Pakistan Blockchain Institute

Pakistan Blockchain Institute

MODULE-1

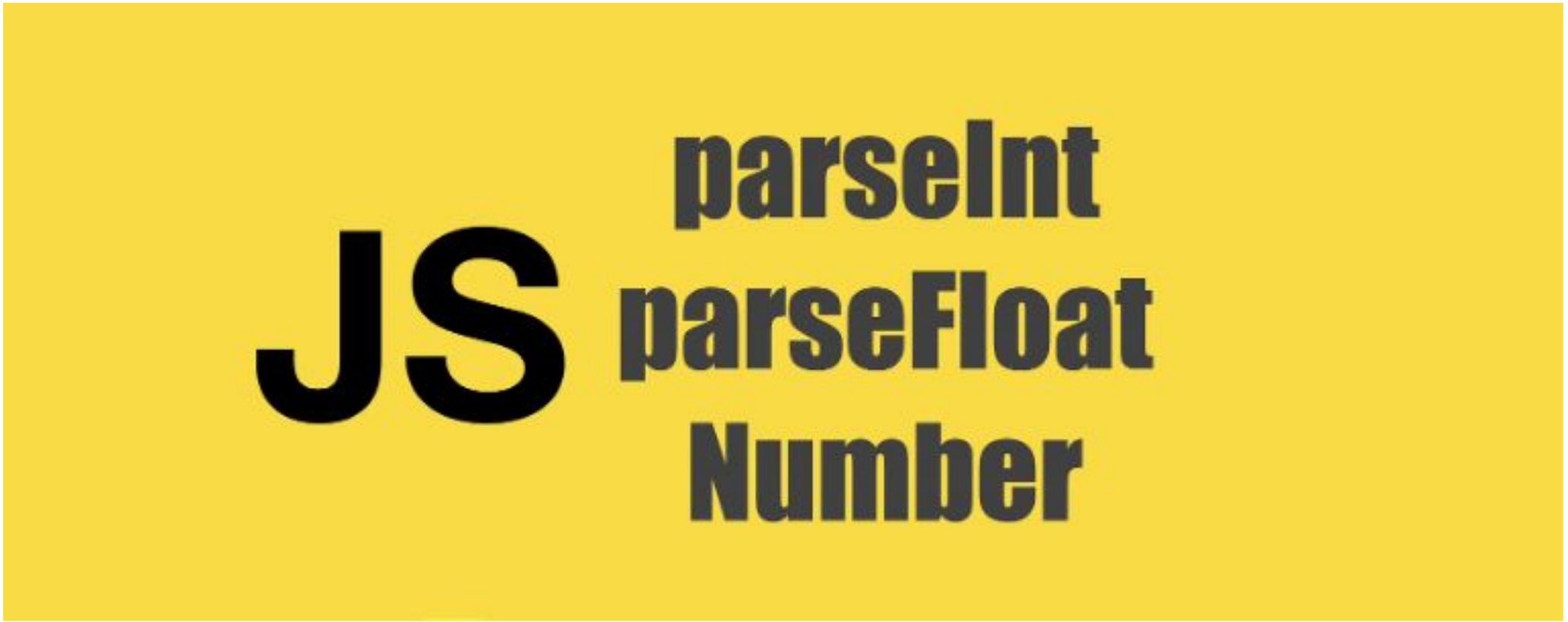
JAVASCRIPT CRASH COURSE

Class-4

Edversity.

Raja Rizwan Saleem
Lead Blockchain Trainer

Convert string to integer



Convert string to integer

- `parseInt` is a built-in JavaScript function that converts a string to an integer.
- It is useful for converting user input or data from other sources into a numeric format.

Convert string to integer

parseFloat is a function that is used to parse a string and extract a floating-point number(a number with decimal places). It will read from the beginning of the string until it encounters a character that is not a valid part of a floating-point number.

parseFloat will always return a floating-point number, even if the input string contains integer values. This is a great choice when you need to parse strings that might contain decimals that you want to preserve

Convert string to integer

parseInt() stops parsing as soon as it encounters a non-numeric character and returns the parsed integer value up to that point. **parseFloat()** continues parsing until it reaches the end of the string or encounters an invalid character, and it returns the parsed floating-point value up to that point

Convert string to integer

Financial Analysts: Those working in the finance sector, such as financial analysts and accountants who require precise calculations involving currency or financial data. This ensures the accuracy when dealing with monetary values.

