ETHEREUM 2.0 MASTERY PROGRAM

Instructor: Raja Rizwan Saleem

Module

ONE

JAVASCRIPT CRASH COURSE

Instructor: Raja Rizwan Saleem

SESSION-6

String functions

String functions

During development you will often require to manipulate string and for that string have many functions to do the job

1. toLowerCase()
2. toUpperCase()
3. slice()
4. indexOf()
5. lastIndexOf()
6. charAt()
7. replace()
8. split()

toLowerCase() function

1. toLowerCase function convert string in lowercase letters
2. It is useful when comparing user input

```
var food = "MUHAMMADSHAHIDAKRAM";
var updatedFood = food.toLowerCase();
console.log(food);                  // sANDWiCh
console.log(updatedFood);           // sandwich
```

toUpperCase() function

1. toUpperCase function convert string in uppercase letters
2. It is useful when comparing user input

```
var food = "sANdWiCh";
var updatedFood = food.toUpperCase();
console.log(food);                    // sANDWiCh
console.log(updatedFood);             // SANDWICH
```

slice() function

1. slice() extracts a part of a string and returns the extracted part in a new string. The method takes 2 parameters: the
2. start position, and the end position (end not included).
   String character count starts from 0 same as Arrays
3. First character at zero, second at 1
4. `var a = "Hello World"; // H=0,e=1,l=2,l=3,o=4,space=5,W=6...`

slice() function

1. This example extract portion from 6th index to 8th index

```
var a = "Hello World"; // 6 to (9-1)
var b = a.slice(6,9);

                                //returns Wor
```

slice() function

1. If we omit second parameter it will return rest of the string

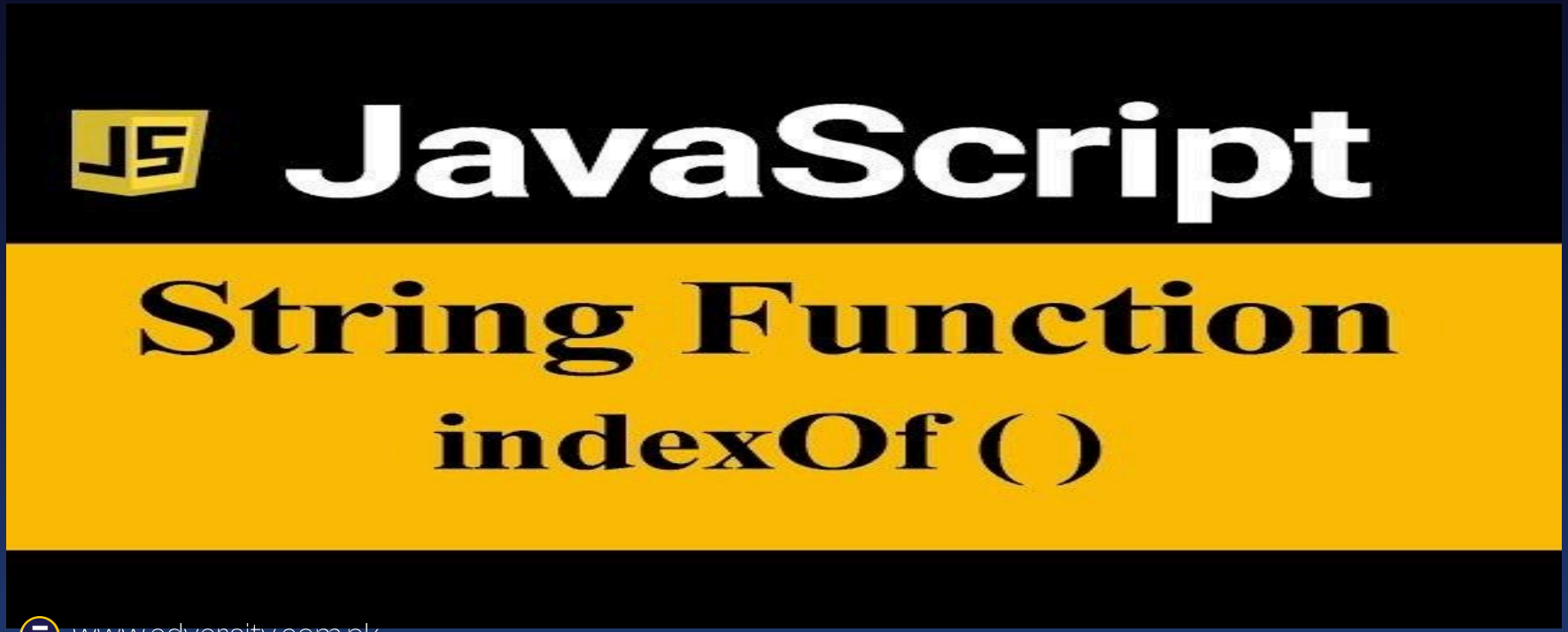
```
var a = "Hello World"; // 6 to end
var b = a.slice(6);

                                //returns World
```

slice() function

1. If a parameter is negative, the position is counted from the end of the string.

```
var a = "Hello World";
//d=-1,l=-2,r=-3,o=-4,W=-5,space=-6,o=-7...
// -5 to -3
var b = a.slice(-5,-2);          //returns Wor
```

