Data Type Conversion

David Tucker

CTO Consultant

@_davidtucker_ | davidtucker.net

Data Type Conversion

Even though JavaScript leverages dynamic typing, we still need to understand how types work in the language. This includes knowing how to convert one type to another type. The process of changing from one type to another is called conversion.

Conversion Use Cases

- Joining a non-string value with a string
- Formatting a number to be displayed as currency
- Formatting an object, such as Date, for display
- Exporting an object to a portable format
- Evaluating an expression to a boolean value

JSON

JSON

JavaScript Object Notation (JSON) enables developers to convert a JavaScript object into a string. This string can be passed between applications, stored on the local file system, or loaded at runtime.

Using JSON

sample.json

```
{
  "firstName": "David",
  "lastName": "Tucker",
  "birthDate": "1982-01-01T05:00:00.000Z",
  "isActive": true,
  "numYearsEmployment": 7
}
```

JSON

- Similar to the object literal syntax
- Requires double quotes for property names
- Requires any string values to be in double quotes
- Won't include undefined properties or functions
- JavaScript has methods for converting to JSON and from JSON to an object

Related Topics

Standard built-in objects

JSON

▼ Constructor

▼ Methods

JSON.parse()

JSON.stringify()

Inheritance:

Object

▶ Constructor

▶ Properties

JSON

The `JSON` object contains methods for parsing JavaScript Object Notation (JSON) and converting values to JSON. It can't be called or constructed.

Description

Unlike most global objects, `JSON` is not a constructor. You cannot use it with a `new` operator or invoke the `JSON` object as a function. All properties and methods of `JSON` are static (just like the `Math` object).

JavaScript and JSON differences

JSON is a syntax for serializing objects, arrays, numbers, strings, booleans, and `null`. It is based upon JavaScript syntax, but is distinct from JavaScript: most of

Formatting Numbers

Number Formatting Use Cases

Rounding a number to an integer value

Limiting the number of decimal places to display

Output a number using a format specific to a certain geographic area

Outputting a number for a specific currency format for an area

Locale

In computing, a locale is a set of parameters that defines the user's language, region and any special variant preferences that the user wants to see in their user interface. Usually a locale identifier consists of at least a language code and a country/region code. Locale is an important aspect of i18n (internationalization).

Wikipedia

Basic Locale Code

Example Locale Codes

Locale Code	Description
en-US	English, United States
en-GB	English, Great Britain
en-PR	English, Puerto Rico
zh-HK	Chinese, Hong Kong
zh-SG	Chinese, Singapore
fr-FR	French, France
fr-HT	French, Haiti
fr-GA	French, Gabon
fr-TG	French, Togo

Locale Code	Description
ar-BH	Arabic, Bahrain
ar-IQ	Arabic, Iraq
ar-LB	Arabic, Lebanon
ja-JP	Japanese, Japan
de-DE	German, Germany
hi-IN	Hindi, India
id-ID	Indonesian, Indonesia
it-IT	Italian, Italy
la-VA	Latin, Vatican City

Math

`Math` is a built-in object that has properties and methods for mathematical constants and functions. It's not a function object.

`Math` works with the `Number` type. It doesn't work with `BigInt`.

Description

Unlike many other global objects, `Math` is not a constructor. All properties and methods of `Math` are static. You refer to the constant pi as `Math.PI` and you call the sine function as `Math.sin(x)`, where `x` is the method's argument. Constants are defined with the full precision of real numbers in JavaScript.

Note: Many `Math` functions have a precision that's *implementation-*

Intl.NumberFormat

The `Intl.NumberFormat` object enables language-sensitive number formatting.

Try it

JavaScript Demo: Intl.NumberFormat

```
1  const number = 123456.789;
2
3  console.log(new Intl.NumberFormat('de-DE', { style: 'currency', currency:
4  // Expected output: "123.456,79 €"
5
6  // The Japanese yen doesn't use a minor unit
7  console.log(new Intl.NumberFormat('ja-JP', { style: 'currency', currency:
8  // Expected output: "￥123,457"
9
10 // Limit to three significant digits
11 console.log(new Intl.NumberFormat('en-IN', { maximumSignificantDigits: 3
12 // Expected output: "1,23,000"
13
```

Formatting Dates