Engineers: Those working in the engineering sector often work with data that includes real-world measurements. They may use **parseFloat** to ensure accuracy with coordinates, angles, or simulations.

Convert string to integer

1. If you are sure that string has number, then you have to convert it into number to perform addition
2. We need to use **parseInt** function for conversion

```
var age = prompt("What is your age");//input 5
var num = 4;
var sum = parseInt(age) + num;
console.log(sum); // result, 9
```

Convert string to integer `parseInt`

1. If value is not number then it will return NaN

```
var age = prompt("What is your age");//input abc
var num = 4;
var sum = parseInt(ag)   + num;

console.log(sum); // result, NaN
```

Convert string to integer `parseInt` = Example

```
var usdollar=parseInt(prompt("Plz enter an amount in dollars"));

var riyal=parseInt(prompt("Plz enter amount in saudi riyal"));

document.write("<br>");

document.write("Amount of in PKR is :
",((usdollar*177)+(riyal*48)));
```

Convert string to decimal (parseFloat)

1. If your string is number with decimal places then you can use parseFloat for conversion

```
var age = prompt("What is your age");//input 5.5
var num = 4;
var sum = parseFloat(age) + num;
console.log(sum); // result, 9.5
```

KeyPoints and Differences (parseInt vs parseFloat)

Key Points

- **Common Features**:
  - Both parseInt and parseFloat parse strings and convert them to numbers.
  - Both functions ignore leading whitespace.
  - Parsing stops at the first non-numeric character.
  - If the string cannot be converted to a number, both functions return NaN.
- **Differences**:
  - parseInt parses integers and can take an optional radix to specify the number base.
  - parseFloat parses floating-point numbers and does not take a radix parameter.

Convert string to Number [Number()]

1. The Number() function converts the object argument to a number that represents the object's value.
2. If the value cannot be converted to a legal number, NaN is returned.

```
var age = prompt("What is your age");//input 5.5
var num = 4;
var sum = age + Number(num);
console.log(sum); // result, 5.54
```

Convert string to Number (Practice)

1. In case of Number() function you don't have to use separate parseInt or parseFloat function for conversion

```
var a = Number(true);          //returns 1
var b = Number(false);         //returns 0
var c = Number("999");         //returns 999
var d = Number("999 888");     //returns NAN
var e = Number("Hello");       //returns NAN
```

Convert Number to string toFixed()

The toFixed() method converts a number into a string, rounding to a specified number of decimals.

Note: *if the desired number of decimals are higher than the actual number, zeros are added to create the desired decimal length.*

```
var a = 5.56789;
var num = a.toFixed();
document.write(num);
```

Format a number to a specified length toPrecision()

The toPrecision() method formats a number to a specified length. A decimal point and nulls are added (if needed), to create the specified length.

```
let num = 25.3852;
let a = num.toPrecision();
let b = num.toPrecision(2);
let c = num.toPrecision(3);
let d = num.toPrecision(10);
let n = a + "<br>" + b + "<br>" + c + "<br>" + d;
document.write(n);
```

Comparison Operators

Comparison operators

1. Comparison operators are used in logical statements to determine equality or difference between variables or values. They will result in **true** or **false** only

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type

Operator	Description
>	greater than
<	less than
>=	greater than and equal to
<=	less than and equal to

Comparison operators

```
var a = 3;
console.log(a == 3);          //return false
console.log(a === 6);         //return false
console.log(a != 6);          //return true
console.log(a !== 6);         //return true
console.log(a > 6);           //return false
console.log(a < 6);           //return true
console.log(a >= 6);          //return false
console.log(a <= 6);          //return true
```

