@m34ra

AN ADVENTURE IN JAVASCRIPT
ADDITION AND TYPE COERCION

HOW DOES JAVASCRIPT MATH?

Secure https://www.google.com/search?q=javacript+addition&oq=javacript+addition&aqs=chrome..69i57j0l5

Google javascript addition

All Videos Images News Shopping More Settings Tools

About 91,800,000 results (0.37 seconds)

JavaScript Operators - W3Schools

https://www.w3schools.com/js/js_operators.asp

Adding Strings and Numbers. Adding two numbers, will return the **sum**, but adding a number and a string will return a string: ...

JavaScript Arithmetic - W3Schools

https://www.w3schools.com/js/js_arithmetic.asp

Is the result of example above the same as 150 * 3, or is it the same as 100 + 150? Is the **addition** or the multiplication done first? As in traditional school ...

Addition is not working in JavaScript - Stack Overflow

stackoverflow.com/questions/8377410/addition-is-not-working-in-javascript

Dec 4, 2011 - I am trying to learn **Javascript**. Here I am confused with the following ... One or both of the variables is a string instead of a number. This makes the ...

math - Javascript (+) sign concatenates instead of giving sum of ...

stackoverflow.com/.../javascript-sign-concatenates-instead-of-giving-sum-of-variables

May 11, 2011 - Why when i use this: (assuming i=1) divID = "question-" + i+1; ... Use this instead: var divID = "question-" + (i+1). It's a fairly common problem and ...

Arithmetic operators - JavaScript | MDN

https://developer.mozilla.org › ... › JavaScript reference › Expressions and operators

Jump to **Addition (+)** - The **addition** operator produces the **sum** of numeric operands or ... Number + Number -> **addition** 1 + 2 // 3 // Boolean + Number ...

stackoverflow.com/questions/8377410/addition-is-not-working-in-javascript

stackoverflow Questions Jobs Documentation BETA Tags Users Search...

Addition is not working in JavaScript

12

I am trying to learn Javascript. Here I am confused with the following code.

http://rendera.heroku.com/usercode/eae2b0f40cf503b36ee346f5c511b0e29fc82f9e

When I put `x+y` in the function it is going wrong. For example `2+2=22` , `5+7=57`

But `/` , `*` , `-` are working. Why is `+` not working? Please help me. Thanks a lot in advance

3

javascript addition

```
failbowl:~(master!?) $ jsc
> [] + []

> [] + {}
[object Object]
> {} + []
0
> {} + {}
NaN
>
```

"Wat" - @garybernhardt

GIF

HI, I'M MEARA

@m34ra

A BRIEF INTRODUCTION

OBJECT TYPES

PRIMITIVE TYPES

- Boolean
- Number
- String
- Object
- Null
- Undefined

BOOLEAN

- true
- false

NUMBER

- 0
- 102.20349587
- 8
- -24.2
- 537
- NaN
- +Infinity
- -Infinity

STRING

- “apples”
- “2”
- “table, noun – a piece of furniture with a flat top and one or more legs”
- “pretty much anything you put in quotes”
- “/2p234u a;sldfj09345;”
- “;D”

OBJECT

```
{
    firstName: "Meara",
    lastName: "Charnetzki",
    company: "TableXI",
    numberOfToes: 10,
    fullName: function() {return this.firstName + " " + this.lastName;}
}
```

NULL

- null

UNDEFINED

- undefined

DATA STRUCTURES

ARRAY

- [1, 2, 3, 4]
- [“apples”, 2, true, null]

LET'S DO MATH!

ADDING NUMBERS AND STRINGS

- 2 + 2
- 4
- “apples” + “bananas”
- “applesbananas”
- “2” + “2”
- “22”

“+” ACTUALLY MEANS TWO DIFFERENT THINGS

NUMBER ADDITION

2 + 2 = 4

STRING CONCATENATION

“Hello” + “ ” + “Midwest JS!”

=

“Hello Midwest JS!”

ADDING NUMBERS TO STRINGS

- "2" + 2
- 2 + "2"
- 2 + "apples"

- "22"
- "22"
- "2apples"

COERCION, N. — IMPLICITLY CONVERTING A VALUE FROM ONE TYPE TO ANOTHER

THE ADDITION OPERATOR (+)

$$a + b$$

- If either *a* or *b* is a string
  - We're doing string concatenation (with coercion if necessary)
- Otherwise, if both *a* and *b* are NOT strings
  - We're doing number addition

WHAT IF I WANT TO ADD OTHER THINGS?

