**JavaScript Generating Output**

## Generating Output in JavaScript

There are certain situations in which you may need to generate output from your JavaScript code. For example, you might want to see the value of variable, or write a message to browser console to help you debug an issue in your running JavaScript code, and so on.

In JavaScript there are several different ways of generating output including writing output to the browser window or browser console, displaying output in dialog boxes, writing output into an HTML element, etc. We'll take a closer look at each of these in the following sections.

## Writing Output to Browser Console

You can easily outputs a message or writes data to the browser console using the console.log() method. This is a simple, but very powerful method for generating detailed output. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=write-into-the-browser-console)

// Printing a simple text message

console.log("Hello World!"); // Prints: Hello World!

// Printing a variable value

let x = 10;

let y = 20;

let sum = x + y;

console.log(sum); // Prints: 30

**Tip:** To access your web browser's console, first press F12 key on the keyboard to open the developer tools then click on the console tab. It looks something like the [screenshot here](https://www.tutorialrepublic.com/lib/images/chrome-browser-console.png).

## Displaying Output in Alert Dialog Boxes

You can also use alert dialog boxes to display the message or output data to the user. An alert dialog box is created using the alert() method. Here's is an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=write-into-an-alert-dialog-box)

// Displaying a simple text message

alert("Hello World!"); // Outputs: Hello World!

// Displaying a variable value

let x = 10;

let y = 20;

let sum = x + y;

alert(sum); // Outputs: 30

## Writing Output to the Browser Window

You can use the document.write() method to write the content to the current document only while that document is being parsed. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=write-into-the-browser-window)

// Printing a simple text message

document.write("Hello World!"); // Prints: Hello World!

// Printing a variable value

let x = 10;

let y = 20;

let sum = x + y;

document.write(sum); // Prints: 30

If you use the document.write() method method after the page has been loaded, it will overwrite all the existing content in that document. Check out the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=problem-with-document-write-method)

<h1>This is a heading</h1>

<p>This is a paragraph of text.</p>

<button type="button" onclick="document.write('Hello World!')">Click Me</button>

## Inserting Output Inside an HTML Element

You can also write or insert output inside an HTML element using the element's innerHTML property. However, before writing the output first we need to [select the element](https://www.tutorialrepublic.com/javascript-tutorial/javascript-dom-selectors.php) using a method such as getElementById(), as demonstrated in the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=write-into-an-html-element)

<p id="greet"></p>

<p id="result"></p>

<script>

// Writing text string inside an element

document.getElementById("greet").innerHTML = "Hello World!";

// Writing a variable value inside an element

let x = 10;

let y = 20;

let sum = x + y;

document.getElementById("result").innerHTML = sum;

</script>

You will learn about manipulating HTML element in detail in [JavaScript DOM manipulation](https://www.tutorialrepublic.com/javascript-tutorial/javascript-dom-manipulation.php) chapter.

# JavaScript Data Types

## Data Types in JavaScript

Data types basically specify what kind of data can be stored and manipulated within a program.

There are six basic data types in JavaScript which can be divided into three main categories: primitive (or primary), composite (or reference), and special data types. String, Number, and Boolean are primitive data types. Object, Array, and Function (which are all types of objects) are composite data types. Whereas Undefined and Null are special data types.

Primitive data types can hold only one value at a time, whereas composite data types can hold collections of values and more complex entities. Let's discuss each one of them in detail.

## The String Data Type

The string data type is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters, as shown below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=string-data-type)

let a = 'Hi there!'; // using single quotes

let b = "Hi there!"; // using double quotes

You can include quotes inside the string as long as they don't match the enclosing quotes.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=include-quotes-inside-the-string)

let a = "Let's have a cup of coffee."; // single quote inside double quotes

let b = 'He said "Hello" and left.'; // double quotes inside single quotes

let c = 'We\'ll never give up.'; // escaping single quote with backslash

You will learn more about the strings in [JavaScript strings](https://www.tutorialrepublic.com/javascript-tutorial/javascript-strings.php) chapter.

## The Number Data Type

The number data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. 1.5e-4 (equivalent to 1.5x10-4).

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=number-data-type)

let a = 25; // integer

let b = 80.5; // floating-point number

let c = 4.25e+6; // exponential notation, same as 4.25e6 or 4250000

let d = 4.25e-6; // exponential notation, same as 0.00000425

The Number data type also includes some special values which are: Infinity, -Infinity and NaN. Infinity represents the mathematical Infinity ∞, which is greater than any number. Infinity is the result of dividing a nonzero number by 0, as demonstrated below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=infinity)

alert(16 / 0); // Output: Infinity

alert(-16 / 0); // Output: -Infinity

alert(16 / -0); // Output: -Infinity

While NaN represents a special Not-a-Number value. It is a result of an invalid or an undefined mathematical operation, like taking the square root of -1 or dividing 0 by 0, etc.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=not-a-number)

alert("Some text" / 2); // Output: NaN

alert("Some text" / 2 + 10); // Output: NaN

alert(Math.sqrt(-1)); // Output: NaN

You will learn more about the numbers in [JavaScript numbers](https://www.tutorialrepublic.com/javascript-tutorial/javascript-numbers.php) chapter.

## The Boolean Data Type

The Boolean data type can hold only two values: true or false. It is typically used to store values like yes (true) or no (false), on (true) or off (false), etc. as demonstrated below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=boolean-data-type)

let isReading = true; // yes, I'm reading

let isSleeping = false; // no, I'm not sleeping

Boolean values also come as a result of comparisons in a program. The following example compares two variables and shows the result in an alert dialog box:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=comparisons)

let a = 2, b = 5, c = 10;

alert(b > a) // Output: true

alert(b > c) // Output: false

You will learn more about the comparisons in [JavaScript if/else](https://www.tutorialrepublic.com/javascript-tutorial/javascript-if-else-statements.php) chapter.

