading External JavaScript File

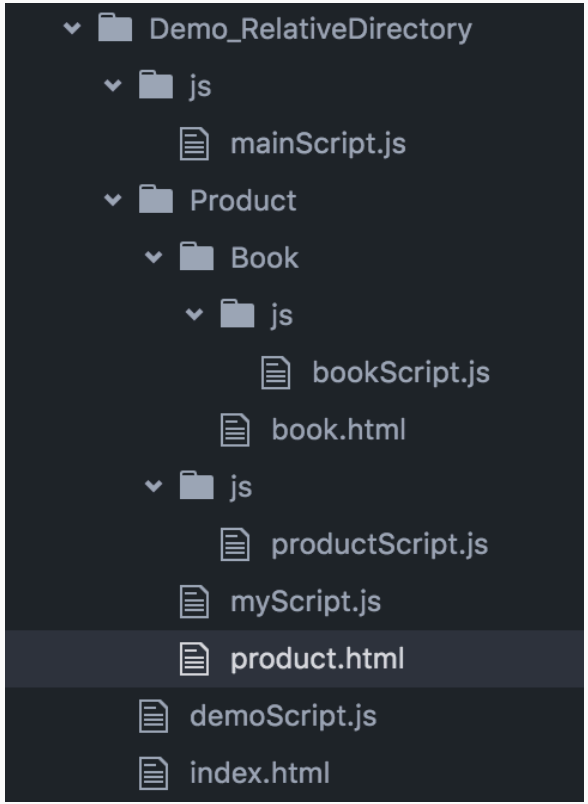
Loading External JavaScript File

Loading any javascript (.js) file present at different directory level.

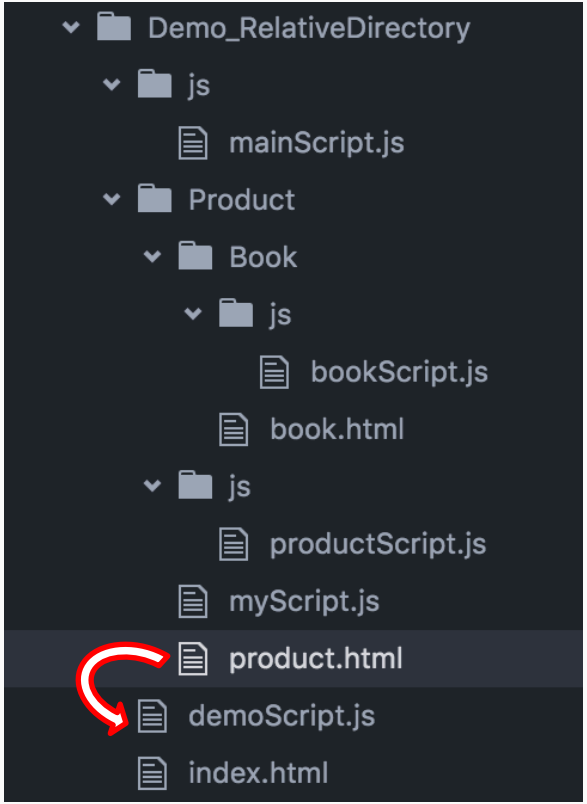
Also known as **relative directory**

Can be used to locate any href link, image src or css file.

Loading External JavaScript File



Loading External JavaScript File

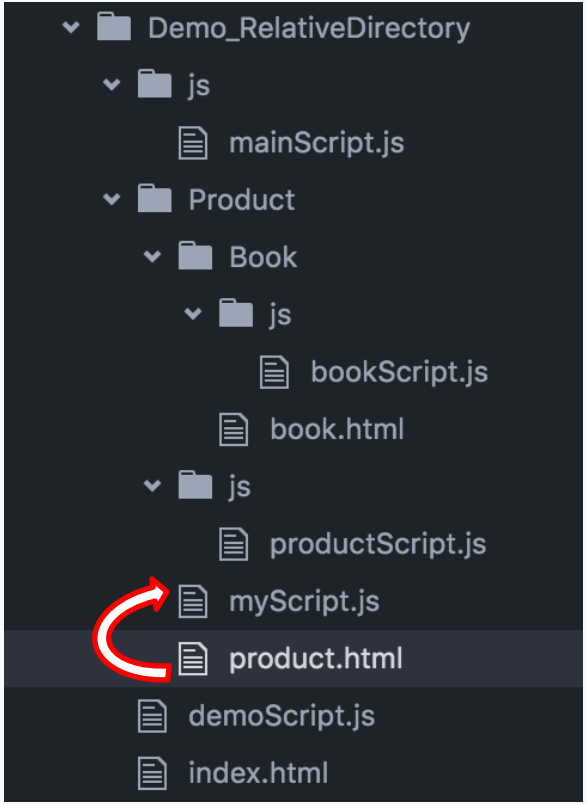


product.html

```
<!-- To goto upper level directory -->

<script src="../demoScript.js"></script>
```

