# M02 - JavaScript Fundamentals

•••

## Data Types

## Data Types

1. Definition
2. Number
3. BigInt
4. String
5. Boolean
6. Null and Undefined
7. Object and Symbol
8. Type Conversions

## Data Types

1. Definition

# Data Types

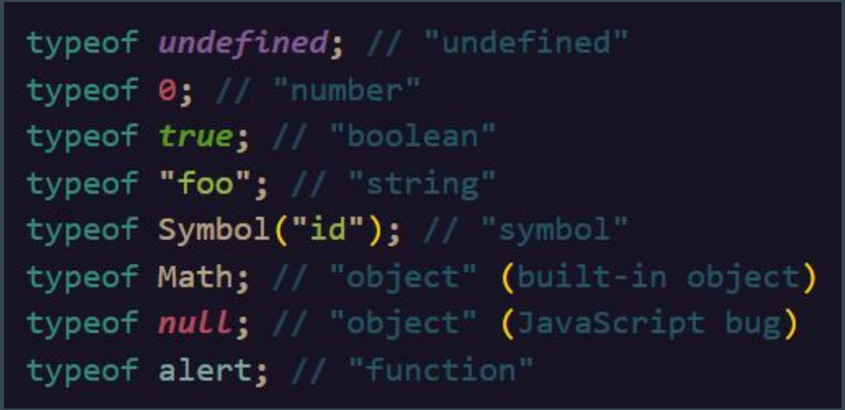
## 1. Definition

- A value in JavaScript is always of a certain type (e.g. a string, number)
- There are eight basic data types in JavaScript:
  - Number for numbers of any type: integer or floating point
  - BigInt to represent integers of arbitrary length
  - String for texts. Can have one or more characters, there is no separate single character type
  - Boolean for true/false
  - null for unknown values: an independent type that has a single null value
  - undefined for unassigned values: an independent type that has a single undefined value
  - Object for more complex data structures
  - Symbol for unique identifiers

## Data Types

### 1. Definition

- Type checking is done by the typeof operator
- Returns the type of the argument. It is useful when we want to process values of different types differently or just do a quick check.
- Supports two forms of syntax:
  - As an operator: typeof x
  - As a function: typeof(x)
- The return is 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" (built-in object)
typeof null; // "object" (JavaScript bug)
typeof alert; // "function"
```

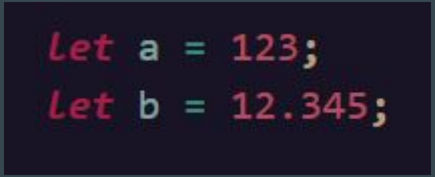
## Data Types

2. Number

# Data Types

## 2. Number

- The number type represents integers and floating point numbers



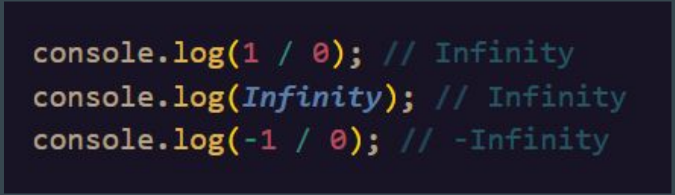
```
let a = 123;
let b = 12.345;
```

- In addition to regular numbers, there are so-called “special numeric values” that also belong to this type of data: Infinity, -Infinity and NaN

# Data Types

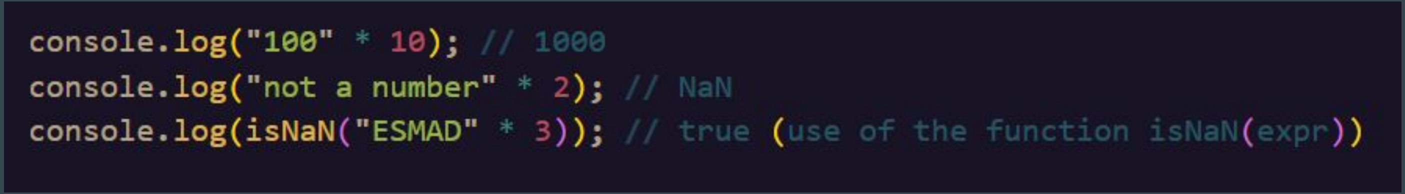
## 2. Number

- The Infinity value represents the mathematical infinity ∞. It is a special value that is greater than any number. We can get it as a result of a division by zero:



```
console.log(1 / 0); // Infinity
console.log(Infinity); // Infinity
console.log(-1 / 0); // -Infinity
```

- The NaN (Not a Number) value represents a computational error - incorrect or undefined mathematical operation:



```
console.log("100" * 10); // 1000
console.log("not a number" * 2); // NaN
console.log(isNaN("ESMAD" * 3)); // true (use of the function isNaN(expr))
```

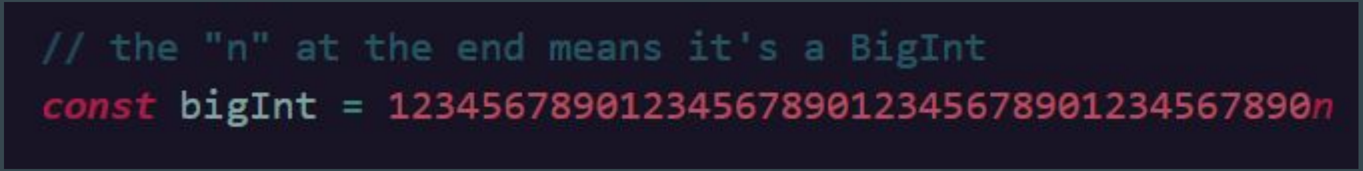
# Data Types

3. BigInt

# Data Types

## 3. BigInt

- In JavaScript, the “number” type cannot represent integer values larger than ($2^{53}$-1) (that’s 9007199254740991), or less than -($2^{53}$-1) for negatives
- For most purposes that’s quite enough, but sometimes we need really big numbers, e.g. for cryptography or microsecond-precision timestamps.
- BigInt type represents integers of arbitrary length.
- A BigInt value is created by appending n to the end of an integer:



```
// the "n" at the end means it's a BigInt
const bigInt = 1234567890123456789012345678901234567890n
```

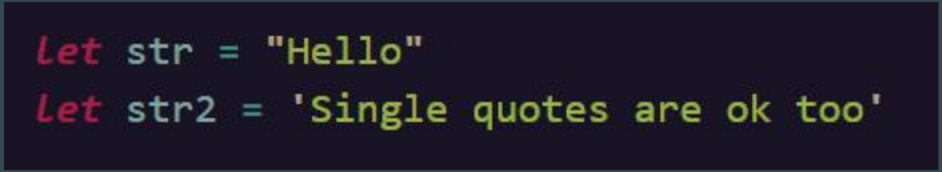
# Data Types

## 4. String

# Data Types

## 4. String

- Used to store and manipulate text (string of alphanumeric characters)
- A string value can be between:
  - Double quotes: "Olá"
  - Single quotes: 'Olá'
  - Backticks: `Olá`
- There is no difference between double quotes and single quotes in JavaScript

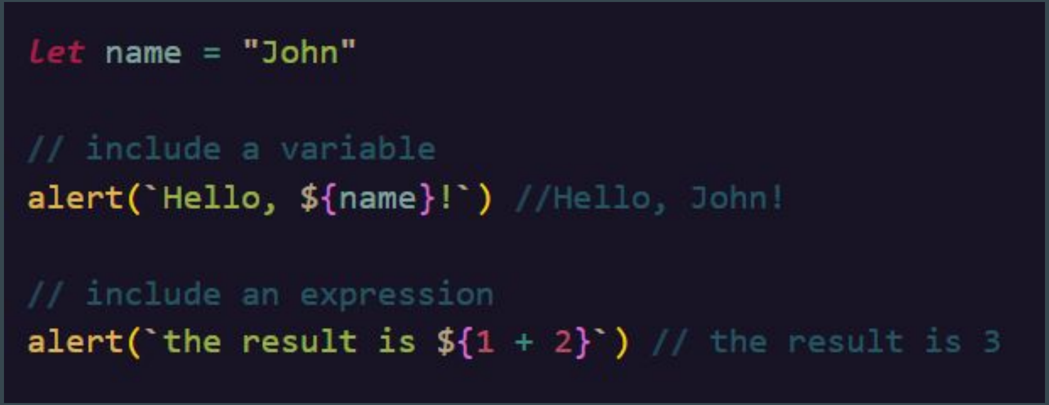


```
let str = "Hello"
let str2 = 'Single quotes are ok too'
```

# Data Types

## 4. String

- Backticks are “extended functionality” quotes because they allow you to incorporate variables and expressions in a string, wrapping them in ${...}



```
let name = "John"