Comparison operators

1. == and === are used for equality check
2. == does not consider data type of values being compared and
3. == tries to convert one of the value and compare based on that
4. === also check datatype of the values and datatype of value on both side should be same otherwise it will return false

Comparison operators

```
var a = 3;
console.log(a == 3);          //return true
console.log(a == "3");        //return true
console.log(a == 6);          //return false


console.log(a === 3);         //return true
console.log(a === "3");       //return false
console.log(a === 6);         //return false
```

Comparison operators

1. Be careful JavaScript is dynamic language and many decision taken at start are still causing confusion
2. Comparing different type of values will result in answer which is not easily understandable
3. E.g: When comparing a string with a number, JavaScript will convert the string to a number while doing the comparison. An empty string converts to 0. A non-numeric string converts to NaN which is always false.

Comparison operators (Class Practice)

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
===	equal value and equal type	x === "5"	false
		x === 5	true
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== "5"	true
		x !== 5	false
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	*true*

Logical Operators

Short-circuit Logical Operators

1. Logical operators are used to determine the logic between variables or values.
2. Logical operators required boolean operands on both side of operator
3. However, the && and || operators actually return the value of one of the specified operands, so if these operators are used with non-Boolean values, they will return a non-Boolean value.

Logical Operators

1. There are three logical operators in JavaScript:
    a. && (AND)
    b. || (OR)
    c. ! (NOT)

&& Logical Operator

1. This logical operator is used with two or more values (operands), and only evaluates to true if all the operands are true. It will return the false value if at least one value is false.

```
alert( true && true );    // true
alert( false && true );   // false
alert( true && false );   // false
alert( false && false ); // false
```

&& Logical Operator

```
var a = 60;
var b = a > 50 && a < 70;

alert(b); // return true
```

```
var c = 80;
var d = c > 50 && c < 70;

alert(d); // return false
```

|| Logical Operator

1. The logical operator || (OR) also is used with two or more values, but it evaluates to true if any of the operands (values) are true, so only evaluates to false if both operands are false.

```
alert( true || true );    // true
alert( false || true );   // true
alert( true || false );   // true
alert( false || false ); // false
```

|| Logical Operator

```
var a = 60;
var b = a < 50 || a > 70;

alert(b); // return false



var c = 80;
var d = c < 50 || c > 70;

alert(d); // return true
```

! Logical NOT

1. Using the ! operator in front of a boolean will convert it to opposite value. It means that a true value will return false, and a false will return true. This method is known as negation:

```
alert( !true ); // false
alert( !false ); // true
```

! Logical NOT

```
var   a  =  60;
var   b  =  !(a  <  50);
alert(b);                  //   return   true


var   c  =  80;
var   d  =  !(c  >  50);
alert(d);                  //   return   false
```

! Logical NOT

1. Using the ! operator in front of a other value will convert it to a Boolean and return an opposite value.

```
alert( !1 ); // false
alert( !0); // true
```

2. Here 1 will be converted to boolean first and it will be true and because of ! operator opposite will return
3. Same for 0, it will be converted to false and ! will negate it to true

!! Double NOT

1. Using the ! operator twice in front of a value will convert it to a Boolean and negate it twice, useful when you want to convert value to boolean

```
alert( !!1 ); //
true   alert( !!0);
// true
```

2. Here 1 will be converted to boolean first and it will be true and because of first ! operator it will convert to false and then because of second ! operator it will be

Why they are called short-circuit

1. && and || operator stops evaluation of expression once they find desired value
2. && stops evaluation as soon as it finds false and returns false
3. && returns true if all values are true
4. || stops evaluation as soon as it finds true and returns true
5. || returns false if all values are false

Why they are called short-circuit

```
//returns false, evaluation stops at first value
var a = false && true && false;


//returns false, evaluation stops at last value
var b = true && true && false;

//returns false, evaluation stops at second
value  var c = true && false && true;
```

Why they are called short-circuit

```
//returns true, evaluation stops at second value
var d = false || true || false;

//returns true, evaluation stops at first value
var e = true || true || false;

//returns true, evaluation stops at last value
var f = false || false || true
```

Thanks
End of Module-2 (Class-4)