STT-NF

Pemrograman Web

Sirojul Munir | rojulman@nurulfikri.ac.id

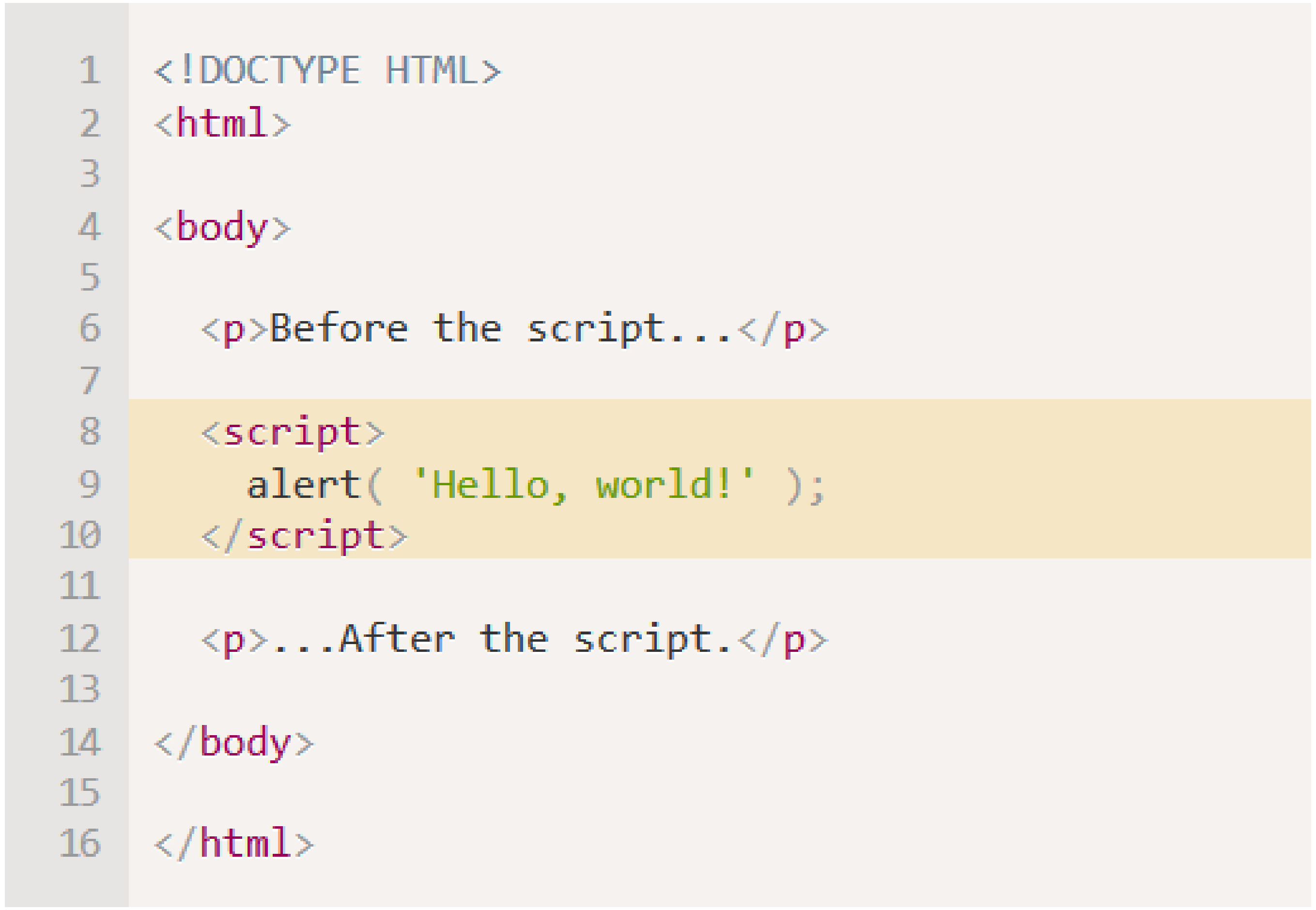
Basic JavaScript Programming

Sirojul Munir | rojulman@nurulfikri.ac.id

Outline Materi JavaScript

1. Pengantar JavaScript
2. Penulisan Program – Standard Output
3. Basic JavaScript: Statement, Data Type, Operator
4. Bekerja dengan Array & Object
5. Struktur Conditional dan Looping
6. Bekerja dengan Document Object Model
7. JavaScript Function
8. JavaScript Event Handler

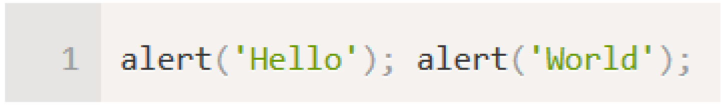
Hello World



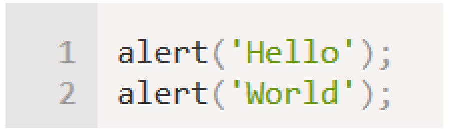
```html
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script>
    alert( 'Hello, world!' );
  </script>

  <p>...After the script.</p>

</body>

</html>
```

JavaScript Statement

- Statement JavaScript diakhiri tanda titik koma **(;)**.



```
alert('Hello'); alert('World');
```



```
alert('Hello');
alert('World');
```

- JavaScript is Case Sensitive
- Komentar program

  gunakan

  //this is comment

  atau

  /* this is comment */

Variable : digunakan menyimpan informasi / data

Most of the time, a JavaScript application needs to work with information. Here are two examples:

1. An online shop – the information might include goods being sold and a shopping cart.
2. A chat application – the information might include users, messages, and much more.

Variables are used to store this information.

```
let message;
message = 'Hello!';

alert(message); // shows the variable content
```

```
let user = 'John', age = 25, message = 'Hello';
```

Tidak di rekomendasikan ya !!

```
let user = 'John';
let age = 25;
let message = 'Hello';
```

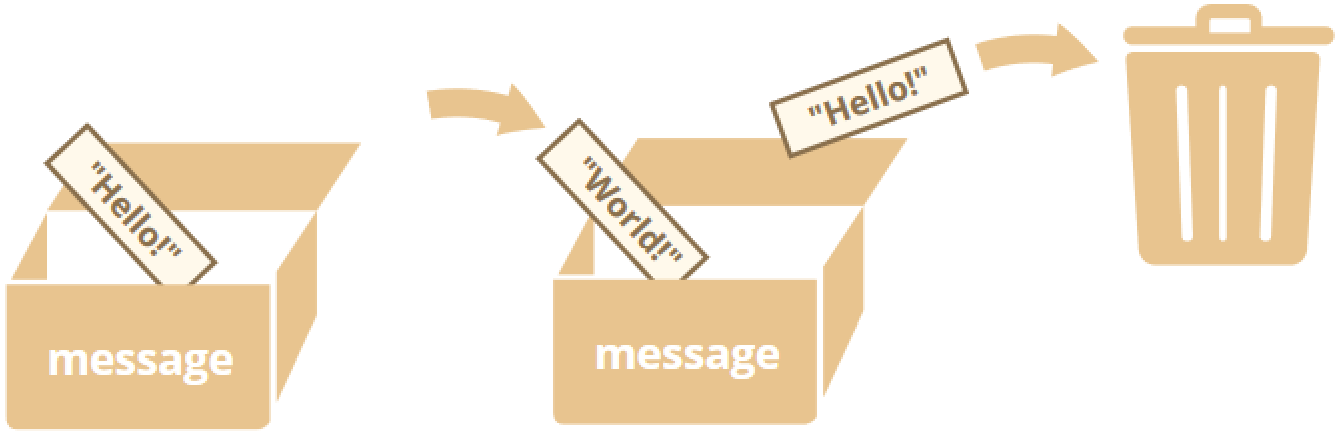
https://javascript.info

Konsep Variable

We can easily grasp the concept of a "variable" if we imagine it as a "box" for data, with a uniquely-named sticker on it.

