WEB TECHNOLOGIES

JavaScript –Built in Objects

Prof.Vinay Joshi and Dr.Sarasvathi V
Department of
Computer Science and Engineering

JavaScript – Built-in Objects
Global Object

- When the JavaScript interpreter starts up, there is always a single global object instance created. All other objects are properties of this object.
- Also, all other variables and functions are properties of this global object.
- For client-side JavaScript, this object is the instance **window** of the constructor Window.
- . Since everything is a property of the window object, there is a relaxation of the dot notation when referring to properties. Thus, each reference to a variable, function or object is not required to start with "window.“
- Both these lines have the same effect:

```
window.status = "this is a test";

status = "this is a test";
```

JavaScript – Built-in Objects
Global Objects supported

- Math
- Number
- String
- Array
- Date
- window
- document

JavaScript – Built-in Objects
Array

- Arrays are lists of elements indexed by a numerical value starting with 0 to length - 1
- Arrays can be created using
  - The new Array method
    - let arr = new Array(100) – creates an array of 100 elements
    - let arr = new Array(10, 20) – creates an array of 2 elements
  - Literal arrays using square brackets
    - var alist = [1, “ii”, “gamma”, “4”];

JavaScript – Built-in Objects

Array Length

- Array length property can be modified at runtime
- Hence, the length property does not necessarily indicate the number of defined values in the array

```
const arr = [1, 2];
console.log(arr);
// [ 1, 2 ]

arr.length = 7; // set array length to 5 while currently 2.
console.log(arr);
// [ 1, 2, <5 empty items> ]

for (i in arr)
   console.log(typeof i + i));
// string 0
// string 1
```

Array Elements Can Be Objects

JavaScript variables can be objects. Arrays are special kinds of objects. We can have objects in an Array. We can have functions in an Array. We can have arrays in an Array.

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

JavaScript – Built-in Objects
Array Methods

- push – Add to the end
- pop – Remove from the end
- shift – Remove from the front
- unshift – Add to the front
- join – return a string with array elements
- indexOf – return the index of array
- sort – sort an array in ascending order by default
- concat – concatenate two arrays
- slice – returns a subset if of the array

JavaScript Arrays - Methods

Adding Array Elements

Both methods adds a new element (WT) to Courses

var courses=['HTML' , 'CSS', 'Javascript', 'React']
courses.push('WT')
Or
courses[Courses.length] = “WT”

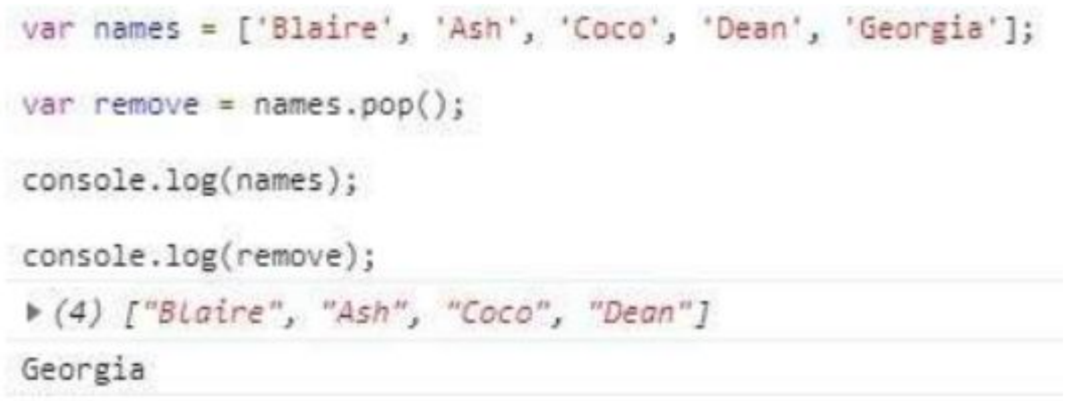
JavaScript Arrays method

1) Converting Arrays to Strings
const array1 = [1, 2, 'a', '1a’];
console.log(array1.toString());   // expected output: "1,2,a,1a"
2) The join() method also joins all array elements into a string.
const elements = ['Fire', 'Air', 'Water’];
console.log(elements.join());// expected output: "Fire,Air,Water“

