Type Conversion

- In JavaScript there are 5 different data types that can contain values:

String,Number,Boolean,Object,function

- There are 6 types of objects:

String,Number,Boolean,Array,Object,Date

- And 2 data types that cannot contain values:

Null,Undefined

- NAN (Not a number)

Type Conversion

- Convert Number, Boolean and date to string by global method String() or toString()

- Convert String to Numbers by Number(), ParseFloat(), ParseInt()

Null vs undefined vs NaN

- **Javascript null** represents the intentional absence of any object value and the data type of null is an object, also Null is falsy

- **Javascript undefined** property indicates that the variable has not been assigned a value, or not declared at all

- **Javascript NaN** property represents a “Not-a-Number” value

typeOf

- The typeofoperator is not a variable. It is an operator. Operators (+ - * /) do not have any data type.

- But, the typeof operator always returns a string (containing the type of the operand)

Note : You cannot use “typeof” to determine if a JavaScript object is an array (or a date).

- The constructor property returns the constructor function for all JavaScript variables.

Error Handling

- The **try** statement lets you test a block of code for errors.
- The **catch** statement lets you handle the error.
- The **throw** statement lets you create custom errors.
- The **finally** statement lets you execute code, after try and catch, regardless of the result.

Strict Mode

- The "use strict" directive was new in ECMAScript version 5.
- It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.
- The purpose of "use strict" is to indicate that the code should be executed in "strict mode".
- With strict mode, you can not, for example, use undeclared variables.

Note: Strict mode is declared by adding "use strict"; to the beginning of a script or a function.

Debugging

- There are JavaScript debugging tools built right into the browsers.
- Some are:
  - Firefox: Ctrl+Shift+I
  - Chrome: Ctrl+Shift+I
  - Opera: Ctrl+Shift+I
  - Safari: Ctrl+Alt+I
  - IE: F12
  - Firefox’s Firebug: F12
- Inside these tools we can execute our script step by step, watch for variables values and more.

Debugging

- **Console.Log() method**

If your browser supports debugging, you can use console.log() to display JavaScript values in the debugger window

- **Setting Breakpoints**

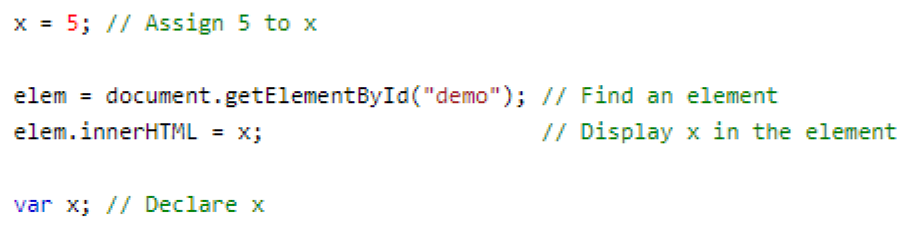
In the debugger window, you can set breakpoints in the JavaScript code.At each breakpoint, JavaScript will stop executing, and let you examine JavaScript values.

- **The debugger keyword**

The debugger keyword stops the execution of JavaScript, and calls (if available) the debugging function.

Hoisting

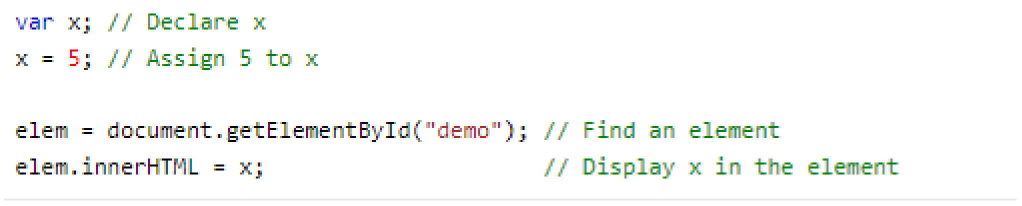
- Hoisting is JavaScript's default behavior of moving declarations to the top.



```
x = 5; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x;                     // Display x in the element

var x; // Declare x
```



```
var x; // Declare x
x = 5; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x;                     // Display x in the element
```

Hoisting

- JavaScript only hoists declarations, not initializations.