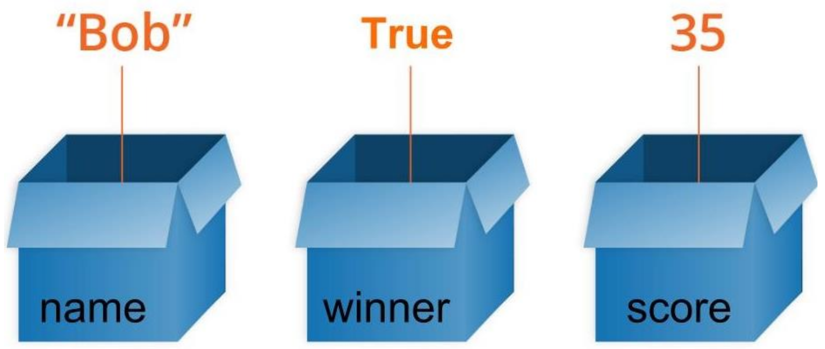
For instance, the variable `message` can be imagined as a box labeled `"message"` with the value `"Hello!"` in it:

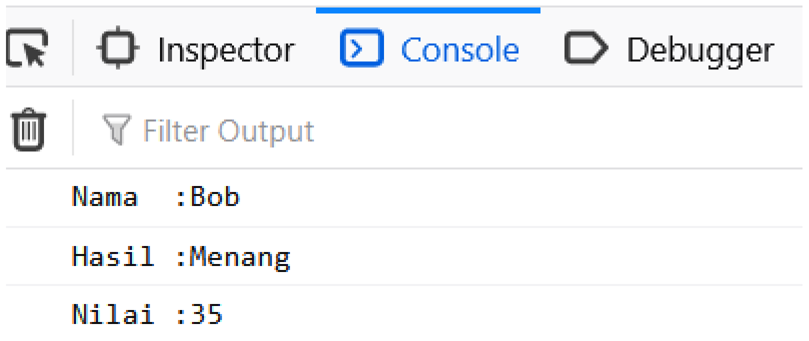
```
let message;

message = 'Hello!';

message = 'World!'; // value changed

alert(message);
```



https://javascript.info

Variable – Box Name



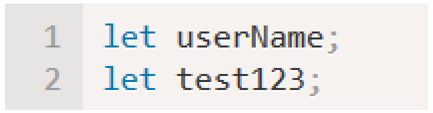


```
Nama   :Bob
Hasil  :Menang
Nilai  :35
```

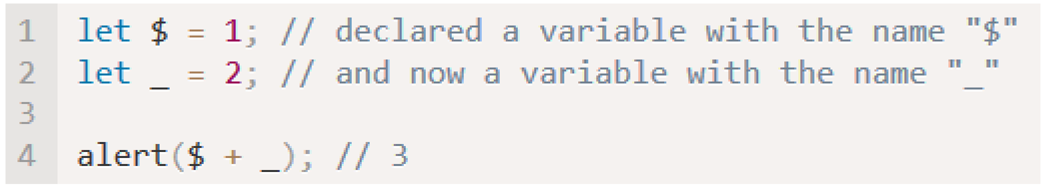
```html
<!DOCTYPE html>
<html lang="id">
    <head>
        <title>JS</title>
        <script>
            let name = "Bob";
            let winner = true;
            let score = 35;
            console.log("Nama  :" + name);
            let hasil = (winner==true)? "Menang": "Kalah";
            console.log("Hasil :" + hasil);
            console.log("Nilai :" + score);
        </script>
    </head>
    <body>

    </body>
</html>
```

Penamaan Variabel

- Karakter pertama dari identifier (name of variable, keywords, functions, and labels) harus huruf, underscore (_), atau tanda dolar ($).

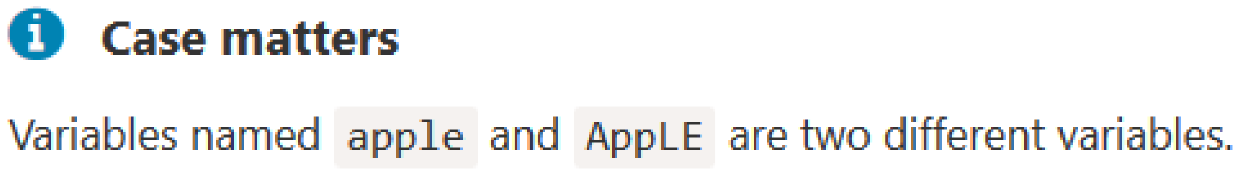


```
let userName;
let test123;
```



```
let $ = 1; // declared a variable with the name "$"
let _ = 2; // and now a variable with the name "_"

alert($ + _); // 3
```

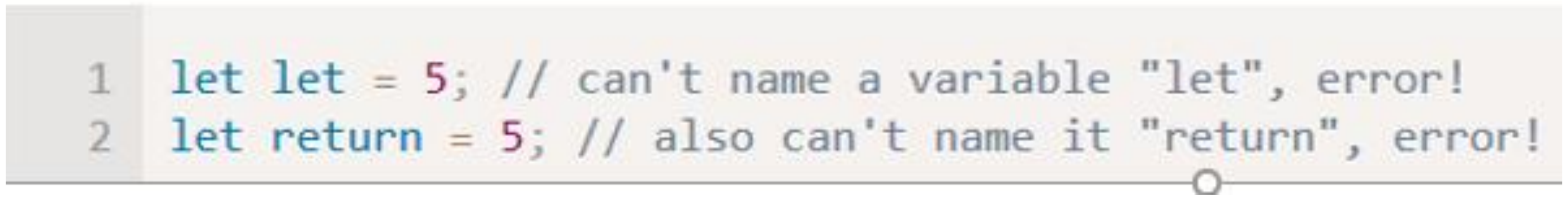
- JavaScript is Case Sensitive



Case matters

Variables named `apple` and `AppLE` are two different variables.

- List Keyword JavaScript: tidak boleh digunakan



```
let let = 5; // can't name a variable "let", error!
let return = 5; // also can't name it "return", error!
```

https://javascript.info

Variable Konstan

To declare a constant (unchanging) variable, use `const` instead of `let`:

```
const myBirthday = '18.04.1982';
```

Variables declared using `const` are called "constants". They cannot be reassigned. An attempt to do so would cause an error:

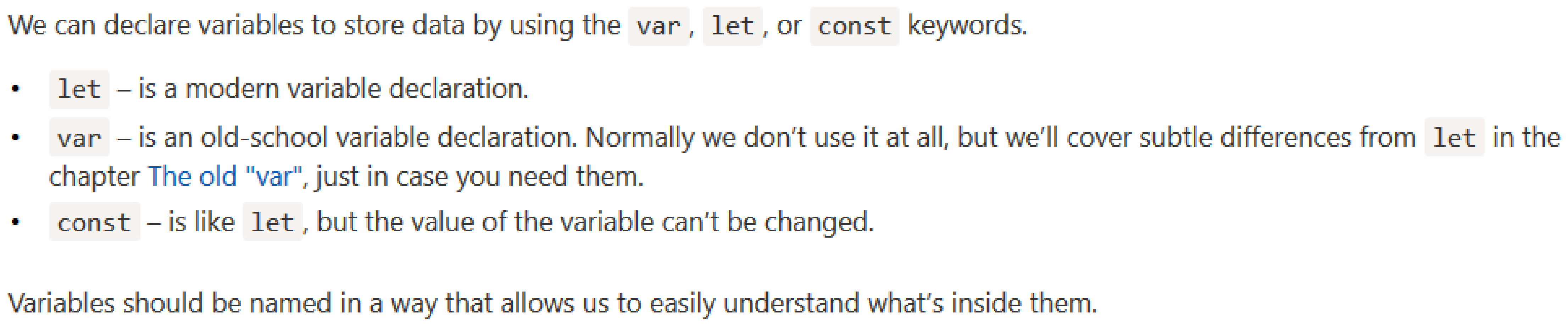
```
const myBirthday = '18.04.1982';

myBirthday = '01.01.2001'; // error, can't reassign the constant!
```

```
const COLOR_RED = "#F00";
const COLOR_GREEN = "#0F0";
const COLOR_BLUE = "#00F";
const COLOR_ORANGE = "#FF7F00";

// ...when we need to pick a color
let color = COLOR_ORANGE;
alert(color); // #FF7F00
```

Uppercase constants

https://javascript.info

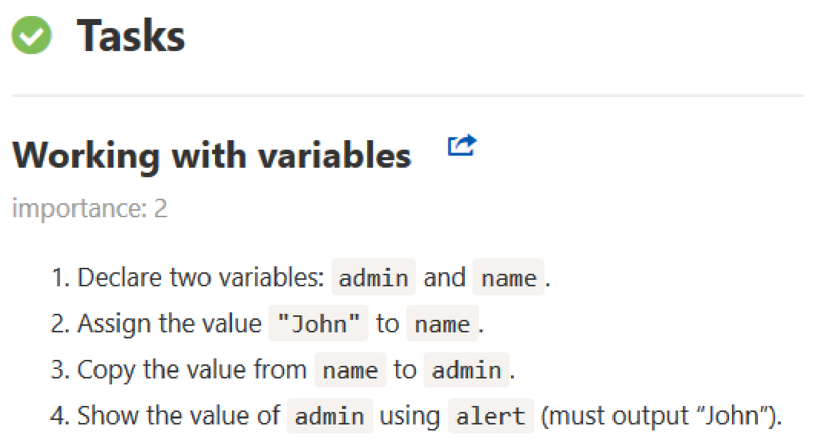
Variable - Summary



We can declare variables to store data by using the `var`, `let`, or `const` keywords.

- `let` – is a modern variable declaration.
- `var` – is an old-school variable declaration. Normally we don't use it at all, but we'll cover subtle differences from `let` in the chapter The old "var", just in case you need them.
- `const` – is like `let`, but the value of the variable can't be changed.

