

JavaScript Essentials

Numbers and Operators



Table of Contents

1. Overview
2. Arithmetic operators
3. Assignment operators
4. Increment/Decrement operators
5. Comparison operators

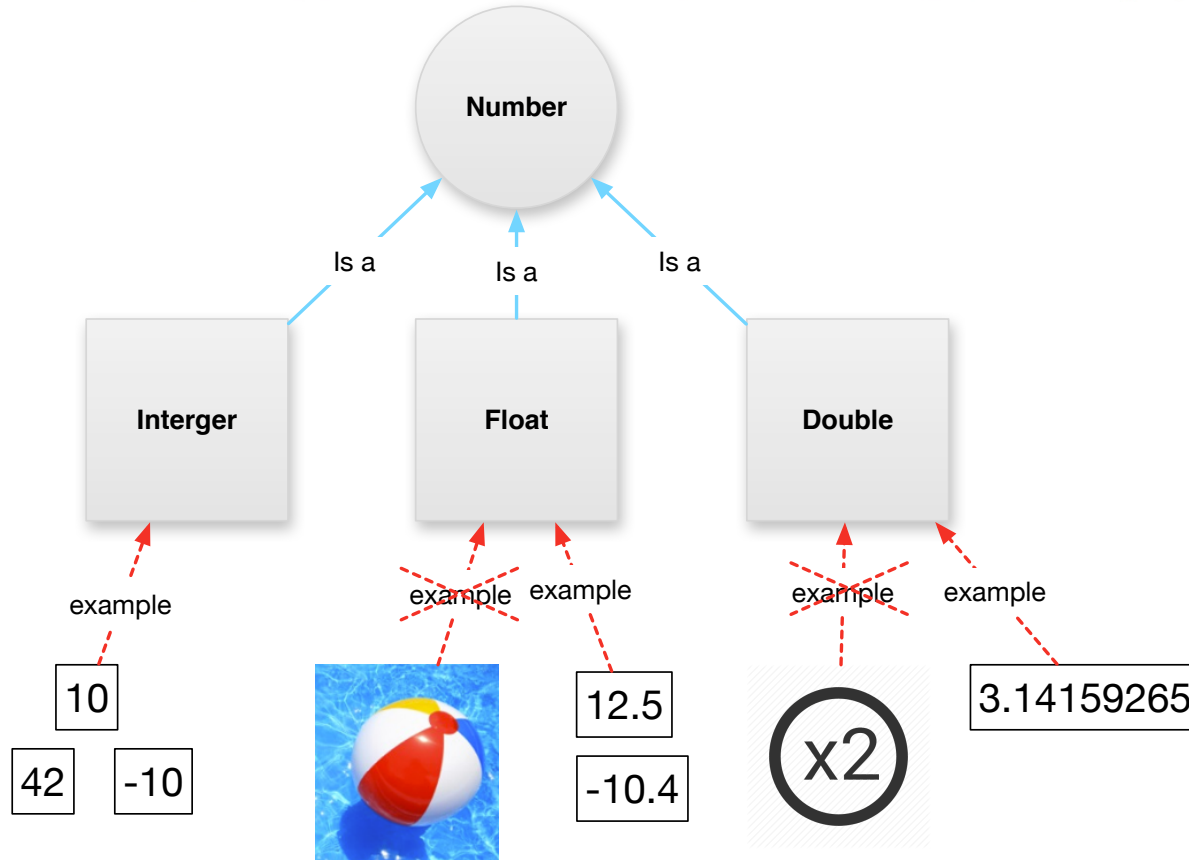
Lesson Objectives

- Understand Numbers in JavaScript
- Able to compute numbers using Arithmetic operators
- Able to use assignment operators for cleaner code
- Able to compare numbers with comparison operators