Example 1 does **not** give the same result as **Example 2**

```
var x = 5; // Initialize x
var y = 7; // Initialize y

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y
```

```
var x = 5; // Initialize x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x + " " + y;           // Display x and y

var y = 7; // Initialize y
```

JSON

- JSON is a format for storing and transporting data.
- JSON is often used when data is sent from a server to a web page.

What is JSON?

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

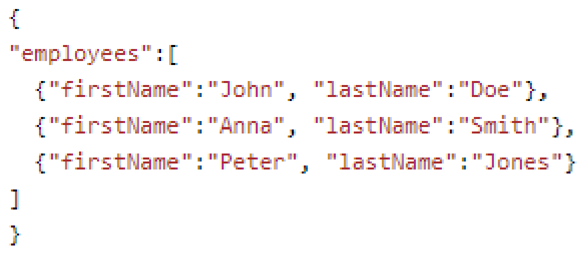
Note: The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only. Code for reading and generating JSON data can be written in any programming language.

JSON format

- The JSON format is syntactically identical to the code for creating JavaScript objects.

JSON Syntax Rules

- Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

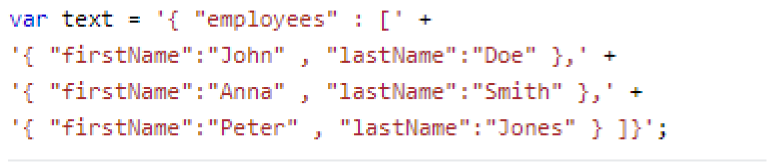


```
{
"employees":[
  {"firstName":"John", "lastName":"Doe"},
  {"firstName":"Anna", "lastName":"Smith"},
  {"firstName":"Peter", "lastName":"Jones"}
]
}
```

JSON parse() and stringify()

- Converting a JSON Text to a JavaScript Object

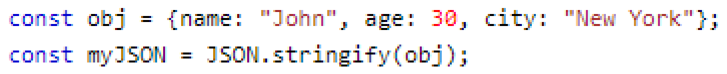
JSON Syntax Rules



```
var text = '{ "employees" : [' +
'{ "firstName":"John" , "lastName":"Doe" },' +
'{ "firstName":"Anna" , "lastName":"Smith" },' +
'{ "firstName":"Peter" , "lastName":"Jones" } ]}';
```

var obj = JSON.parse(text);

- Convert a JavaScript object into a string with JSON.stringify().



```
const obj = {name: "John", age: 30, city: "New York"};
const myJSON = JSON.stringify(obj);
```

const myJSON = JSON.stringify(obj);

JavaScript Array methods

- **Array.isArray()**

The isArray() method checks whether an object is an array.

- **Array.forEach()**

The forEach() method calls a function once for each array element.

- **Array.map()**

The forEach() method creates a new array by performing a function on each array element

- **Array.reduce()**

The reduce method runs a function on each array element to produce (reduce it to) a single value.

- **Array.every()**

The every() method check if all array values pass a test.

- **Array.some()**

The some() method check if some array values pass a test.

JS scope Local and Global

Scope determines the accessibility (visibility) of variables.

JavaScript Function Scope

In JavaScript there are two types of scope:

- **Local scope**
- **Global scope**

- JavaScript has function scope: Each function creates a new scope.
- Scope determines the accessibility (visibility) of these variables.
- Variables defined inside a function are not accessible (visible) from outside the function.

JS scope Local and Global

Local JavaScript Variables

- Variables declared within a JavaScript function, become LOCAL to the function.
- Local variables have Function scope: They can only be accessed from within the function.

Global JavaScript Variables

- A variable declared outside a function, becomes GLOBAL.
- A global variable has global scope: All scripts and functions on a web page can access it.

In JavaScript, objects and functions are also variables. Scope determines the accessibility of variables, objects, and functions from different parts of the code.

“this” keyword

The JavaScript this keyword refers to the object it belongs to.

It has different values depending on where it is used:

- In a method, this refers to the owner object.
- Alone, this refers to the global object.
- In a function, this refers to the global object.
- In a function, in strict mode, this is undefined.
- In an event, this refers to the element that received the event.
- Methods like call(), and apply() can refer this to any object.