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

// include an expression
alert(`the result is ${1 + 2}`) // the result is 3
```

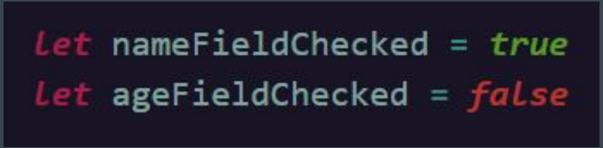
# Data Types

## 5. Boolean

## Data Types

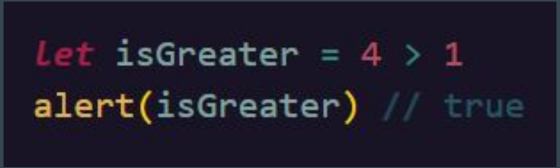
### 5. Boolean

- 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"



```
let nameFieldChecked = true
let ageFieldChecked = false
```

- Boolean values also appear as a result of comparisons:



```
let isGreater = 4 > 1
alert(isGreater) // true
```

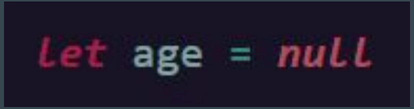
## Data Types

6. Null and Undefined

## Data Types

### 6. Null and Undefined

- Null
  - The null value does not belong to any of the types described previously
  - It forms a separate type that contains only the null value:



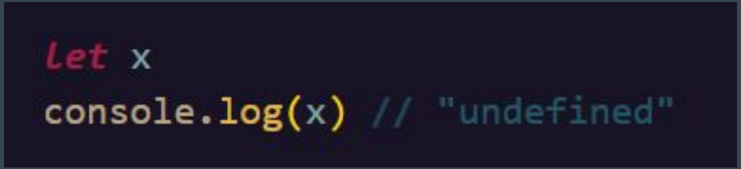
```
let age = null
```

  - It is not a “reference to a non-existent object” or “null pointer” as in other languages
  - It is just a special value that represents "nothing", "empty" or "unknown value"
  - The above code indicates that the age is unknown or is empty for some reason

## Data Types

### 6. Null and Undefined

- Undefined
  - Means “unassigned value”
  - If a variable is declared, but no value is assigned, its value is undefined:



```
let x
console.log(x) // "undefined"
```

  - Good practices:
    - use null to assign a value "empty" or "unknown" to a variable
    - use undefined to check if a value has been assigned to a variable

# Data Types

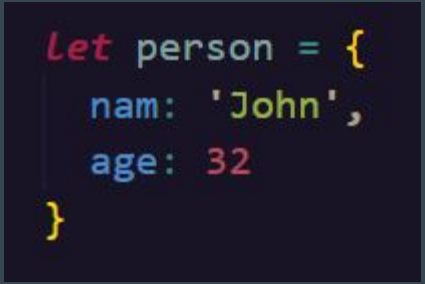
## 7. Object and Symbol

# Data Types

## 7. Object and Symbol

- Object
  - The object type is special
  - All other types are called "primitives" because their values can contain only one thing (be it a string or a number...)
  - Objects are used to store collections of data/complex entities
- Symbol
  - Special type used to create unique identifiers for objects

- Both types will be discussed later in these slides



```
let person = {
  nam: 'John',
  age: 32
}
```

# Data Types

## 8. Type Conversions

## Data Types

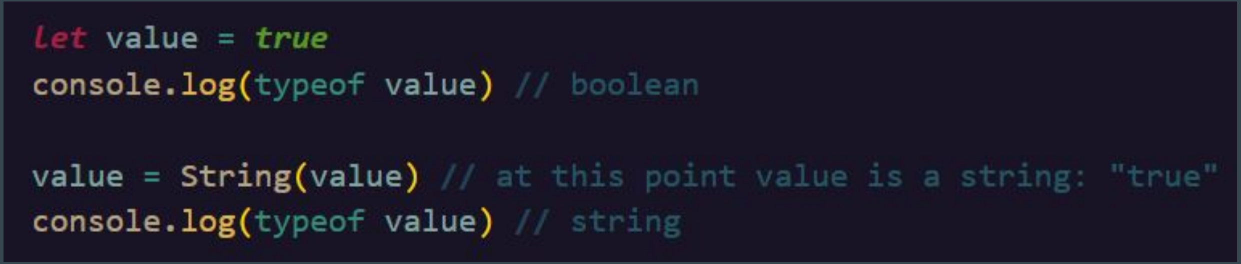
### 8. Type Conversions

- Most of the time, operators and functions automatically convert the values assigned to them to the correct type. This is called type conversion.
- For example, the alert automatically converts any value to a string in order to display it. Mathematical operations convert values into numbers.
- The three most used type conversions are:
  - conversion to string
  - conversion to number
  - conversion to boolean

# Data Types

## 8. Type Conversions

- Conversion to string
  - String conversion happens when we need the string form of a value
  - For example, alert(value) does this to show the value
  - We can also call the String(value) function to convert a value to a string:



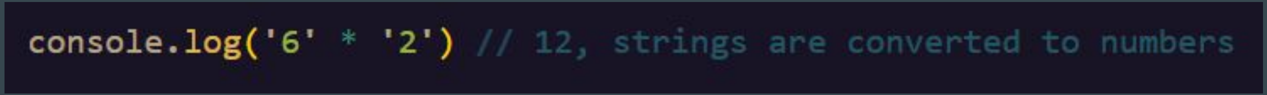
```
let value = true
console.log(typeof value) // boolean

