한국외국어대학교
HANKUK UNIVERSITY OF FOREIGN STUDIES

Web programming

Web 프로그래밍

Lecture 09

JavaScript (JS) (Part 4)

Mohsen Ali Alawami, Ph.D.
Assistant Professor
Department of Computer Engineering
College of Engineering
Hankuk University of Foreign Studies (HUFS)

Fall Semester – 2024

Today's schedule

- JS Sets
  - JS Sets Methods
  - JS Task 1

- JS Maps
  - JS Maps Methods
  - JS Task 2

- JS Errors
  - JS Errors - The Error Object

JavaScript Sets

- A JavaScript Set is a collection of unique values.
- Each value can only occur once in a Set.
- The values can be of any type, values, or objects.

How to Create a Set

You can create a JavaScript Set by:

1. Passing an array to new Set()
2. Create an empty set and use add() to add values

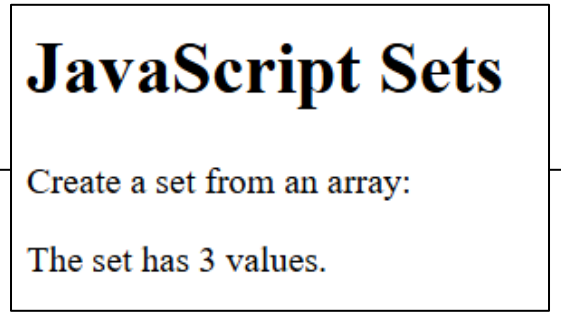
```
// Create a Set
const letters = new Set(["a","b","c"]);
```

```
// Create a Set
const letters = new Set();

// Add Values to the Set
letters.add("a");
letters.add("b");
letters.add("c");
```

JavaScript Sets

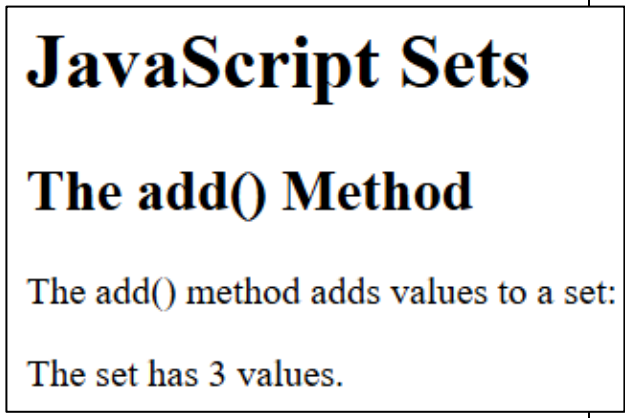
Example 1:

```
<h1>JavaScript Sets</h1>
<p>Create a set from an array:</p>
<p id="demo"></p>

<script>
// Create a Set
const letters = new Set(["a","b","c"]);

// Display set.size
document.getElementById("demo").innerHTML =
"The set has " + letters.size + " values.";
</script>
```



Example 2:

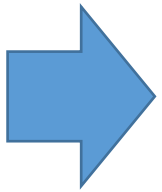
```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>The add() method adds values to a set:</p>
<p id="demo"></p>

<script>
// Create a Set
const letters = new Set();

// Add Values to the Set
letters.add("a");
letters.add("b");
letters.add("c");

// Display the Size
document.getElementById("demo").innerHTML =
"The set has " + letters.size + " values.";
</script>
```

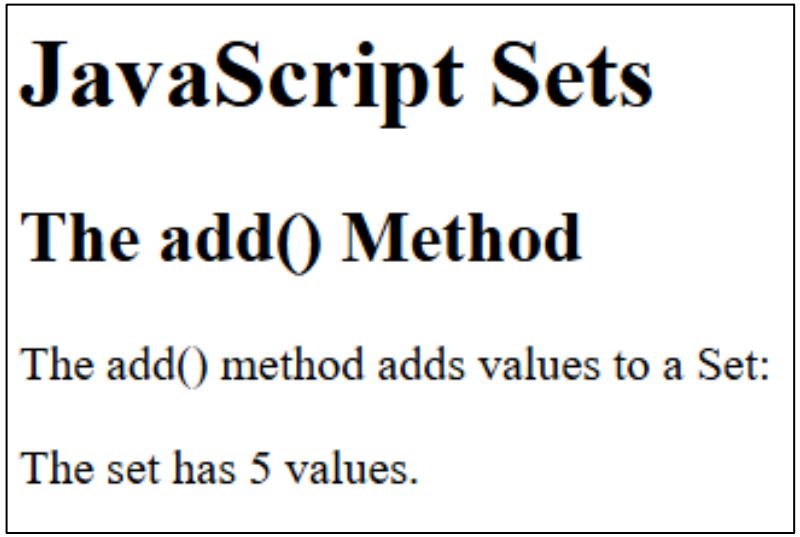


JavaScript Sets

Example 3: Add new elements

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>The add() method adds values to a Set:</p>
<p id="demo"></p>

<script>
// Create a new Set
const letters = new Set(["a","b","c"]);

// Add a new Element
letters.add("d");
letters.add("e");

// Display set.size
document.getElementById("demo").innerHTML =
"The set has " + letters.size + " values.";

</script>
```





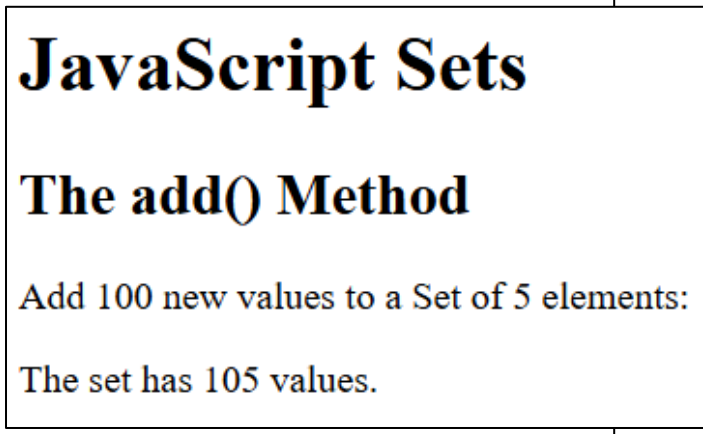
JavaScript Sets

Example 4: Add new elements using for loop

1) add x numbers (x = 100, 500, …)

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>Add 100 new values to a Set of 5 elements:</p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["a","b","c"]);
// Add a new Element
letters.add("d");
letters.add("e");
```

```
// Display set.size
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values.";
</script>
```