“this” keyword

The JavaScript this keyword refers to the object it belongs to.

It has different values depending on where it is used:

- In a method, this refers to the owner object.
- Alone, this refers to the global object.
- In a function, this refers to the global object.
- In a function, in strict mode, this is undefined.
- In an event, this refers to the element that received the event.
- Methods like call(), and apply() can refer this to any object.

JavaScript Functions

JavaScript functions are defined with the function keyword.
You can use a function declaration or a function expression.

Function Declarations

```
function functionName(parameters) {
  // code to be executed
}
```

Note:- Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are invoked (called upon).

JavaScript Self Invoking Functions

Function expressions can be made "self-invoking".

- A self-invoking expression is invoked (started) automatically, without being called.
- Function expressions will execute automatically if the expression is followed by ().
- You cannot self-invoke a function declaration.
- You have to add parentheses around the function to indicate that it is a function expression:

```
(function () {
  let x = "Hello!!";  // I will invoke myself
})();
```

JavaScript Function Call()

With the call() method, you can write a method that can be used on different objects.

- The call() method is a predefined JavaScript method.
- It can be used to invoke (call) a method with an owner object as an argument (parameter).

JavaScript Function Apply()

With the apply() method, you can write a method that can be used on different objects.

The JavaScript apply() Method

- The apply() method is similar to the call() method

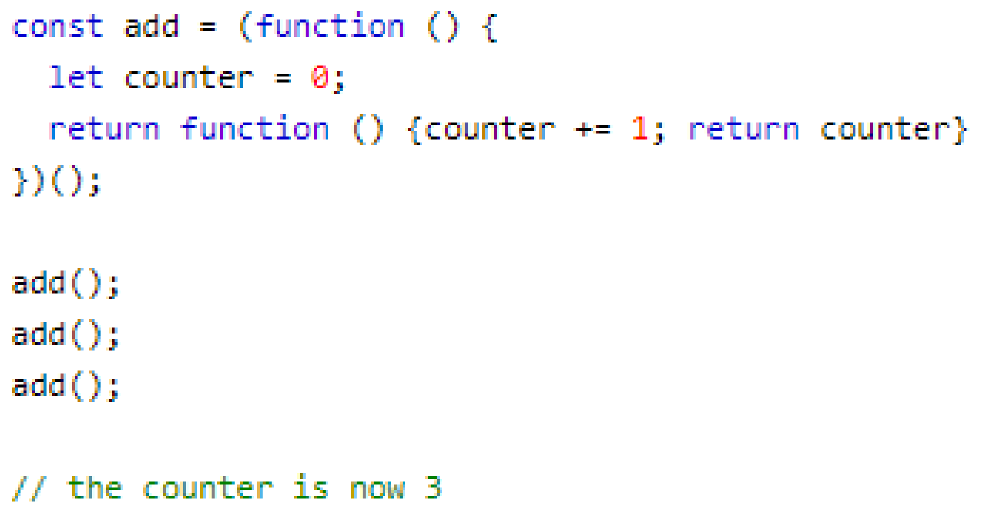
The Difference Between call() and apply()

The call() method takes arguments separately.
The apply() method takes arguments as an array.

JavaScript Function Closures

- JavaScript variables can belong to the **local** or **global** scope.
- Global variables can be made local (private) with **closures**.

Note:-
Variables created without a declaration keyword (var, let, or const) are always global, even if they are created inside a function.



```
const add = (function () {
  let counter = 0;
  return function () {counter += 1; return counter}
})();

add();
add();
add();

// the counter is now 3
```

JavaScript OOPS

- JavaScript supports Object Oriented Programming but not in the same way as other OOP languages(C++, php, Java, etc.) do.
- The main difference between JavaScript and the other languages is that, there are no Classes in JavaScript(upto ES-5) whereas Classes are very important for creating objects.
- However there are ways through which we can simulate the Class concept in JavaScript(ES-5).
- Another important difference is Data Hiding. There is no access specifier like (public, private and protected) in JavaScript but we can simulate the concept using variable scope in functions.