James Luterek

/jamesluterek
/jamesluterek

elasticpath

Do you know these 10 JavaScript features?

elasticpath

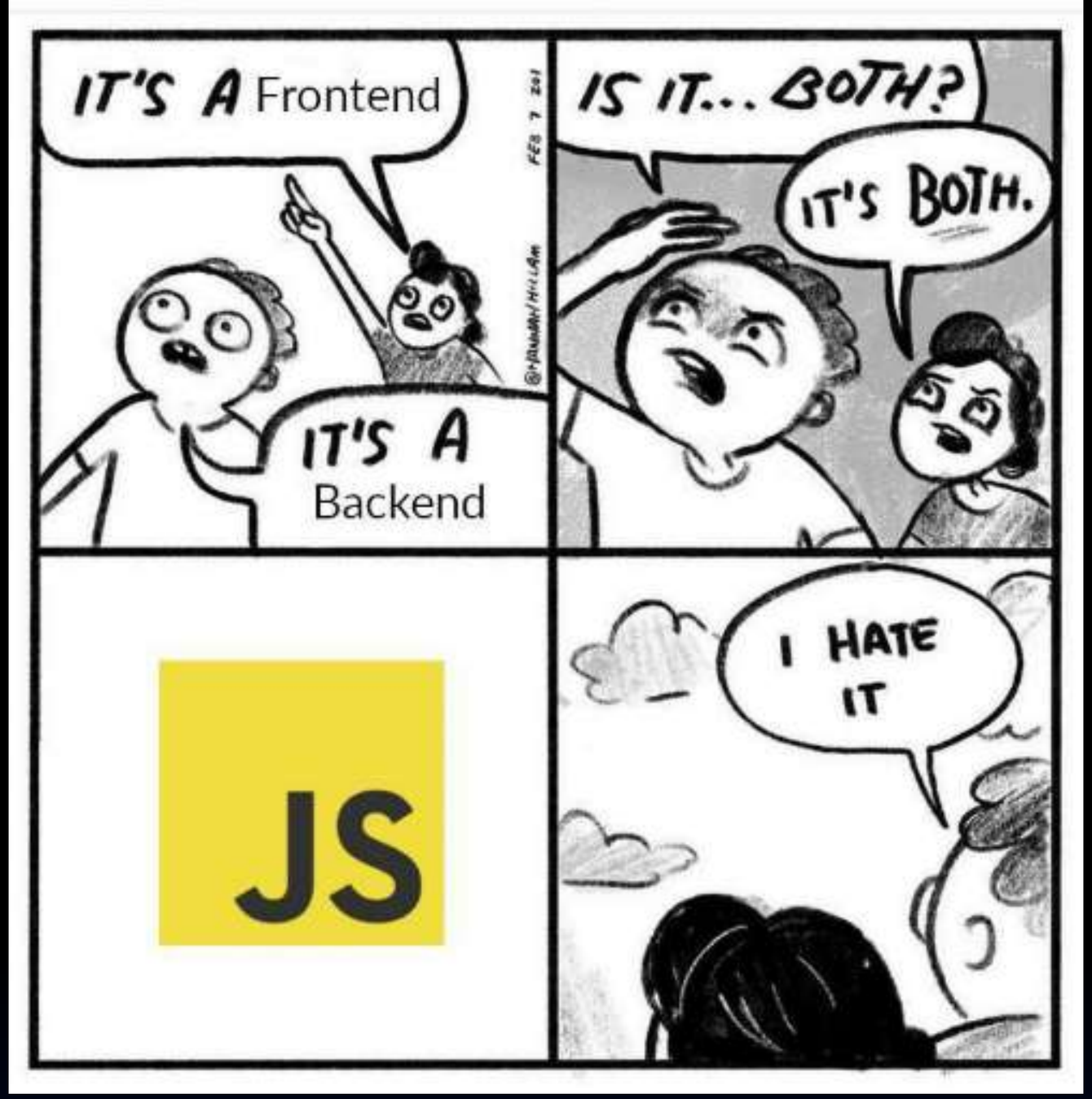
Do you know these 10 JavaScript features?