3) **Popping and Pushing**

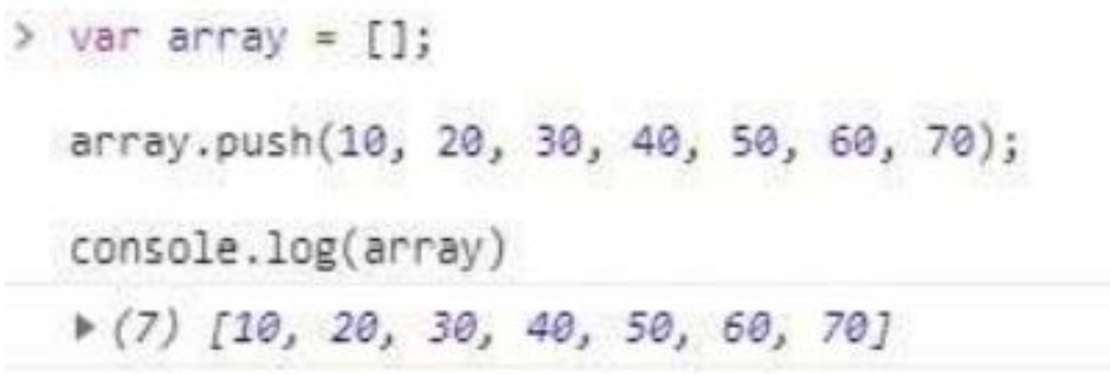
The pop() method removes the last element from an array:

Eg1:



```
var names = ['Blaire', 'Ash', 'Coco', 'Dean', 'Georgia'];
var remove = names.pop();
console.log(names);
console.log(remove);
▸ (4) ["Blaire", "Ash", "Coco", "Dean"]
Georgia
```

Eg 2:

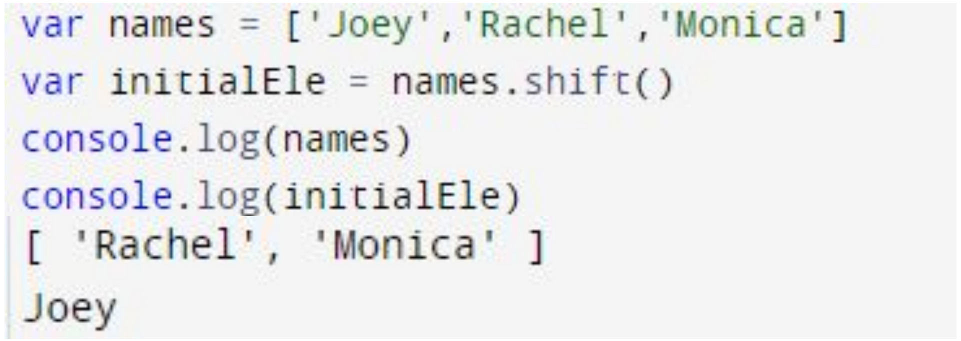


```
> var array = [];
  array.push(10, 20, 30, 40, 50, 60, 70);
  console.log(array)
  ▸ (7) [10, 20, 30, 40, 50, 60, 70]
```

JavaScript Arrays - Methods

4) Shifting Elements

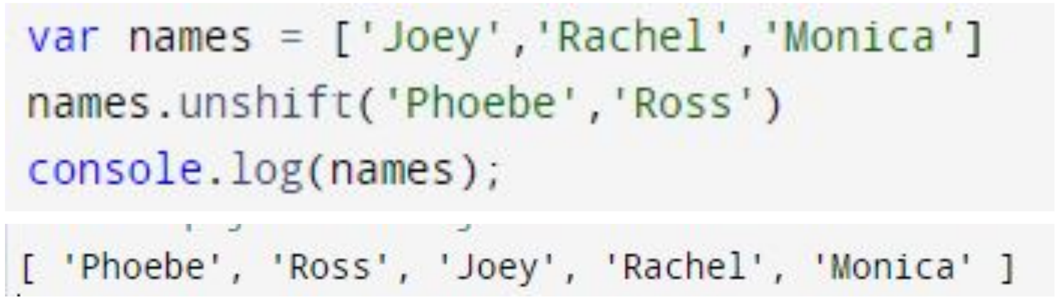
The shift() method removes the first array element and "shifts" all other elements to a lower index. Example:



```
var names = ['Joey','Rachel','Monica']
var initialEle = names.shift()
console.log(names)
console.log(initialEle)
[ 'Rachel', 'Monica' ]
Joey
```