value = String(value) // at this point value is a string: "true"
console.log(typeof value) // string
```

## Data Types

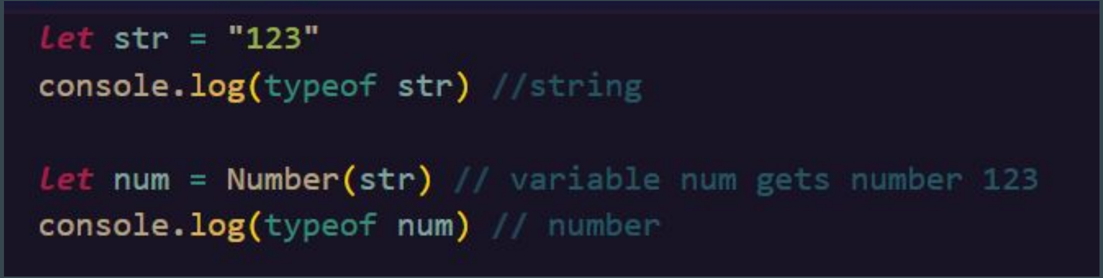
### 8. Type Conversions

- Conversion to number
  - Numeric conversion happens automatically in mathematical functions and expressions
  - For example, when multiplication * is applied to non-numbers:
  -



```
console.log('6' * '2') // 12, strings are converted to numbers
```

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



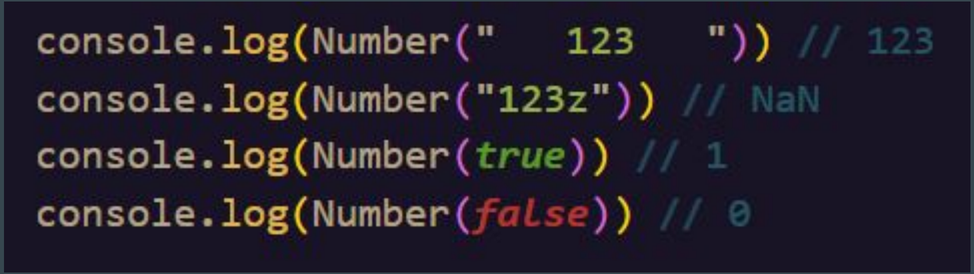
```
let str = "123"
console.log(typeof str) //string

let num = Number(str) // variable num gets number 123
console.log(typeof num) // number
```

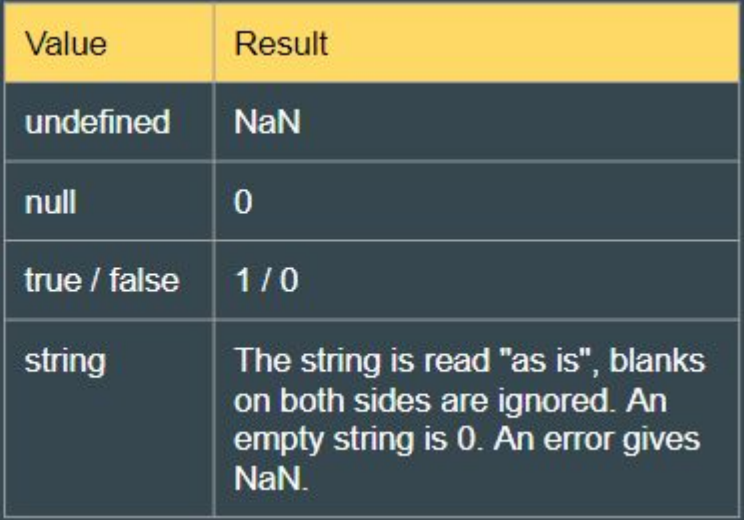
## Data Types

### 8. Type Conversions

- Conversion to number (rules)