## The Undefined Data Type

The undefined data type can only have one value-the special value undefined. If a variable has been declared, but has not been assigned a value, has the value undefined.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=undefined-data-type)

let a;

let b = "Hello World!";

alert(a) // Output: undefined

alert(b) // Output: Hello World!

## The Null Data Type

This is another special data type that can have only one value-the null value. A null value means that there is no value. It is not equivalent to an empty string ("") or 0, it is simply nothing.

A variable can be explicitly emptied of its current contents by assigning it the null value.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=null-data-type)

let a = null;

alert(a); // Output: null

let b = "Hello World!";

alert(b); // Output: Hello World!

b = null;

alert(b) // Output: null

## The Object Data Type

The object is a complex data type that allows you to store collections of data.

An object contains properties, defined as a key-value pair. A property key (name) is always a string, but the value can be any data type, like strings, numbers, booleans, or complex data types like arrays, function and other objects. You'll learn more about objects in upcoming chapters.

The following example will show you the simplest way to create an object in JavaScript.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=object-data-type)

let emptyObject = {};

let person = {"name": "Clark", "surname": "Kent", "age": "36"};

// For better reading

let car = {

"modal": "BMW X3",

"color": "white",

"doors": 5

}

You can omit the quotes around property name if the name is a valid JavaScript name. That means quotes are required around "first-name" but are optional around firstname. So the car object in the above example can also be written as:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=object-properties-names-without-quotes)

let car = {

modal: "BMW X3",

color: "white",

doors: 5

}

You will learn more about the objects in [JavaScript objects](https://www.tutorialrepublic.com/javascript-tutorial/javascript-objects.php) chapter.

## The Array Data Type

An array is a type of object used for storing multiple values in single variable. Each value (also called an element) in an array has a numeric position, known as its index, and it may contain data of any data type-numbers, strings, booleans, functions, objects, and even other arrays. The array index starts from 0, so that the first array element is arr[0] not arr[1].

The simplest way to create an array is by specifying the array elements as a comma-separated list enclosed by square brackets, as shown in the example below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=array-data-type)

let colors = ["Red", "Yellow", "Green", "Orange"];

let cities = ["London", "Paris", "New York"];

alert(colors[0]); // Output: Red

alert(cities[2]); // Output: New York

You will learn more about the arrays in [JavaScript arrays](https://www.tutorialrepublic.com/javascript-tutorial/javascript-arrays.php) chapter.

## The Function Data Type

The function is callable object that executes a block of code. Since functions are objects, so it is possible to assign them to variables, as shown in the example below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=function-data-type)

let greeting = function(){

return "Hello World!";

}

// Check the type of greeting variable

alert(typeof greeting) // Output: function

alert(greeting()); // Output: Hello World!

In fact, functions can be used at any place any other value can be used. Functions can be stored in variables, objects, and arrays. Functions can be passed as arguments to other functions, and functions can be returned from functions. Consider the following function:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=function-passed-as-argument-to-other-function)

function createGreeting(name){

return "Hello, " + name;

}

function displayGreeting(greetingFunction, userName){

return greetingFunction(userName);

}

let result = displayGreeting(createGreeting, "Peter");

alert(result); // Output: Hello, Peter

You will learn more about the functions in [JavaScript functions](https://www.tutorialrepublic.com/javascript-tutorial/javascript-functions.php) chapter.

## The typeof Operator

The typeof operator can be used to find out what type of data a variable or operand contains. It can be used with or without parentheses (typeof(x) or typeof x).

The typeof operator is particularly useful in the situations when you need to process the values of different types differently, but you need to be very careful, because it may produce unexpected result in some cases, as demonstrated in the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=typeof-operator)

// Numbers

typeof 15; // Returns: "number"

typeof 42.7; // Returns: "number"

typeof 2.5e-4; // Returns: "number"

typeof Infinity; // Returns: "number"

typeof NaN; // Returns: "number". Despite being "Not-A-Number"

// Strings

typeof ''; // Returns: "string"

typeof 'hello'; // Returns: "string"

typeof '12'; // Returns: "string". Number within quotes is typeof string

// Booleans

typeof true; // Returns: "boolean"

typeof false; // Returns: "boolean"

// Undefined

typeof undefined; // Returns: "undefined"

typeof undeclaredVariable; // Returns: "undefined"

// Null

typeof Null; // Returns: "object"

// Objects

typeof {name: "John", age: 18}; // Returns: "object"

// Arrays

typeof [1, 2, 4]; // Returns: "object"

// Functions

typeof function(){}; // Returns: "function"

As you can clearly see in the above example when we test the null value using the typeof operator (*line no-22*), it returned "object" instead of "null".

This is a long-standing bug in JavaScript, but since lots of codes on the web written around this behavior, and thus fixing it would create a lot more problem, so idea of fixing this issue was rejected by the committee that design and maintains JavaScript.

**JavaScript Operators**

## What are Operators in JavaScript

Operators are symbols or keywords that tell the JavaScript engine to perform some sort of actions. For example, the addition (+) symbol is an operator that tells JavaScript engine to add two variables or values, while the equal-to (==), greater-than (>) or less-than (<) symbols are the operators that tells JavaScript engine to compare two variables or values, and so on.

The following sections describe the different operators used in JavaScript.

