

Course Name : Web Technologies



Course Instructor :

J.Dhiviya Rose

Assistant Professor-Selection Grade

Department of Cybernetics

School of Computer Science

Email : dhiviyarj@ddn.upes.ac.in

Mobile : 9410188296

JavaScript Array and Exception Handling

Course Instructor

Dhiviya Rose J . Asst. Prof. Selection Grade

AI Cluster | SOCS | UPES

Need of array:

- In real time projects array is used to collect same type of objects to send all values with single method call.

Problem of primitive data types:

- We can not store values in continuous memory locations using primitive data types .
- We have two problems due to this limitation

www.InstanceOfJava.com

1. we cant store multiple values:

- If we want to store multiple values , say 1 to 10 , we must create 10 variables .
- All those 10 variables are created at different locations.

2. In single method call we can not pass multiple values :

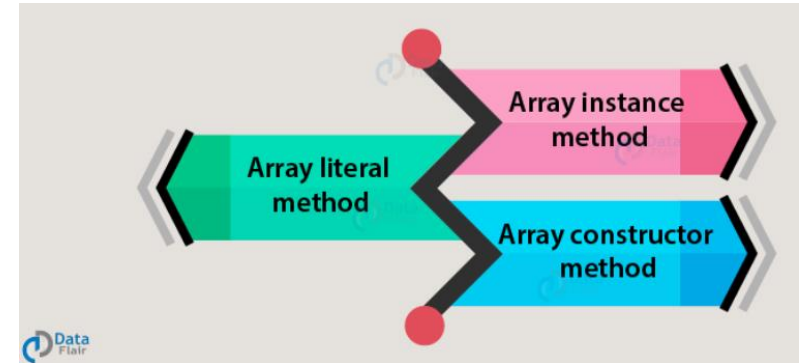
- we can not pass all values to the remote computer with single network call or method call. Using primitive variables , which increases burden on network and also increase number of lines of code in program.

Create Javascript Array?

1. Array Literal Method

The syntax to create an array literal is as follows:

```
1. var arrayName = [val1, val2, val3, ..., valN];
```



2. Array Instance Method

creating an array using this method is as follows:

```
1. var arrayName = new Array( );
```

3. Array Constructor Method

The syntax to create an array with the help of the constructor method is as follows:

```
1. var arrayName = new Array(val1, val2, val3, ..., valN);
```

```
<html>
<body>

<script>
    //creating an array with literal method
    var array1 = ["Neha", "DataFlair Web Services", 26, 3.5, true];
    //accessing the values of the array
    document.write("Employee: " + array1[0] + "<br>");
    document.write("Working for: " + array1[1] + "<br>");
    document.write("Age: " + array1[2] + "<br>");
    document.write("Work Experience: " + array1[3] + "<br>");
    document.write("Still Working?: " + array1[4] + "<br>");
</script>

</body>
</html>
```

```
<html>
<body>

<script>
    //creating an array with constructor method
    var array3 = new Array("Neha", "DataFlair Web Services", 26, 3.5, true);
    //accessing the values of the array
    document.write("Employee: " + array3[0] + "<br>");
    document.write("Working for: " + array3[1] + "<br>");
    document.write("Age: " + array3[2] + "<br>");
    document.write("Work Experience: " + array3[3] + "<br>");
    document.write("Still Working?: " + array3[4] + "<br>");
</script>

</body>
</html>
```

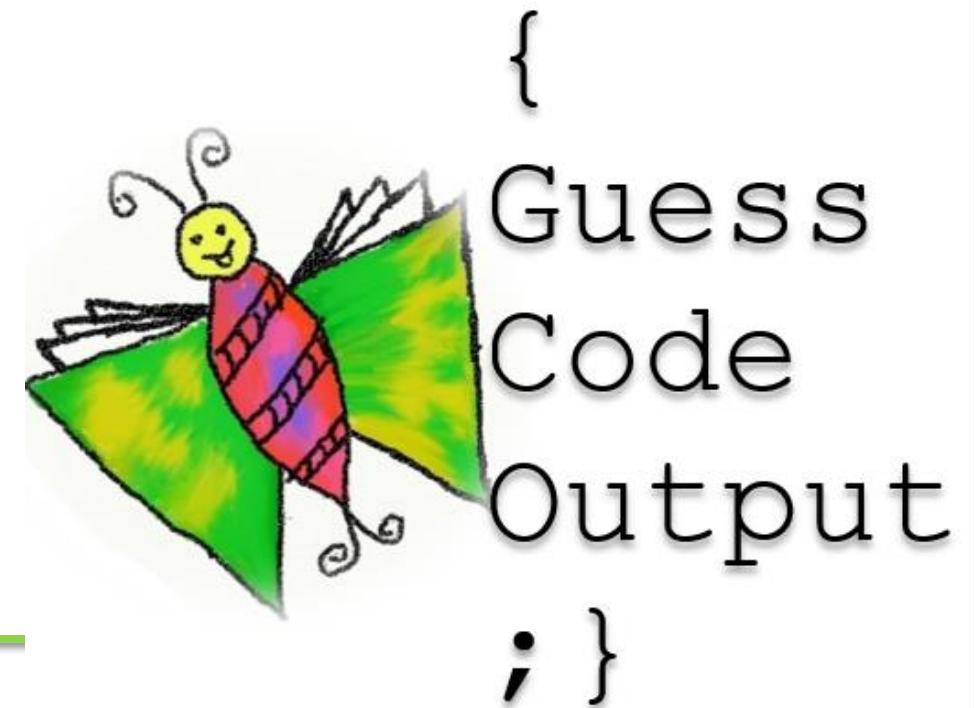
```
<html>
<body>

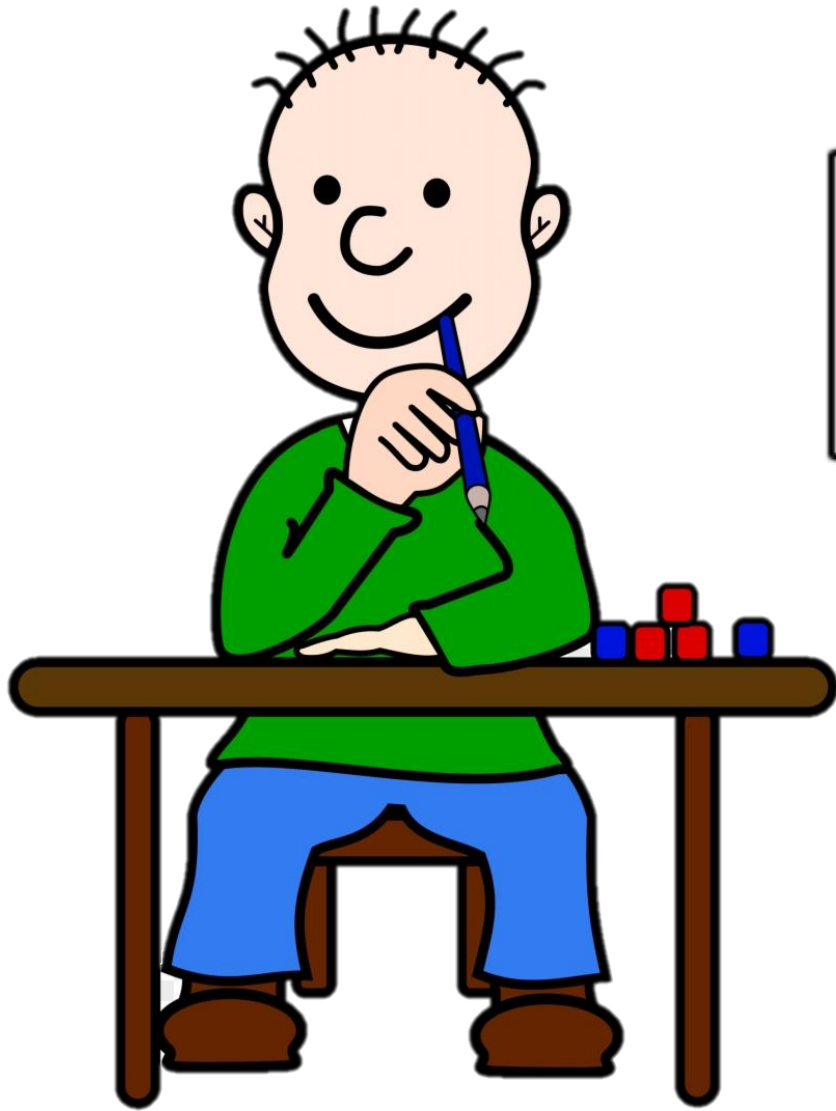
<script>
    var array2 = new Array(); //creating an instance of the array
    //adding values
    array2[0] = "Neha";
    array2[1] = "DataFlair Web Services";
    array2[2] = 26;
    array2[3] = 3.5;
    array2[4] = true;
    //accessing the values of the array
    document.write("Employee: " + array2[0] + "<br>");
    document.write("Working for: " + array2[1] + "<br>");
    document.write("Age: " + array2[2] + "<br>");
    document.write("Work Experience: " + array2[3] + "<br>");
    document.write("Still Working?: " + array2[4] + "<br>");
</script>

</body>
</html>
```