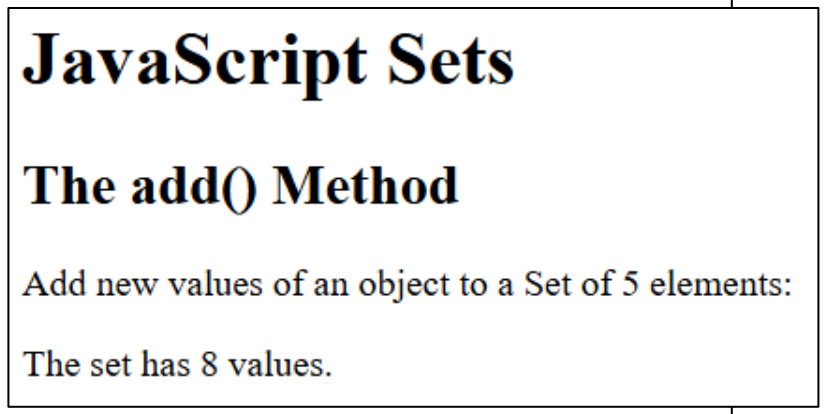
JavaScript Sets

Example 5: Add new elements from an object using for loop

2) Add elements of an Object to a Set. → `const person = {fname:"John", lname:"Doe", age:25};`

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>Add new values of an object to a Set of 5 elements:</p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["a","b","c"]);
// Add a new Element
letters.add("d");
letters.add("e");
// Add object to Set Elements
```

```
// Display set.size
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values.";
</script>
```



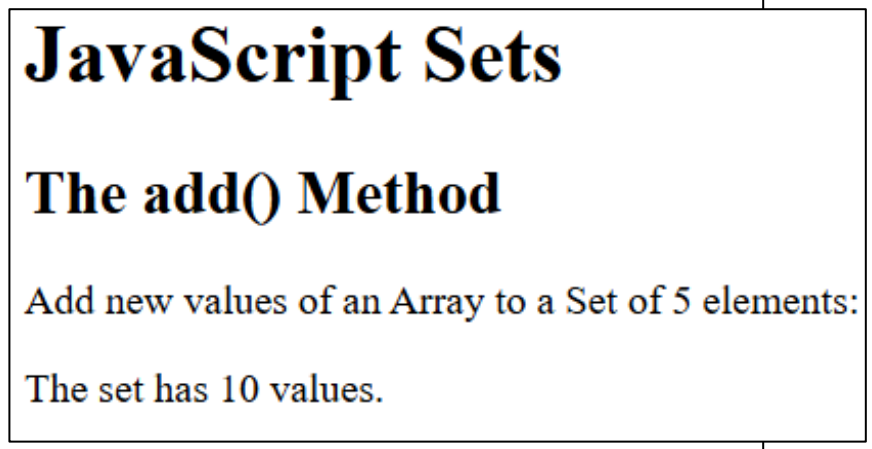
JavaScript Sets

Example 6: Add new elements from an array using for loop

3) Add elements of an Array to a Set. → `const cars = ["BMW", "Volvo", "Kia", "Toyota", "Hyundai"];`

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>Add new values of an Array to a Set of 5 elements:</p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["a","b","c"]);
// Add a new Element
letters.add("d");
letters.add("e");
// Add Array to Set Elements
```

```
// Display set.size
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values.";
</script>
```

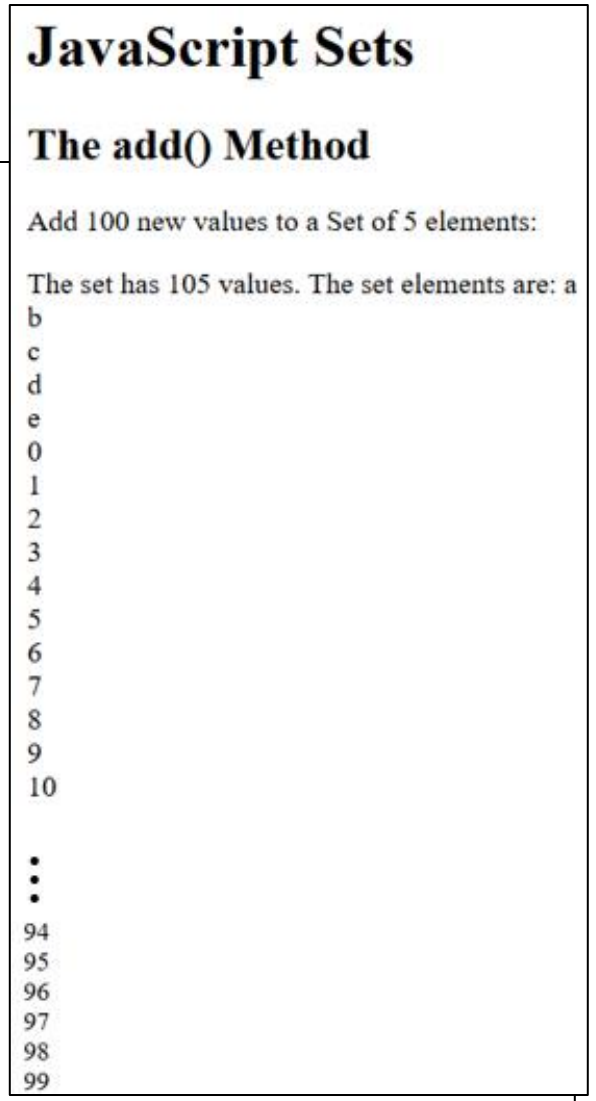


JavaScript Sets – Display the Set elements

Example 7: Display [a,b,c,d,e] and x numbers (x = 100)

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>Add 100 new values to a Set of 5 elements:</p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["a","b","c"]);
// Add a new Element
letters.add("d");
letters.add("e");