Loading External JavaScript File

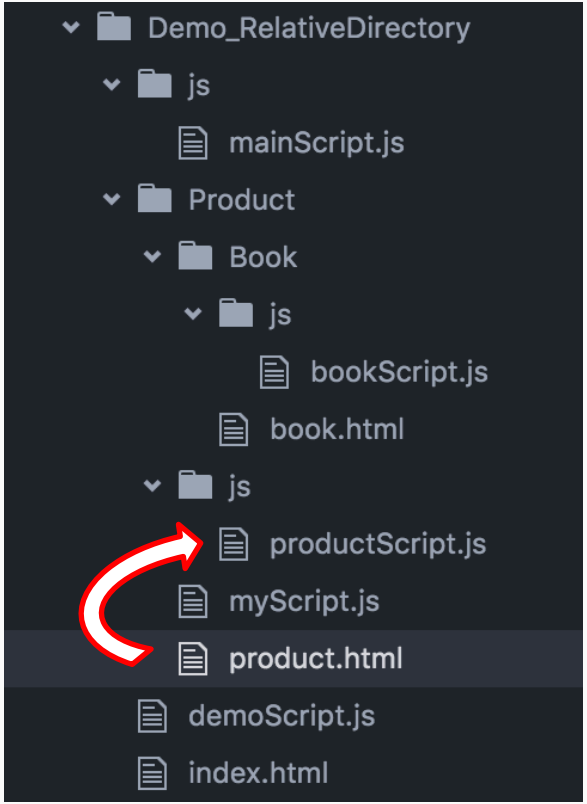


product.html

```
<!-- To load from same level directory -->

<script src="myScript.js"></script>
```

Loading External JavaScript File

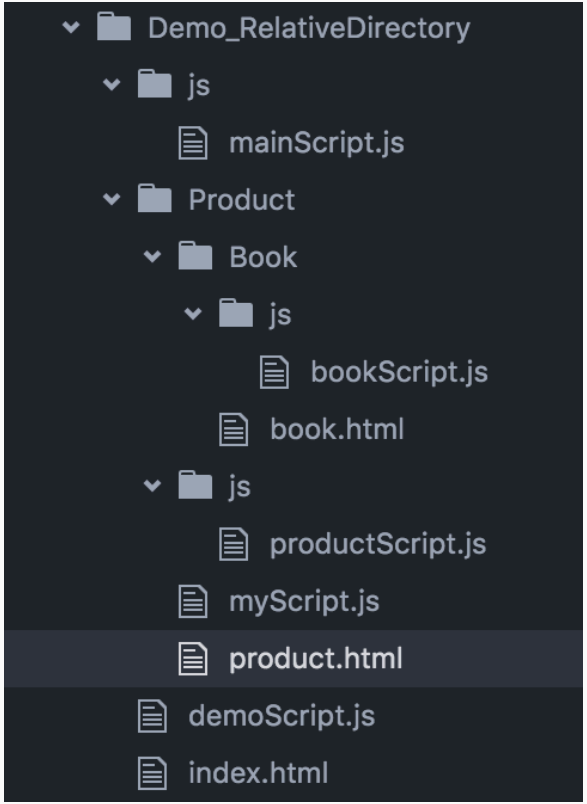


product.html

```
<!-- To goto lower level directory -->

<script src="js/productScript.js"></script>
```

Loading External JavaScript File



product.html

```
<!-- To goto upper level directory -->
<script src="../demoScript.js"></script>

<!-- To goto upper level directory -->
<script src="../js/mainScript.js"></script>

<!-- To load from same level directory -->
<script src="myScript.js"></script>

<!-- To goto lower level directory -->
<script src="js/productScript.js"></script>

<!-- To goto lower level directory -->
<script src="Book/js/bookScript.js"></script>
```

Function Basics – Local and Global Scope

Function Basics – Local and Global Scope

Global Scope : The scope is “global”, which means that variables declared are potentially accessible from everywhere.