Multi-Dimensional Array

```
<script>
  <!-- Demo on JS Multidimensional array function @dhiviyarj-->
  let activities = [
    ['Work', 9],
    ['Eat', 1],
    ['Commute', 2],
    ['Play Game', 1],
    ['Sleep', 7]
  ];
  console.log(activities[0])
  console.log(activities[2][0])
  console.log(activities[2][1])
  console.log(activities)
  console.log("Length is",activities.length)
</script>
```





$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1+9 & 2+8 & 3+7 \\ 4+6 & 5+5 & 6+4 \\ 7+3 & 8+2 & 9+1 \end{bmatrix} \\
 = \begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$$

ydemo.html >  html

```
<html>
  <!-- Demo on JS array function @dhiviyarj-->
  <script>
    let array1=new Array("One", " Two","Three")
    //add element at the end
    array1.push("Four")
    console.log(array1)
    //add element at begining
    array1.unshift("Begin")
    console.log(array1)
    //delete element at end
    array1.pop()
    console.log(array1)
    //delete element at beginning
    array1.shift()
    console.log(array1)
    //add or replace inbetween
    array1.splice(1,0,"Added")
    console.log(array1)
    array1.splice(1,1,"Added Two")
    console.log(array1)
  </script>
</html>
```

Array's built-in method

W3schools

Tutorials ▾ Exercises ▾ Get Certified ▾ Services ▾

Bootcamps

Gift

HTMLCSSJAVASCRIPTSQLPYTHONJAVAPHPHOW TOW3.CSSCC#BOOTSTRAPREACT

JS Array

at()concat()constructorcopyWithin()entries()every()fill()filter()find()findIndex()flat()flatMap()forEach()from()includes()indexOf()

from()

Creates an array from an object

includes()

Check if an array contains the specified element

indexOf()

Search the array for an element and returns its position

isArray()

Checks whether an object is an array

join()

Joins all elements of an array into a string

keys()

Returns a Array Iteration Object, containing the keys of the original array

lastIndexOf()

Search the array for an element, starting at the end, and returns its position

length

Sets or returns the number of elements in an array

map()

Creates a new array with the result of calling a function for each array element

pop()

Removes the last element of an array, and returns that element

prototype

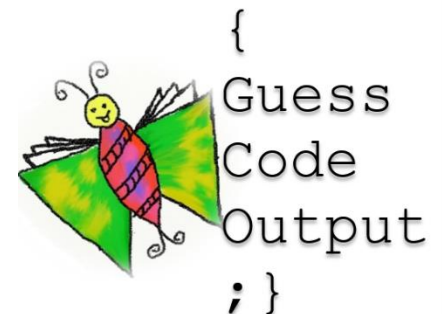
Allows you to add properties and methods to an Array object

push()

Adds new elements to the end of an array, and returns the new length

reduce()