The **unshift()** method adds a new element to an array at the beginning

Example:



```
var names = ['Joey','Rachel','Monica']
names.unshift('Phoebe','Ross')
console.log(names);
```

```
[ 'Phoebe', 'Ross', 'Joey', 'Rachel', 'Monica' ]
```

JavaScript Basics

JavaScript Arrays - Methods

5) **Deleting Elements**

Eg :
```
let arr = [1,2,3,4,5]
delete arr[2]
console.log(arr)
Array(5) [ 1, 2, <1 empty slot>, 4, 5 ]  //replaces with "undefined"
```

6) **Splicing an Array**

The splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place

Eg:
```
arr=[1,2,3,4,5]
let rm = arr.splice(2,1) // remove 1 element from index 2
console.log(rm)
Array [ 3 ] // returns array with removed elements
```

JavaScript Arrays - looping

- The first parameter (2) defines the position where new elements should be added (spliced in).
- The second parameter (2) defines how many elements should be removed.
- The rest of the parameters (“Node", “Express") define the new elements to be added.
- The splice() method returns an array with the deleted items:

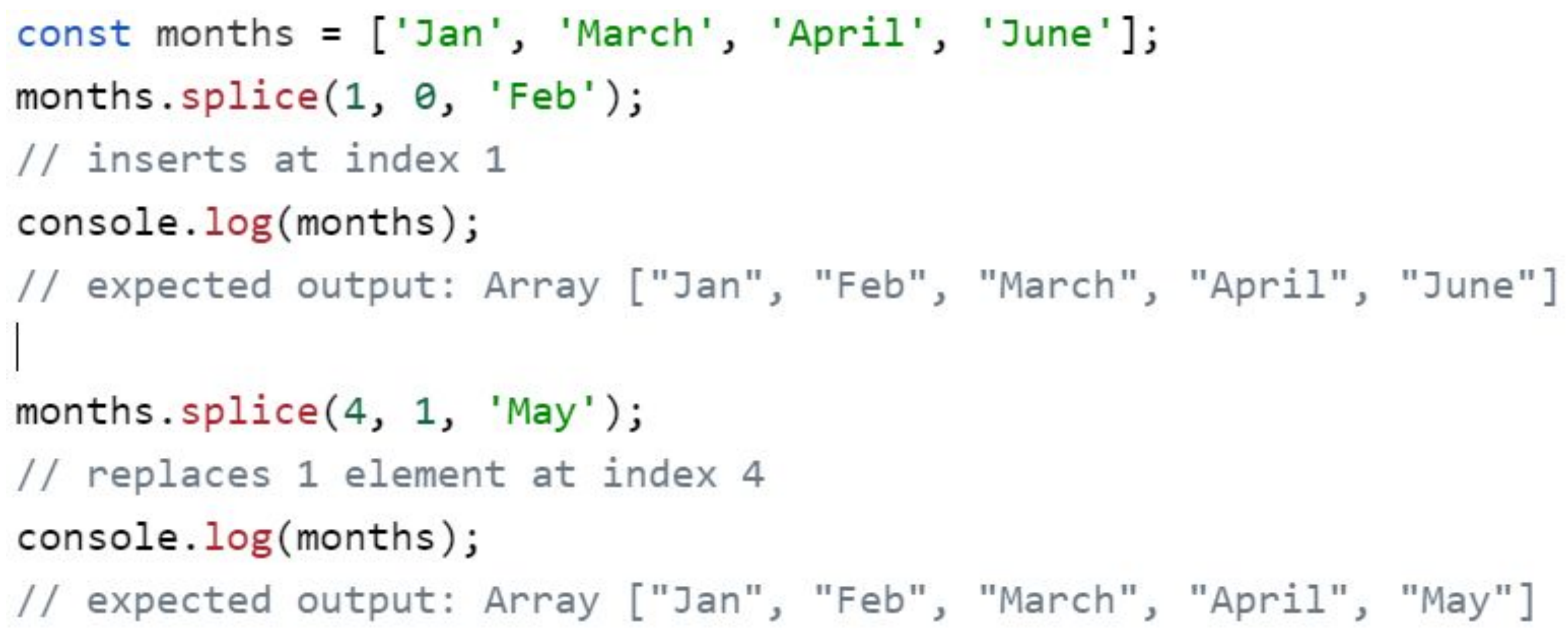
Eg –

var Courses = [“HTML”, “CSS”, “JavaScript”, “React”];
Courses.splice(2, 2, “Node”, “Express”);