- [1, 2, 3, 4] + [5, 6]
- true + true
- undefined + null
- null + null
- [] + {}
- {} + []

- “1,2,3,45,6"
- 2
- NaN
- 0
- "[object Object]”
- 0


7.367
97.367
367
7.367
97.367

11.6.1 The Addition operator (+)

The addition operator either performs string concatenation or numeric addition.

The production *AdditiveExpression* : *AdditiveExpression* + *MultiplicativeExpression* is evaluated as follows:

1. Let *lref* be the result of evaluating AdditiveExpression.
2. Let *lval* be GetValue(*lref*).
3. Let *rref* be the result of evaluating MultiplicativeExpression.
4. Let *rval* be GetValue(*rref*).
5. Let *lprim* be ToPrimitive(*lval*).
6. Let *rprim* be ToPrimitive(*rval*).
7. If Type(*lprim*) is String or Type(*rprim*) is String, then
    a. Return the String that is the result of concatenating ToString(*lprim*) followed by ToString(*rprim*)
8. Return the result of applying the addition operation to ToNumber(*lprim*) and ToNumber(*rprim*). See the Note below 11.6.3.

NOTE 1 No hint is provided in the calls to ToPrimitive in steps 5 and 6. All native ECMAScript objects except Date objects handle the absence of a hint as if the hint Number were given; Date objects handle the absence of a hint as if the hint String were given. Host objects may handle the absence of a hint in some other manner.

NOTE 2 Step 7 differs from step 3 of the comparison algorithm for the relational operators (11.8.5), by using the logical-or operation instead of the logical-and operation.

THE ADDITION OPERATOR (+)

$$a + b$$

- If either *a* or *b* is a string (OR its primitive's default value is a string)
  - string concatenation
  - a.toString() + b.toString()

THE ADDITION OPERATOR (+)

$$a + b$$

- Otherwise, if both *a* and *b* are NOT strings
  - number addition
  - a.toNumber() + b.toNumber()

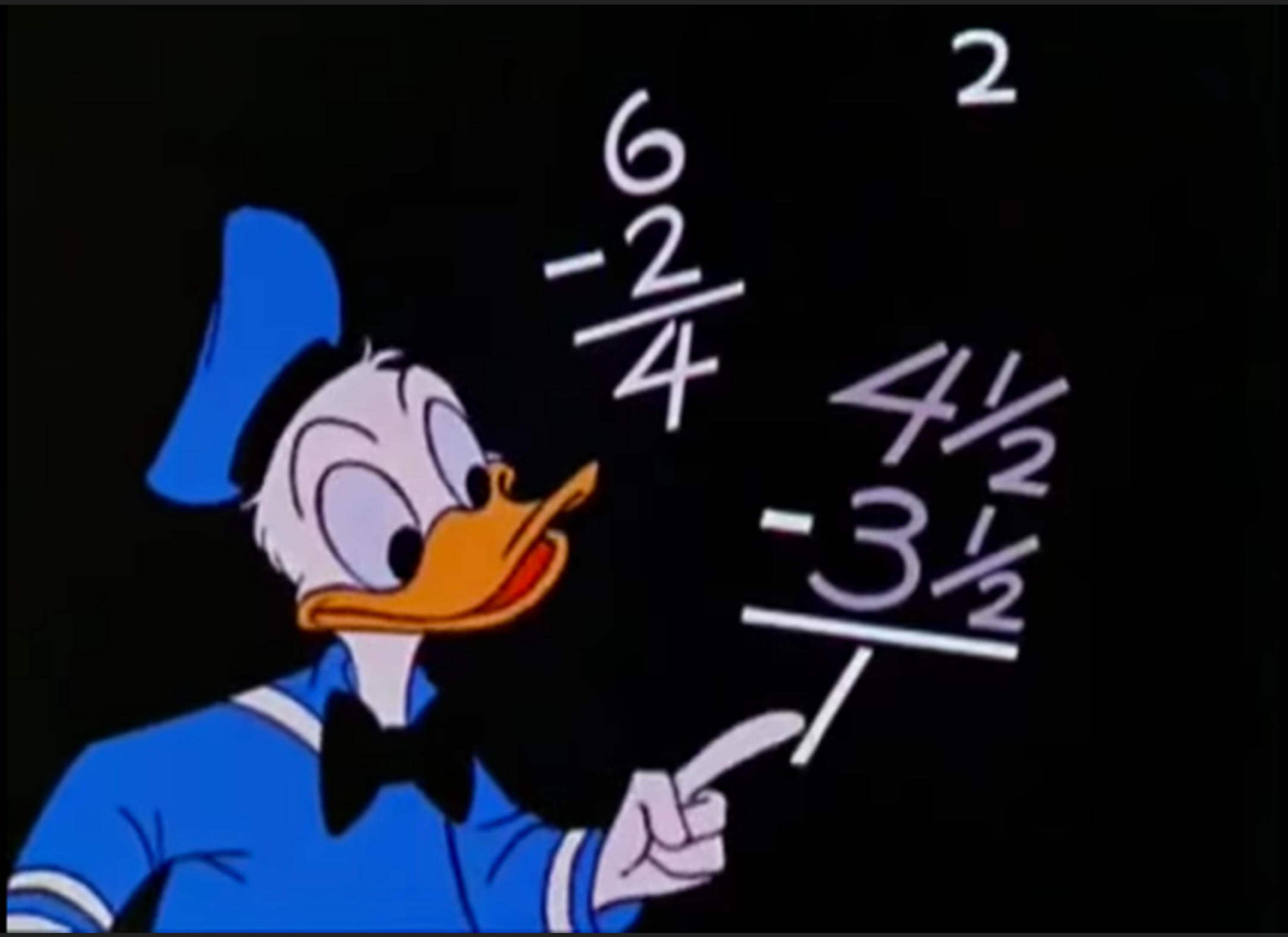
WHAT IF I WANT TO ADD OTHER THINGS?