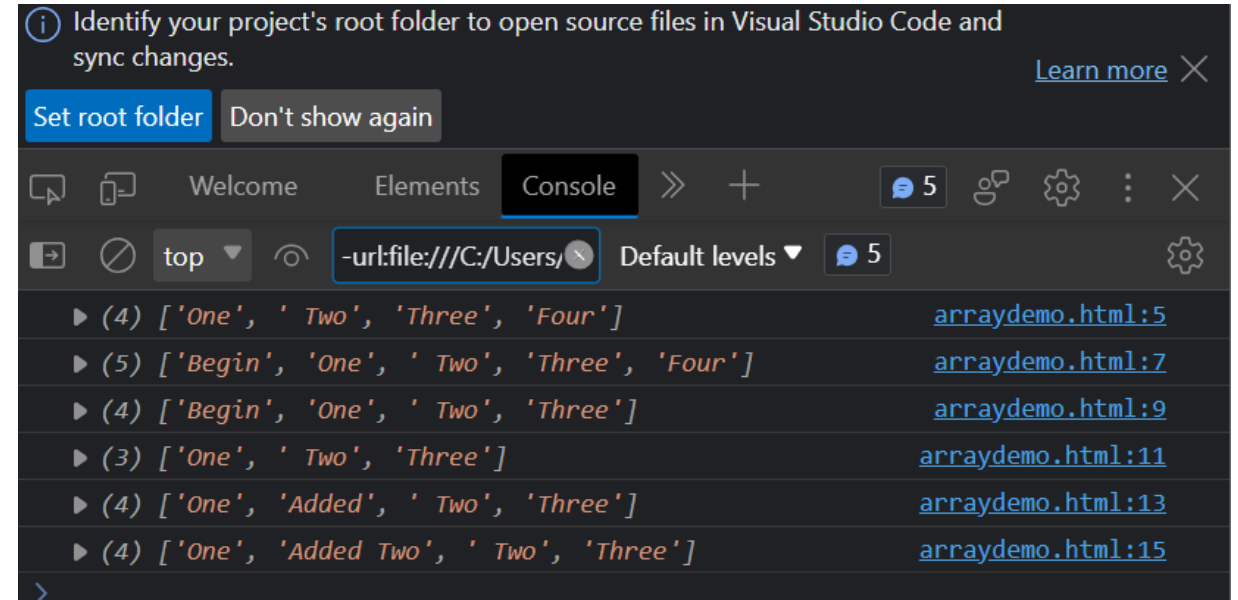
Reduces the values of an array to a single value



arraydemo.html > html

```
<html>
  <!-- Demo on JS array function @dhiviyarj-->
  <script>
    let array1=new Array("One", " Two","Three")
    //add element at the end
    array1.push("Four")
    console.log(array1)
    //add element at beginning
    array1.unshift("Begin")
    console.log(array1)
    //delete element at end
    array1.pop()
    console.log(array1)
    //delete element at beginning
    array1.shift()
    console.log(array1)
    //add or replace inbetween
    array1.splice(1,0,"Added")
    console.log(array1)
    array1.splice(1,1,"Added Two")
    console.log(array1)
  </script>
</html>
```

Array's built-in method



```
<script>
```

```
<!-- Demo on JS Array function @dhiviyarj-->
//Two Dimentional Array
var myarr1=[1,2,2,3,4]
var myarr2=[10,20,30,30,50,60,70,70]
//Concat Function
console.log(myarr2.concat(myarr1))
```

















```
{
  Guess
  Code
  Output
; }
```






```
//map function
let numbers = [1,2,3,4];
let newArr = numbers.map(myFunction)
console.log(newArr)
function myFunction(num) {
  return num * 10;
}
```





```
//Filter Function
let ages = [32, 33, 16, 40];
let result = ages.filter(checkValue);
console.log(result)
function checkValue(agey) {
  return agey >= 30;
}
```








JavaScript Array Methods CheatSheet 🍕





   .map( → ) →   





   .filter() →  

   .find() → 

   .indexOf() → 1











   .fill(1, ) →   







   .some() → true






   .every() → false


















 RammCodes






    .map( => ) =>    







    .filter() => 






    .every() => false

    .some() => true

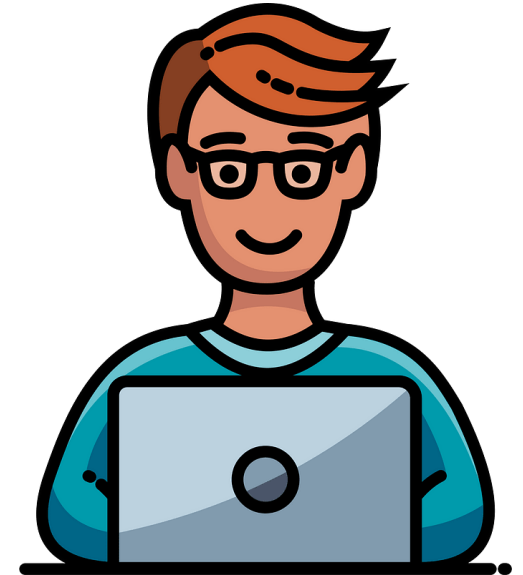
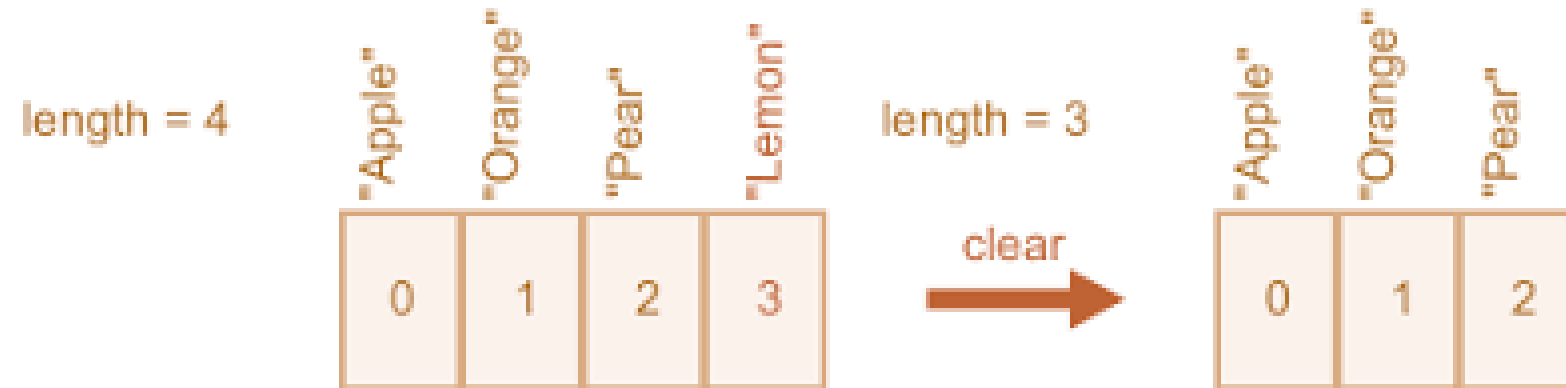
    .fill( , 1) =>    