Variables should be named in a way that allows us to easily understand what's inside them.



Tasks

Working with variables

importance: 2

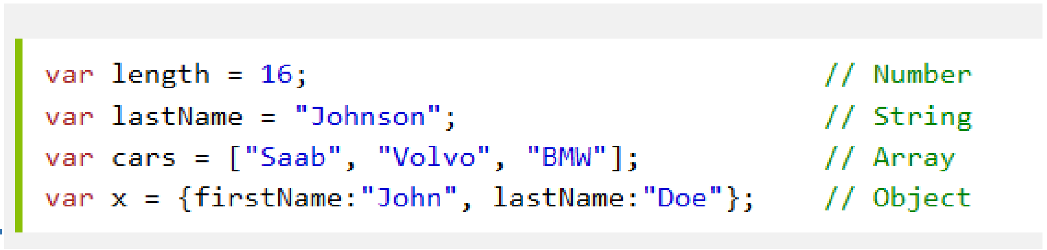
1. Declare two variables: `admin` and `name`.
2. Assign the value `"John"` to `name`.
3. Copy the value from `name` to `admin`.
4. Show the value of `admin` using `alert` (must output "John").

https://javascript.info

Variable – Data Type

There are 7 basic data 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.



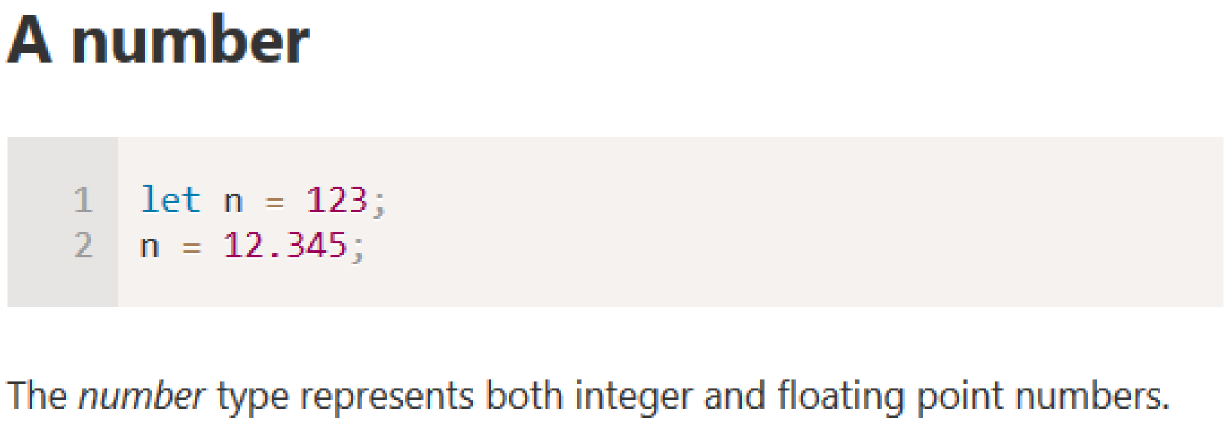
```
var length = 16;                                   // Number
var lastName = "Johnson";                          // String
var cars = ["Saab", "Volvo", "BMW"];               // Array
var x = {firstName:"John", lastName:"Doe"};        // Object
```

The `typeof` operator allows us to see which type is stored in a variable.

- Two forms: `typeof x` or `typeof(x)`.
- Returns a string with the name of the type, like `"string"`.
- For `null` returns `"object"` – this is an error in the language, it's not actually an object.

https://javascript.info

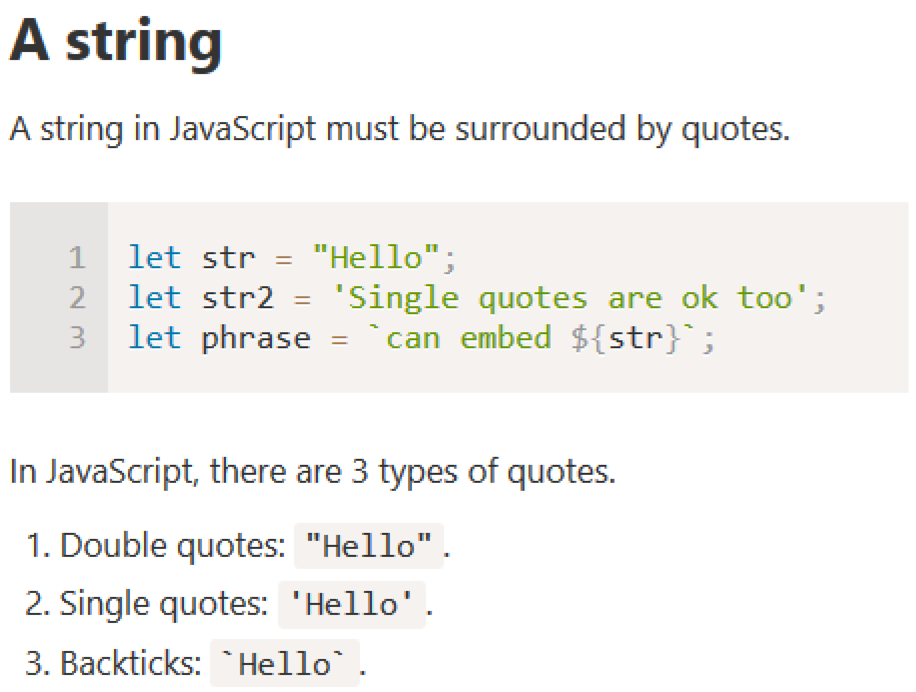
Data Type



A number

```javascript
let n = 123;
n = 12.345;
```

The *number* type represents both integer and floating point numbers.



A string

A string in JavaScript must be surrounded by quotes.

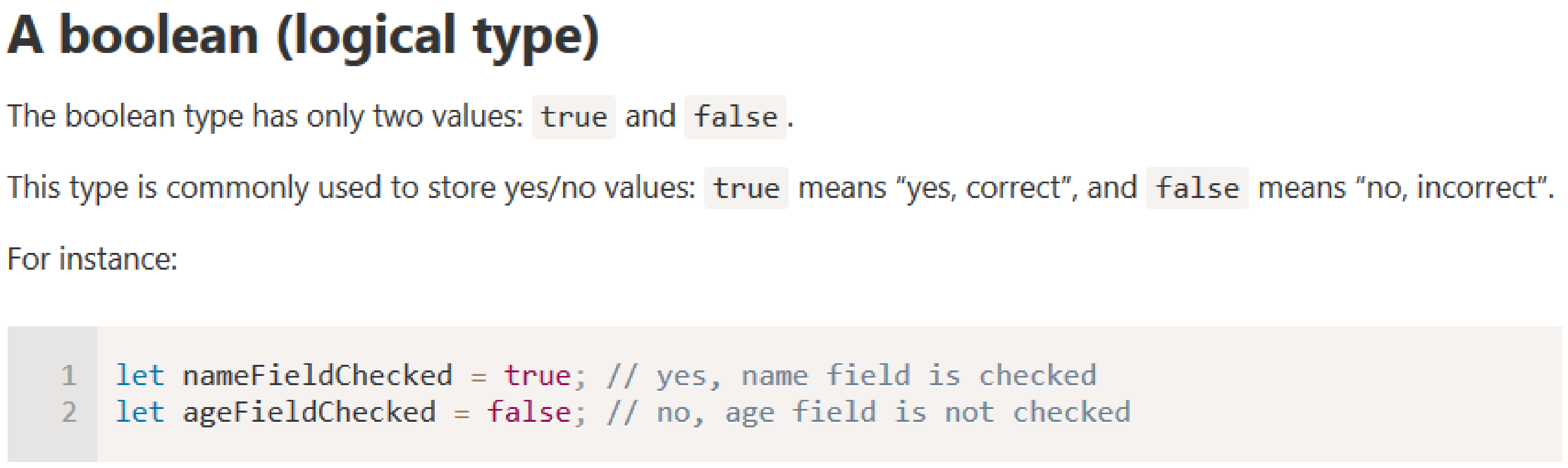
```javascript
let str = "Hello";
let str2 = 'Single quotes are ok too';
let phrase = `can embed ${str}`;
```

In JavaScript, there are 3 types of quotes.

