

JAVA SCRIPT





#ThinkBig

Trust of 1 Lakh + Students.

Celebrating more than 1 Lakh Followers on Social Media.

Ratings : ★★★★★ ★★★★★ **4.8**



Data Types

- A variable in JavaScript can contain any data. A variable can at one moment be a string and later receive a numeric value
- Programming languages that allow such things are called
“dynamically typed”
- meaning that there are data types, but variables are not bound to any of them.

```
var length = 16;           // Number
var lastName = "Johnson"; // String
var x = {firstName:"John", lastName:"Doe"}; // Object
```

Basic Data Types In Javascript

There are 7 basic types in JavaScript.

- **number** for numbers of any kind: integer or floating-point.
- **string** for strings. A string may have one or more characters, there's no separate single-character type.
- **boolean** for true/false.
- **null** for unknown values – a standalone type that has a single value null.
- **undefined** for unassigned values – a standalone type that has a single value undefined.
- **object** for more complex data structures.
- **symbol** for unique identifiers.

- The *number* type serves both for integer and floating point numbers.
- Besides regular numbers, there are so-called “special numeric values” which also belong to that type: Infinity, -Infinity and NaN. <body>

A number

An embedded page at run.plnkr.co says
Infinity

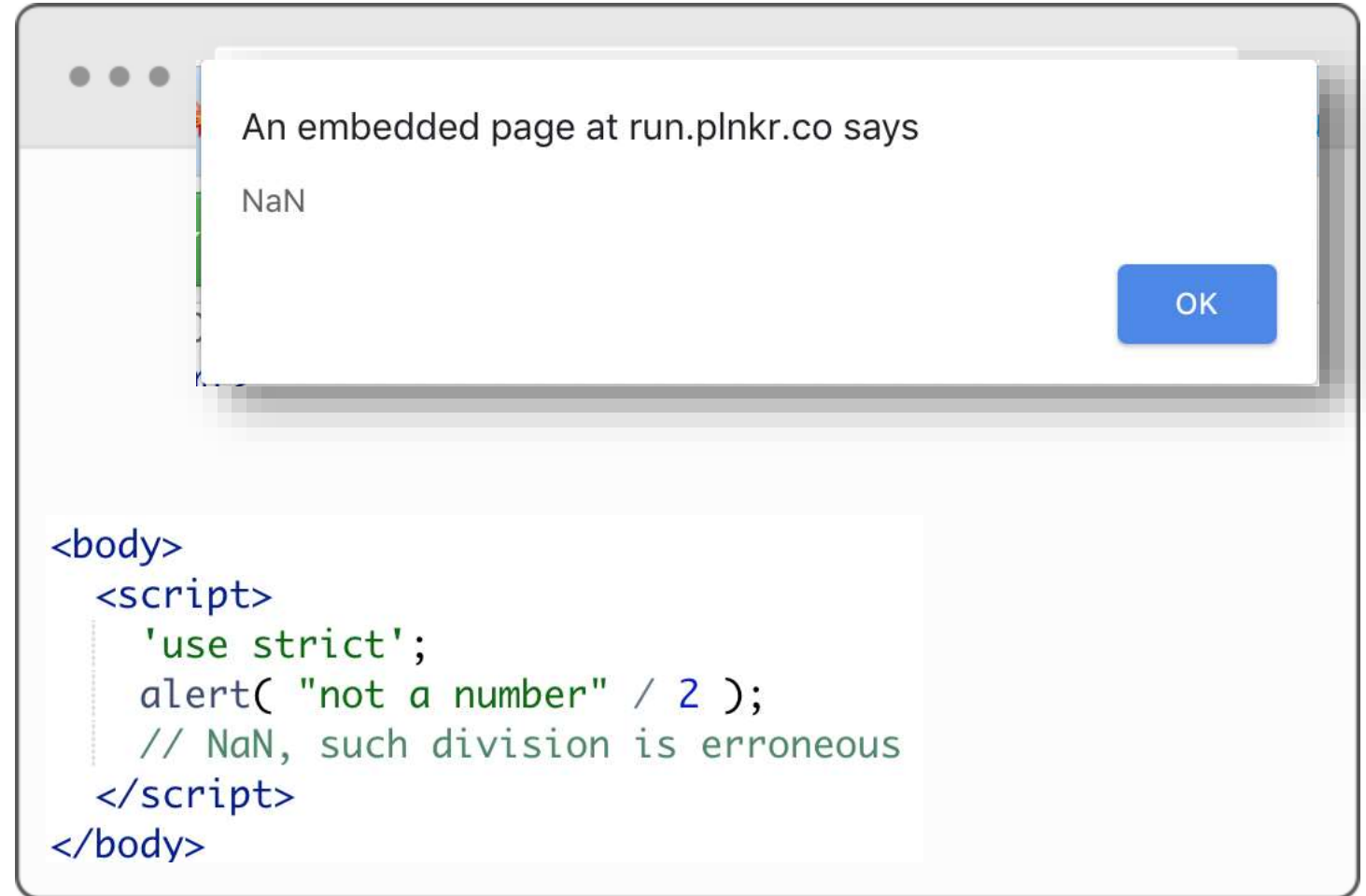
OK

```
<body>
  <script>
    'use strict';
    alert( 1 / 0 ); // Infinity
  </script>
</body>
```