elasticpath

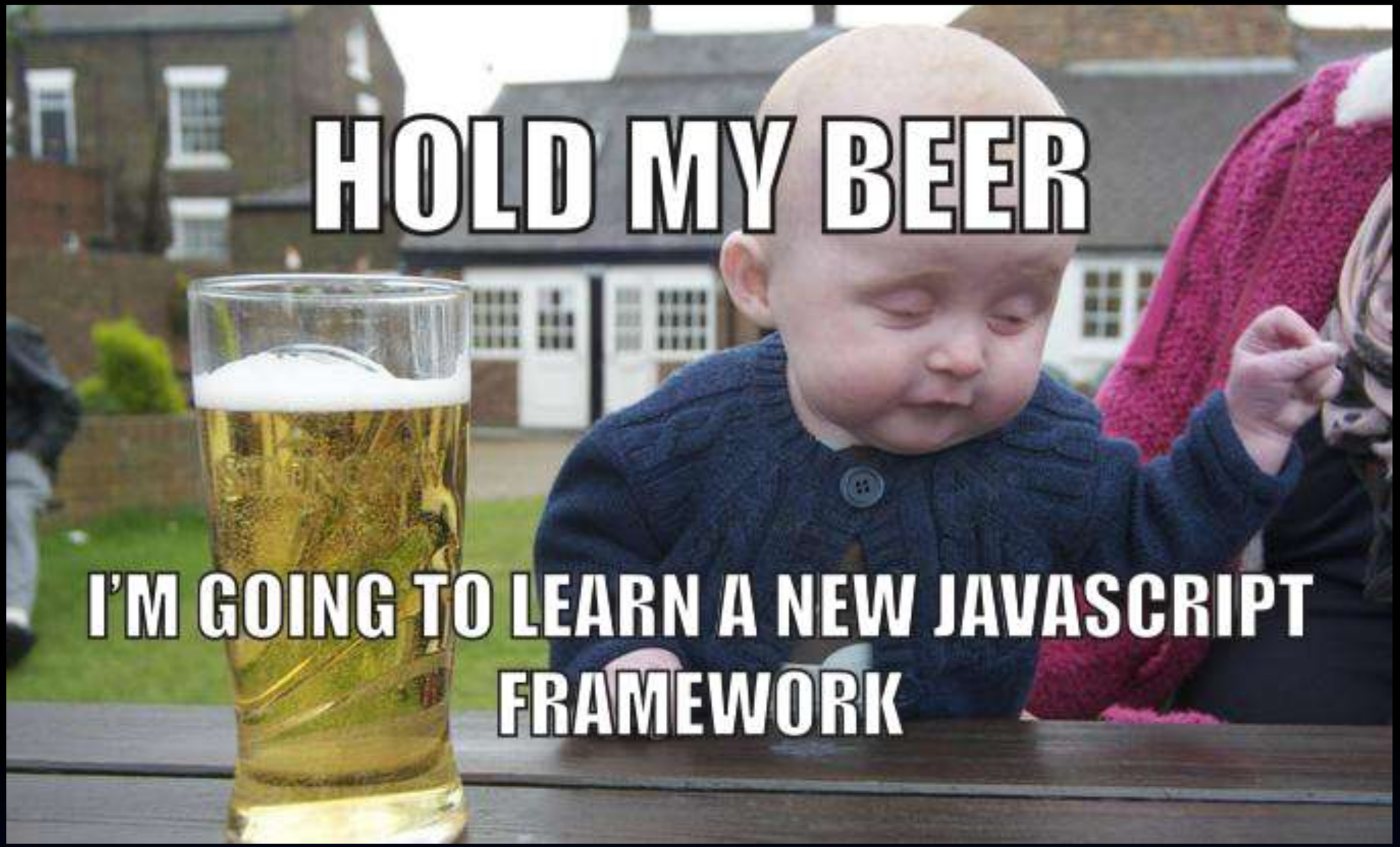
RANT


JAVA
JAVASCRIPT
imgflip.com


0 == "0"
true
JS


IT'S A Frontend
IT'S A Backend
IS IT... BOTH?
IT'S BOTH.
JS
I HATE IT


NOTICE
EMPLOYEES MUST WASH HANDS AFTER USING JAVASCRIPT


HOLD MY BEER
I'M GOING TO LEARN A NEW JAVASCRIPT FRAMEWORK


What is the world's most popular language?
A: French
B: English
C: Spanish
D: JavaScript

JavaScript is a BEAST!


JAVASCRIPT
EVERYWHERE
memegenerator.net

Brendan Eich on Creating JavaScript in 10 Days, and What He'd Do Differently Today

elasticpath

Class Features

Public / Private Fields

elasticpath

```
class Counter {
    constructor() {
      this.name = 'Counter';
      this.count = 0;
    }

    inc() {
      this.count++;
    }
  }
```

```
class Counter {
    name = 'Counter';
    #count = 0;  // private field!
    inc() {
        this.#count++;
    }
}
```

Private Methods

elasticpath

```
class Counter {
    name = 'Counter';
    #count = 0;  // private field!
    inc() {
        this.#setCount(this.#count++)
    }
    #setCount(cnt) {
        this.#count = cnt;
    }
}
```

Static

elasticpath

```
class Counter {
    static staticProperty = 'StaticValue';
    static staticMethod() {
        console.log('Class static block called');
    }
}

console.log(Counter.staticProperty);
// output: "StaticValue“

console.log(Counter.staticMethod());
// output: "Class static block called"
```

elasticpath

Array Functions

Array.prototype.at()

elasticpath

```
const array1 = [a, b, c, d, e, z];

console.log(array1[2]);
// output: c


console.log(array1[array1.length - 1]);
// output: z
```

```
const array1 = [a, b, c, d, e, z];

console.log(array1.at(2));
// output: c


console.log(array1.at(-1));
// output: z
```