    .findIndex(el => el === ) => 2

    .find() => 

    .reduce((acc, cur) => acc + cur) => 

Activity Time



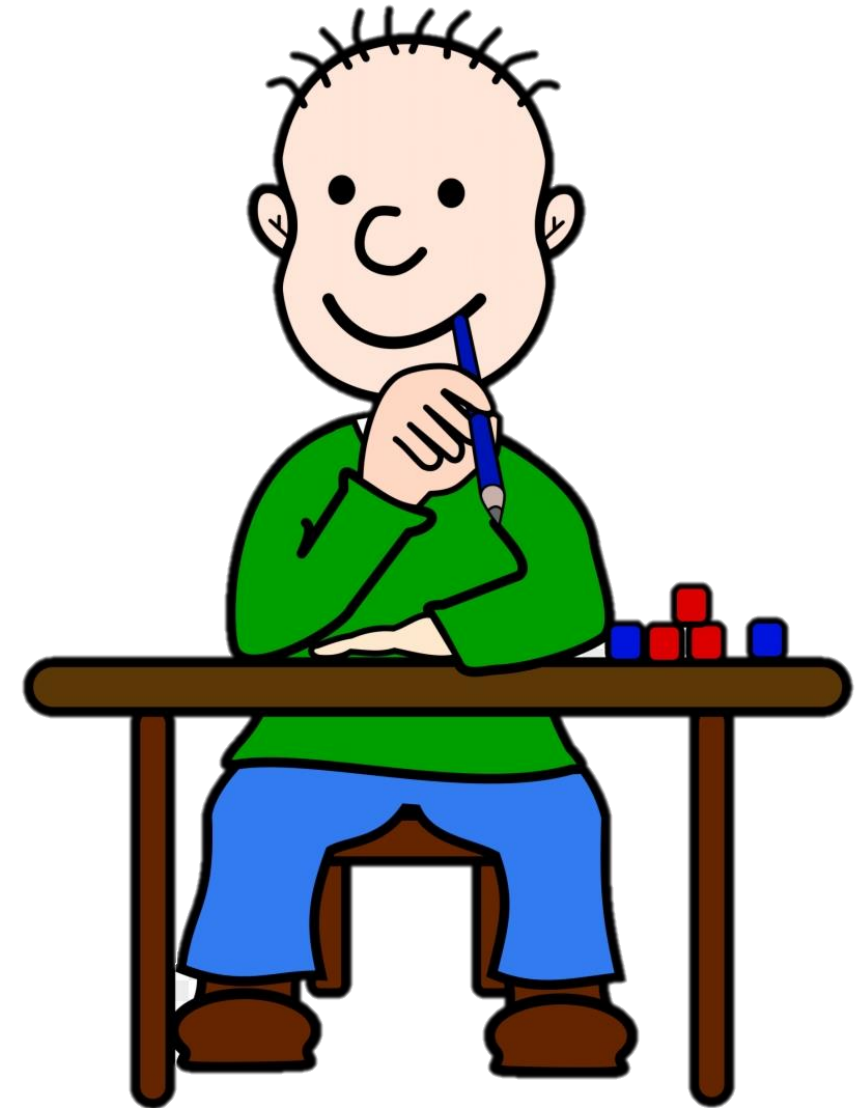
[←](#)
[→](#)
[↻](#)
[w3resource.com/javascript-exercises/](#)

w3resource

[Validation with Regular expression](#)
[Validation without Regular expression](#)
[Sorting Algorithm](#)
[Searching Algorithm](#)
[..More to come..](#)

List of JavaScript Exercises :

- [JavaScript Basic \[150 Exercises with Solution \]](#)
- [JavaScript Fundamental \(ES6 version\) Part-I \[150 Exercises with Solution \]](#)
- [JavaScript Fundamental \(ES6 version\) Part-II \[116 Exercises with Solution \]](#)
- [JavaScript Error Handling \[13 Exercises with Solution \]](#)
- [JavaScript Functions \[29 Exercises with Solution \]](#)
- [JavaScript Recursion \[13 Exercises with Solution \]](#)
- [JavaScript Conditional Statements and loops \[12 Exercises with Solution \]](#)
- [JavaScript Event Handling \[10 exercises with solution \]](#)
- [JavaScript Asynchronous Programming \[9 exercises with solution \]](#)
- [JavaScript Object-Oriented Programming \[12 exercises with solution \]](#)



Associative Array

- Associative Arrays are dynamic objects that redefine as per the user's needs.
- These arrays comprise of **key: value** pairs
- When we assign values to keys in an array variable, the array transforms into an object, losing its properties and methods as Array.

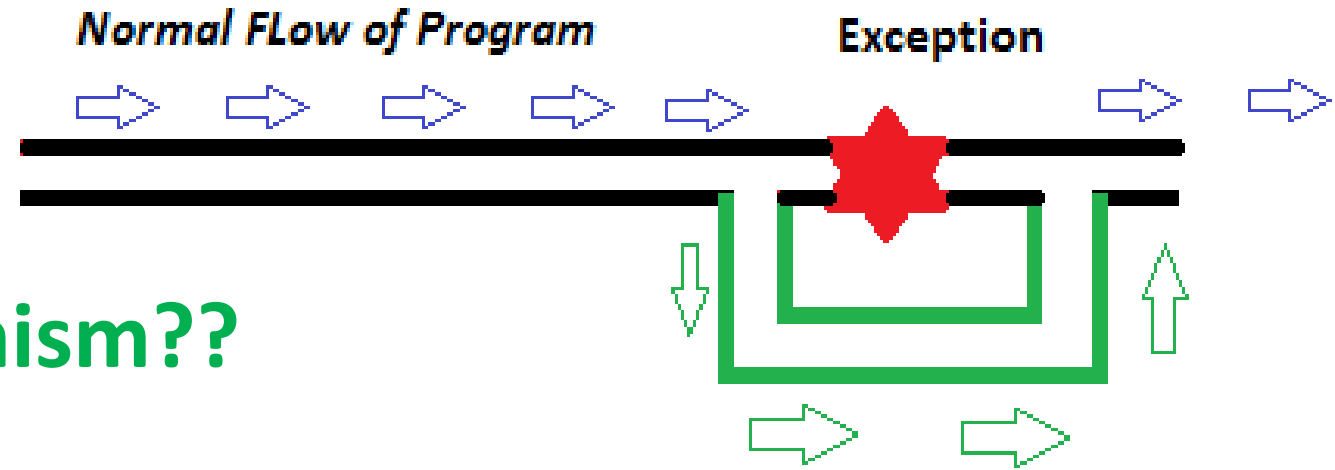
```
var arrayName = {key1: val1, key2: val2, ..., keyN: valN};
```

```
<script>
  //creating an associative array
  var associativeArray = {employee: "Neha", workingFor:"DataFlair Web Services", age: 26, workExpe
  //accessing the values of the array
  for (arrayValue in associativeArray){
    document.write(arrayValue + ": " + associativeArray[arrayValue] + "<br>");
  }
</script>

</body>
</html>
```


Exceptions??