- [1, 2, 3, 4] + [5, 6]
- true + true
- undefined + null
- null + null
- [] + {}
- {} + []

- “1,2,3,45,6"
- 2
- NaN
- 0
- "[object Object]”
- 0

Table 12 — To Number Conversions

Argument Type	Result
Undefined	**NaN**
Null	**+0**
Boolean	The result is **1** if the argument is **true**. The result is **+0** if the argument is **false**.
Number	The result equals the input argument (no conversion).
String	See grammar and note below.
Object	Apply the following steps: 1. Let *primValue* be ToPrimitive(*input argument*, hint Number). 2. Return ToNumber(*primValue*).


2
6
-2
4
4 1/2
-3 1/2

WHAT IF I WANT TO ADD OTHER THINGS?

- [1, 2, 3, 4] + [5, 6]
- ✓ true + true
- ✓ undefined + null
- ✓ null + null
- [] + {}
- {} + []

- “1,2,3,45,6"
- 2
- NaN
- 0
- "[object Object]”
- 0

THE ADDITION OPERATOR (+)

$$a + b$$

- If either *a* or *b* is a string (OR its primitive's default value is a string)
  - string concatenation
  - a.toString() + b.toString()

Table 10 — ToPrimitive Conversions

Input Type	Result
Undefined	The result equals the *input* argument (no conversion).
Null	The result equals the *input* argument (no conversion).
Boolean	The result equals the *input* argument (no conversion).
Number	The result equals the *input* argument (no conversion).
String	The result equals the *input* argument (no conversion).
Object	Return a default value for the Object. The default value of an object is retrieved by calling the [[DefaultValue]] internal method of the object, passing the optional hint *PreferredType*. The behaviour of the [[DefaultValue]] internal method is defined by this specification for all native ECMAScript objects in 8.12.8.

8.12.8 [[DefaultValue]] (hint)

When the [[DefaultValue]] internal method of *O* is called with hint String, the following steps are taken:

1. Let *toString* be the result of calling the [[Get]] internal method of object *O* with argument "`toString`".
2. If IsCallable(*toString*) is **true** then,
   a. Let *str* be the result of calling the [[Call]] internal method of *toString*, with *O* as the **this** value and an empty argument list.
   b. If *str* is a primitive value, return *str*.
3. Let *valueOf* be the result of calling the [[Get]] internal method of object *O* with argument "`valueOf`".
4. If IsCallable(*valueOf*) is **true** then,
   a. Let *val* be the result of calling the [[Call]] internal method of *valueOf*, with *O* as the this value and an empty argument list.
   b. If *val* is a primitive value, return *val*.
5. Throw a **TypeError** exception.