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

https://javascript.info

Data Type



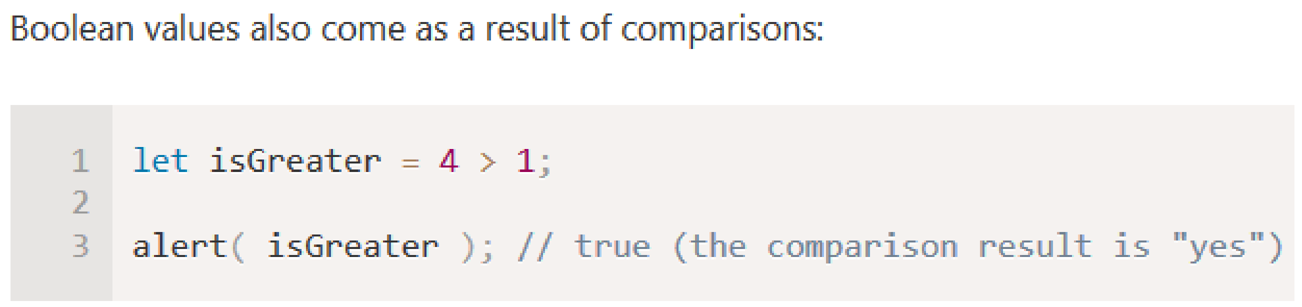
A boolean (logical type)

The boolean type has only two values: `true` and `false`.

This type is commonly used to store yes/no values: `true` means "yes, correct", and `false` means "no, incorrect".

For instance:

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



Boolean values also come as a result of comparisons:

```
let isGreater = 4 > 1;

alert( isGreater ); // true (the comparison result is "yes")
```

https://javascript.info

Data Type

A boolean (logical type)

The boolean type has only two values: `true` and `false`.

This type is commonly used to store yes/no values: `true` means "yes, correct", and `false` means "no, incorrect".

For instance:

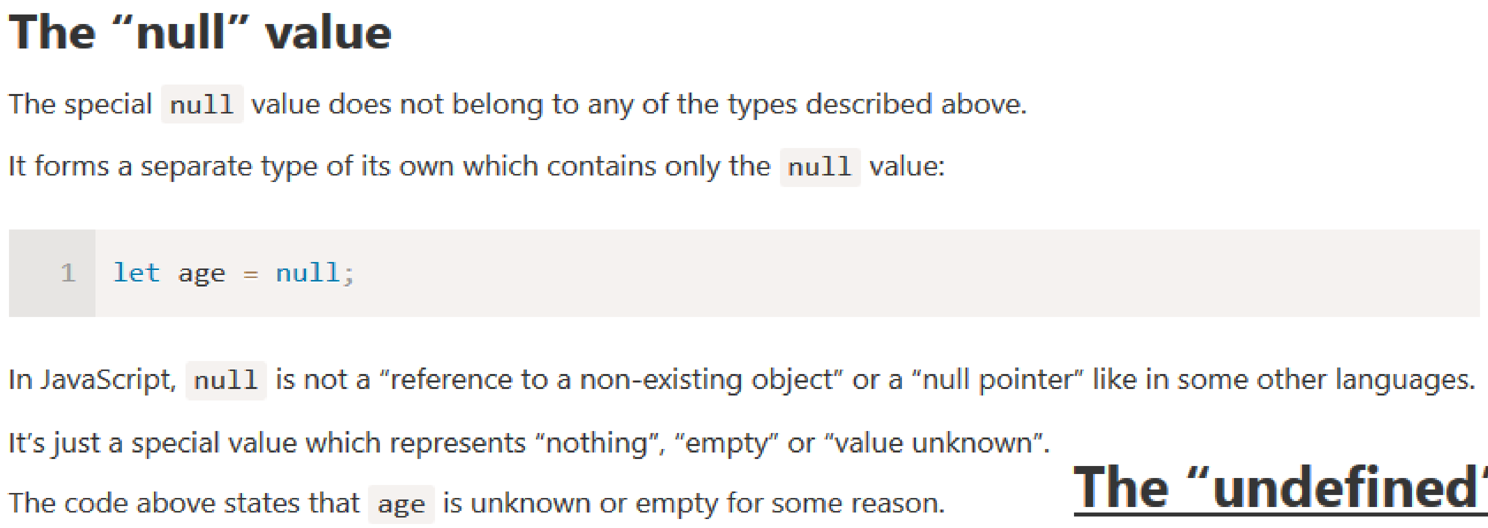
```
let nameFieldChecked = true; // yes, name field is checked
let ageFieldChecked = false; // no, age field is not checked
```

Boolean values also come as a result of comparisons:

```
let isGreater = 4 > 1;

alert( isGreater ); // true (the comparison result is "yes")
```

https://javascript.info

Data Type



The “null” value

The special `null` value does not belong to any of the types described above.

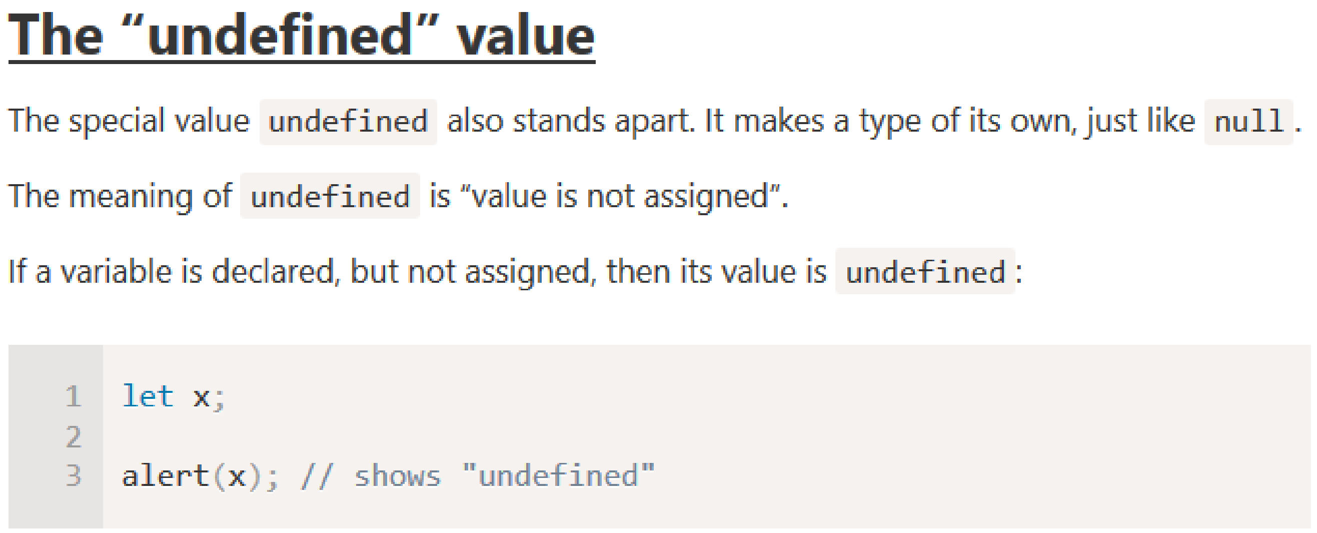
It forms a separate type of its own which contains only the `null` value:

```
let age = null;
```

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 represents “nothing”, “empty” or “value unknown”.

The code above states that `age` is unknown or empty for some reason.



The “undefined” value

The special value `undefined` also stands apart. It makes a type of its own, just like `null`.

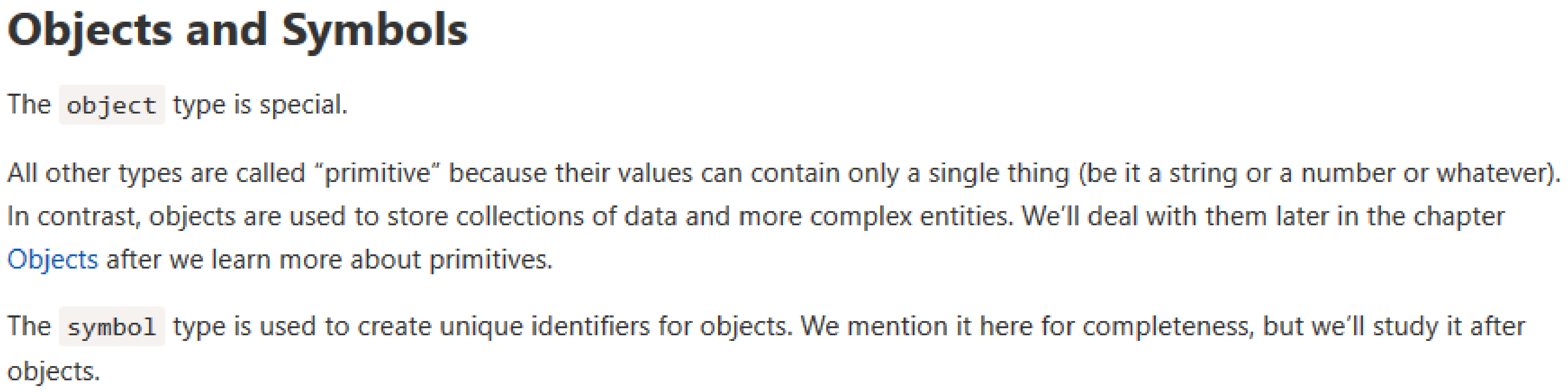
The meaning of `undefined` is “value is not assigned”.