Section 1

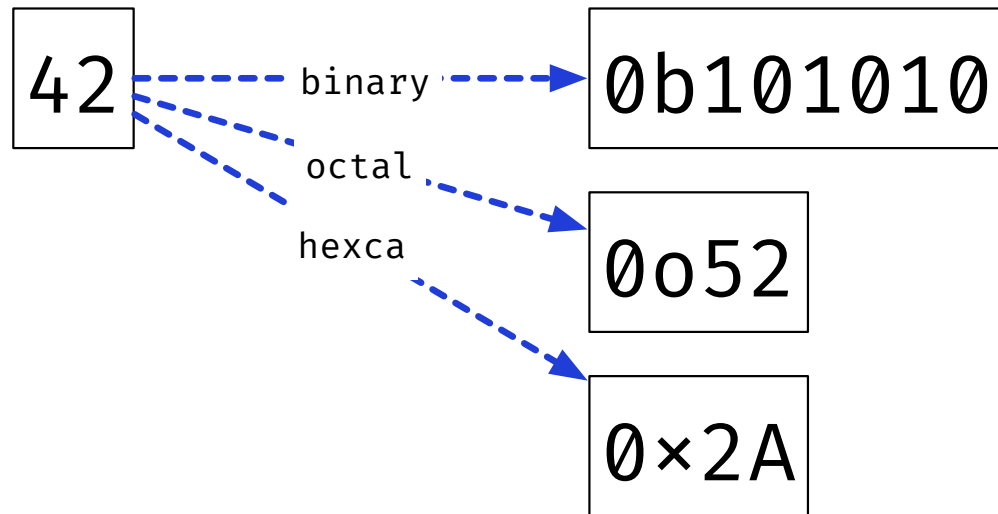
Overview

Overview – Types of numbers



We even have different types of number systems:

- **Binary** (lowest level)
- **Octal** (base 8)
- **Decimal** (base 10)
- **Hexadecimal** (base 16)



Overview – It's all numbers

```
> var n = 42;
```

```
< undefined
```

```
> var PI = 3.1415;
```

```
< undefined
```

```
> typeof n;
```

```
< "number"
```

```
> typeof PI;
```

```
< "number"
```

```
> |
```

Different type

Same "kind" in
JavaScript

Overview – Useful Number methods

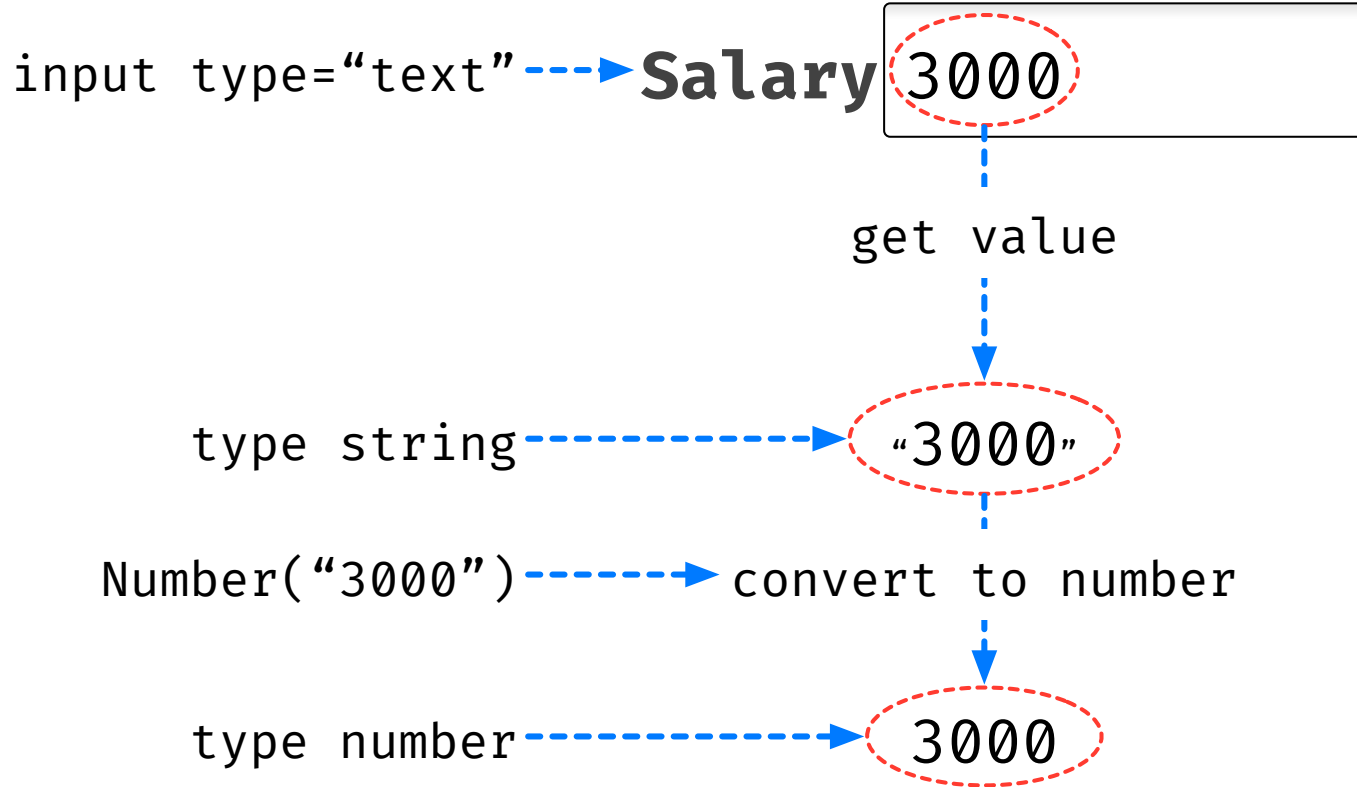
```
> let lotsOfDecimal = 1.766584958675746364;  
< undefined  
  
> lotsOfDecimal  
< 1.7665849586757463  
  
> let twoDecimalPlaces = lotsOfDecimal.toFixed(2);  
< undefined  
  
> twoDecimalPlaces  
< "1.77"
```