Exception Handling Mechanism??



- An **exception** is an event that occurs during **the runtime** of a program **that disrupts the normal flow of instructions** during the execution of a program.

E.g. An attempt to divide by zero.

- **Exception Handlers** provide a way to transfer control from one part of a program to another path
 - **Prevents abnormal terminations**

Exception Handlers @ JS

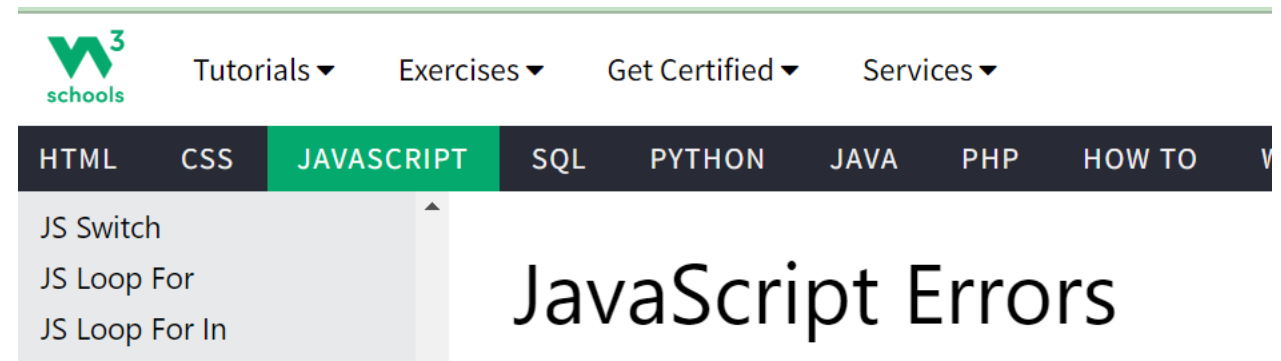
Throw, and Try...Catch...Finally

The `try` statement defines a code block to run (to try).

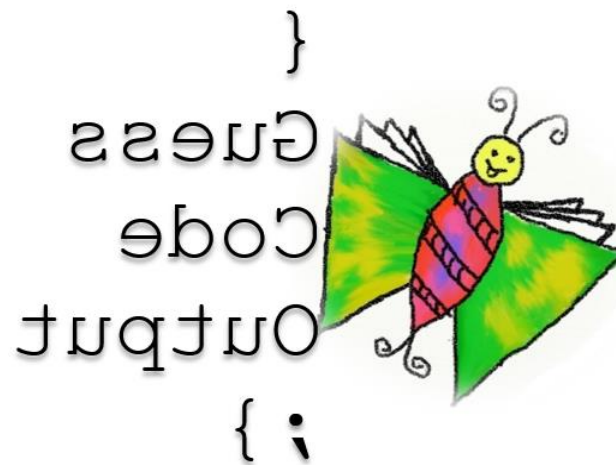
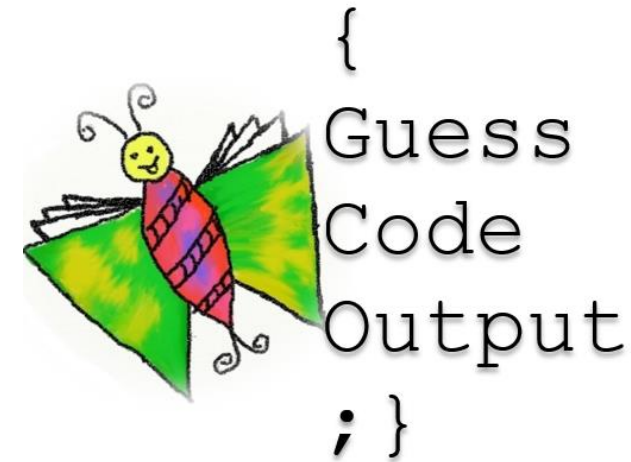
The `catch` statement defines a code block to handle any error.

The `finally` statement defines a code block to run regardless of the result.

The `throw` statement defines a custom error.

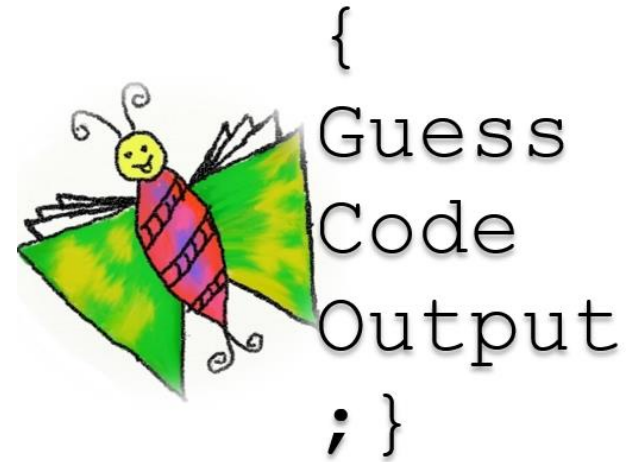


```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      alert("Welcome")
      document.write("Hi All Well!!!!!!")
    </script>
  </body>
</html>
```