7) **Merging (Concatenating) Arrays**
The concat() method creates a new array by merging (concatenating) existing arrays
var arr1 = ["Cecilie", "Lone"];
var myChildren = arr1.concat(["Emil", "Tobias", "Linus"]);

JavaScript Basics

JavaScript Arrays - looping



```
const months = ['Jan', 'March', 'April', 'June'];
months.splice(1, 0, 'Feb');
// inserts at index 1
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "June"]

months.splice(4, 1, 'May');
// replaces 1 element at index 4
console.log(months);
// expected output: Array ["Jan", "Feb", "March", "April", "May"]
```

JavaScript Arrays - looping

8) **Sorting an Array**

The sort() method sorts an array alphabetically.

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();        // Sorts the elements of fruits
```

9) The **reverse()** method reverses the elements in an array.

```
    fruits.reverse();
```

JavaScript – Built-in Objects
Array – Sort

- *arr*.sort(*[compareFunction]*)
- If compareFunction is not specified,array is sorted as strings in ascending order
- Arr = [1, 2, 11, 12, 22]
- Console.log(arr.sort())
- // 1, 11, 12, 2, 22

- compareFunction takes two parameters say a and b and returns
    - 1 if a > b
    - 0 if a = b
    - -1 if a < b
- To reverse the order modify the condition for returning 1, 0 and -1