## JavaScript Arithmetic Operators

The arithmetic operators are used to perform common arithmetical operations, such as addition, subtraction, multiplication etc. Here's a complete list of JavaScript's arithmetic operators:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Result** |
| + | Addition | x + y | Sum of x and y |
| - | Subtraction | x - y | Difference of x and y. |
| \* | Multiplication | x \* y | Product of x and y. |
| / | Division | x / y | Quotient of x and y |
| % | Modulus | x % y | Remainder of x divided by y |

The following example will show you these arithmetic operators in action:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=arithmetic-operators)

let x = 10;

let y = 4;

alert(x + y); // 0utputs: 14

alert(x - y); // 0utputs: 6

alert(x \* y); // 0utputs: 40

alert(x / y); // 0utputs: 2.5

alert(x % y); // 0utputs: 2

## JavaScript Assignment Operators

The assignment operators are used to assign values to variables.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Is The Same As** |
| = | Assign | x = y | x = y |
| += | Add and assign | x += y | x = x + y |
| -= | Subtract and assign | x -= y | x = x - y |
| \*= | Multiply and assign | x \*= y | x = x \* y |
| /= | Divide and assign quotient | x /= y | x = x / y |
| %= | Divide and assign modulus | x %= y | x = x % y |

The following example will show you these assignment operators in action:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=assignment-operators)

let x; // Declaring Variable

x = 10;

alert(x); // Outputs: 10

x = 20;

x += 30;

alert(x); // Outputs: 50

x = 50;

x -= 20;

alert(x); // Outputs: 30

x = 5;

x \*= 25;

alert(x); // Outputs: 125

x = 50;

x /= 10;

alert(x); // Outputs: 5

x = 100;

x %= 15;

alert(x); // Outputs: 10

## JavaScript String Operators

There are two operators which can also used be for strings.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Result** |
| + | Concatenation | str1 + str2 | Concatenation of str1 and str2 |
| += | Concatenation assignment | str1 += str2 | Appends the str2 to the str1 |

The following example will show you these string operators in action:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=string-operators)

let str1 = "Hello";

let str2 = " World!";

alert(str1 + str2); // Outputs: Hello World!

str1 += str2;

alert(str1); // Outputs: Hello World!

## JavaScript Incrementing and Decrementing Operators

The increment/decrement operators are used to increment/decrement a variable's value.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Name** | **Effect** |
| ++x | Pre-increment | Increments x by one, then returns x |
| x++ | Post-increment | Returns x, then increments x by one |
| --x | Pre-decrement | Decrements x by one, then returns x |
| x-- | Post-decrement | Returns x, then decrements x by one |

The following example will show you how increment and decrement operators actually work:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=increment-decrement-operators)

let x; // Declaring Variable

x = 10;

alert(++x); // Outputs: 11

alert(x); // Outputs: 11

x = 10;

alert(x++); // Outputs: 10

alert(x); // Outputs: 11

x = 10;

alert(--x); // Outputs: 9

alert(x); // Outputs: 9

x = 10;

alert(x--); // Outputs: 10

alert(x); // Outputs: 9

## JavaScript Logical Operators

The logical operators are typically used to combine conditional statements.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| && | And | x && y | True if both x and y are true |
| || | Or | x || y | True if either x or y is true |
| ! | Not | !x | True if x is not true |

The following example will show you how these logical operators actually work:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=logical-operators)

let year = 2018;

// Leap years are divisible by 400 or by 4 but not 100

if((year % 400 == 0) || ((year % 100 != 0) && (year % 4 == 0))){

alert(year + " is a leap year.");

} else{

alert(year + " is not a leap year.");

}

You will learn about conditional statements in [JavaScript if/else](https://www.tutorialrepublic.com/javascript-tutorial/javascript-if-else-statements.php) chapter.

## JavaScript Comparison Operators

The comparison operators are used to compare two values in a Boolean fashion.

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Name** | **Example** | **Result** |
| == | Equal | x == y | True if x is equal to y |
| === | Identical | x === y | True if x is equal to y, and they are of the same [type](https://www.tutorialrepublic.com/javascript-tutorial/javascript-data-types.php) |
| != | Not equal | x != y | True if x is not equal to y |
| !== | Not identical | x !== y | True if x is not equal to y, or they are not of the same type |
| < | Less than | x < y | True if x is less than y |
| > | Greater than | x > y | True if x is greater than y |
| >= | Greater than or equal to | x >= y | True if x is greater than or equal to y |
| <= | Less than or equal to | x <= y | True if x is less than or equal to y |

The following example will show you these comparison operators in action:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=comparison-operators)

let x = 25;

let y = 35;

let z = "25";

alert(x == z); // Outputs: true

alert(x === z); // Outputs: false

alert(x != y); // Outputs: true

alert(x !== z); // Outputs: true

alert(x < y); // Outputs: true

alert(x > y); // Outputs: false

alert(x <= y); // Outputs: true

alert(x >= y); // Outputs: false

**JavaScript Events**

## Understanding Events and Event Handlers

An event is something that happens when user interact with the web page, such as when he clicked a link or button, entered text into an input box or textarea, made selection in a select box, pressed key on the keyboard, moved the mouse pointer, submits a form, etc. In some cases, the Browser itself can trigger the events, such as the page load and unload events.

When an event occur, you can use a JavaScript event handler (or an event listener) to detect them and perform specific task or set of tasks. By convention, the names for event handlers always begin with the word "on", so an event handler for the click event is called onclick, similarly an event handler for the load event is called onload, event handler for the blur event is called onblur, and so on.

There are several ways to assign an event handler. The simplest way is to add them directly to the start tag of the HTML elements using the special event-handler attributes. For example, to assign a click handler for a button element, we can use onclick attribute, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=attaching-event-handlers-inline)

<button type="button" onclick="alert('Hello World!')">Click Me</button>

However, to keep the JavaScript seperate from HTML, you can set up the event handler in an external JavaScript file or within the <script> and </script> tags, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=attaching-event-handlers-in-an-external-file)

<button type="button" id="myBtn">Click Me</button>

<script>

function sayHello() {

alert('Hello World!');

}

document.getElementById("myBtn").onclick = sayHello;

</script>

**Note:** Since HTML attributes are case-insensitive so onclick may also be written as onClick, OnClick or ONCLICK. But its value is case-sensitive.

In general, the events can be categorized into four main groups — [mouse events](https://www.tutorialrepublic.com/javascript-tutorial/javascript-events.php#mouse-events), [keyboard events](https://www.tutorialrepublic.com/javascript-tutorial/javascript-events.php#keyboard-events), [form events](https://www.tutorialrepublic.com/javascript-tutorial/javascript-events.php#form-events) and [document/window events](https://www.tutorialrepublic.com/javascript-tutorial/javascript-events.php#document-and-window-events). There are many other events, we will learn about them in later chapters. The following section will give you a brief overview of the most useful events one by one along with the real life practice examples.