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

```
let x;

alert(x); // shows "undefined"
```

https://javascript.info

Data Type



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. We’ll deal with them later in the chapter Objects after we learn more about primitives.

The `symbol` type is used to create unique identifiers for objects. We mention it here for completeness, but we’ll study it after objects.

https://javascript.info

Operator TypeOf

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 do a quick check.

It supports two forms of syntax:

1. As an operator: `typeof x`.
2. As a function: `typeof(x)`.

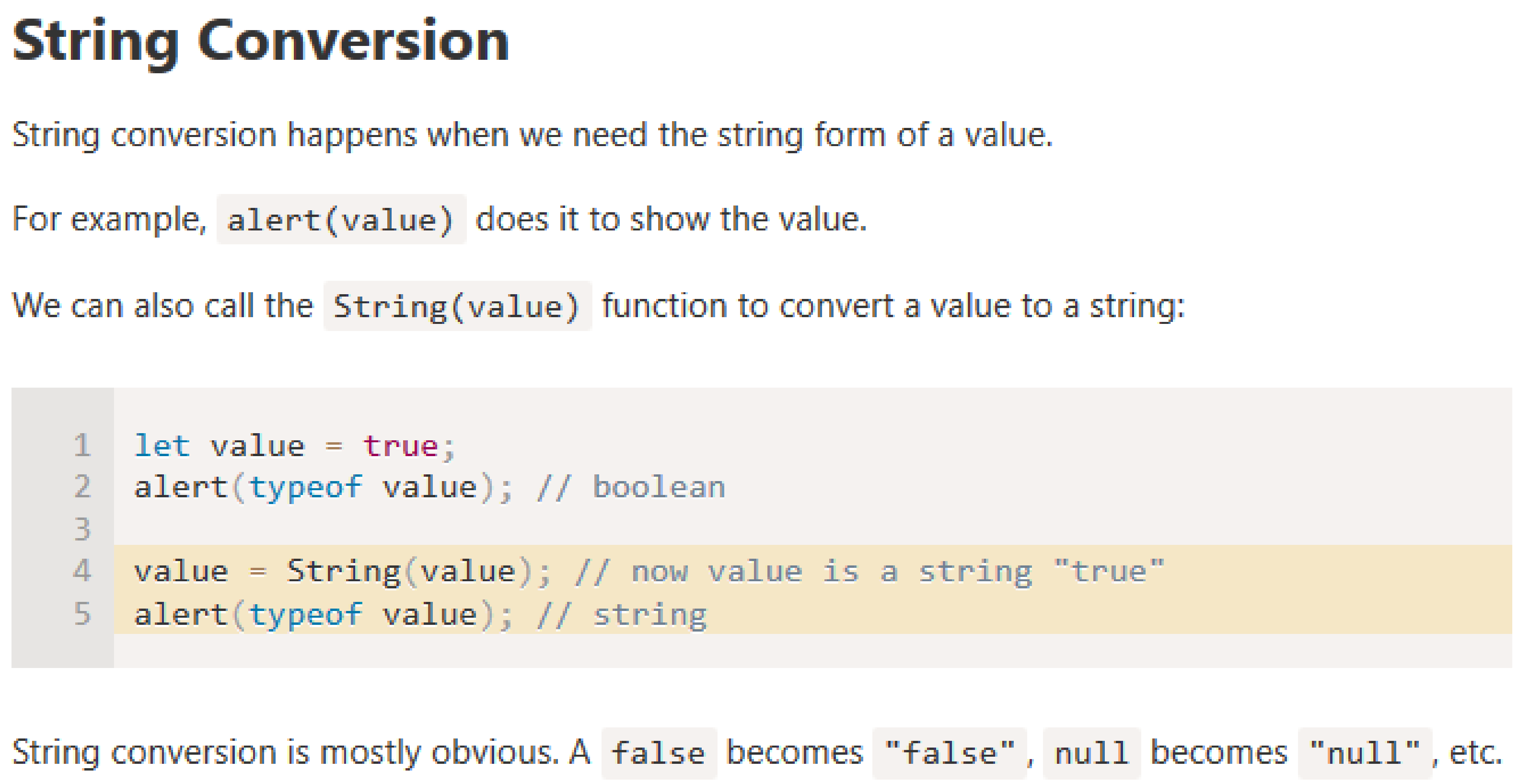
In other words, it works with parentheses or without them. The result is the same.

The call to `typeof x` returns a string with the type name:

```
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)
```

https://javascript.info

Type Conversions



String Conversion

String conversion happens when we need the string form of a value.

For example, `alert(value)` does it to show the value.

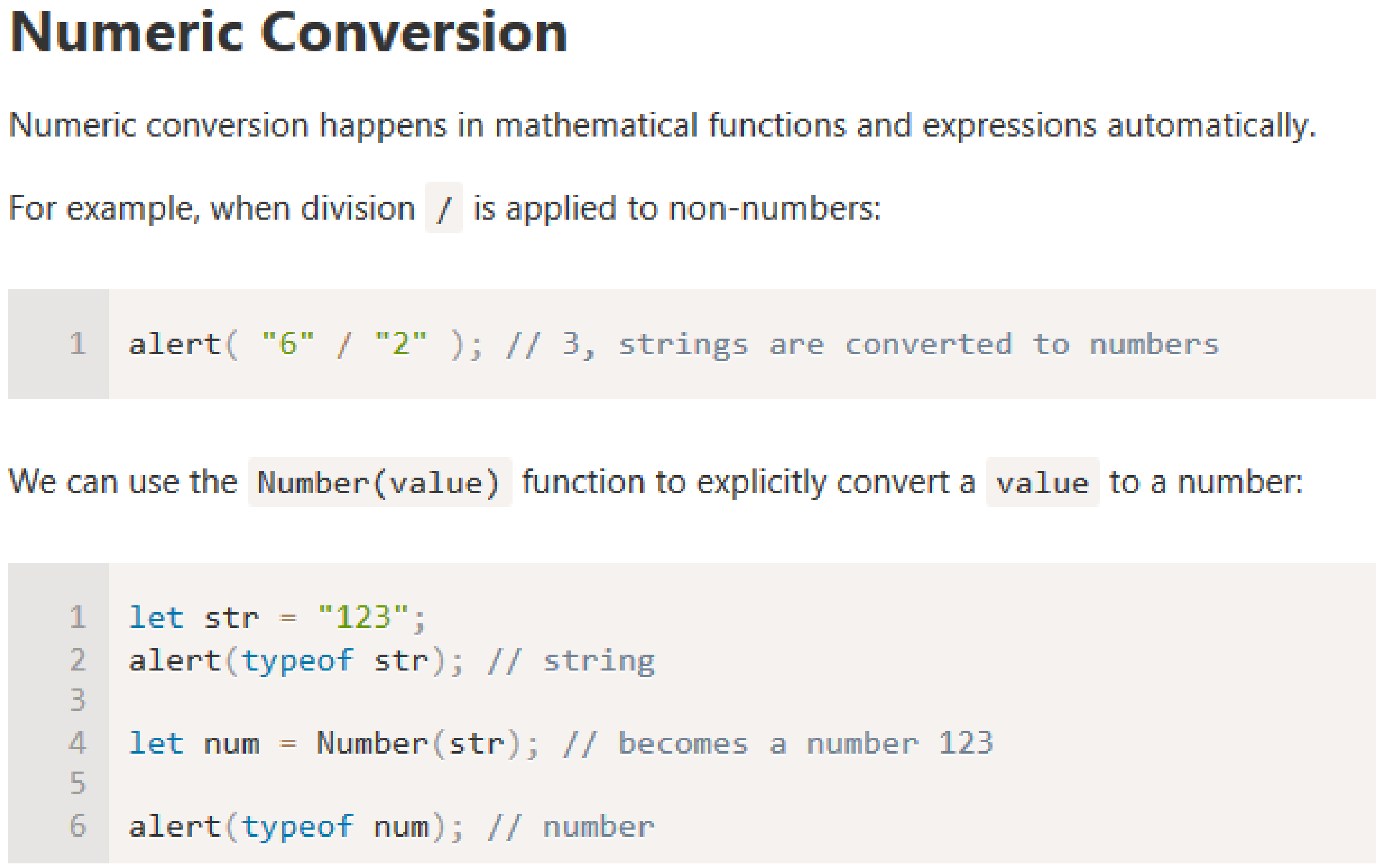
We can also call the `String(value)` function to convert a value to a string:

```
let value = true;
alert(typeof value); // boolean

value = String(value); // now value is a string "true"
alert(typeof value); // string
```

String conversion is mostly obvious. A `false` becomes `"false"`, `null` becomes `"null"`, etc.

https://javascript.info

Type Conversions



Numeric Conversion

Numeric conversion happens in mathematical functions and expressions automatically.

