PHP প্রোগ্রামিং ভাষ Associative Array declaration যেমন ভাবে করা হয় JavaScript প্রোগ্রামিং ভাষ সেই ভাববে declaration করা যাই না। JavaScript প্রোগ্রামিং ভাষই Associative Array ব্যাবহার করার জন JavaScript Object Or New Map() declare করা হয়।

JavaScript doesn’t have passing by reference only passing by value, even through it looks like passing by reference.

**02 JavaScript Fundamentals – Part 1**

1. Typeof(true).Check the data type.
2. কোন error আসলে আগে data type sure হতে হবে এর পরে operator precedence check korte hobe. আর জদ

Some() and every() return Boolean value(true or false) and it is the most similar function but have one big difference, difference is some() check the negative and positive all value (Example: const movements=[200, 450, -400, 3000,-650]) but every() check the only above zero value that’s main positive value (Example: const movements=[200, 450, 400, 3000, 650]).

First you should know what JSON is:

* It is *language agnostic* data-interchange format.

The syntax of JSON was inspired by the JavaScript Object Literal notation, but there are differences between them.

For example, in JSON all *keys* must be quoted, while in object literals this is not necessary:

// JSON:

{ "foo": "bar" }

// Object literal:

var o = { foo: "bar" };

The quotes are mandatory on JSON because in JavaScript (more exactly in ECMAScript 3rd. Edition), the usage of *reserved words* as property names is disallowed, for example:

var o = { if: "foo" }; // SyntaxError in ES3

While, using a string literal as a property name (quoting the property name) gives no problems:

var o = { "if": "foo" };

So for *"compatibility"* (and easy eval'ing maybe?) the quotes are mandatory.

The data types in JSON are also restricted to the following values:

* string
* number
* object
* array
* A literal as:
  + true
  + false
  + null

The grammar of Strings changes. They *have to* be delimited by *double quotes*, while in JavaScript, you can use single or double quotes interchangeably.

// Invalid JSON:

{ "foo": 'bar' }

The accepted JSON grammar of Numbers also changes, in JavaScript you can use Hexadecimal Literals, for example 0xFF, or (the infamous) Octal Literals e.g. 010. In JSON you can use only Decimal Literals.

// Invalid JSON:

{ "foo": 0xFF }