```
var numbers = [4, 2, 5, 1, 3];
numbers.sort(function(a, b) { return
b - a; });
console.log(numbers);
// [5,4,3,2,1]
```

Array.sort- Compare function

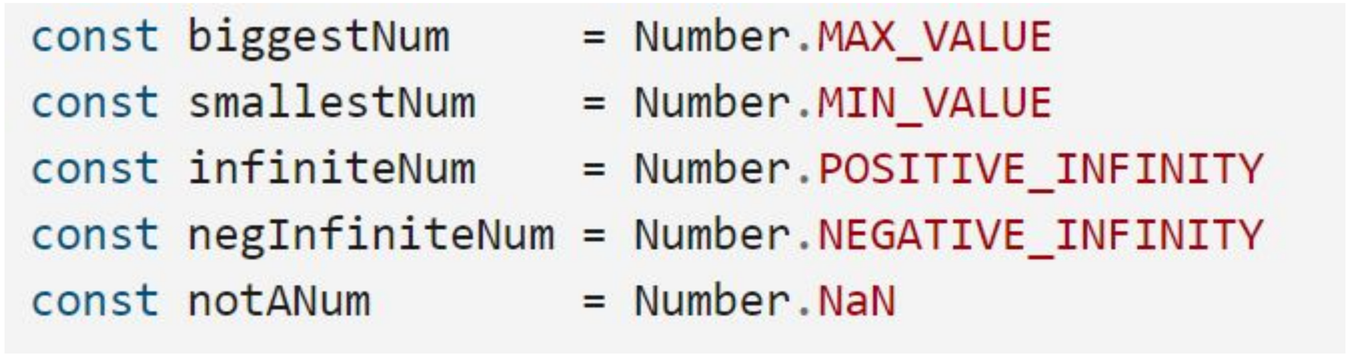
- **compareFunction(Optional):** A function that defines an alternative sort order. The function should return a negative, zero, or positive value, depending on the arguments, like:

  function(a, b)

  {return a-b}
- When the sort() method compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.
- Example:
- When comparing 40 and 100, the sort() method calls the compare function(40,100).
- The function calculates 40-100, and returns -60 (a negative value).
- The sort function will sort 40 as a value lower than 100.

JavaScript – Built-in Objects
Number

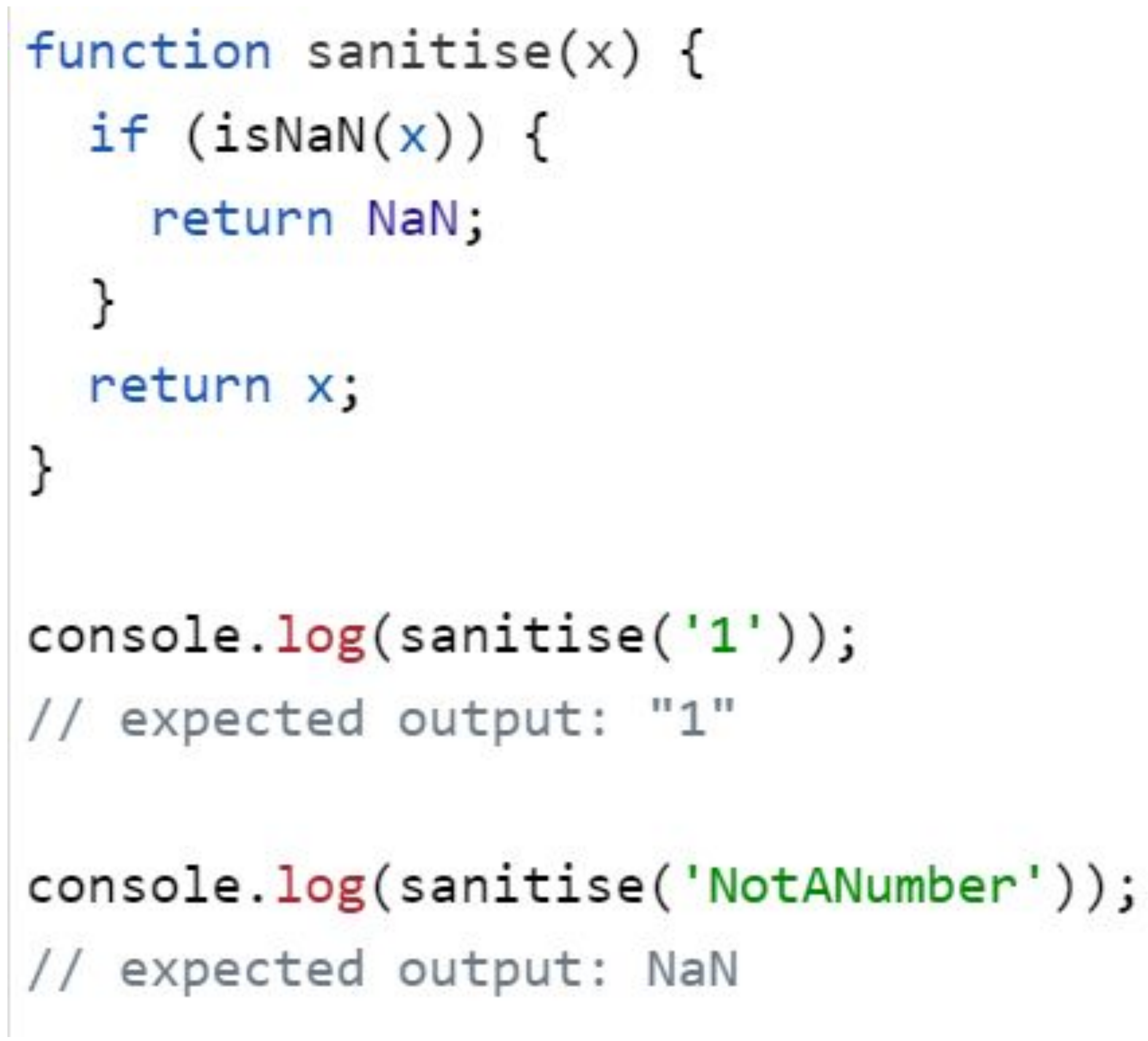
- MAX_VALUE : represents the maximum numeric value representable in JavaScript.
- MIN_VALUE
- NaN – Not a Number
- POSITIVE_INFINITY
- NEGATIVE_INFINITY
- Operations resulting in errors return NaN
  - Use isNaN(a) to test if a is NaN
- toString method converts a number to string