NaN

NaN represents a computational error. It is a result of an incorrect or an undefined mathematical operation,

if there's NaN somewhere in a mathematical expression, it propagates to the whole result.



An embedded page at run.plnkr.co says

NaN

OK

```
<body>
  <script>
    'use strict';
    alert( "not a number" / 2 );
    // NaN, such division is erroneous
  </script>
</body>
```

A String

```
let str = "Hello";  
let str2 = 'Single quotes are ok too';  
let phrase = `can embed ${str}`;
```

In JavaScript, there are 3 types of quotes.

1. Double quotes: "Hello".
2. Single quotes: 'Hello'.
3. Backticks: `Hello`.

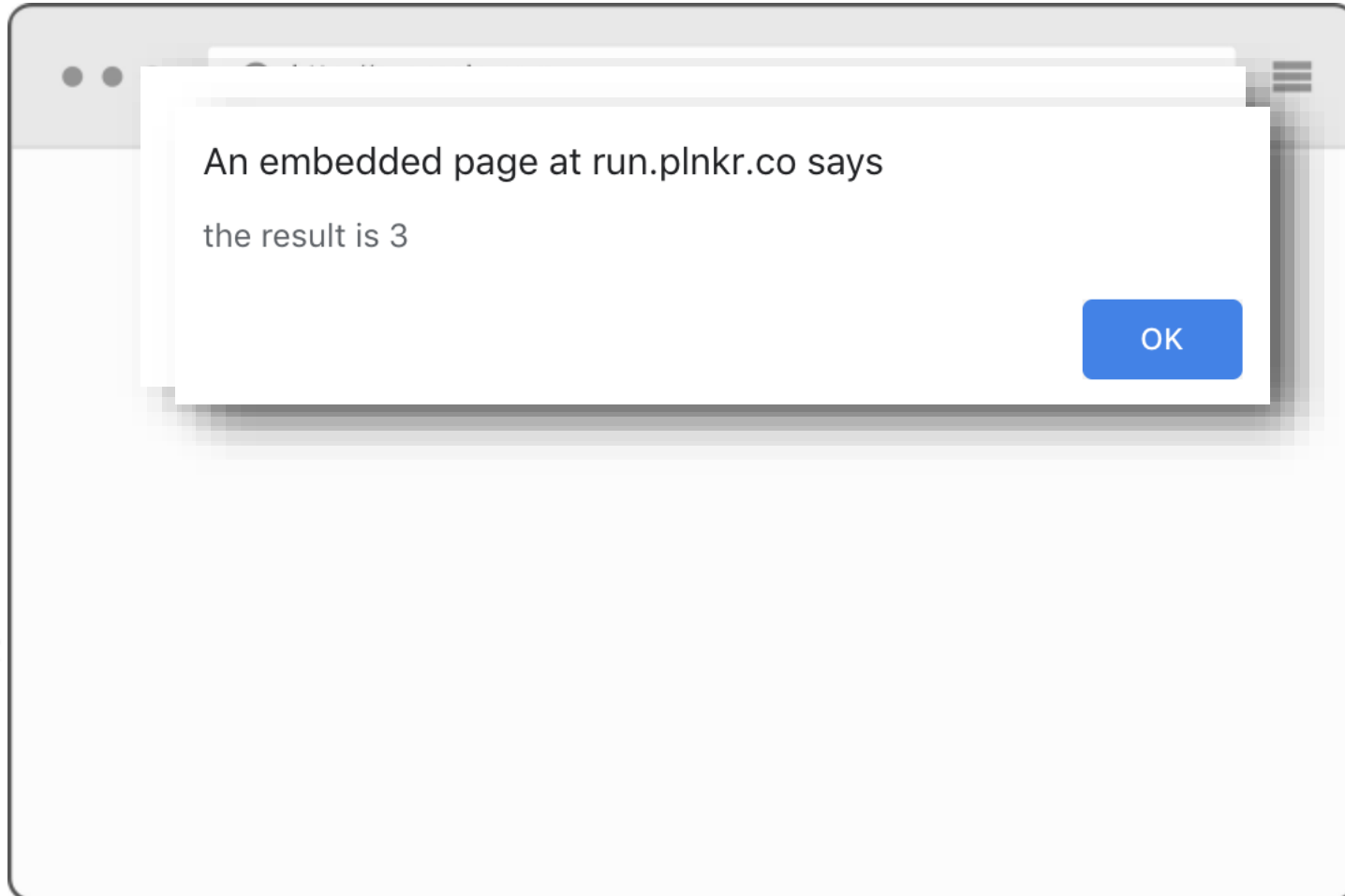
Double and single quotes are “simple” quotes. There’s no difference between them in JavaScript.