Result after rounded up

Return a string not number

Take 2 number after '.'

Overview – Converting to number data types



- Different types of number such as **Integers, Float, Doubles**
- Different systems to represent number: **Binary, Octal, Decimal, Hexadecimal**
- In JavaScript, it's all numbers
- Use `.toFixed()` to round your number to a fixed number of decimal places
- Use `Number()` to convert text to number

Section 2

Arithmetic operators

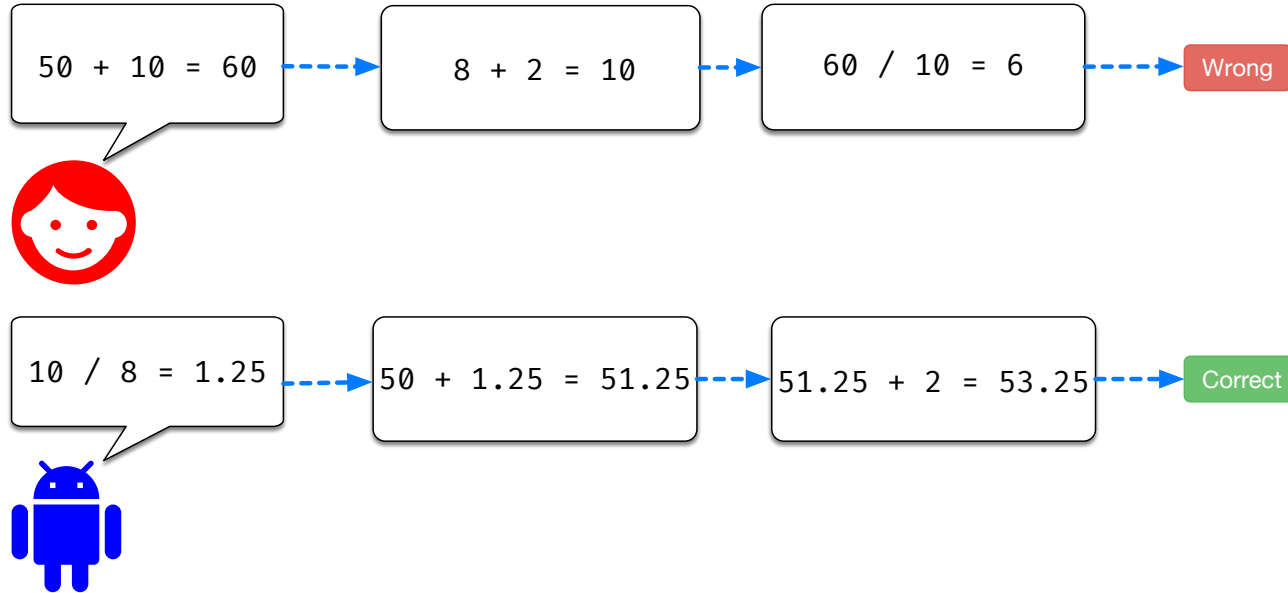
Operator	Name	Purpose	Example
+	Addition	Adds two numbers together	$12 + 30$
-	Subtraction	Subtracts the right number from the left	$20 - 15$
*	Multiplication	Multiplies two numbers together.	$3 * 7$
%	Remainder (modulo)	Returns the remainder left over after you've divided the left number by the right number.	$8 \% 3 = 2$
**	Exponent	Raises a base number to the exponent power	$5 ** 2 = 25$