```
const biggestNum     = Number.MAX_VALUE
const smallestNum    = Number.MIN_VALUE
const infiniteNum    = Number.POSITIVE_INFINITY
const negInfiniteNum = Number.NEGATIVE_INFINITY
const notANum        = Number.NaN
```

JavaScript – Built-in Objects

NaN- Not a Number



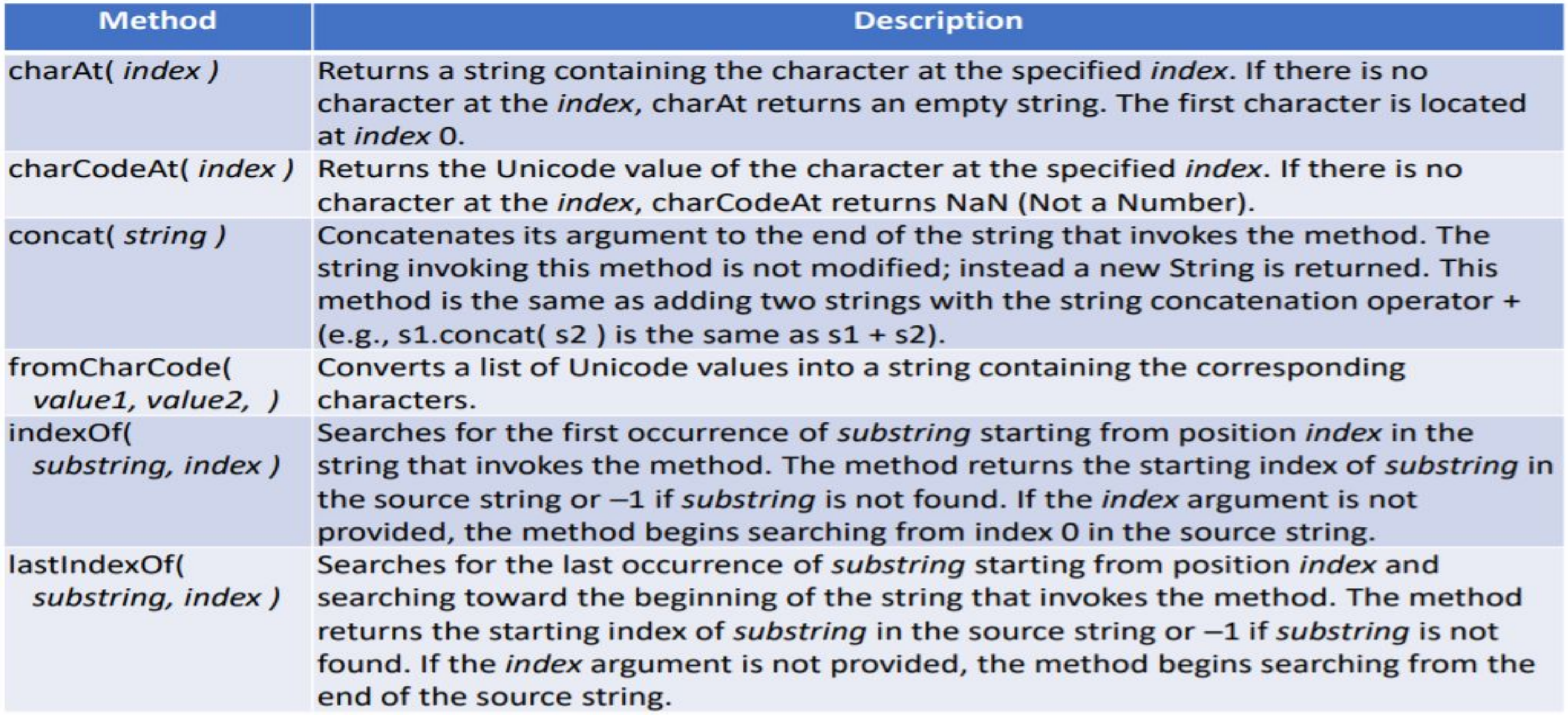
```
function sanitise(x) {
  if (isNaN(x)) {
    return NaN;
  }
  return x;
}

console.log(sanitise('1'));
// expected output: "1"