## Mouse Events

A mouse event is triggered when the user click some element, move the mouse pointer over an element, etc. Here're some most important mouse events and their event handler.

## The Click Event (onclick)

The click event occurs when a user clicks on an element on a web page. Often, these are form elements and links. You can handle a click event with an onclick event handler.

The following example will show you an alert message when you click on the elements.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-click-event)

<button type="button" onclick="alert('You have clicked a button!');">Click Me</button>

<a href="#" onclick="alert('You have clicked a link!');">Click Me</a>

## The Contextmenu Event (oncontextmenu)

The contextmenu event occurs when a user clicks the right mouse button on an element to open a context menu. You can handle a contextmenu event with an oncontextmenu event handler.

The following example will show an alert message when you right-click on the elements.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-contextmenu-event)

<button type="button" oncontextmenu="alert('You have right-clicked a button!');">Right Click on Me</button>

<a href="#" oncontextmenu="alert('You have right-clicked a link!');">Right Click on Me</a>

## The Mouseover Event (onmouseover)

The mouseover event occurs when a user moves the mouse pointer over an element.

You can handle the mouseover event with the onmouseover event handler. The following example will show you an alert message when you place mouse over the elements.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-mouseover-event)

<button type="button" onmouseover="alert('You have placed mouse pointer over a button!');">Place Mouse Over Me</button>

<a href="#" onmouseover="alert('You have placed mouse pointer over a link!');">Place Mouse Over Me</a>

## The Mouseout Event (onmouseout)

The mouseout event occurs when a user moves the mouse pointer outside of an element.

You can handle the mouseout event with the onmouseout event handler. The following example will show you an alert message when the mouseout event occurs.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-mouseout-event)

<button type="button" onmouseout="alert('You have moved out of the button!');">Place Mouse Inside Me and Move Out</button>

<a href="#" onmouseout="alert('You have moved out of the link!');">Place Mouse Inside Me and Move Out</a>

## Keyboard Events

A keyboard event is fired when the user press or release a key on the keyboard. Here're some most important keyboard events and their event handler.

## The Keydown Event (onkeydown)

The keydown event occurs when the user presses down a key on the keyboard.

You can handle the keydown event with the onkeydown event handler. The following example will show you an alert message when the keydown event occurs.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-keydown-event)

<input type="text" onkeydown="alert('You have pressed a key inside text input!')">

<textarea onkeydown="alert('You have pressed a key inside textarea!')"></textarea>

## The Keyup Event (onkeyup)

The keyup event occurs when the user releases a key on the keyboard.

You can handle the keyup event with the onkeyup event handler. The following example will show you an alert message when the keyup event occurs.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-keyup-event)

<input type="text" onkeyup="alert('You have released a key inside text input!')">

<textarea onkeyup="alert('You have released a key inside textarea!')"></textarea>

## The Keypress Event (onkeypress)

The keypress event occurs when a user presses down a key on the keyboard that has a character value associated with it. For example, keys like Ctrl, Shift, Alt, Esc, Arrow keys, etc. will not generate a keypress event, but will generate a keydown and keyup event.

You can handle the keypress event with the onkeypress event handler. The following example will show you an alert message when the keypress event occurs.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-keypress-event)

<input type="text" onkeypress="alert('You have pressed a key inside text input!')">

<textarea onkeypress="alert('You have pressed a key inside textarea!')"></textarea>

## Form Events

A form event is fired when a form control receive or loses focus or when the user modify a form control value such as by typing text in a text input, select any option in a select box etc. Here're some most important form events and their event handler.

## The Focus Event (onfocus)

The focus event occurs when the user gives focus to an element on a web page.

You can handle the focus event with the onfocus event handler. The following example will highlight the background of text input in yellow color when it receives the focus.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-focus-event)

<script>

function highlightInput(elm){

elm.style.background = "yellow";

}

</script>

<input type="text" onfocus="highlightInput(this)">

<button type="button">Button</button>

**Note:** The value of this keyword inside an event handler refers to the element which has the handler on it (i.e. where the event is currently being delivered).

## The Blur Event (onblur)

The blur event occurs when the user takes the focus away from a form element or a window.

You can handle the blur event with the onblur event handler. The following example will show you an alert message when the text input element loses focus.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-blur-event)

<input type="text" onblur="alert('Text input loses focus!')">

<button type="button">Submit</button>

To take the focus away from a form element first click inside of it then press the tab key on the keyboard, give focus on something else, or click outside of it.

## The Change Event (onchange)

The change event occurs when a user changes the value of a form element.

You can handle the change event with the onchange event handler. The following example will show you an alert message when you change the option in the select box.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-change-event)

<select onchange="alert('You have changed the selection!');">

<option>Select</option>

<option>Male</option>

<option>Female</option>

</select>

## The Submit Event (onsubmit)

The submit event only occurs when the user submits a form on a web page.

You can handle the submit event with the onsubmit event handler. The following example will show you an alert message while submitting the form to the server.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-submit-event)

<form action="action.php" method="post" onsubmit="alert('Form data will be submitted to the server!');">

<label>First Name:</label>

<input type="text" name="first-name" required>

<input type="submit" value="Submit">

</form>

## Document/Window Events

Events are also triggered in situations when the page has loaded or when user resize the browser window, etc. Here're some most important document/window events and their event handler.

## The Load Event (onload)

The load event occurs when a web page has finished loading in the web browser.

You can handle the load event with the onload event handler. The following example will show you an alert message as soon as the page finishes loading.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-load-event)

<body onload="window.alert('Page is loaded successfully!');">

<h1>This is a heading</h1>

<p>This is paragraph of text.</p>

</body>

## The Unload Event (onunload)

The unload event occurs when a user leaves the current web page.