Practice Arithmetic operators

Arithmetic operators – Operator precedence

What is the result ?

> 50 + 10 / 8 + 2 ;



- 6 arithmetic operators: $+$, $-$, $*$, $/$, $\%$, $**$
- $\%$ has many commonly use cases. Make sure you understand it
- Take note of operator precedence else you won't get the correct result

Section 3

Assignment operators

Assignment operators

Operator	Name	Purpose	Example	Shortcut for
<code>+=</code>	Addition assignment	Adds the value on the right to the variable value on the left, then returns the new variable value	<code>x = 3; x += 4;</code>	<code>x = 3; x = x + 4;</code>
<code>-=</code>	Subtraction assignment	Subtracts the value on the right from the variable value on the left, and returns the new variable value	<code>x = 6; x -= 3;</code>	<code>x = 6; x = x - 3;</code>
<code>*=</code>	Multiplication assignment	Multiplies the variable value on the left by the value on the right, and returns the new variable value	<code>x = 2; x *= 3;</code>	<code>x = 2; x = x * 3;</code>
<code>/=</code>	Division assignment	Divides the variable value on the left by the value on the right, and returns the new variable value	<code>x = 10; x /= 5;</code>	<code>x = 10; x = x / 5;</code>

- Assignment operators provide useful shortcuts to keep our code cleaner and more efficient
- Can use other variables on the right hand side of each expression as well

Section 4

Increment/Decrement operators

- Sometimes you'll want to repeatedly add or subtract one to or from a numeric variable value.
- This can be conveniently done using the increment (++) and decrement(--) operators.

Increment/Decrement operators

```
> var count = 10;
```

```
< undefined
```

```
> count++;
```

Increment expression
(return then increment)

returned value
from expression

10

```
> count;
```

variable access expression

11

Increment/Decrement operators

```
> var count = 10;
```

```
< undefined
```

```
> ++count;
```

Increment expression
(increment then return)

```
< 11
```

returned value
from expression

```
> count;
```

variable access expression

```
< 11
```

- Provide a convenient mechanism to repeatedly add or subtract one to or from a numeric value
- **Syntax:** variable++, ++variable
- variable++ is same as variable += 1 then return the value **before** increment its value
- ++variable is same as variable += 1 then return the value **after** increment its value

Section 5

Comparison operators

Comparison operators

Operator	Name	Purpose	Example
===	Strict equality	Tests whether the left and right are identical	$5 === 2 + 4$
!==	Strict-non-equality	Tests whether the left and right are not identical	$5 !== 2 + 3$
<	Less than	Tests whether the left value is smaller than the right one.	$10 < 6$
>	Greater than	Tests whether the left value is greater than the right one.	$10 > 20$
<=	Less than or equal to	Tests whether the left value is smaller than or equal to the right one.	$3 <= 2$
>=	Greater than or equal to	Tests whether the left value is greater than or equal to the right one.	$5 >= 4$

- If you want to compare numeric number use Comparison operators
- The result of a comparison is always a **Boolean**
- Always use strict comparison operator as it test the equality of both the values and their datatypes

Thank you

Q&A