console.log(sanitise('NotANumber'));
// expected output: NaN
```

JavaScript – Built-in Objects

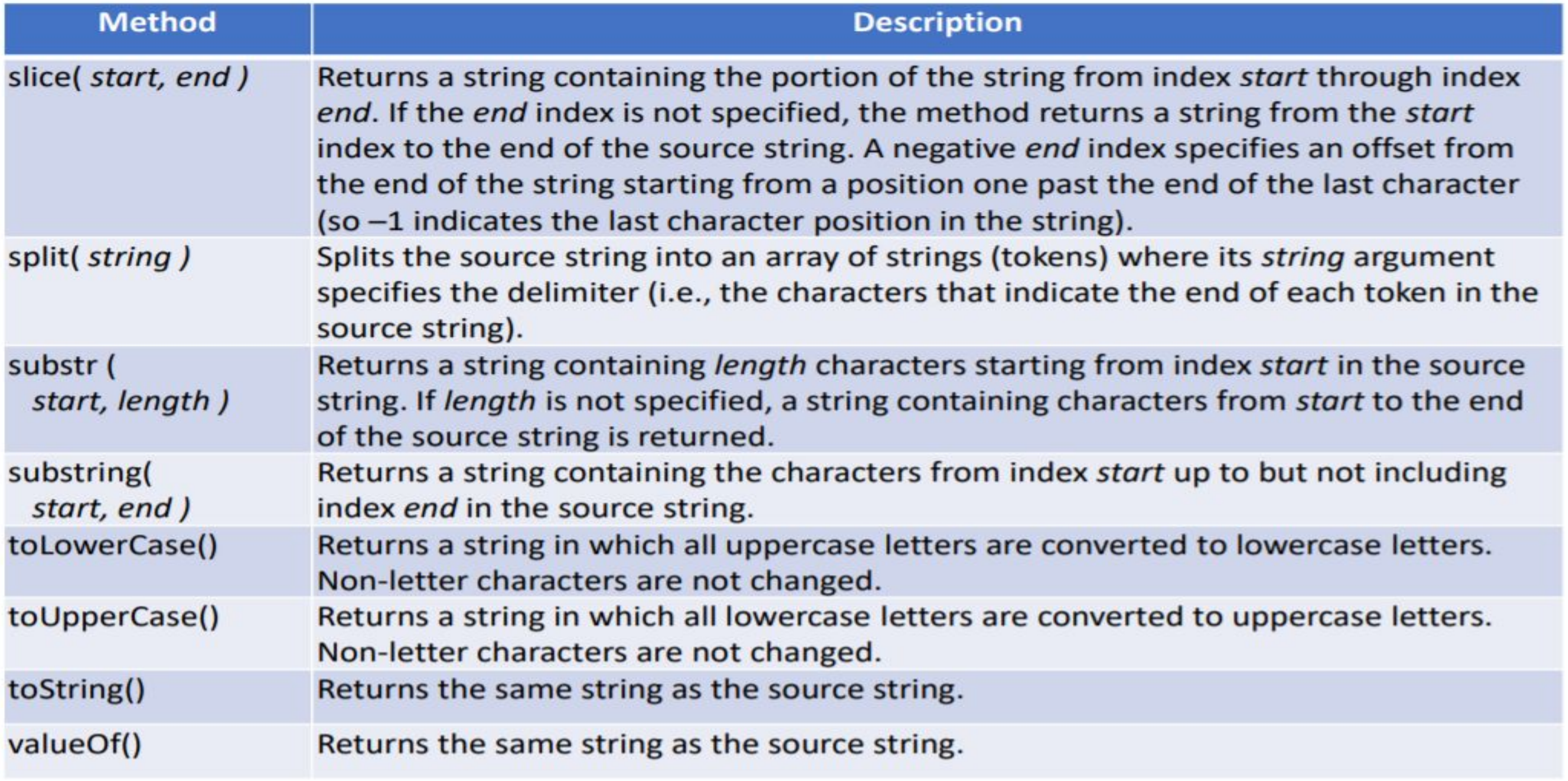
String Object



Method	Description
charAt(*index*)	Returns a string containing the character at the specified *index*. If there is no character at the *index*, charAt returns an empty string. The first character is located at *index* 0.
charCodeAt(*index*)	Returns the Unicode value of the character at the specified *index*. If there is no character at the *index*, charCodeAt returns NaN (Not a Number).
concat(*string*)	Concatenates its argument to the end of the string that invokes the method. The string invoking this method is not modified; instead a new String is returned. This method is the same as adding two strings with the string concatenation operator + (e.g., s1.concat(s2) is the same as s1 + s2).
fromCharCode(*value1, value2,*)	Converts a list of Unicode values into a string containing the corresponding characters.
indexOf(*substring, index*)	Searches for the first occurrence of *substring* starting from position *index* in the string that invokes the method. The method returns the starting index of *substring* in the source string or –1 if *substring* is not found. If the *index* argument is not provided, the method begins searching from index 0 in the source string.
lastIndexOf(*substring, index*)	Searches for the last occurrence of *substring* starting from position *index* and searching toward the beginning of the string that invokes the method. The method returns the starting index of *substring* in the source string or –1 if *substring* is not found. If the *index* argument is not provided, the method begins searching from the end of the source string.

JavaScript – Built-in Objects

String Object



Method	Description
slice(*start, end)*	Returns a string containing the portion of the string from index *start* through index *end*. If the *end* index is not specified, the method returns a string from the *start* index to the end of the source string. A negative *end* index specifies an offset from the end of the string starting from a position one past the end of the last character (so –1 indicates the last character position in the string).
split(*string)*	Splits the source string into an array of strings (tokens) where its *string* argument specifies the delimiter (i.e., the characters that indicate the end of each token in the source string).
substr (*start, length)*	Returns a string containing *length* characters starting from index *start* in the source string. If *length* is not specified, a string containing characters from *start* to the end of the source string is returned.
substring(*start, end)*	Returns a string containing the characters from index *start* up to but not including index *end* in the source string.
toLowerCase()	Returns a string in which all uppercase letters are converted to lowercase letters. Non-letter characters are not changed.
toUpperCase()	Returns a string in which all lowercase letters are converted to uppercase letters. Non-letter characters are not changed.
toString()	Returns the same string as the source string.
valueOf()	Returns the same string as the source string.

JavaScript – Built-in Objects

Date Methods