// Add 100 numbers to Set Elements
for (let i = 0; i< 100; i++){
letters.add(i);
}
// Display all 105 Elements
```

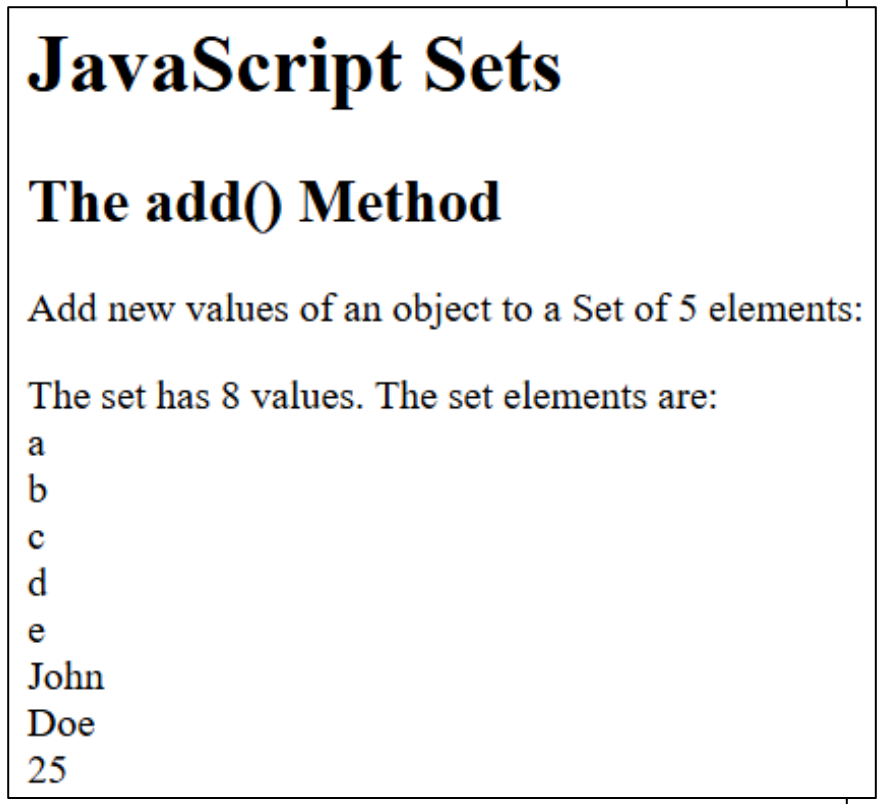
```
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values.  " + "The set elements are: "
+  text;
</script>
```



JavaScript Sets – Display the Set elements

Example 8: Display [a,b,c,d,e] and 3 object elements (John, Doe, 25)

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>Add new values of an object to a Set of 5 elements:</p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["a","b","c"]);
// Add a new Element
letters.add("d");
letters.add("e");

const person = {fname:"John", lname:"Doe", age:25};
// Add all object's Elements
for (let x in person) {
    letters.add(person[x]);
}

// Display all the 8 Elements
[redacted]
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values.  " + "The set elements are: " +
"<br>" +  text;
</script>
```

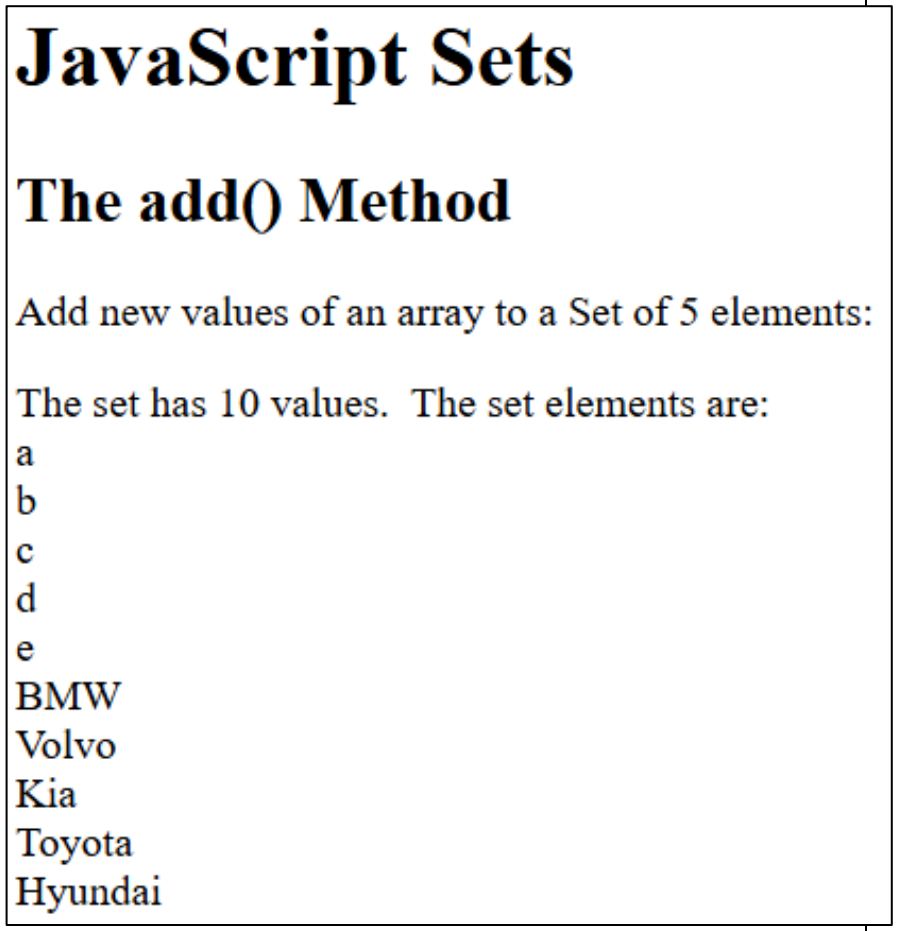


JavaScript Sets – Display the Set elements

Example 9: Display [a,b,c,d,e] and 5 array elements `["BMW", "Volvo", "Kia", "Toyota", "Hyundai"];`

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p>Add new values of an array to a Set of 5 elements:</p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["a","b","c"]);
// Add a new Element
letters.add("d");
letters.add("e");
// Add Array to Set Elements
const cars = ["BMW", "Volvo", "Kia", "Toyota", "Hyundai"];
for (let x of cars) {
letters.add(x);
}
// Display all the 10 Elements
```

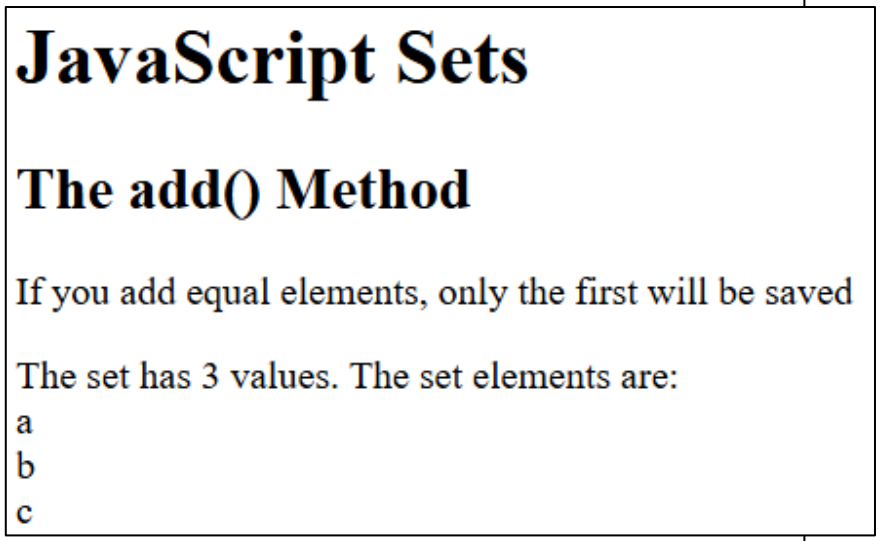
```
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values.  " + "The set elements are: " +
"<br>" +  text;
</script>
```



JavaScript Sets – Adding equal elements

Example 10: If you add equal elements, only one of them will be saved