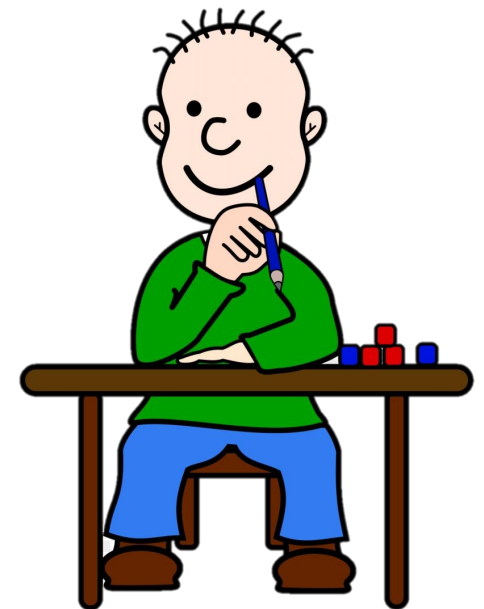
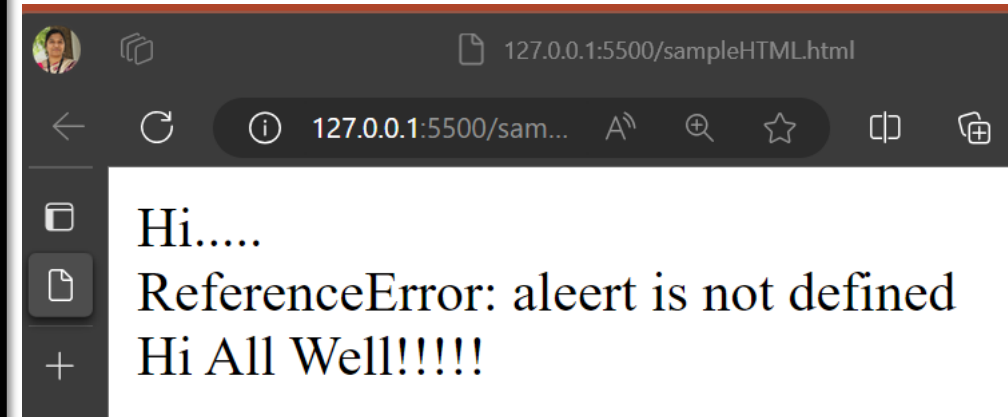


```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....")
      alert("Welcome")
      document.write("Hi All Well!!!!!!")
    </script>
  </body>
</html>
```

```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        alert("Welcome")
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```



```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        aleert("Welcome")
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```



The throw Statement

The `throw` statement allows you to create a custom error.

Technically you can **throw an exception (throw an error)**.

The exception can be a JavaScript `String`, a `Number`, a `Boolean` or an `Object` :

```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

If you use `throw` together with `try` and `catch`, you can control program flow and generate custom error messages.



Tutorials ▼

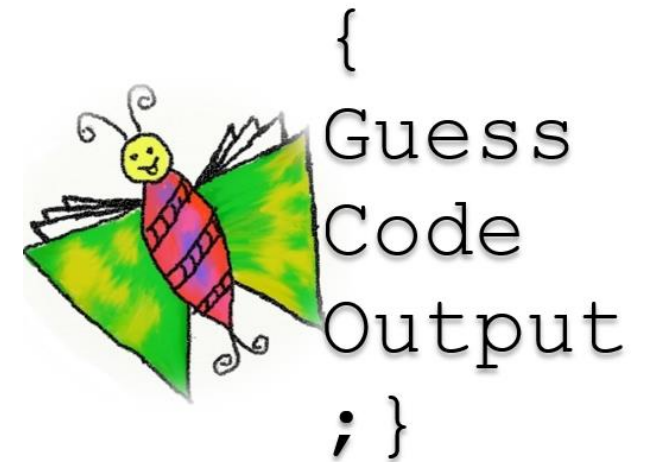
Exercises

HTML

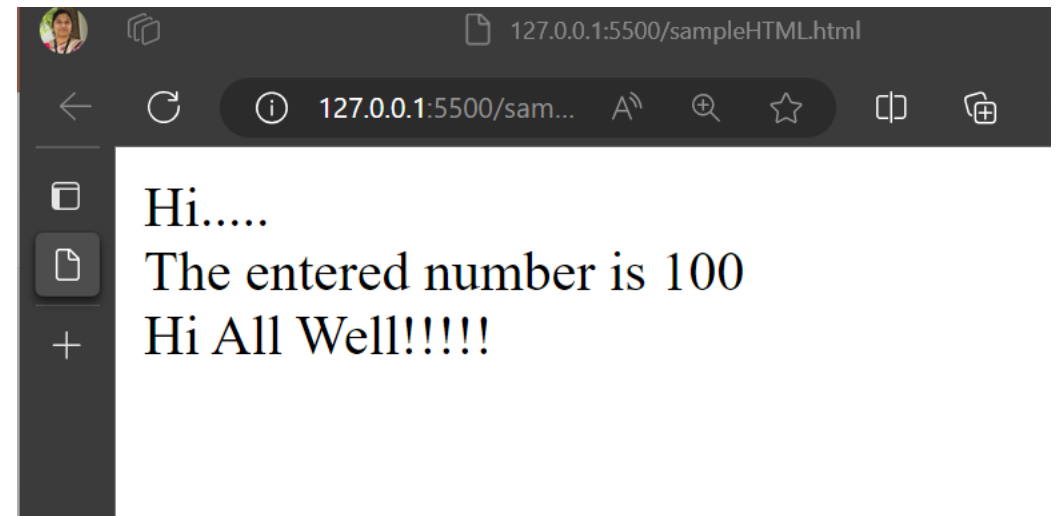
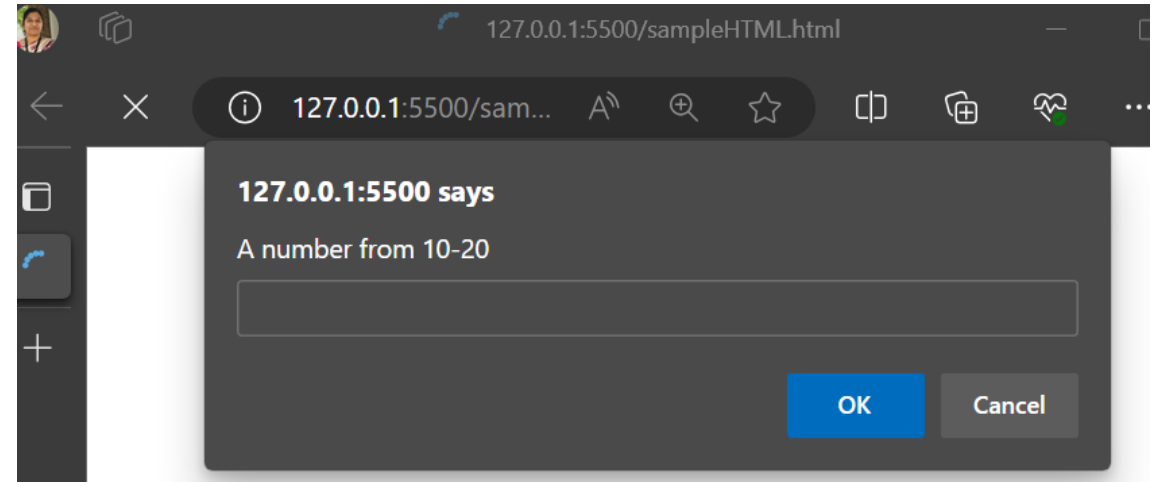
CSS

JAVASCRIPT

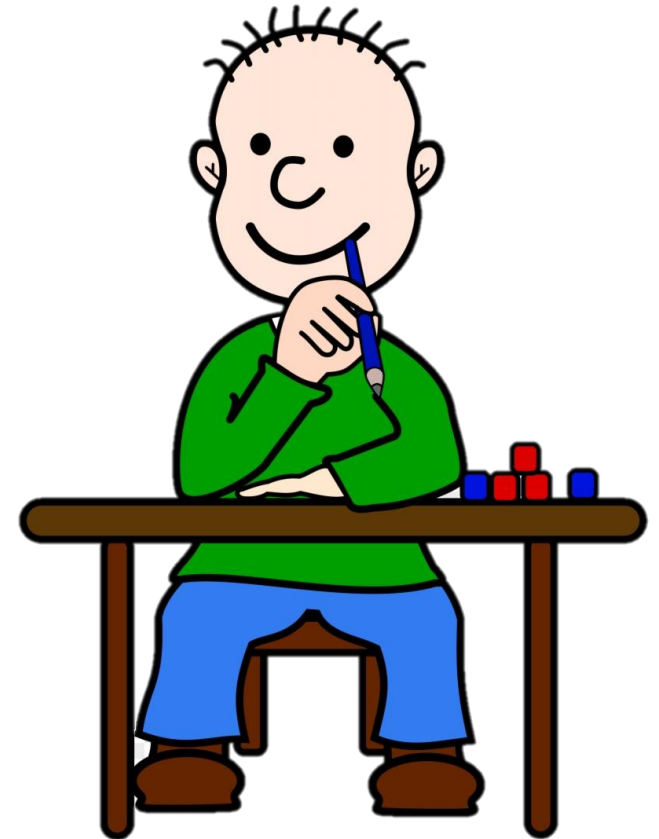
```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        document.write("The entered number is ",no)
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```



```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        document.write("The entered number is ",no)
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```



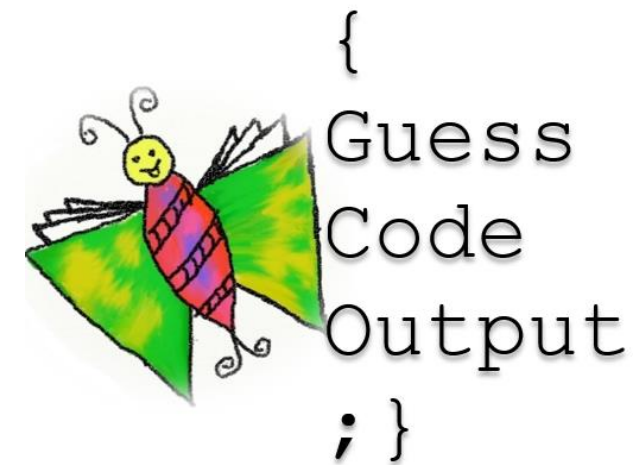