```
console.log(Number("   123   ")) // 123
console.log(Number("123z")) // NaN
console.log(Number(true)) // 1
console.log(Number(false)) // 0
```

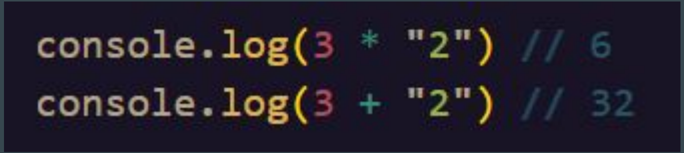


| Value | Result |
|---|---|
| undefined | NaN |
| null | 0 |
| true / false | 1 / 0 |
| string | The string is read "as is", blanks on both sides are ignored. An empty string is 0. An error gives NaN. |

# Data Types

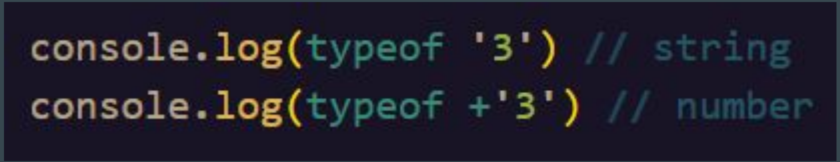
## 8. Type Conversions

- Conversion to number
    - Almost all mathematical operations convert values into numbers.
    - A notable exception is the addition +. If one of added values is a string, the other is also converted to a string by joining them:



```
console.log(3 * "2") // 6
console.log(3 + "2") // 32
```

- Using the + sign immediately before a string it will try to convert to number

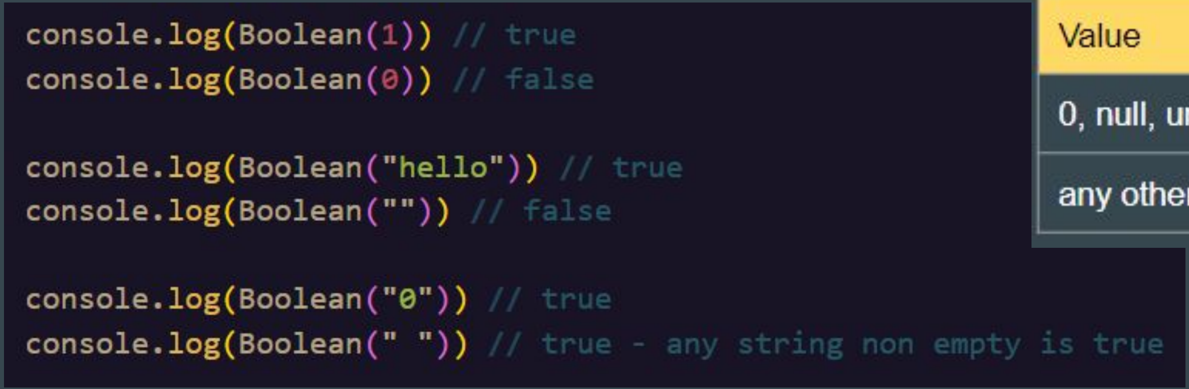


```
console.log(typeof '3') // string
console.log(typeof +'3') // number
```

# Data Types

## 8. Type Conversions

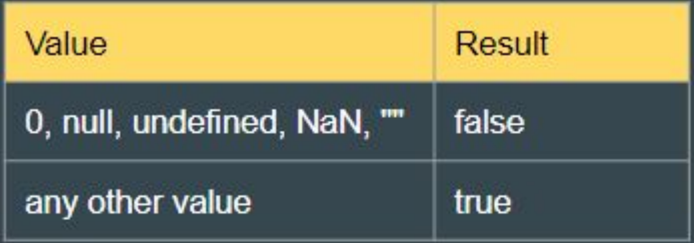
- Conversion to boolean
    - In logical operations (later, we will learn about condition tests and similar things)
    - It can also be performed explicitly with the Boolean(value) function



```
console.log(Boolean(1)) // true
console.log(Boolean(0)) // false

console.log(Boolean("hello")) // true
console.log(Boolean("")) // false

console.log(Boolean("0")) // true
console.log(Boolean(" ")) // true - any string non empty is true
```



| Value | Result |
|---|---|
| 0, null, undefined, NaN, "" | false |
| any other value | true |