```
<h1>JavaScript Sets</h1>
<h2>The add() Method</h2>
<p id="demo"></p>
<script>
// Create a Set
const letters = new Set();
// Add Values to the Set
letters.add("a");
letters.add("b");
letters.add("b");
letters.add("c");
letters.add("c");
letters.add("c");
letters.add("c");
letters.add("c");
letters.add("c");
// Display all Elements
let text = "";
for (const x of letters) {
text += x + "<br>";
}
document.getElementById("demo").innerHTML = "The set has " + letters.size + " values. " + "The set elements are: "
+ "<br>" + text;;
</script>
```



JavaScript Sets – The values() and Keys() methods

The values() method returns an Iterator object with the values in a Set

The keys() method returns an Iterator object with the values in a Set

Note

A Set has no keys, so `keys()` returns the same as `values()`.

Change in the previous examples and check the output

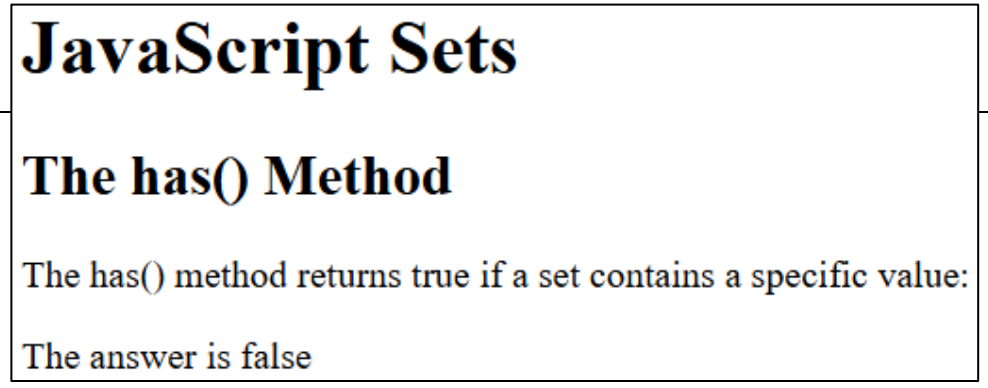
```
// List all Elements
let text = "";
for (const x of letters.values()) {
  text += x + "<br>";
}
```

```
// List all Elements
let text = "";
for (const x of letters.keys()) {
  text += x + "<br>";
}
```

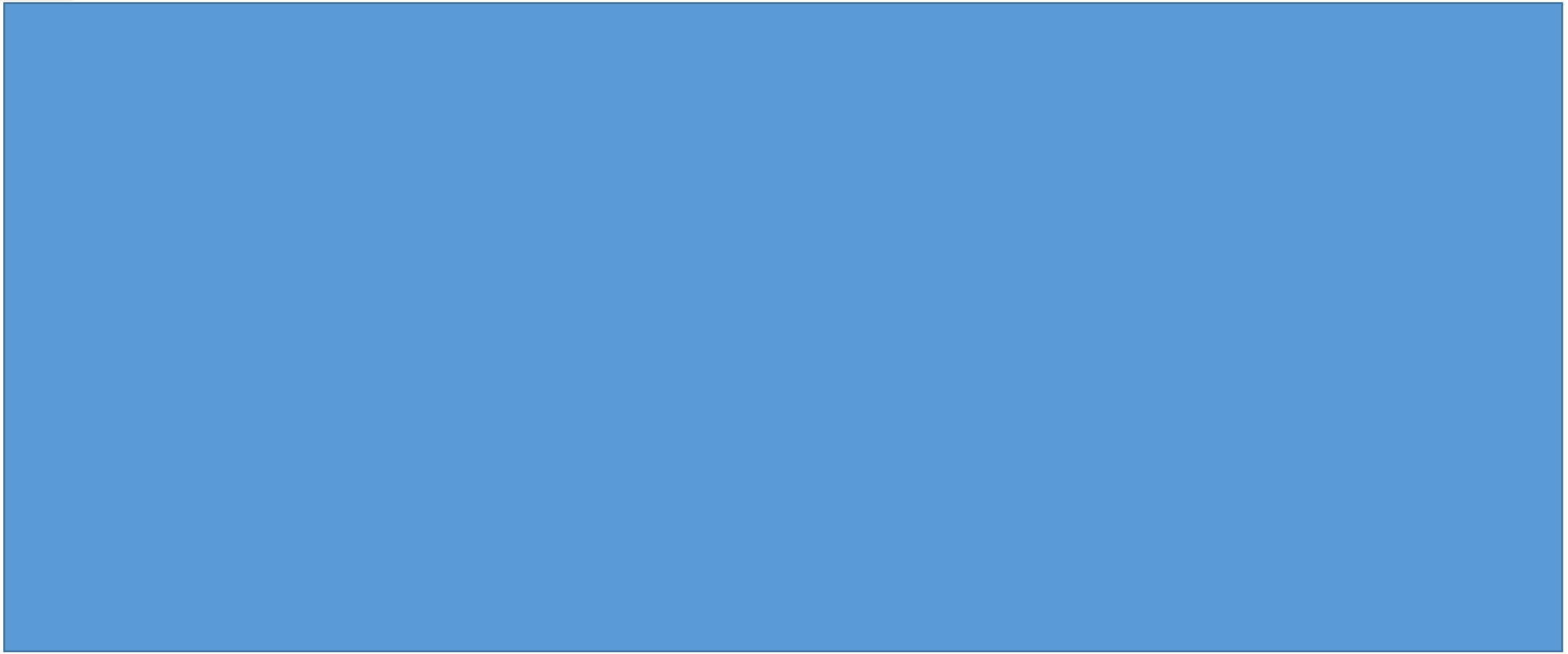
JavaScript Sets – The has() method

Example 11: The has() method returns true if a specified value exists in a set.

```
<h1>JavaScript Sets</h1>
<h2>The has() Method</h2>
<p>The has() method returns true if a set contains a specific value: </p>
<p id="demo"></p>
<script>
// Create a new Set
const letters = new Set(["BMW", "Volvo", "Kia", "Toyota", "Hyundai"]);

// Does the Set contain "Jeep"?
answer = letters.has("Jeep");
document.getElementById("demo").innerHTML = "The answer is " + answer;