You can handle the unload event with the onunload event handler. The following example will show you an alert message when you try to leave the page.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-unload-event)

<body onunload="alert('Are you sure you want to leave this page?');">

<h1>This is a heading</h1>

<p>This is paragraph of text.</p>

</body>

## The Resize Event (onresize)

The resize event occurs when a user resizes the browser window. The resize event also occurs in situations when the browser window is minimized or maximized.

You can handle the resize event with the onresize event handler. The following example will show you an alert message when you resize the browser window to a new width and height.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=handling-the-resize-event)

<p id="result"></p>

<script>

function displayWindowSize() {

let w = window.outerWidth;

let h = window.outerHeight;

let txt = "Window size: width=" + w + ", height=" + h;

document.getElementById("result").innerHTML = txt;

}

window.onresize = displayWindowSize;

</script>

**JavaScript Strings**

## What is String in JavaScript

A string is a sequence of letters, numbers, special characters and arithmetic values or combination of all. Strings can be created by enclosing the string literal (i.e. string characters) either within single quotes (') or double quotes ("), as shown in the example below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=create-strings)

let myString = 'Hello World!'; // Single quoted string

let myString = "Hello World!"; // Double quoted string

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=using-quotes-inside-a-string)

let str1 = "it's okay";

let str2 = 'He said "Goodbye"';

let str3 = "She replied 'Calm down, please'";

let str4 = 'Hi, there!"; // Syntax error - quotes must match

However, you can still use single quotes inside a single quoted strings or double quotes inside double quoted strings by escaping the quotes with a backslash character (\), like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=escaping-quotes-inside-a-string)

let str1 = 'it\'s okay';

let str2 = "He said \"Goodbye\"";

let str3 = 'She replied \'Calm down, please\'';

The backslash (\) is called an escape character, whereas the sequences \' and \" that we've used in the example above are called escape sequences.

## JavaScript Escape Sequences

Escape sequences are also useful for situations where you want to use characters that can't be typed using a keyboard. Here are some other most commonly used escape sequences.

* \n is replaced by the newline character
* \t is replaced by the tab character
* \r is replaced by the carriage-return character
* \b is replaced by the backspace character
* \\ is replaced by a single backslash (\)

Here's an example to clarify the how escape sequences actually works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=escape-sequences)

let str1 = "The quick brown fox \n jumps over the lazy dog.";

document.write("<pre>" + str1 + "</pre>"); // Create line break

let str2 = "C:\Users\Downloads";

document.write(str2); // Prints C:UsersDownloads

let str3 = "C:\\Users\\Downloads";

document.write(str3); // Prints C:\Users\Downloads

## Performing Operations on Strings

JavaScript provides several properties and methods to perform operations on string values. Technically, only objects can have properties and methods. But in JavaScript primitive data types can act like objects when you refer to them with the property access notation (i.e. dot notation).

JavaScript making it possible by creating a temporary wrapper object for primitive data types. This process is done automatically by the JavaScript interpreter in the background.

## Getting the Length of a String

The length property returns the length of the string, which is the number of characters contained in the string. This includes the number of special characters as well, such as \t or \n.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=get-the-length-of-a-string)

let str1 = "This is a paragraph of text.";

document.write(str1.length); // Prints 28

let str2 = "This is a \n paragraph of text.";

document.write(str2.length); // Prints 30, because \n is only one character

**Note:** Since length is a property, not a function, so don't use parentheses after it like str.length(). Instead just write str.length, otherwise it will produce an error.

## Finding a String Inside Another String

You can use the indexOf() method to find a substring or string within another string. This method returns the index or position of the first occurrence of a specified string within a string.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=find-the-position-a-substring-within-a-string)

let str = "If the facts don't fit the theory, change the facts.";

let pos = str.indexOf("facts");

alert(pos); // 0utputs: 7

Similarly, you can use the lastIndexOf() method to get the index or position of the last occurrence of the specified string within a string, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=find-the-position-of-last-occurrence-of-a-substring-within-a-string)

let str = "If the facts don't fit the theory, change the facts.";

let pos = str.lastIndexOf("facts");

alert(pos); // 0utputs: 46

Both the indexOf(), and the lastIndexOf() methods return -1 if the substring is not found. Both methods also accept an optional integer parameter which specifies the position within the string at which to start the search. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=find-the-position-of-a-substring-within-a-string-from-specific-index)

let str = "If the facts don't fit the theory, change the facts.";

// Searching forwards

let pos1 = str.indexOf("facts", 20);

alert(pos1); // 0utputs: 46

// Searching backwards

let pos2 = str.lastIndexOf("facts", 20);

alert(pos2); // 0utputs: 7

**Note:** Characters in a string are indexed from left to right. The index of the first character is 0, and the index of the last character of a string called myStr is myStr.length - 1.

## Searching for a Pattern Inside a String

You can use the search() method to search a particular piece of text or pattern inside a string.

Like indexOf() method the search() method also returns the index of the first match, and returns -1 if no matches were found, but unlike indexOf() method this method can also take a regular expression as its argument to provide advanced search capabilities.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=search-text-or-pattern-inside-a-string)

let str = "Color red looks brighter than color blue.";

// Case sensitive search

let pos1 = str.search("color");

alert(pos1); // 0utputs: 30

// Case insensitive search using regexp

let pos2 = str.search(/color/i);

alert(pos2); // 0utputs: 0

**Note:** The search() method does not support global searches; it ignores the g flag or modifier (i.e. /pattern/g) of its regular expression argument.

You will learn more about regular expressions in the upcoming chapters.

## Extracting a Substring from a String

You can use the slice() method to extract a part or substring from a string.

This method takes 2 parameters: start index (index at which to begin extraction), and an optional end index (index before which to end extraction), like str.slice(startIndex, endIndex).