indexOf() function



indexOf() function

1. indexOf() returns the index of the first occurrence of a specified text in a string
2. If not found, -1 is returned.

```
var a = "To be or not to be";
var b = a.indexOf("be");//returns 3
```

indexOf() function

1. Second optional argument in indexOf() specify position to begin search in string. If not provided its default to 0

```
var a = "To be or not to be";
var b = a.indexOf("be",       10);      //returns 16
```

lastIndexOf() function

1. lastIndexOf() returns the index of the last occurrence of a specified text in a string.
2. It searches backwards, from the end to the beginning
3. If not found, -1 is returned

```
var a = "To be or not to be";
var b = a.lastIndexOf("be");
```

//returns 16

lastIndexOf() function

1. Second optional argument in lastIndexOf() specify position to to begin search in string. If not provided its default to last index

```
var a = "To be or not to be";
var b = a.lastIndexOf("be",10);          //returns 3
```

2. It will start looking for text ‘be’ from index 10 to backwards

charAt() function

1. slice() function extract the portion of string provided the starting and ending positions charAt() function
2. takes single index input and return character at that index Returns empty string if index does not exists on
3. negative index provided

```
var a = "To be or not to be";


var b = a.charAt(7);          //returns r
```

replace() function

1. The replace() function replaces a specified value with another value in a string
2. The replace() function does not change the string it is called on. It returns a new string.

```
var str = "To be or not to be";
var b = str.replace("be","hello");
// result "To hello or not to be"
```

replace() function

1. To replace case insensitive, use a regular expression with an /i flag (insensitive)

```
var str = "To be or not to be";
var b = str.replace(/to/i,"hello");
// returns "hello be or not to be"
```

replace() function

1. Combine both g and i flog to replace all matches and case insensitive

```
var str = "To be or not to be";
var b = str.replace(/To/gi,"hello");
// returns "hello be or not hello be"
```

split() function

1. The split() function is used to split a string into an array of substrings, and returns the new array.

```
var str = "To be or not to be";
var b = str.split(" ");// split with space
// returns array: ["To", "be", "or", "not",
"to", "be"]
```

split() function (practice)

1. Split can be done with commas, spaces or any character

```
var str = "To,be or|not to,be";
var a = str.split(",");// Split oncommas
var b = str.split(" ");// Split onspaces
var c = str.split("|");// Split onpipe
```

Other String functions (Practice)

There are few more string functions you can learn

1. charCodeAt()
2. concat()
3. endsWith()
4. includes()
5. match()
6. repeat()
7. replace()
8. search()
9. startsWith()
10. substr()
11. substring()
12. trim()

Math

Math class provides many functions that allows you to perform mathematical tasks on numbers

Math.round() function

Math.round() function

1. Math.round(x) returns the value of x rounded to its nearest integer
2. E.g calculating average score will result number with decimal places and you need to round them

```
var average = (15 + 23 + 39) /3 ; // 25.6666 var
roundedAverage = Math.round(average); // 26
console.log(roundedAverage);
```

Math.round() function

```
var a = Math.round(4.7); var     //5
b = Math.round(4.1);             //4
var c = Math.round(4.5);         //5
var d = Math.round(-4.1);        //-4
var e = Math.round(-4.7);        //-5
var f = Math.round(-4.5);        //-4
var g = Math.round(5);           //5
```

Math.ceil() function

1.Math.ceil(x) returns the value of x rounded **up** to its nearest integer

```
var a = Math.ceil(4.7);          //5
var b = Math.ceil(4.1);          //5
var c = Math.ceil(-4.1);         //-4
var d = Math.ceil(-4.7);         //-4
```

Difference between Math.round() & Math.ceil()

Math.ceil() and **Math.round()** methods differ in a way that the former round off a number to its nearest integer in upward direction of rounding(towards the greater value) whereas the latter round off a number to its nearest integer in downward direction of rounding(towards lower value)

```
var a = 9.4889;
var b = Math.round(a);
console.log(b);// will Return 9
```

```
var a = 9.4889;
var b = Math.ceil(a);
console.log(b);// will Return 10
```

Math.floor() function

1.Math.floor(x) returns the value of x rounded **down** to its nearest integer

```
var a = Math.floor(4.7);        //4
var b = Math.floor(4.1);        //4
var c = Math.floor(-4.1);       //-5
var d = Math.floor(-4.7);       //-5
```

Math.random() function

1. Suppose you want to simulate the throw of a die. In the simulation, you want it to randomly come up 1, 2, 3, 4, 5, or 6
2. If you want build a game that will allow user to guess a number
3. Math.random() returns a random number between 0 (inclusive), and 1 (exclusive)
4. Everytime you execute this function it will return random value

Math.random() function

```
var num = Math.random(); // result will be like 0.5251908043871081
```

If you want to generate random number between some range then you have to add some calculations like:

```
var num = Math.random(); var num2 = (num * 6) + 1;

var dice = Math.floor(num2); // 1 to 6
```