</script>
```



JavaScript Sets – Task 1



JavaScript Sets – Task 1 (code)

Today's schedule

- ~~JS Sets~~
  - ~~JS Sets Methods~~
  - ~~JS Task 1~~

- JS Maps
  - JS Maps Methods
  - JS Task 2

- JS Errors
  - JS Errors - The Error Object

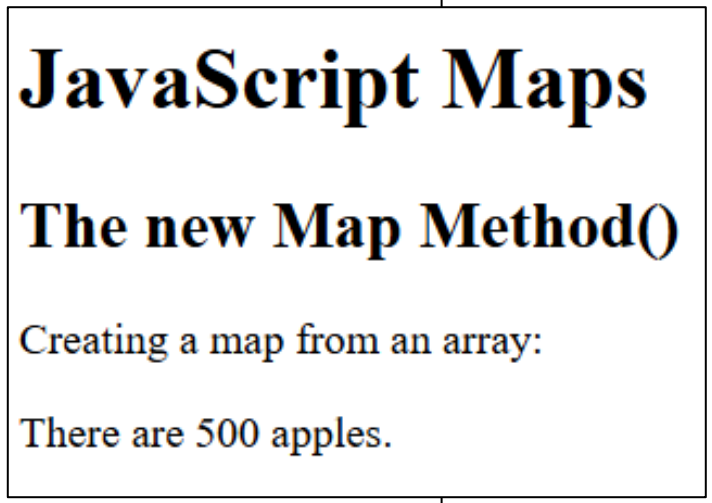
JavaScript Maps

A Map holds key-value pairs where the keys can be any datatype.

- You can create a JavaScript Map by:

  (1) Passing an Array to new Map()

```
<h1>JavaScript Maps</h1>
<h2>The new Map Method()</h2>
<p>Creating a map from an array:</p>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

let numb = fruits.get("apples");
document.getElementById("demo").innerHTML = "There are " + numb + " apples.";
</script>
```



I need "" !!

```
// In this example, the key should be a string
 → try call fruits.get(apples);
```

JavaScript Maps

- You can create a JavaScript Map by:
  (2) Create a Map and use Map.set()

```
<h1>JavaScript Maps</h1>
<h2>The set() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map();

// Set Map Values
fruits.set("apples", 500);
fruits.set("bananas", 300);
fruits.set("oranges", 200);

let numb = fruits.get("apples");
document.getElementById("demo").innerHTML = "There are " + numb + " apples.";

</script>
```

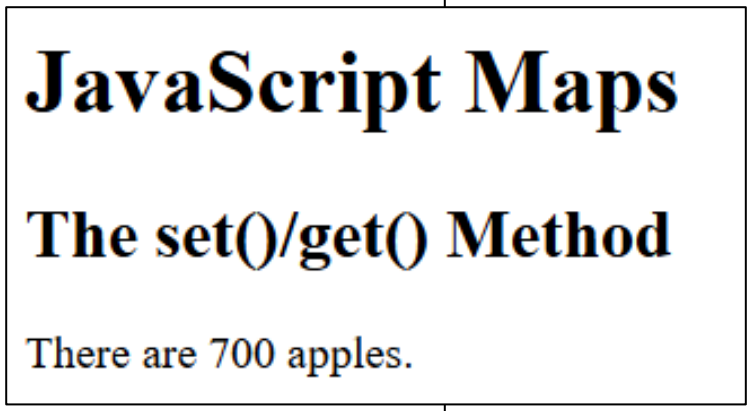

```
// In this example, the key should be a string
 → try call fruits.get(apples);
```

JavaScript Maps – set()/get() Map values

The set() method can also be used to change existing Map values

The get() method gets the value of a key in a Map

```
<h1>JavaScript Maps</h1>
<h2>The set()/get() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

fruits.set("apples", 700);// change apples value to 700
let numb = fruits.get("apples");
document.getElementById("demo").innerHTML = "There are " + numb + " apples.";
</script>
```



JavaScript Maps – Maps are objects

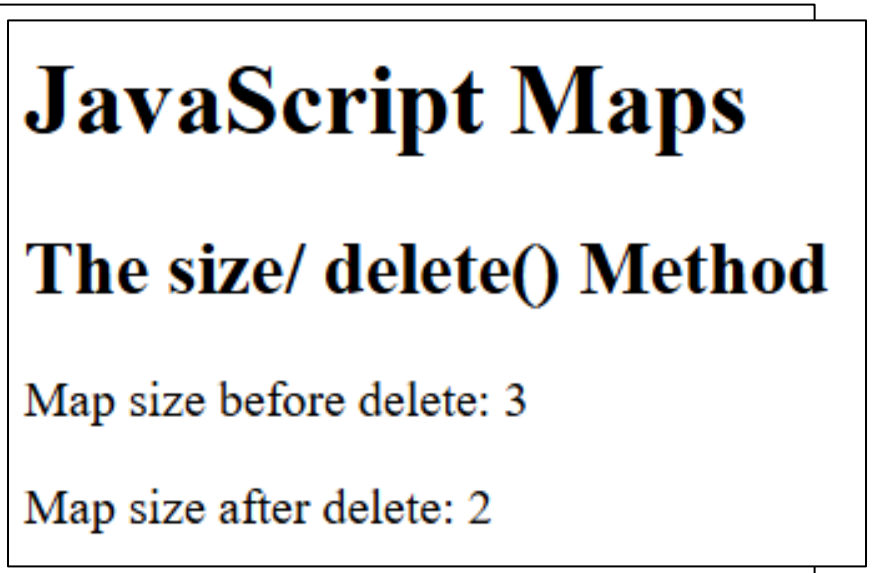
1) typeof returns object

2) instanceof Map returns true

```
<h1>JavaScript Maps</h1>
<h2>typeof Map</h2>
<p id="demo"></p>
<p id="demo1"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

document.getElementById("demo").innerHTML = "Maps is : " + typeof fruits;
document.getElementById("demo1").innerHTML = fruits instanceof Map;
</script>
```

JavaScript Maps – size/ delete() methods

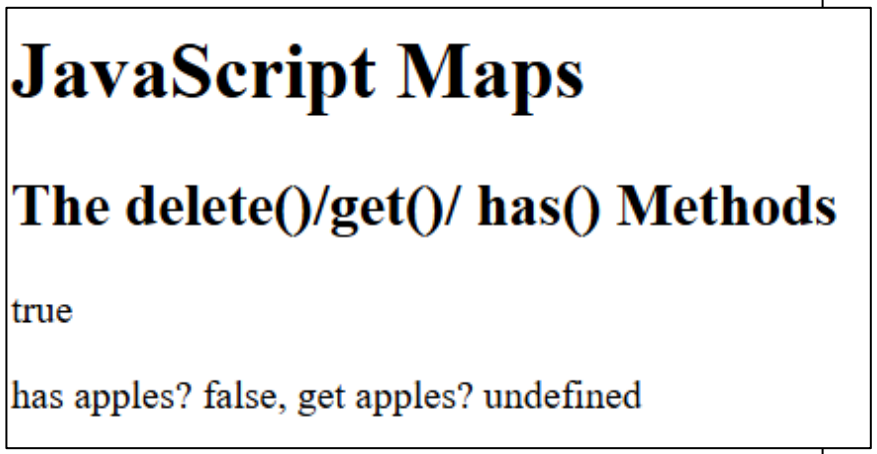
```
<h1>JavaScript Maps</h1>
<h2>The size/ delete() Method</h2>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