The following example slices out a portion of a string from position 4 to position 15:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=slice-out-a-portion-of-a-string)

let str = "The quick brown fox jumps over the lazy dog.";

let subStr = str.slice(4, 15);

document.write(subStr); // Prints: quick brown

You can also specify negative values. The negative value is treated as strLength + startIndex, where strLength is the length of the string (i.e. str.length), for example, if startIndex is -5 it is treated as strLength - 5. If startIndex is greater than or equal to the length of the string, slice() method returns an empty string. Also, if optional endIndex is not specified or omitted, the slice() method extracts to the end of the string.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=slice-strings-using-negative-indexes)

let str = "The quick brown fox jumps over the lazy dog.";

document.write(str.slice(-28, -19)); // Prints: fox jumps

document.write(str.slice(31)); // Prints: the lazy dog.

You can also use the substring() method to extract a section of the given string based on start and end indexes, like str.substring(startIndex, endIndex). The substring() method is very similar to the slice() method, except few differences:

* If either argument is less than 0 or is NaN, it is treated as 0.
* If either argument is greater than str.length, it is treated as if it were str.length.
* If startIndex is greater than endIndex, then substring() will swap those two arguments; for example, str.substring(5, 0) == str.substring(0, 5).

The following example will show you how this method actuallty works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=extract-substring-from-a-string)

let str = "The quick brown fox jumps over the lazy dog.";

document.write(str.substring(4, 15)); // Prints: quick brown

document.write(str.substring(9, 0)); // Prints: The quick

document.write(str.substring(-28, -19)); // Prints nothing

document.write(str.substring(31)); // Prints: the lazy dog.

### Extracting a Fixed Number of Characters from a String

JavaScript also provide the substr() method which is similar to slice() with a subtle difference, the second parameter specifies the number of characters to extract instead of ending index, like str.substr(startIndex, length). If length is 0 or a negative number, an empty string is returned. The following example demonstrates how it works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=extract-fixed-number-of-characters-from-a-string)

let str = "The quick brown fox jumps over the lazy dog.";

document.write(str.substr(4, 15)); // Prints: quick brown fox

document.write(str.substr(-28, -19)); // Prints nothing

document.write(str.substr(-28, 9)); // Prints: fox jumps

document.write(str.substr(31)); // Prints: the lazy dog.

## Replacing the Contents of a String

You can use the replace() method to replace part of a string with another string. This method takes two parameters a regular expression to match or substring to be replaced and a replacement string, like str.replace(regexp|substr, newSubstr).

This replace() method returns a new string, it doesn't affect the original string that will remain unchanged. The following example will show you how it works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=replace-part-of-string-with-another-string)

let str = "Color red looks brighter than color blue.";

let result = str.replace("color", "paint");

alert(result); // 0utputs: Color red looks brighter than paint blue.

By default, the replace() method replaces only the first match, and it is case-sensitive. To replace the substring within a string in a case-insensitive manner you can use a regular expression (regexp) with an i modifier, as shown in the example below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=find-and-replace-text-in-a-string-case-insensitively)

let str = "Color red looks brighter than color blue.";

let result = str.replace(/color/i, "paint");

alert(result); // 0utputs: paint red looks brighter than color blue.

Similarly, to replace all the occurrences of a substring within a string in a case-insensitive manner you can use the g modifier along with the i modifier, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=replace-all-occurrences-of-a-substring-in-a-string)

let str = "Color red looks brighter than color blue.";

let result = str.replace(/color/ig, "paint");

alert(result); // 0utputs: paint red looks brighter than paint blue.

## Converting a String to Uppercase or Lowercase

You can use the toUpperCase() method to convert a string to uppercase, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=convert-a-string-to-uppercase-characters)

let str = "Hello World!";

let result = str.toUpperCase();

document.write(result); // Prints: HELLO WORLD!

Similarly, you can use the toLowerCase() to convert a string to lowercase, like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=convert-a-string-to-lowercase-characters)

let str = "Hello World!";

let result = str.toLowerCase();

document.write(result); // Prints: hello world!

## Concatenating Two or More Strings

You can concatenate or combine two or more strings using the + and += [assignment operators](https://www.tutorialrepublic.com/javascript-tutorial/javascript-operators.php#assignment).

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=join-two-or-more-strings)

let hello = "Hello";

let world = "World";

let greet = hello + " " + world;

document.write(greet); // Prints: Hello World

let wish = "Happy";

wish += " New Year";

document.write(wish); // Prints: Happy New Year

JavaScript also provides concat() method to combine strings, but it is not recommended.

## Accessing Individual Characters from a String

You can use the charAt() method to access individual character from a string, like str.charAt(index). The index specified should be an integer between 0 and str.length - 1. If no index is provided the first character in the string is returned, since the default is 0.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=extract-a-single-character-from-a-string)

let str = "Hello World!";

document.write(str.charAt()); // Prints: H

document.write(str.charAt(6)); // Prints: W

document.write(str.charAt(30)); // Prints nothing

document.write(str.charAt(str.length - 1)); // Prints: !

There is even better way to do this. Since ECMAScript 5, strings can be treated like read-only arrays, and you can access individual characters from a string using square brackets ([]) instead of the charAt() method, as demonstrated in the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=extract-a-single-character-from-a-string-using-square-brackets)

let str = "Hello World!";

document.write(str[0]); // Prints: H

document.write(str[6]); // Prints: W

document.write(str[str.length - 1]); // Prints: !

document.write(str[30]); // Prints: undefined

**Note:** The only difference between accessing the character from a string using the charAt() and square bracket ([]) is that if no character is found, [] returns [undefined](https://www.tutorialrepublic.com/javascript-tutorial/javascript-data-types.php#undefined), whereas the charAt() method returns an empty string.

## Splitting a String into an Array

The split() method can be used to splits a string into an array of strings, using the syntax str.split(separator, limit). The seperator argument specifies the string at which each split should occur, whereas the limit arguments specifies the maximum length of the array.