```
var now= new Date();
```

- This creates a Date object for the time at which it was created (stored as the number of milliseconds since January 1, 1970, UTC - Epoch)

toLocaleString	A string of the Date information
(get/set)Date	The day of the month
(get/set)Month	The month in the range of 0 to 11
(get/set)Day	The day of the week in the range of 0 to 6
(get/set)FullYear	The year
(get/set)Time	The number of milliseconds since January 1, 1970
(get/set)Hours	The number of the hour in the range of 0 to 23
(get/set)Minutes	The number of the minute in the range of 0 to 59
(get/set)Seconds	The number of the second in the range of 0 to 59
(get/set)Milliseconds	The number of the millisecond in the range of 0 to 999

JavaScript – Built-in Objects

Math Methods

Provides **static** mathematical constants and functions

Math.E	Euler’s Constant (Approx. 2.718)
Math.PI	Value of PI (Approx. 3.1416)
Math.SQRT2	Square root of 2 (Approx. 1.414)
Math.abs(x)	Returns the absolute value of x
Math.ceil(x)	Returns the smallest integer greater than or equal to x
Math.floor(x)	Returns the largest integer less than or equal to x.
Math.max([x[, y[, …]]])	Returns the largest of zero or more numbers.
Math.min([x[, y[, …]]])	Returns the smallest of zero or more numbers.
Math.pow(x, y)	Returns base x to the exponent power y (that is, x^y).
Math.random()	Returns a pseudo-random number between 0 and 1.

JavaScript – Built-in Objects

Window – Properties and Methods

- Global object containing global variables and functions declared in the page. For example, *var x;* can also be accessed as *window.x*

location	object containing location details like href, path etc.
history	object containing the browser history
localStorage	object containing a local cache for storing user info
innerHeight, innerWidth	dimensions of the display area of the browser
alert(text)	method to display a dialog box with message
prompt(text,default)	method to seek input from user, returns string
confirm(text)	method to show a confirmation dialog
setInterval, clearInterval	start/stop performing action repeatedly after an interval
setTimeout, clearTimeout	start/stop performing action once after a timeout period

JavaScript – Built-in Objects

setTimeout and setInterval

- **setTimeout** allows us to run a function once after the interval of time.
- The syntax: let timerId = setTimeout(func|code, [delay], [arg1], [arg2], ...) where delay is in milliseconds

Example:
```
function sayHi(phrase, who) {
    alert( phrase + ', ' + who );
    }
    setTimeout(sayHi, 1000, "Hello", "John"); // Hello, John
```

- **setInterval** allows us to run a function repeatedly, starting after the interval of time, then repeating continuously at that interval.
- The setInterval method has the same syntax as setTimeout:
  let timerId = setInterval(func|code, [delay], [arg1], [arg2], ...)
- Example:
```
// repeat with the interval of 2 seconds
    let timerId = setInterval(() => alert('tick'), 2000);
    // after 5 seconds stop
    setTimeout(() => { clearInterval(timerId); alert('stop'); }, 5000);
```

PES UNIVERSITY

CELEBRATING 50 YEARS

THANK YOU

Prof.Vinay Joshi and Dr.Sarasvathi V

Department of Computer Science and Engineering