Backticks are “extended functionality” quotes. They allow us to embed variables and expressions into a string by wrapping them in `${...}`


```
<body>
  <script>
    'use strict';
    let name = "DigiiMento";

    // embed a variable
    alert( `Hello, ${name}!` );
    // Hello, John!

    // embed an expression
    alert( `the result is ${1 + 2}` );
    // the result is 3
  </script>
</body>
```



- The expression inside `${...}` is evaluated and the result becomes a part of the string. We can put anything there: a variable like name or an arithmetical expression like `1 + 2` or something more complex.
- Please note that this can only be done in backticks. Other quotes do not allow such embedding!



An embedded page at run.plnkr.co says
the result is `${1 + 2}`

OK

```
<body>  
  <script>  
    'use strict';  
    alert( "the result is ${1 + 2}" );  
    // the result is ${1 + 2} (double quotes do nothing)  
  </script>  
</body>
```

There is no *character* type

- In some languages, there is a special “character” type for a single character. For example, in the C language and in Java it is char.
- In JavaScript, there is no such type. There’s only one type: string. A string may consist of only one character or many of them.

A boolean (logical type)

The boolean type has only two values: true and false

```
1 let nameFieldChecked = true; // yes, name field is checked
2 let ageFieldChecked = false; // no, age field is not checked
```

Boolean values also come as a result of comparisons:

```
1 let isGreater = 4 > 1;
2
3 alert( isGreater ); // true (the comparison result is "yes")
```

The “null” value

- The special null value does not belong to any type of those described above.
- It forms a separate type of its own, which contains only the null value
- In JavaScript null is not a “reference to a non-existing object” or a “null pointer” like in some other languages.
- It’s just a special value which has the sense of “nothing”, “empty” or “value unknown”.

```
let age = null;
```

The “undefined” value

- The special value undefined stands apart. It makes a type of its own, just like null.
- The meaning of undefined is “value is not assigned”.
- If a variable is declared, but not assigned, then its value is exactly undefined

If a variable is declared, but not assigned, then its value is exactly `undefined` :

```
1 let x;  
2  
3 alert(x); // shows "undefined"
```

Technically, it is possible to assign `undefined` to any variable:

```
1 let x = 123;  
2  
3 x = undefined;  
4  
5 alert(x); // "undefined"
```

Objects and Symbols

- The object type is special.
- All other types are called “primitive”, because their values can contain only a single thing (be it a string or a number or whatever). In contrast, objects are used to store collections of data and more complex entities.

The typeof operator

- The typeof operator returns the type of the argument. It's useful when we want to process values of different types differently, or just want to make a quick check.


```
typeof undefined // "undefined"
```

```
typeof 0 // "number"
```

```
typeof true // "boolean"
```

```
typeof "foo" // "string"
```

```
typeof Symbol("id") // "symbol"
```

```
typeof Math // "object" (1)
```

```
typeof null // "object" (2)
```

```
typeof alert // "function" (3)
```

The result of `typeof` `null` is "object". That's wrong. It is an officially recognized error in `typeof`, kept for compatibility. Of course, `null` is not an object. It is a special value with a separate type of its own. So, again, that's an error in the language

```
typeof undefined // "undefined"  
typeof 0 // "number"  
typeof true // "boolean"  
typeof "foo" // "string"  
typeof Symbol("id") // "symbol"  
typeof Math // "object" (1)  
typeof null // "object" (2)  
typeof alert // "function" (3)
```