```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        if(no>=10 && no<=20)
          document.write("The entered number is ",no)
        else
          throw "The number should be between 10-20"
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```





Case 1: Input Number 13
Case 2: Input Number 113

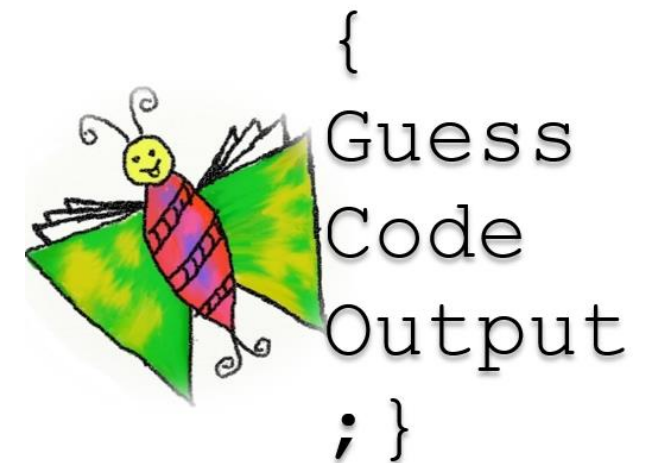
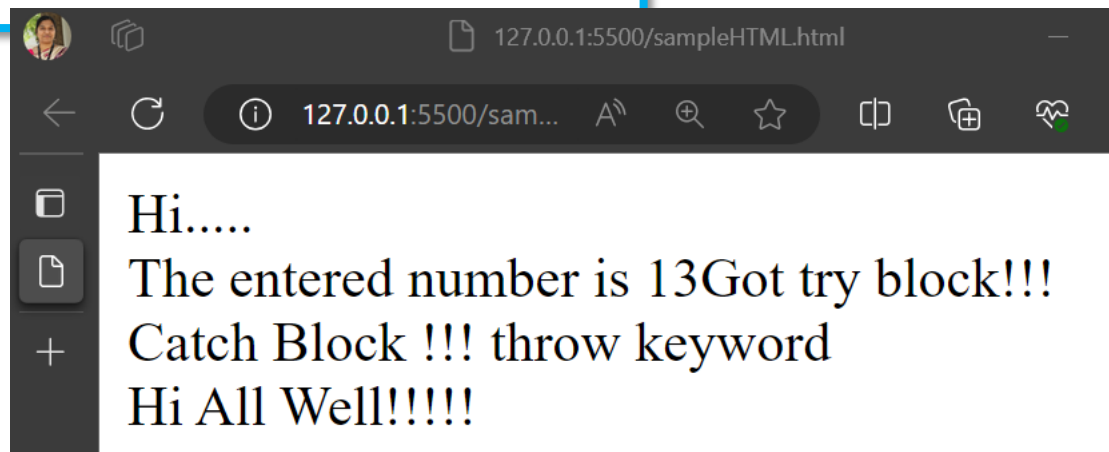
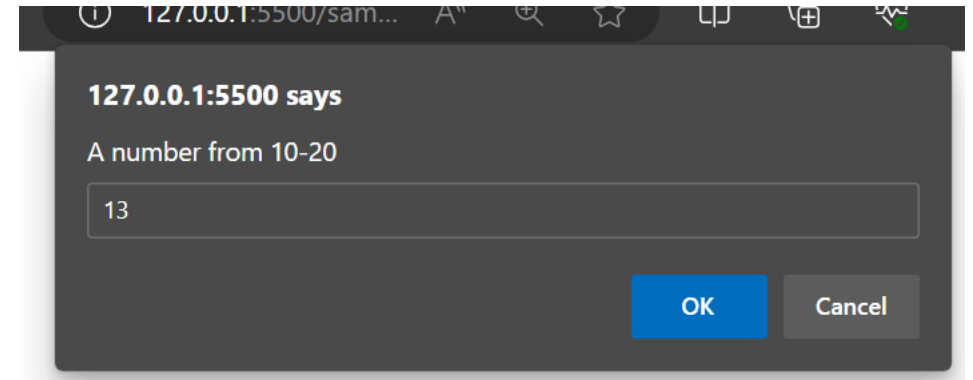
```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        if(no>=10 && no<=20)
        |   document.write("The entered number is ",no)
        else
        |   throw "The number should be between 10-20"
        document.write("Got try block!!! Catch Block !!! throw keyword")
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```