If separator argument is omitted or not found in the specified string, the entire string is assigned to the first element of the array. The following example shows how it works:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=split-a-string-into-an-array)

let fruitsStr = "Apple, Banana, Mango, Orange, Papaya";

let fruitsArr = fruitsStr.split(", ");

document.write(fruitsArr[0]); // Prints: Apple

document.write(fruitsArr[2]); // Prints: Mango

document.write(fruitsArr[fruitsArr.length - 1]); // Prints: Papaya

// Loop through all the elements of the fruits array

for(let i in fruitsArr) {

document.write("<p>" + fruitsArr[i] + "</p>");

}

To split a string into an array of characters, specify an empty string ("") as a separator.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=split-a-string-into-an-array-of-characters)

let str = "INTERSTELLAR";

let strArr = str.split("");

document.write(strArr[0]); // Prints: I

document.write(strArr[1]); // Prints: N

document.write(strArr[strArr.length - 1]); // Prints: R

// Loop through all the elements of the characters array and print them

for(let i in strArr) {

document.write("<br>" + strArr[i]);

}

**JavaScript Numbers**

## Working with Numbers

JavaScript supports both integer and floating-point numbers that can be represented in decimal, hexadecimal or octal notation. Unlike other languages, JavaScript does not treat integer and floating-point numbers differently. All numbers in JavaScript are represented as floating-point numbers. Here's an example demonstrating the numbers in different formats:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=numbers)

let x = 2; // integer number

let y = 3.14; // floating-point number

let z = 0xff; // hexadecimal number

Extra large numbers can be represented in exponential notation e.g. 6.02e+23 (same as 6.02x1023).

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=representing-numbers-in-exponential-notation)

let x = 1.57e4; // same as 15700

let y = 4.25e+6; // same as 4.25e6 or 4250000

let z = 4.25e-6; // same as 0.00000425

**Tip:** The biggest safe integer in JavaScript is 9007199254740991 (253-1), whereas the smallest safe integer is -9007199254740991 (-(253-1)).

Numbers can also be represented in hexadecimal notation (base 16). Hexadecimal numbers are prefixed with 0x. They are commonly used to represent [colors](https://www.tutorialrepublic.com/css-reference/css-color-values.php). Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=representing-numbers-in-hexadecimal-notation)

let x = 0xff; // same as 255

let y = 0xb4; // same as 180

let z = 0x00; // same as 0

**Note:** Integers can be represented in decimal, hexadecimal, and octal notation. Floating-point numbers can be represented in decimal or exponential notation.

## Operating on Numbers and Strings

As you know from the previous chapters, the + operator is used for both addition and concatenation. So, performing mathematical operations on numbers and strings may produce interesting results. The following example will show you what happens when you add numbers and strings:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=adding-numbers-and-strings)

let x = 10;

let y = 20;

let z = "30";

// Adding a number with a number, the result will be sum of numbers

console.log(x + y); // 30

// Adding a string with a string, the result will be string concatenation

console.log(z + z); // '3030'

// Adding a number with a string, the result will be string concatenation

console.log(x + z); // '1030'

// Adding a string with a number, the result will be string concatenation

console.log(z + x); // '3010'

// Adding strings and numbers, the result will be string concatenation

console.log("The result is: " + x + y); // 'The result is: 1020'

// Adding numbers and strings, calculation performed from left to right

console.log(x + y + z); // 'The result is: 3030'

If you observe the above example carefully, you will find that the result of the last operation is not just a simple string concatenation, because operators with the same precedence are evaluated from left to right. That's why, since variables x and y both are numbers they are added first then the result is concatenated with the variable z which is a string, hence final result is 30 + "30" = "3030".

But, if you perform other mathematical operations like multiplication, division, or subtraction the result will be different. JavaScript will automatically convert numeric strings (i.e. strings containing numeric values) to numbers in all numeric operations, as shown in the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=automatic-conversion-of-numeric-strings-to-numbers)

let x = 10;

let y = 20;

let z = "30";

// Subtracting a number from a number

console.log(y - x); // 10

// Subtracting a number from a numeric string

console.log(z - x); // 20

// Multiplying a number with a numeric string

console.log(x \* z); // 300

// Dividing a number with a numeric string

console.log(z / x); // 3

Moreover, if you try to multiply or divide numbers with strings that are not numeric, it returns NaN (Not a Number). Also, if you use NaN in a mathematical operation, the result will also be NaN.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=performing-mathematical-operation-on-non-numeric-values)

let x = 10;

let y = "foo";

let z = NaN;

// Subtracting a number from a non-numeric string

console.log(y - x); // NaN

// Multiplying a number with a non-numeric string

console.log(x \* y); // NaN

// Dividing a number with a non-numeric string

console.log(x / y); // NaN

// Adding NaN to a number

console.log(x + z); // NaN

// Adding NaN to a string

console.log(y + z); // fooNaN

## Representing Infinity

Infinity represents a number too big for JavaScript to handle. JavaScript has special keyword Infinity and -Infinity to represent positive and negative infinity respectively. For example, dividing by 0 returns Infinity, as demonstrated below:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=representing-infinity)

let x = 5 / 0;

console.log(x); // Infinity

let y = -5 / 0;

console.log(y); // -Infinity

**Note:** Infinity is a special value that represents the mathematical Infinity ∞, which is greater than any number. The [typeof](https://www.tutorialrepublic.com/javascript-tutorial/javascript-data-types.php" \l "typeof) operator return number for an Infinity value.

## Avoiding Precision Problems

Sometimes, operations on floating-point numbers produce unexpected results, as shown here:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=adding-floating-point-numbers)

let x = 0.1 + 0.2;

console.log(x) // 0.30000000000000004

As you can see the result is 0.30000000000000004 rather than the expected 0.3. This difference is called representation error or roundoff error. It occurs because JavaScript and many other languages uses binary (base 2) form to represent decimal (base 10) numbers internally. Unfortunately, most decimal fractions can't be represented exactly in binary form, so small differences occur.