Local Scope : Variables which are declared within a function, as well as the function parameters have local scope. That means, they are only visible within that function.

Function Basics – Local and Global Scope

```
var a = 10;
var b = 20;

function add(x, y) {
    var a = x + y;
    return a;
}

function multiply(x, y) {
    var a = x * y;
    return a;
}

console.log(add(a, b));
console.log(multiply(a, b));
console.log(a);
console.log(b);
```

The variable a inside function add only exists in the function's local scope. Because it has been declared with var, it doesn't modify the same-named variable "outside" the function.

If the x were not declared with var, it "shadows" the same-named variable from the nearest external scope!

Function Basics – Local and Global Scope

```
var a = 10;
var b = 20;

function add(x, y) {
    var a = x + y;
    return a;
}

function multiply(x, y) {
    var a = x * y;
    return a;
}

console.log(add(a, b));
console.log(multiply(a, b));
console.log(a);
console.log(b);
```

What will be output of this ?

Function Basics – Local and Global Scope

```
var a = 10;
var b = 20;

function add(x, y) {
    var a = x + y;
    return a;
}

function multiply(x, y) {
    var a = x * y;
    return a;
}

console.log(add(a, b));
console.log(multiply(a, b));
console.log(a);
console.log(b);
```

Output

```
30
200
10
20
```

Function Basics – Local and Global Scope

```
1   var a = 10;
2   var b = 20;
3
4   function add(x, y) {
5       var a = x + y;
6       return a;
7   }
8
9   function multiply(x, y) {
10      a = x * y;
11      return a;
12  }
13
14  console.log(add(a, b));
15  console.log(multiply(a, b));
16  console.log(a);
17  console.log(b);
```

What will be output of this ?

Function Basics – Local and Global Scope

```
1   var a = 10;
2   var b = 20;
3
4   function add(x, y) {
5       var a = x + y;
6       return a;
7   }
8
9   function multiply(x, y) {
10      a = x * y;
11      return a;
12  }
13
14  console.log(add(a, b));
15  console.log(multiply(a, b));
16  console.log(a);
17  console.log(b);
```

Output

```
30
200
200
20
```

Function Expression

Function Expression

A function expression is very similar to and has almost the same syntax as a function statement.

The main difference between a function expression and a function statement is the *function name,* which can be omitted in function expressions to create **anonymous functions**.

A function expression can be used as a **IIFE** (Immediately Invoked Function Expression) which runs as soon as it is defined.

Source : developer.mozilla.org

Function Expression

Normal Function Declaration

```
function add(x, y) {
    var a = x + y;
    return a;
}

console.log(add(a, b));
```

Function Expression

```
var addNumbers = function add(x, y) {
    var a = x + y;
    return a
};

console.log(addNumbers(5, 10));
```

Function Expression

```
var addNumbers = function add(x, y) {
    var a = x + y;
    return a
};

console.log(addNumbers(5, 10));
```

We can also write it as :

```
var addNumbers = function (x, y) {
    var a = x + y;
    return a
};

console.log(addNumbers(5, 10));
```

Here function is *anonymous*

Note : Both the function expression are valid.

Function Expression : Scope

The function name is only local to the function body.

Function Expression : Scope

What is the output ?

```
var findFactorial = function factorial(n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return n * factorial(n - 1);
    }
};

console.log(factorial(5));
```

Function Expression : Scope

What is the output ?

```
var findFactorial = function factorial(n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return n * factorial(n - 1);
    }
};

console.log(factorial(5));
```

Output

```
Uncaught ReferenceError:
factorial is not defined
```

Note : factorial is local to factorial function

Function Expression : Scope

What is the output ?

```
var findFactorial = function factorial(n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return n * findFactorial(n - 1);
    }
};

console.log(findFactorial(5));
```

Function Expression : Scope

What is the output ?

```
var findFactorial = function factorial(n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return n * findFactorial(n - 1);
    }
};

console.log(findFactorial(5));
```

Output

```
120
```

Note : findFactorial is having global scope

Function Expression : Scope

What is the output ?

```
var findFactorial = function factorial(n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return n * factorial(n - 1);
    }
};

console.log(findFactorial(5));
```