```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        if(no>=10 && no<=20)
        |   document.write("The entered number is ",no)
        else
        |   throw "The number should be between 10-20"
        document.write("Got try block!!! Catch Block !!! throw keyword")
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!<br>")
    </script>
  </body>
</html>
```



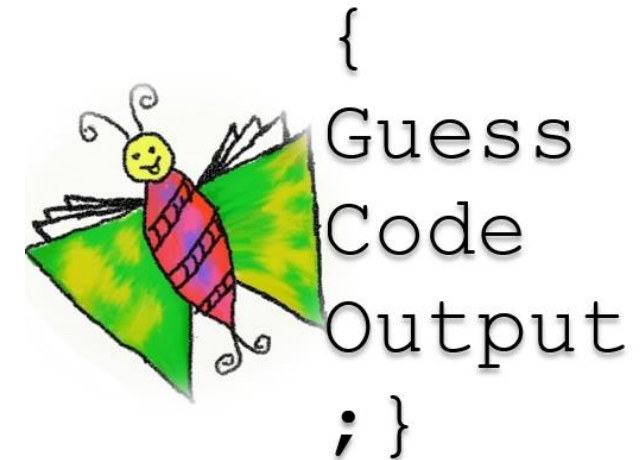
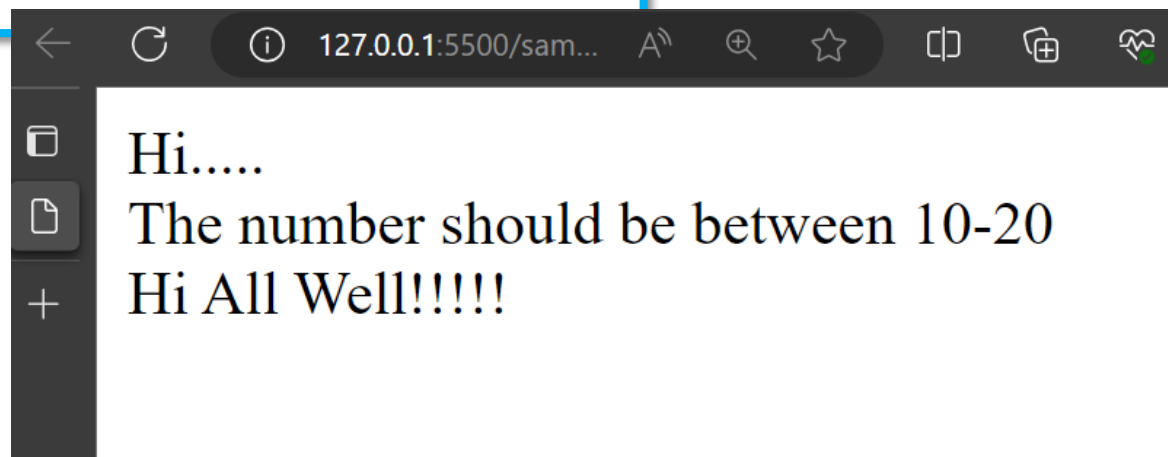
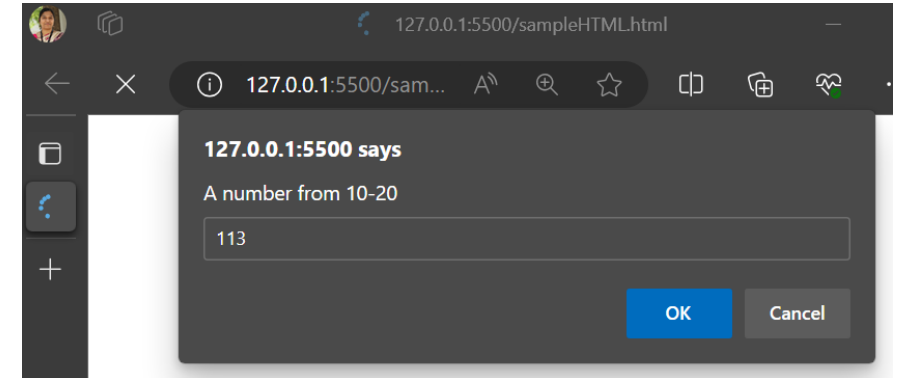
Case 1: Input Number 13



```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        if(no>=10 && no<=20)
        |   document.write("The entered number is ",no)
        else
        |   throw "The number should be between 10-20"
        document.write("Got try block!!! Catch Block !!! throw keyword")
      }
      catch(err)
      {
        document.write(err)
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```



Case 1: Input Number 113



```
<html>
  <!--Exception Handlers by DhiviyaRJ-->
  <body>
    <script>
      document.write("Hi.....<br>")
      try
      {
        let no=prompt("A number from 10-20")
        if(no>=10 && no<=20)
        |   document.write("The entered number is ",no)
        else
        |   throw "The number should be between 10-20"
      }
      catch(err)
      {
        document.write(err)
      }
      finally
      {
        document.write("Got try block!!! Catch Block !!! throw keyword")
      }
      document.write("<br>Hi All Well!!!!!!<br>")
    </script>
  </body>
</html>
```

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
finally {
  Block of code to be executed regardless of the try / catch result
}
```



*Thank
you*