To avoid this problem you can use the solution something like this:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=fixing-roundoff-error)

let x = (0.1 \* 10 + 0.2 \* 10) / 10;

console.log(x) // 0.3

JavaScript round floating-point numbers to 17 digits, which is enough precision or accuracy in most cases. Also, in JavaScript integers (numbers without fractional parts or exponential notation) are accurate is up to 15 digits, as demonstrated in the following example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=precision-problem-with-integers)

let x = 999999999999999;

console.log(x); // 999999999999999

let y = 9999999999999999;

console.log(y); // 10000000000000000

## Performing Operations on Numbers

JavaScript provides several properties and methods to perform operations on number values. As you already know from the previous chapters, in JavaScript primitive data types can act like objects when you refer to them with the property access notation (i.e. dot notation).

In the following sections, we will look at the number methods that are most commonly used.

## Parsing Integers from Strings

The parseInt() method can be used to parse an integer from a string. This method is particularly handy in situations when you are dealing with the values like CSS units e.g. 50px, 12pt, etc. and you would like to extract the numeric value out of it.

If the parseInt() method encounters a character that is not numeric in the specified base, it stops parsing and returns the integer value parsed up to that point. If the first character cannot be converted into a number, the method will return NaN (not a number).

Leading and trailing spaces are allowed. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=parse-integers-from-strings)

console.log(parseInt("3.14")); // 3

console.log(parseInt("50px")); // 50

console.log(parseInt("12pt")); // 12

console.log(parseInt("0xFF", 16)); // 255

console.log(parseInt("20 years")); // 20

console.log(parseInt("Year 2048")); // NaN

console.log(parseInt("10 12 2018")); // 10

**Note:** The parseInt() method truncates numbers to integer values, but it should not be used as a substitute for [Math.floor()](https://www.tutorialrepublic.com/javascript-tutorial/javascript-math-operations.php" \l "floor) method.

Similarly, you can use the parseFloat() method to parse floating-point number from a string. The parseFloat() method works the same way as the parseInt() method, except that it retrieves both integers and numbers with decimals.

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=parse-floating-point-numbers-from-strings)

console.log(parseFloat("3.14")); // 3.14

console.log(parseFloat("50px")); // 50

console.log(parseFloat("1.6em")); // 1.6

console.log(parseFloat("124.5 lbs")); // 124.5

console.log(parseFloat("weight 124.5 lbs")); // NaN

console.log(parseFloat("6.5 acres")); // 6.5

## Converting Numbers to Strings

The toString() method can be used to convert a number to its string equivalent. This method optionally accepts an integer parameter in the range 2 through 36 specifying the base to use for representing numeric values. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=convert-numbers-to-strings)

let x = 10;

let y = x.toString();

console.log(y); // '10'

console.log(typeof y); // string

console.log(typeof x); // number

console.log((12).toString()); // '12'

console.log((15.6).toString()); // '15.6'

console.log((6).toString(2)); // '110'

console.log((255).toString(16)); // 'ff'

## Formatting Numbers in Exponential Notation

You can use the toExponential() method to format or represent a number in exponential notation. This method optionally accepts an integer parameter specifying the number of digits after the decimal point. Also, the returned value is a string not a number. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=format-a-number-in-exponential-notation)

let x = 67.1234;

console.log(x.toExponential()); // 6.71234e+1

console.log(x.toExponential(6)); // 6.712340e+1

console.log(x.toExponential(4)); // 6.7123e+1

console.log(x.toExponential(2)); // 6.71e+1

**Note:** Exponential notation is useful for representing numbers that are either very large or very small in magnitude. For example, 62500000000 can be written as 625e+8 or 6.25e+10.

## Formatting Numbers to Fixed Decimals

You can use the toFixed() method when you want to format a number with a fixed number of digits to the right of the decimal point. The value returned by this method is a string and it has exactly specified number of digits after the decimal point. If the digits parameter is not specified or omitted, it is treated as 0. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=format-numbers-to-fixed-decimal-places)

let x = 72.635;

console.log(x.toFixed()); // '73' (note rounding, no fractional part)

console.log(x.toFixed(2)); // '72.64' (note rounding)

console.log(x.toFixed(1)); // '72.6'

let y = 6.25e+5;

console.log(y.toFixed(2)); // '625000.00'

let z = 1.58e-4;

console.log(z.toFixed(2)); // '0.00' (since 1.58e-4 is equal to 0.000158)

## Formatting Numbers with Precision

If you want most appropriate form of a number, you can use the toPrecision() method instead. This method returns a string representing the number to the specified precision.

If precision is large enough to include all the digits of the integer part of number, then the number is formatted using fixed-point notation. Otherwise, the number is formatted using exponential notation. The precision parameter is optional. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=format-numbers-with-precision)

let x = 6.235;

console.log(x.toPrecision()); // '6.235'

console.log(x.toPrecision(3)); // '6.24' (note rounding)

console.log(x.toPrecision(2)); // '6.2'

console.log(x.toPrecision(1)); // '6'

let y = 47.63;

console.log(y.toPrecision(2)); // '48' (note rounding, no fractional part)

let z = 1234.5;

console.log(z.toPrecision(2)); // '1.2e+3'

## Finding the Largest and Smallest Possible Numbers

The Number object also has several properties associated with it. The Number.MAX\_VALUE and Number.MIN\_VALUE properties of the Number object represent the largest and smallest (closest to zero, not most negative) possible positive numbers that JavaScript can handle. They are constants and their actual values are 1.7976931348623157e+308, and 5e-324, respectively.

A number that falls outside of the range of possible numbers is represented by a constant Number.POSITIVE\_INFINITY or Number.NEGATIVE\_INFINITY. Here's an example:

#### Example

[**Try this code »**](https://www.tutorialrepublic.com/codelab.php?topic=javascript&file=largest-and-smallest-possible-pumbers)

let a = Number.MAX\_VALUE;

console.log(a); // 1.7976931348623157e+308

let b = Number.MIN\_VALUE;

console.log(b); // 5e-324

let x = Number.MAX\_VALUE \* 2;

console.log(x); // Infinity

let y = -1 \* Number.MAX\_VALUE \* 2;

console.log(y); // -Infinity