// size of map before delete
fruits.size
document.getElementById("demo1").innerHTML =  " Map size before delete:  " + fruits.size;

// Delete an Element
fruits.delete("apples");
document.getElementById("demo2").innerHTML = " Map size after delete:  " + fruits.size;
</script>
```




```
// size() gives an error
```

JavaScript Maps – delete()/ has() / get() methods

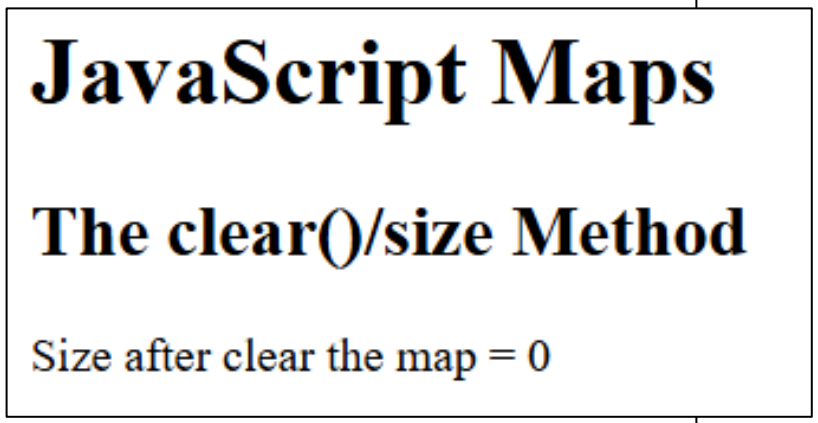
```
<h1>JavaScript Maps</h1>
<h2>The delete()/get()/ has() Methods</h2>
<p id="demo1"></p>
<p id="demo2"></p>
<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

document.getElementById("demo1").innerHTML = fruits.has("apples");

// Delete an Element
fruits.delete("apples");

document.getElementById("demo2").innerHTML = "has apples?  " + fruits.has("apples") +
",  get apples?  "   + fruits.get("apples");
</script>
```



JavaScript Maps – clear()/size methods

The clear() method removes all the elements from a map

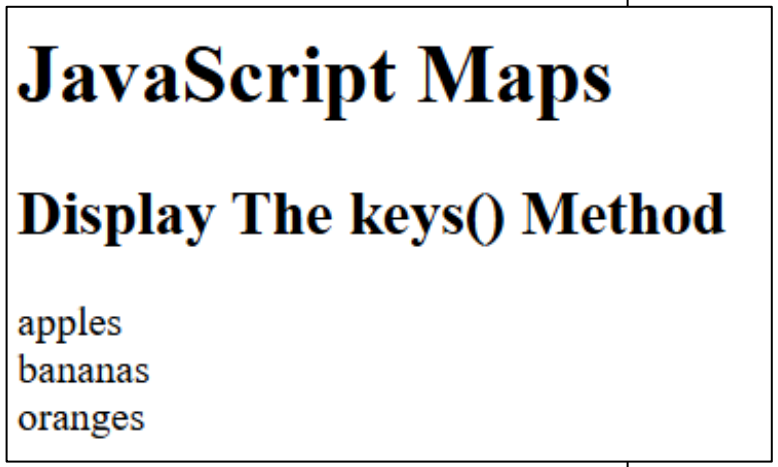
```
<h1>JavaScript Maps</h1>
<h2>The clear()/size Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

// Clear the Map
fruits.clear();
document.getElementById("demo").innerHTML = " Size after clear the map = " + fruits.size;
</script>
```



JavaScript Maps – Display the keys using keys() method

The keys() method returns an iterator object with the keys in a map

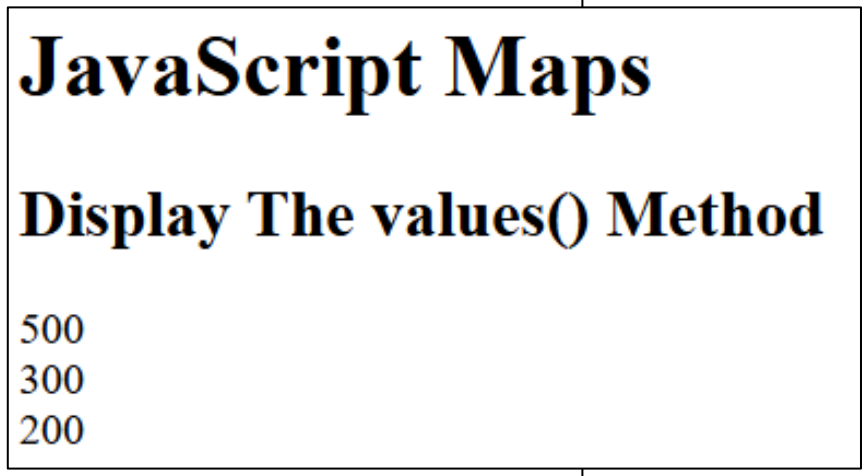
```
<h1>JavaScript Maps</h1>
<h2>Display The keys() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

let text = "";
for (const x of fruits.keys()) {
  text += x + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```



JavaScript Maps – Display the values using values() method

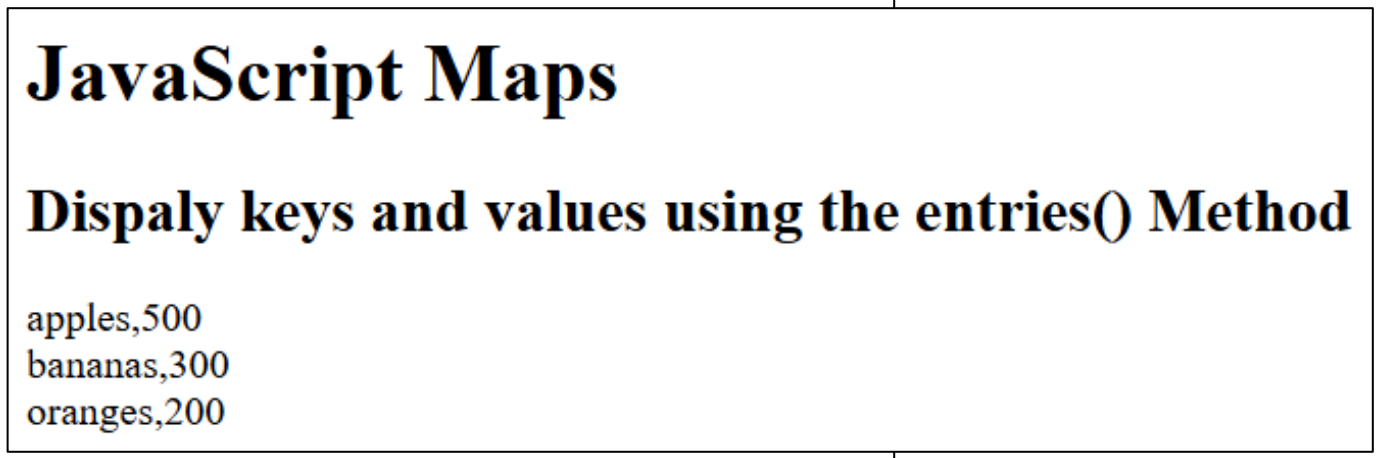
The values() method returns an iterator object with the values in a map

```
<h1>JavaScript Maps</h1>
<h2>Display The values() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

let text = "";
for (const x of fruits.values()) {
  text += x + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```



JavaScript Maps – Display keys/values using entries() method