Function Expression : Scope

What is the output ?

```
var findFactorial = function factorial(n) {
    if (n <= 1) {
        return 1;
    }
    else {
        return n * factorial(n - 1);
    }
};

console.log(findFactorial(5));
```

Output

```
120
```

Note : factorial is local to factorial function

JavaScript is different

JavaScript is different

What will be output

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    return loadNumber();

    function loadNumber() {
        return 200;
    }
}

console.log(getNumber());
```

JavaScript is different

What will be output

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    return loadNumber();

    function loadNumber() {
        return 200;
    }
}

console.log(getNumber());
```

Output

```
200
```

JavaScript is different

Generally in other programming language. Nothing executes after **return** statement.

But in javascript loaders work differently.

Because variable declarations (and declarations in general) are processed before any code is executed, declaring a variable anywhere in the code is equivalent to declaring it at the top. This also means that a variable can appear to be used before it's declared. This behavior is called "hoisting", as it appears that the variable declaration is moved to the top of the function or global code.

Source : developer.mozilla.org

Hoisting

Hoisting

All the declared stuff that needs space in memory is first “hoisted” to the top of scope before any operation code is run.

Hoisting : Example 1

Original Function

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    return loadNumber();

    function loadNumber() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {









}

console.log(getNumber());
```

Hoisting : Example 1

Original Function

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    return loadNumber();

    function loadNumber() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    function loadNumber() {
        return 100;
    }









}

console.log(getNumber());
```

Hoisting : Example 1

Original Function

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    return loadNumber();

    function loadNumber() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    function loadNumber() {
        return 200;
    }


}

console.log(getNumber());
```

Hoisting : Example 1

Original Function

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    return loadNumber();

    function loadNumber() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    function loadNumber() {
        return 100;
    }

    function loadNumber() {
        return 200;
    }

    return loadNumber();
}

console.log(getNumber());
```

Output

```
200
```

Hoisting : Example 2

What will be output

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Hoisting : Example 2

Original Functions

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {









}

console.log(getNumber());
```

Hoisting : Example 2

Original Functions

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;









}

console.log(getNumber());
```

Hoisting : Example 2

Original Functions

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;
    var loadNumber = undefined;




}

console.log(getNumber());
```

Hoisting : Example 2

Original Functions

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;

    var loadNumber = function() {
        return 100;
    }



}

console.log(getNumber());
```

Hoisting : Example 2

Original Functions

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;

    loadNumber = function() {
        return 100;
    }

    return loadNumber();

}

console.log(getNumber());
```

Hoisting : Example 2

Original Functions

```
function getNumber() {
    var loadNumber = function() {
        return 100;
    }

    return loadNumber();

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;

    loadNumber = function() {
        return 100;
    }

    return loadNumber();

}

console.log(getNumber());
```

Output

100

Hoisting : Example 3

What will be output?

```
function getNumber() {
    return loadNumber();

    var loadNumber = function() {
        return 100;
    }

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Hoisting : Example 3

Original Functions

```
function getNumber() {
    return loadNumber();

    var loadNumber = function() {
        return 100;
    }

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {









}

console.log(getNumber());
```

Hoisting : Example 3

Original Functions

```
function getNumber() {
    return loadNumber();

    var loadNumber = function() {
        return 100;
    }

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;









}

console.log(getNumber());
```

Hoisting : Example 3

Original Functions

```
function getNumber() {
    return loadNumber();

    var loadNumber = function() {
        return 100;
    }

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;
    var loadNumber = undefined;




}

console.log(getNumber());
```

Hoisting : Example 3

Original Functions

```
function getNumber() {
    return loadNumber();

    var loadNumber = function() {
        return 100;
    }

    var loadNumber = function() {
        return 200;
    }
}

console.log(getNumber());
```

Loads like this

```
function getNumber() {
    var loadNumber = undefined;
    var loadNumber = undefined;
    return loadNumber();




}

console.log(getNumber());
```

Output

ERROR

“Javascript is the duct tape of the Internet.”

- Charlie Campbell

Thanks!

Contact me:

Rahul Kumar
Webmaster | BMSCE IEEE
Chairman | BMSCE IEEE
chairman@bmsceieee.com
www.bmsceieee.com

https://github.com/rahulcomp24/jQuery-and-Advance-JavaScript-Workshop