For example, when division `/` is applied to non-numbers:

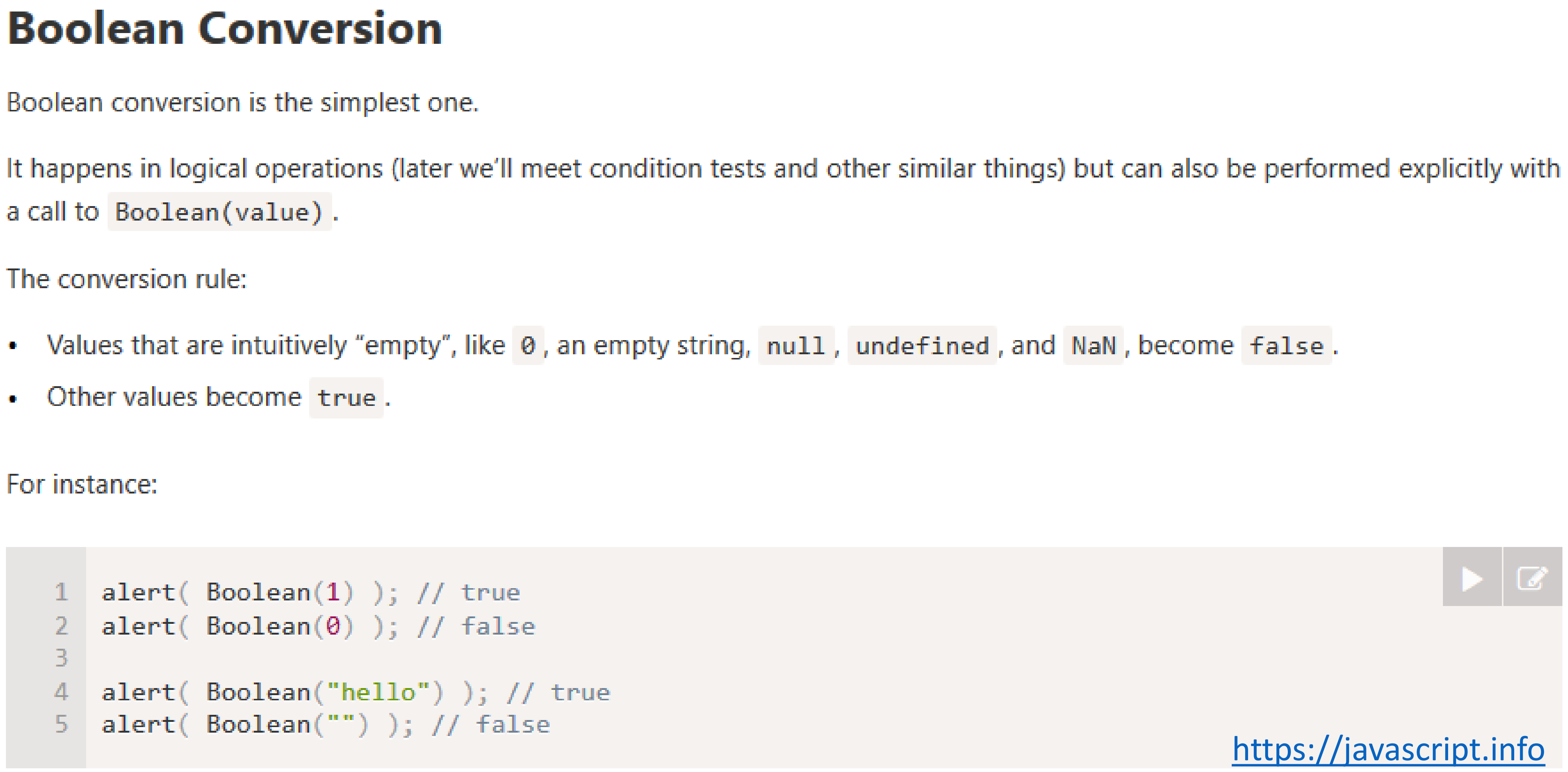
```
alert( "6" / "2" ); // 3, strings are converted to numbers
```

We can use the `Number(value)` function to explicitly convert a `value` to a number:

```
let str = "123";
alert(typeof str); // string

let num = Number(str); // becomes a number 123

alert(typeof num); // number
```

https://javascript.info

Type Conversions



Boolean Conversion

Boolean conversion is the simplest one.

It happens in logical operations (later we'll meet condition tests and other similar things) but can also be performed explicitly with a call to `Boolean(value)`.

The conversion rule:

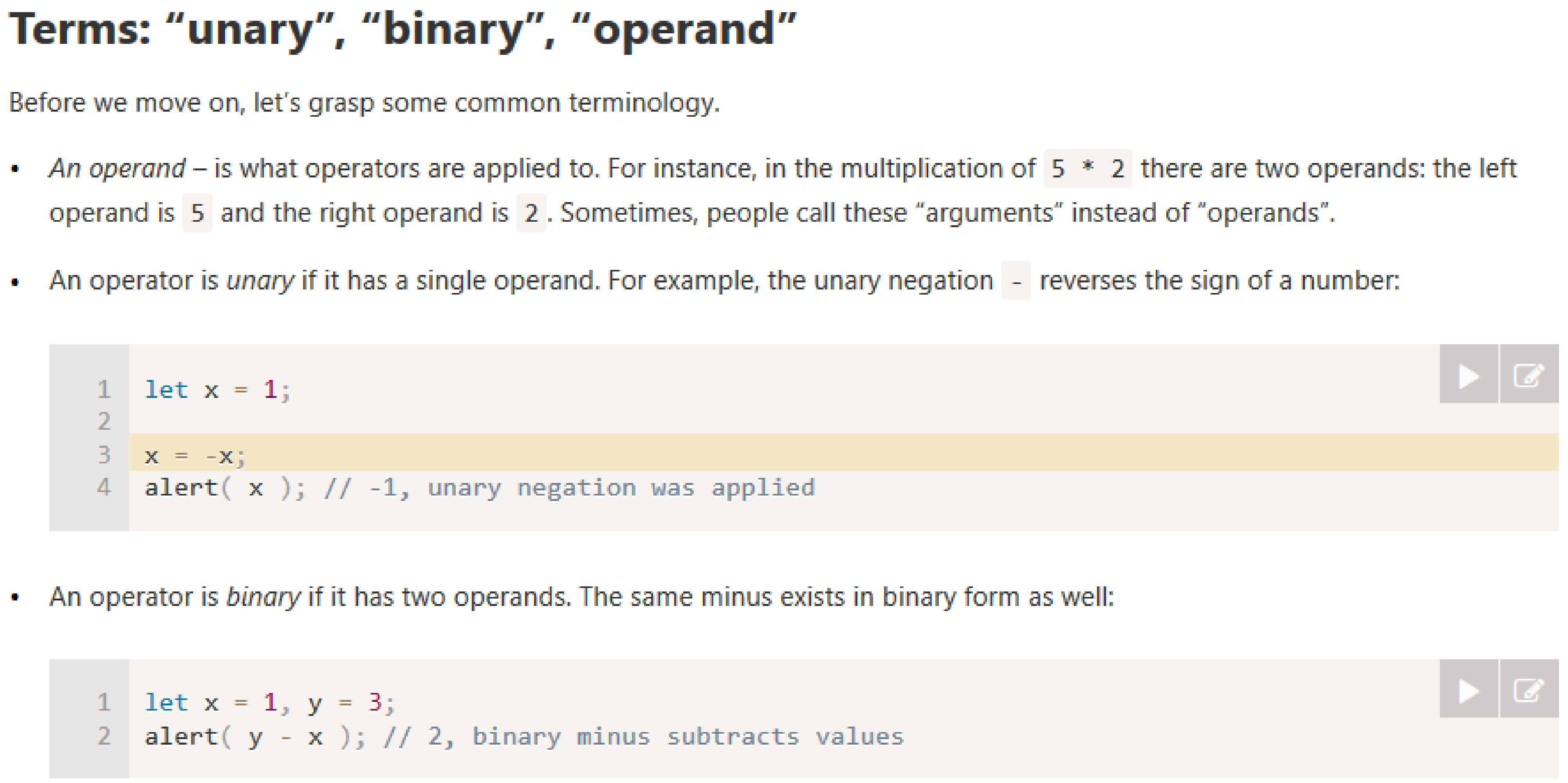
- Values that are intuitively "empty", like `0`, an empty string, `null`, `undefined`, and `NaN`, become `false`.
- Other values become `true`.