```
<h1>JavaScript Maps</h1>
<h2>Display keys and values using the entries() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

let text = "";
for (const x of fruits.entries()) {
  text += x + "<br>";
}

document.getElementById("demo").innerHTML = text;
</script>
```



JavaScript Maps – Display keys/values using forEach() method

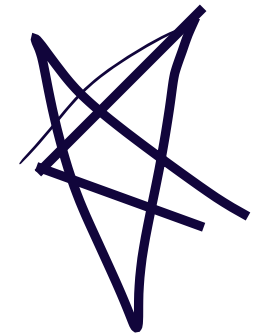
The forEach() method invokes a callback for each key/value pair in a map

```
<h1>JavaScript Maps</h1>
<h2>The forEach() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 3200]
]);

let text = "";
fruits.forEach (function(value, key) {
  text += key + ' = ' + value + "<br>"
})

document.getElementById("demo").innerHTML = text;
</script>
```

JavaScript Maps

The forEach() Method

apples = 500
bananas = 300
oranges = 3200



You can get key or value separately.

Try to replace the locations of key and value

```
fruits.forEach (function(key, value) {
  text += key + ' = ' + value + "<br>"
})
```

Is it the same output?

JavaScript Maps – Display total sum using values() method

```
<h1>JavaScript Maps</h1>
<h2>The values() Method</h2>
<p id="demo"></p>

<script>
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200]
]);

let total = 0;
for (const x of fruits.values()) {
  total += x;
}
document.getElementById("demo").innerHTML = "Total sum of
values = " + total;

</script>
```

JavaScript Maps

The values() Method

Total sum of values = 1000

Try the below cases

```
let total = "    " ;
total += x;
```

```
let total =  0 ;
total += x + "<br>" ;
```

```
// Create a Map
const fruits = new Map([
  ["apples", 500],
  ["bananas", 300],
  ["oranges", 200],
  ["oranges", 100],
  ["oranges", 400],
  ["oranges", 200]
]);
```

Total = 1000 or 1700? Why?

JavaScript Maps – Task 2

JavaScript Maps – Task 2 (code)

Today's schedule

- ~~JS Sets~~
  - ~~JS Sets Methods~~
  - ~~JS Task 1~~

- ~~JS Maps~~
  - ~~JS Maps Methods~~
  - ~~JS Task 2~~

- JS Errors
  - JS Errors - The Error Object

JavaScript Errors

Errors Will Happen!

- When executing JavaScript code, different errors can occur.
- Errors can be coding errors made by the programmer (typos or wrong rules)
- Errors due to wrong input
- Errors due to other unexpected things

Statement to handle errors: 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.

JavaScript Errors - try and catch

The try statement defines a block of code to be tested for errors while it is being executed.

The catch statement defines a block of code to be executed if an error occurs in the try block.

```
try {
  Block of code to try
}
catch(err) {
  Block of code to handle errors
}
```

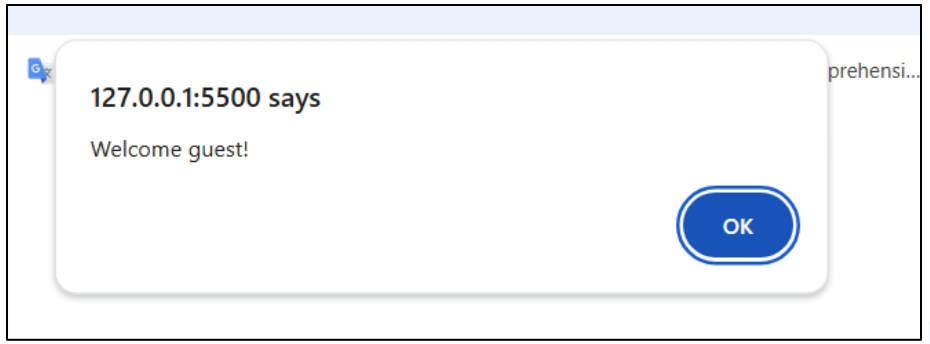
Example 1: When typo (adlert instead of alert)

```
<h2>JavaScript Error Handling</h2>
<p>How to use <b>try-catch</b> to display an error.</p>
<p id="demo"></p>

<script>
try {
  adlert("Welcome guest!"); // typo error (adlert)
}
catch(err) {
  document.getElementById("demo").innerHTML = err.message;
}
</script>
```

JavaScript Error Handling

How to use **try-catch** to display an error.

adlert is not defined



If no error, alert works

JavaScript Errors – try/catch and throw

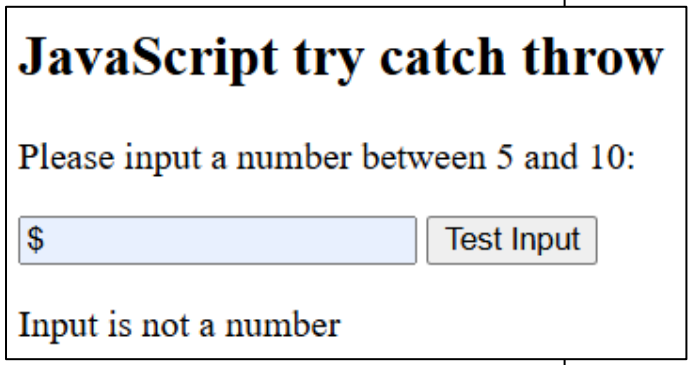
- When an error occurs, JavaScript will normally stop and generate an error message (e.g., adlert is not defined).
- However, the throw statement allows you to create a custom error.
- If you use throw together with try and catch, you can control program flow and generate custom error messages.
- Technically you can throw an exception (throw an error).
- The exception can be a JavaScript String, a Number, or an Object.

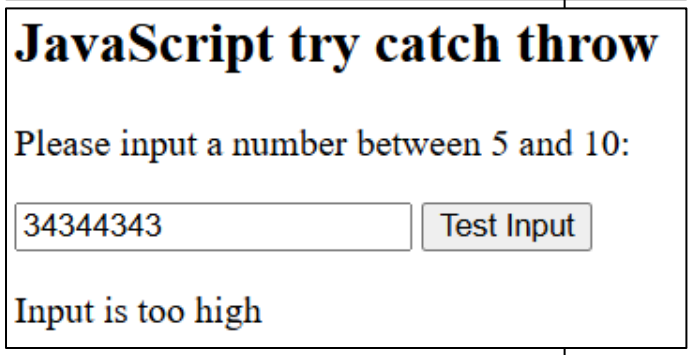
```
throw "Too big";    // throw a text
throw 500;          // throw a number
```

JavaScript Errors – try/catch and throw

User should input a number between 5 and 10, otherwise show the error messages