Other Math functions

There are few more string functions you can learn

1. Math.pow()
2. Math.sqrt()
3. Math.abs()
4. Math.sin()
5. Math.cos()
6. Math.min()
7. Math.max()
8. Math.exp()
9. Math.log()

Controlling the length of decimals

1. In arithmetic operation you may face numbers with many decimal place

```
var average = (15 + 23 + 39) /3 ; // 25.666666666
```

2. To limit decimal places to specified number you can call toFixed() function on number and round last digit

```
var avg = average.toFixed(3);// returns 25.667
```

Date Object

Date

1. In JavaScript, you might have to create a website with a calendar, a train schedule, or an interface to set up appointments.
2. These applications need to show relevant times based on the user's current timezone
3. You might need to use JavaScript to generate a report at a certain time every day

Date

1. The Date object is a built-in object in JavaScript that stores the date and time.
2. It provides a number of built-in methods for formatting and managing that data.
3. Date objects are created with new Date().

Date

```
var date = new Date();
console.log(date);
//Tue Dec 19 2023 10:28:31 GMT+0500 (Pakistan Standard Time)
```

This will be created according to the current computer's system settings. It will show complete date with current timezone, if you change the timezone of your computer it will show different date

Tue Dec 19 2023 10:28:31 GMT+0500 (Pakistan Standard Time)

Looking at the output, we have a date string containing the following:

Day of the Week	Month	Day	Year	Hour	Minute	Second	Timezone
Thu	Nov	07	2019	11	44	50	GMT+0500

Creating Date Objects

1. There are 4 ways to create a new date object

```
new Date()
new Date(year, month, day, hours, minutes,
seconds, milliseconds)
new Date(milliseconds)
new Date(date string)
```

Creating Date Objects

```
new Date()
new Date(2019, 7, 11, 10, 25, 40, 300);
new Date(1565501140300);
new Date("2019/9/8 10:15:15");
new Date("2019/9/8 10:15:15");
new Date("January 12 2019 10:15:15");
```

Unix time

1. JavaScript, understands the date based on a timestamp derived from Unix time, which is a value consisting of the number of milliseconds that have passed since midnight on January 1st, 1970. We can get the timestamp with the getTime() method.

```
var date = new Date();
date.getTime();       // 1573110702109
```

Epoch time

1. Epoch time, also referred to as zero time, is represented by the date string 01 January, 1970 00:00:00 Universal Time (UTC), and by the 0 timestamp
2. Epoch time was chosen as a standard for computers to measure time by in earlier days of programming

```
var date = new Date(0);
console.log(date);
//Thu Jan 01 1970 00:00:00 GMT+0000 (UTC)
```

Retrieving the Date Components

1. Once we have a date, we can access all the components of the date with various built-in methods.
2. The methods will return each part of the date relative to the local timezone.
3. Each of these methods starts with get, and will return the relative number.

Date/Time	Method	Range	Example
Year	`getFullYear()`	YYYY	1970
Month	`getMonth()`	0-11	0 = January
Day (of the month)	`getDate()`	1-31	1 = 1st of the month
Day (of the week)	`getDay()`	0-6	0 = Sunday
Hour	`getHours()`	0-23	0 = midnight
Minute	`getMinutes()`	0-59	
Second	`getSeconds()`	0-59	
Millisecond	`getMilliseconds()`	0-999	
Timestamp	`getTime()`	Milliseconds since Epoch time	

Modifying the Date

1. For all the **get** methods that we learned, there is a corresponding **set** method.
2. Where **get** is used to retrieve a specific component from a date, **set** is used to modify components of a date

Modifying the Date

```
var date = new Date("June 14 2019 10:45:25");
console.log(date);
// Fri Jun 14 2019 10:45:25 GMT+0500 (Pakistan
Standard Time)
date.setFullYear(2017);
console.log(date);
// Wed Jun 14 2017 10:45:25 GMT+0500 (Pakistan
Standard Time)
```

Converting Day of Week to Text

1. getDay() function returns day of week, but it will give number representing day from 0 to 6
2. 0 represent Sunday, 1 represent Monday and so on
3. If you want on convert this value into text representation then you have to do it yourself
4. You create array represent day of week in text format and then map value from getDay() function to array

Converting Day of Week to Text

```
var daysList = ["Sun", "Mon", "Tue", "Wed",
"Thu", "Fri", "Sat"];
var date = new Date("June 14 2019 10:45:25");
var day = date.getDay();// 5

var nameOfDay = daysList[day]; // Fri
```

Calculate Time difference

1. You can calculate time difference between two days using getTime and calculate difference in days
2. getTime returns time in milliseconds so you can find time difference in milliseconds by subtracting out dates
3. Then need find out of milliseconds in a day
   24 hours * 60 minutes * 60 seconds * 1000 milliseconds
4. And divide time difference in milliseconds with milliseconds of a day

Calculate Time difference

```
var d1 = new Date("June 14 2019 10:45:25"); var d2
= new Date("June 28 2019 10:45:25");
var t imeDiff = d2.getTime()      -  d1.getTime();
vardays = timeDiff / (1000        *  60 * 60 * 24);//      14
```

THANK-YOU