When the [[DefaultValue]] internal method of *O* is called with hint Number, the following steps are taken:

1. Let *valueOf* be the result of calling the [[Get]] internal method of object *O* with argument "`valueOf`".
2. If IsCallable(*valueOf*) is **true** then,
   a. Let *val* be the result of calling the [[Call]] internal method of *valueOf*, with *O* as the **this** value and an empty argument list.
   b. If *val* is a primitive value, return *val*.
3. Let *toString* be the result of calling the [[Get]] internal method of object *O* with argument "`toString`".
4. If IsCallable(*toString*) is **true** then,
   a. Let *str* be the result of calling the [[Call]] internal method of *toString*, with *O* as the this value and an empty argument list.
   b. If *str* is a primitive value, return *str*.
5. Throw a **TypeError** exception.

When the [[DefaultValue]] internal method of *O* is called with no hint, then it behaves as if the hint were Number, unless *O* is a Date object (see 15.9.6), in which case it behaves as if the hint were String.

The above specification of [[DefaultValue]] for native objects can return only primitive values. If a host object implements its own [[DefaultValue]] internal method, it must ensure that its [[DefaultValue]] internal method can return only primitive values.

- [1, 2, 3, 4] + [5, 6]
- “1,2,3,4” + “5,6”
- “1,2,3,45,6”

- [] + {}
- “” + “[object Object]”
- “[object Object]”

WHAT IF I WANT TO ADD OTHER THINGS?

- ✅ [1, 2, 3, 4] + [5, 6] → “1,2,3,45,6"
- ✅ true + true → 2
- ✅ undefined + null → NaN
- ✅ null + null → 0
- ✅ [] + {} → "[object Object]”
- {} + [] → 0

WHAT IF I WANT TO ADD OTHER THINGS?

✅ [1, 2, 3, 4] + [5, 6]	▸ “1,2,3,45,6"
✅ true + true	▸ 2
✅ undefined + null	▸ NaN
✅ null + null	▸ 0
✅ [] + {}	▸ "[object Object]”
⁉️ {} + []	▸ 0

BUT WAIT!
THERE'S MORE!

“+” CASTS ANYTHING TO A NUMBER

WHAT IF I WANT TO ADD OTHER THINGS?

- ✓ [1, 2, 3, 4] + [5, 6] → “1,2,3,45,6"
- ✓ true + true → 2
- ✓ undefined + null → NaN
- ✓ null + null → 0
- ✓ [] + {} → "[object Object]”
- → {} + [] → 0

{}

WHAT IF I WANT TO ADD OTHER THINGS?

✓ [1, 2, 3, 4] + [5, 6]	“1,2,3,45,6"
✓ true + true	2
✓ undefined + null	NaN
✓ null + null	0
✓ [] + {}	"[object Object]”
✓ {} + []	0