```
<h2>JavaScript try catch</h2>
<p>Please input a number between 5 and 10:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()" >Test Input</button>
<p id="p1"></p>
<script>
function myFunction() {
  const message = document.getElementById("p1");
  message.innerHTML = ""; // At beginning it is empty message
  let x = document.getElementById("demo").value; // Take input from a user
  try {
    if(x.trim() == "")  throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5)  throw "too low";
    if(x > 10)   throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err; // At the end it is error message
  }
}
</script>
```





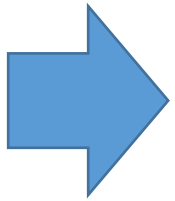
JavaScript Errors – try/catch and finally

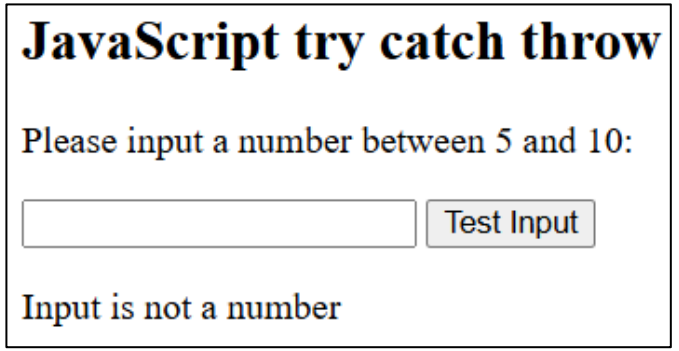
The finally statement lets you execute code, after try and catch, regardless of the result

Syntax

```
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
}
```

```
catch(err) {
  message.innerHTML = "Input is " + err;
}

finally {
  document.getElementById("demo").value = "";
}
```





JavaScript Errors – The Error Object

- JavaScript has a built in error object that provides error information when an error occurs.

Four different errors can be returned by the error object

Error Name	Description
RangeError	A number "out of range" has occurred
ReferenceError	An illegal reference has occurred
SyntaxError	A syntax error has occurred
TypeError	A type error has occurred

JavaScript Errors – (1) Range Error

A RangeError is thrown if you use a number that is outside the range of legal values.

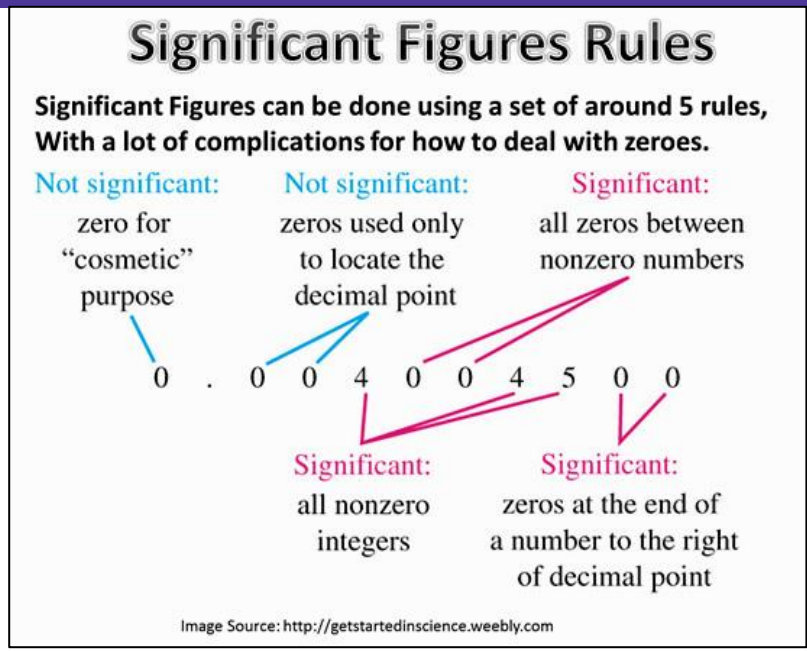
```
<h1>JavaScript Errors</h1>
<h2>The RangeError</h2>
<p>You cannot set the number of significant digits too high:</p>
<p id="demo">

<script>
let num = 5;
try {
  let x = num.toPrecision(500); // 500 is to high
  document.getElementById("demo").innerHTML = x;
}
catch(err) {
  document.getElementById("demo").innerHTML = "Error messgage
=  " + err.name;
}
</script>
```

JavaScript Errors

The RangeError

You cannot set the number of significant digits too high:

Error messgage = RangeError



Number	Number of Significant digits/figures
50000	One
0.008	One
89	Two
340	Two
6700	Two
0.012	Two
1002	Four
4.9210	Five

JavaScript Errors – (2) Reference Error

A ReferenceError is thrown if you use (reference) a variable that has not been declared

```
<h1>JavaScript Errors</h1>
<h2>The ReferenceError</h2>
<p>You cannot use the value of a non-existing variable:</p>
<p id="demo"></p>

<script>
let x = 5;
try {
  x = y + 1; // y has not been declared!
}
catch(err) {
  document.getElementById("demo").innerHTML = "Error messgage =  " + err.name;
}
</script>
```

How to fix this error?

JavaScript Errors – (3) Syntax Error

A SyntaxError is thrown if you try to evaluate code with a syntax error.

```
<h1>JavaScript Errors</h1>
<h2>The SyntaxError</h2>
<p>You cannot evaluate code that contains a syntax error: </p>
<p id="demo"></p>

<script>
try {
  eval("alert('Hello)"); // Missing ' will produce an error

}
catch(err) {
  document.getElementById("demo").innerHTML = "Error messgage =  " + err.name;
}
</script>
```

JavaScript Errors – (4) Type Error

A TypeError is thrown if an operand or argument is incompatible with the type expected by an operator or function.

```
<h1>JavaScript Errors</h1>
<h2>The TypeError</h2>
<p> You cannot convert a number to upper case </p>
<p id="demo"></p>

<script>
let num = 1;
try {
  num.toUpperCase(); //toUpperCase function is expected to convert a string not number

}
catch(err) {
  document.getElementById("demo").innerHTML = "Error messgage =  " + err.name;
}
</script>
```

How to fix this error?

Today's schedule

- ~~JS Sets~~
  - ~~JS Sets Methods~~
  - ~~JS Task 1~~

- ~~JS Maps~~
  - ~~JS Maps Methods~~
  - ~~JS Task 2~~

- ~~JS Errors~~
  - ~~JS Errors – The Error Object~~

Thank You !!

Any Questions!