For instance:

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

alert( Boolean("hello") ); // true
alert( Boolean("") ); // false
```

https://javascript.info

Operators



Terms: “unary”, “binary”, “operand”

Before we move on, let’s grasp some common terminology.

- *An operand* – is what operators are applied to. For instance, in the multiplication of `5 * 2` there are two operands: the left operand is `5` and the right operand is `2`. Sometimes, people call these “arguments” instead of “operands”.
- An operator is *unary* if it has a single operand. For example, the unary negation `-` reverses the sign of a number:

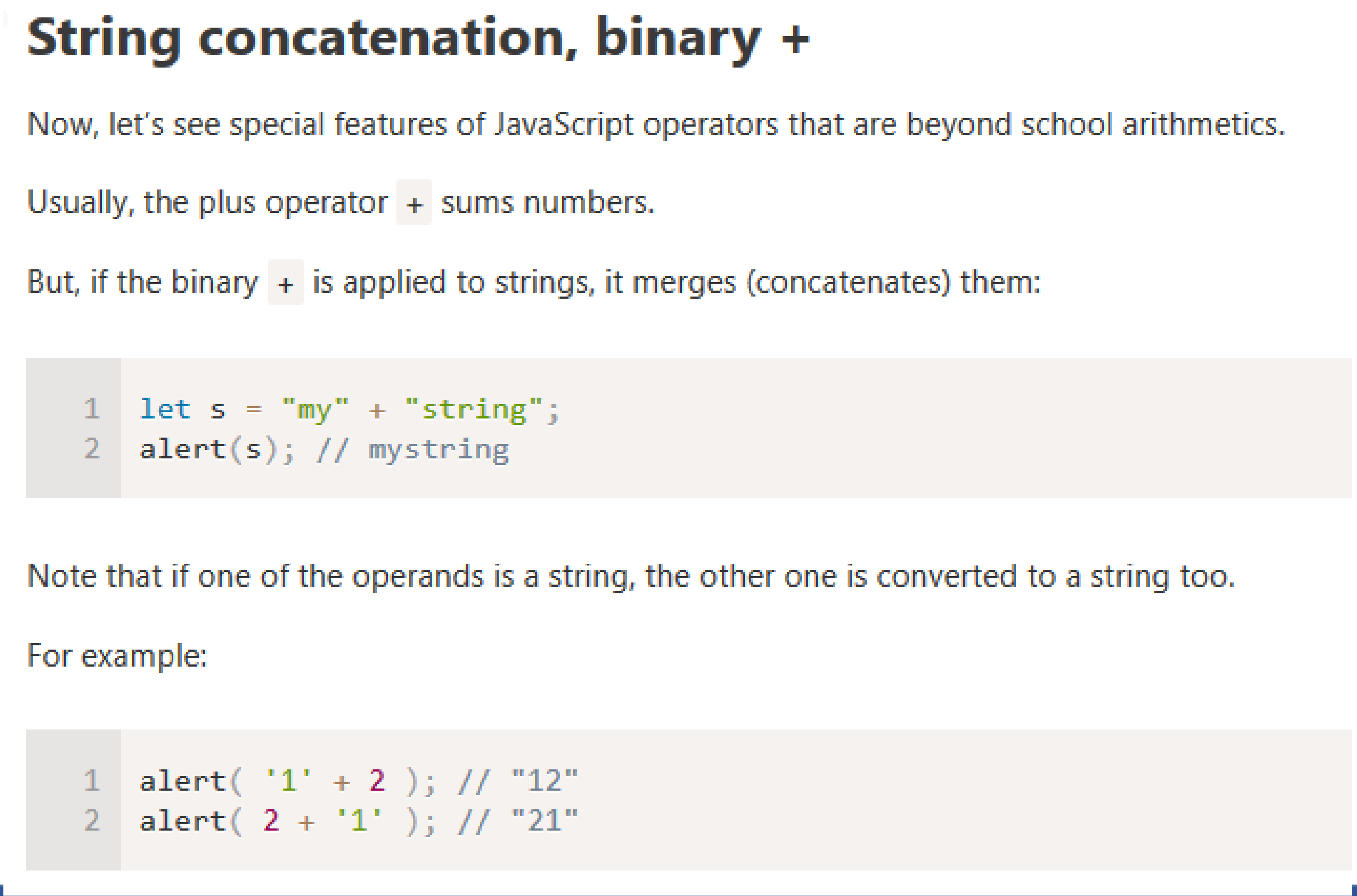
```
let x = 1;

x = -x;
alert( x ); // -1, unary negation was applied
```

- An operator is *binary* if it has two operands. The same minus exists in binary form as well:

```
let x = 1, y = 3;
alert( y - x ); // 2, binary minus subtracts values
```

https://javascript.info

Operators



String concatenation, binary +

Now, let's see special features of JavaScript operators that are beyond school arithmetics.

Usually, the plus operator `+` sums numbers.

But, if the binary `+` is applied to strings, it merges (concatenates) them:

```
let s = "my" + "string";
alert(s); // mystring
```

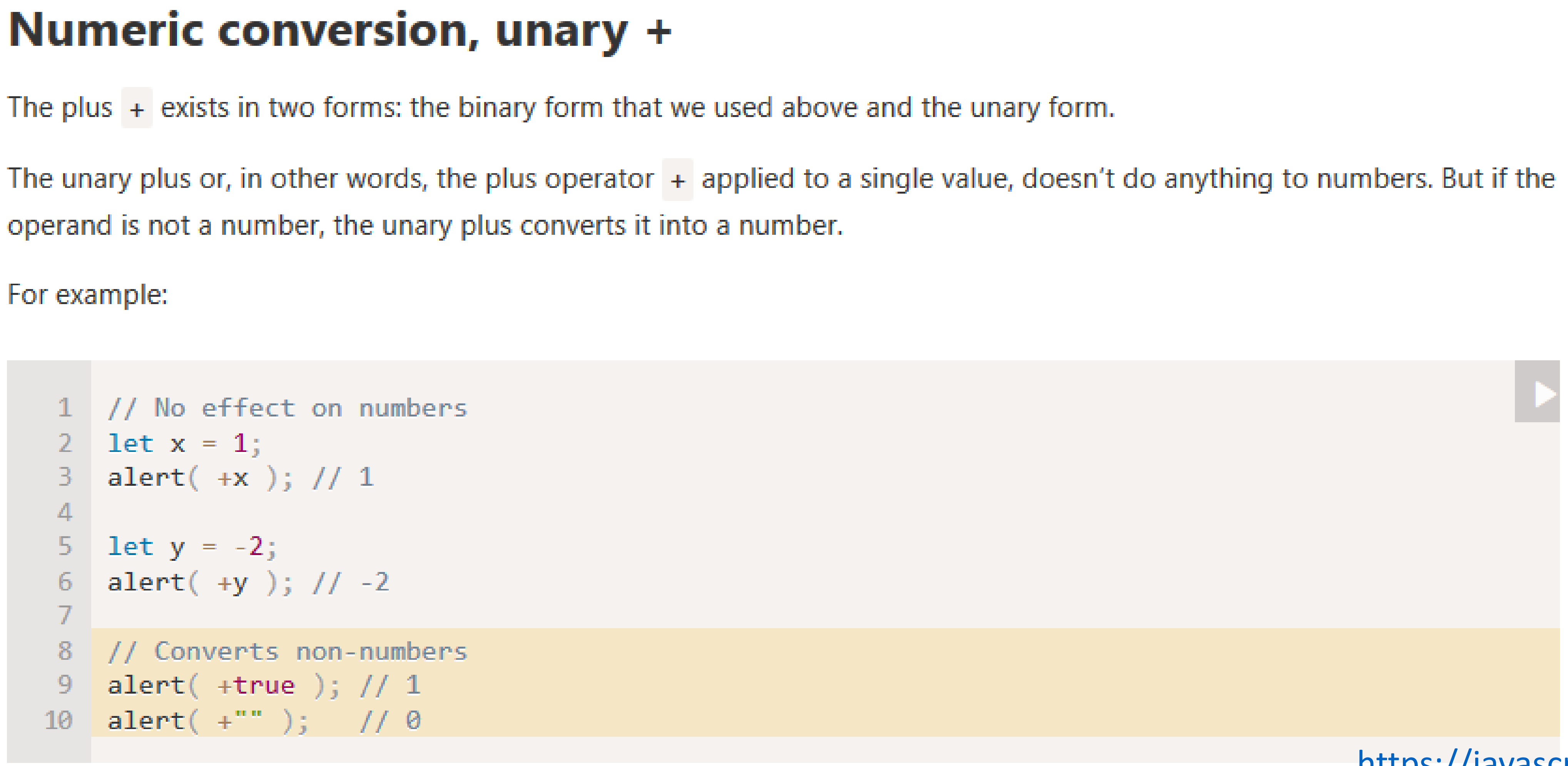
Note that if one of the operands is a string, the other one is converted to a string too.

For example:

```
alert( '1' + 2 ); // "12"
alert( 2 + '1' ); // "21"
```

https://javascript.info

Operators



Numeric conversion, unary +

The plus `+` exists in two forms: the binary form that we used above and the unary form.

The unary plus or, in other words, the plus operator `+` applied to a single value, doesn't do anything to numbers. But if the operand is not a number, the unary plus converts it into a number.

For example:

```javascript
// No effect on numbers
let x = 1;
alert( +x ); // 1

let y = -2;
alert( +y ); // -2

// Converts non-numbers
alert( +true ); // 1
alert( +"" );   // 0
```

https://javascript.info

Operators

The binary plus would add them as strings:

```
let apples = "2";
let oranges = "3";

alert( apples + oranges ); // "23", the binary plus concatenates strings
```

If we want to treat them as numbers, we need to convert and then sum them:

```
let apples = "2";
let oranges = "3";

// both values converted to numbers before the binary plus
alert( +apples + +oranges ); // 5

// the longer variant
// alert( Number(apples) + Number(oranges) ); // 5
```

https://javascript.info

Operators

Assignment

```
let x = 2 * 2 + 1;

alert( x ); // 5
```

Remainder %

The remainder operator `%`, despite its appearance, is not related to percents.

The result of `a % b` is the remainder of the integer division of `a` by `b`.

For instance:

```
alert( 5 % 2 ); // 1 is a remainder of 5 divided by 2
alert( 8 % 3 ); // 2 is a remainder of 8 divided by 3
alert( 6 % 3 ); // 0 is a remainder of 6 divided by 3
```

Exponentiation **

The exponentiation operator `**` is a recent addition to the language.

For a natural number `b`, the result of `a ** b` is `a` multiplied by itself `b` times.

For instance:

```
alert( 2 ** 2 ); // 4  (2 * 2)
alert( 2 ** 3 ); // 8  (2 * 2 * 2)
alert( 2 ** 4 ); // 16 (2 * 2 * 2 * 2)
```

The operator works for non-integer numbers as well.

For instance:

```
alert( 4 ** (1/2) ); // 2 (power of 1/2 is the same as a square root, that's maths)
alert( 8 ** (1/3) ); // 2 (power of 1/3 is the same as a cubic root)
```

https://javascript.info

Operators (Incremental)

Increasing or decreasing a number by one is among the most common numerical operations.

So, there are special operators for it:

- **Increment** `++` increases a variable by 1:

```
let counter = 2;
counter++;        // works the same as counter = counter + 1, but is shorter
alert( counter ); // 3
```

- **Decrement** `--` decreases a variable by 1:

```
let counter = 2;
counter--;        // works the same as counter = counter - 1, but is shorter
alert( counter ); // 1
```

```
let counter = 1;
alert( 2 * ++counter ); // 4
```

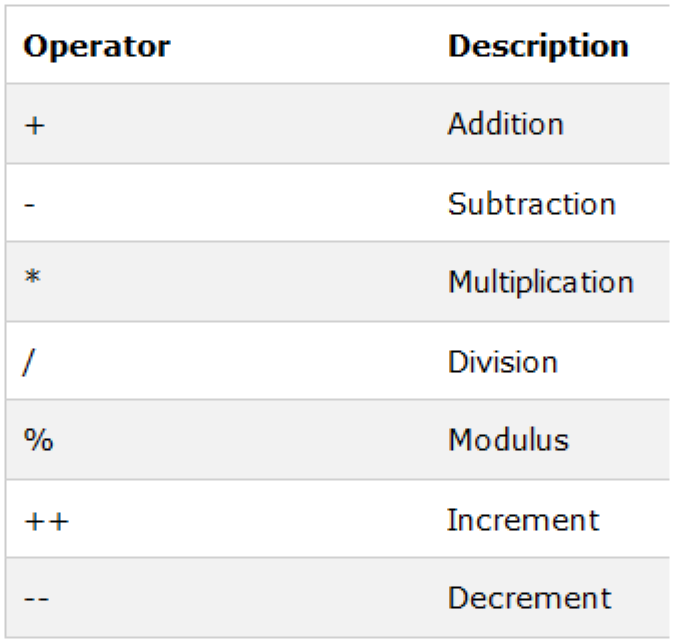
Compare with:

```
let counter = 1;
alert( 2 * counter++ ); // 2, because counter++ returns the "old" value
```

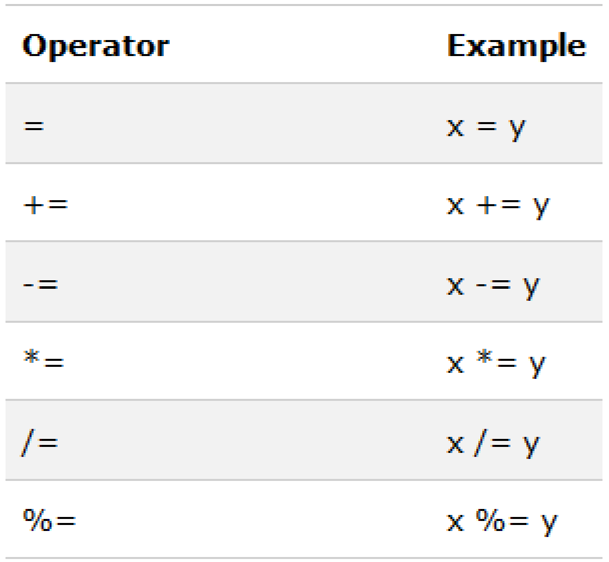
https://javascript.info

Operator Aritmatika

- Sama dengan Bahasa pemrograman lainnya



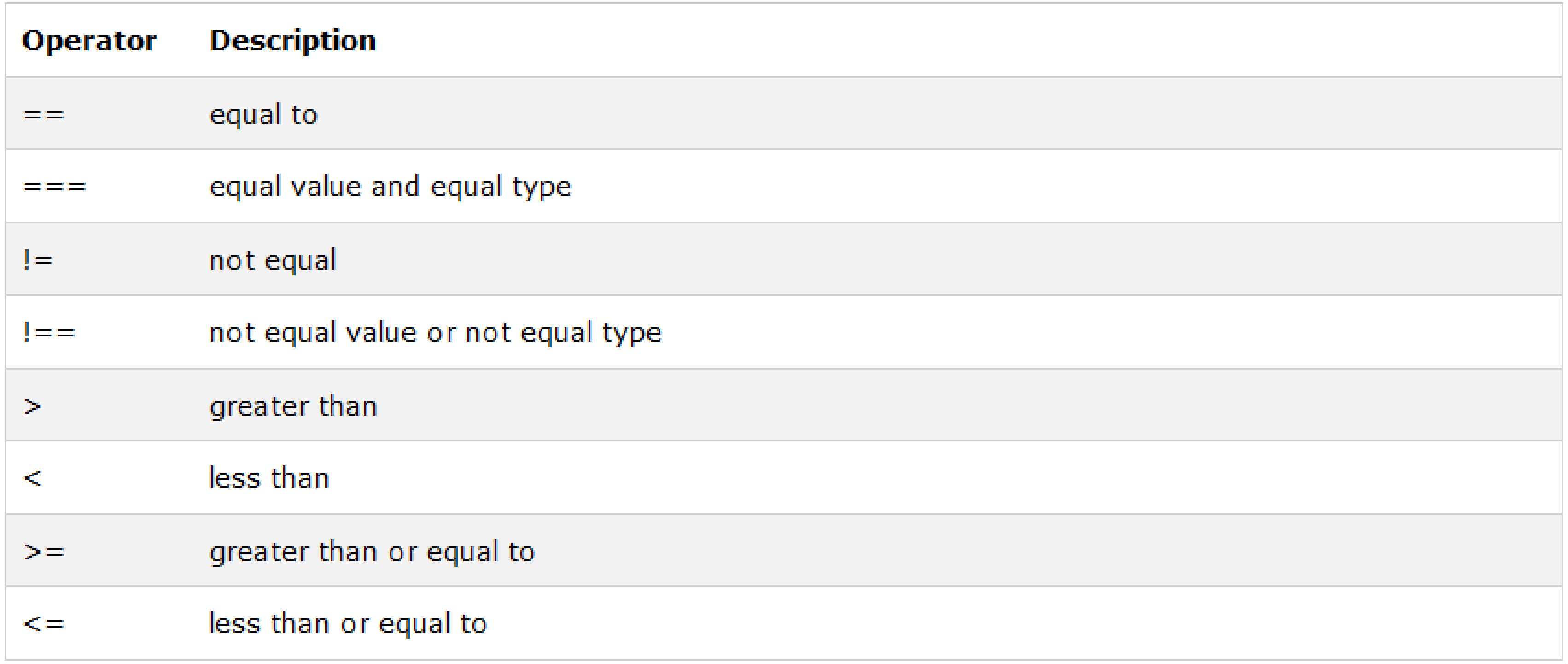
Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement



Operator	Example
=	x = y
+=	x += y
-=	x -= y
*=	x *= y
/=	x /= y
%=	x %= y

Operator Logika

JavaScript Comparison and Logical Operators



Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

Referensi

- This slide content is from http://www.w3schools.com/js
- Slide perkuliahan web UI