11.5 Multiplicative Operators

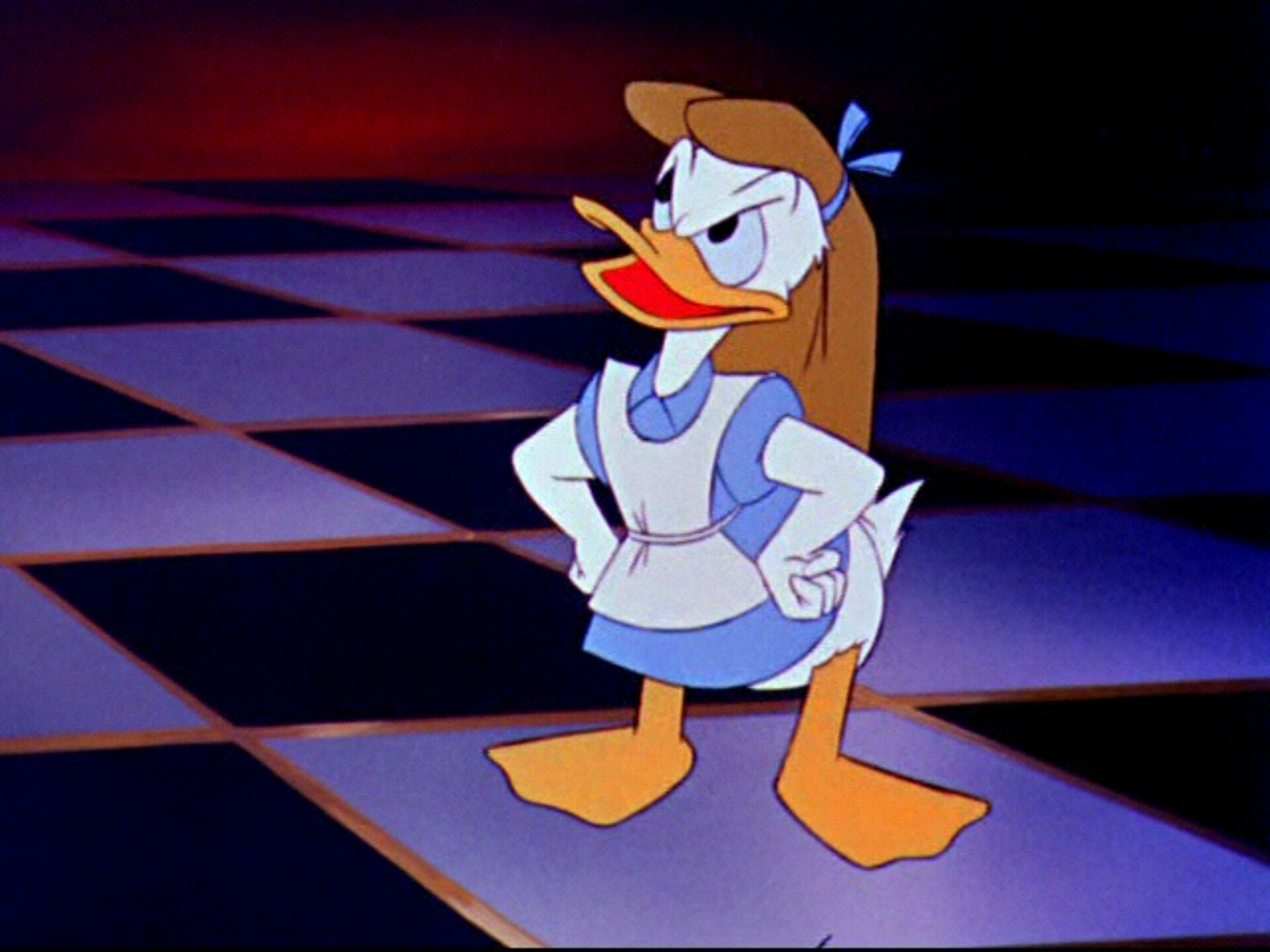
Syntax

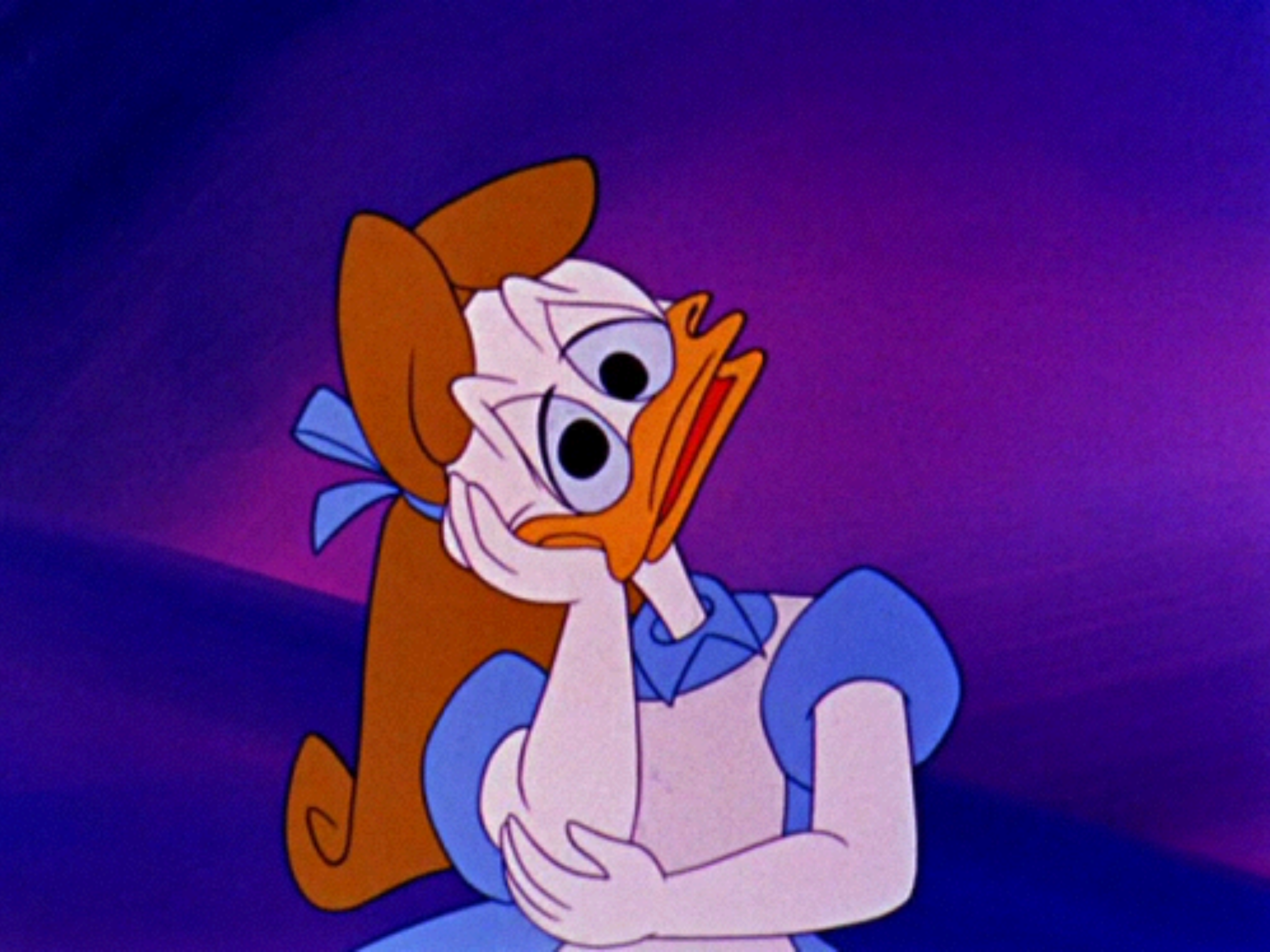
MultiplicativeExpression :
UnaryExpression
MultiplicativeExpression * *UnaryExpression*
MultiplicativeExpression / *UnaryExpression*
MultiplicativeExpression % *UnaryExpression*

Semantics

The production *MultiplicativeExpression* : *MultiplicativeExpression* @ *UnaryExpression* , where @ stands for one of the operators in the above definitions, is evaluated as follows:

1. Let *left* be the result of evaluating MultiplicativeExpression.
2. Let *leftValue* be GetValue(*left*).
3. Let *right* be the result of evaluating UnaryExpression.
4. Let *rightValue* be GetValue(*right*).
5. Let *leftNum* be ToNumber(*leftValue*).
6. Let *rightNum* be ToNumber(*rightValue*).
7. Return the result of applying the specified operation (*, /, or %) to *leftNum* and *rightNum*. See the Notes below 11.5.1, 11.5.2, 11.5.3.





STEP 1:
EXPERIMENT!

STEP 2: GUESS

STEP 3: ITERATE AND RESEARCH

STEP 4:
UNDERSTAND

ACKNOWLEDGEMENTS

tablexi

REFERENCES

- “Wat” by Gary Bernhardt, https://www.destroyallsoftware.com/talks/wat
- You Don’t Know JS by Kyle Simpson, http://shop.oreilly.com/category/get/kyle-simpson-kit.do
- ES5 Specification, http://ecma-international.org/ecma-262/5.1/
- Disney’s *Mathmagic Land*, https://www.youtube.com/watch?v=U_ZHsk0-eF0

Thanks!

@m34ra

@m34ra

AN ADVENTURE IN JAVASCRIPT
ADDITION AND TYPE COERCION

HOW DOES JAVASCRIPT MATH?