Array.prototype.group()

elasticpath

```
const myData = [
    { type: 'food', name: 'Pizza' },
    { type: 'drink', name: 'Coffee' },
    { type: 'food', name: 'Hot Dog' }
];
```

```
const result = {
    food: [
        { type: 'food', name: 'Pizza' },
        { type: 'food', name: 'Hot Dog' }
    ],
    drink: [
        { type: 'drink', name: 'Coffee' }
    ]
}
```

```
const result = myData.group((item) => item.type);
```

elasticpath

Optional Chaining

Optional Chaining (?.)

elasticpath

```
const speaker = {
    name: 'James',
    dog: {
        name: 'Delta'
    }
};
const catName = speaker.cat.name;
```

Error: speaker.cat is undefined

```
if (speaker && speaker.cat &&
speaker.cat.name) {
    catName = speaker.cat.name;
}
else {
    catName = null;
}

catName = (speaker.cat) ? speaker.cat.name : null;
```

```
const catName = speaker?.cat?.name;
// catName set to undefined
```

Nullish coalescing operator (??)

elasticpath

```
function setDefault(val) {
    // Set a default value if null or undefined
    return val || 'default';
}

setDefault(null);  //default
setDefault(undefined);  //default
setDefault('String Value');  //String Value
setDefault(true);  //true
setDefault(false);  //default
setDefault('');  //default
setDefault(0);  //default
```

Nullish coalescing operator (??)

elasticpath

```
function setDefault(val) {
    // Set a default value if null or undefined
    return val ?? 'default';
}

setDefault(null);  //default
setDefault(undefined);  //default
setDefault('String Value');  //String Value
setDefault(true);  //true
setDefault(false);  //false
setDefault('');  //
setDefault(0);  //0
```

elasticpath

Checking for Properties

hasOwnProperty, hasOwn, in

elasticpath

```
const test = {
    name: '1',
    age: 2
};

//General Properties
console.log('name' in test); //true
console.log(Object.hasOwn(test, 'name')); //true
console.log(test.hasOwnProperty('name')); //true

//Inherited Properties
console.log('constructor' in test); //true
console.log(Object.hasOwn(test, 'constructor')); //false
console.log(test.hasOwnProperty('constructor')); //false
```

hasOwnProperty, hasOwn, in

elasticpath

```
// Object not inheriting Object.prototype
const newObject = Object.create(null);
newObject.name = 'James';


console.log('name' in newObject); //true
console.log(Object.hasOwn(newObject, 'name')); //true
console.log(newObject.hasOwnProperty('name'));
```

TypeError: newObject.hasOwnProperty is not a function

elasticpath

Error Cause

Error.prototype.cause

elasticpath

```
// Rethrowing error with new message
try {
    connectToDatabase();
} catch (err) {
    throw new Error('Connecting to database failed.', { cause: err });
}

// Adding structured data to cause
throw new Error('Requires integer inputs.', {
    cause: { code: 'NonInteger', values: [p, q] },
});
```

elasticpath

JSON.stringify

JSON.stringify

elasticpath

```
const speaker = {
    firstName: 'James',
    lastName: 'Luterek',
    company: 'Elastic Path',
    winning: undefined
};
console.log(JSON.stringify(speaker));
```

```
{"firstName":"James","lastName":"Luterek","company":"Elastic Path"}
```

JSON.stringify

elasticpath

```
const speaker = {
    firstName: 'James',
    lastName: 'Luterek',
    company: 'Elastic Path',
    winning: undefined
};
console.log(JSON.stringify(speaker, ['firstName']));
```

```
{"firstName":"James"}
```

JSON.stringify

elasticpath

```
const speaker = {
    firstName: 'James',
    lastName: 'Luterek',
    company: 'Elastic Path',
    winning: undefined
};
console.log(JSON.stringify(speaker, (key, value) => {
    if (value === undefined) {
        return false;
    }
    return value;
}));
```

```
{"firstName":"James","lastName":"Luterek","company":"Elastic Path","winning":false}
```

JSON.stringify

elasticpath

```
const speaker = {
    firstName: 'James',
    lastName: 'Luterek',
    company: 'Elastic Path',
    winning: undefined
};
console.log(JSON.stringify(speaker, null, 2));
```

```
{
  "firstName": "James",
  "lastName": "Luterek",
  "company": "Elastic Path"
}
```

JSON.stringify

elasticpath

```
const speaker = {
    firstName: 'James',
    lastName: 'Luterek',
    company: 'Elastic Path',
    winning: undefined
};
console.log(JSON.stringify(speaker, null, `  ◉_◉  `));
```

```
{
  ◉_◉  "firstName": "James",
  ◉_◉  "lastName": "Luterek",
  ◉_◉  "company": "Elastic Path"
}
```

elasticpath

Underscore Numeric Seperator

Underscore Separator

elasticpath

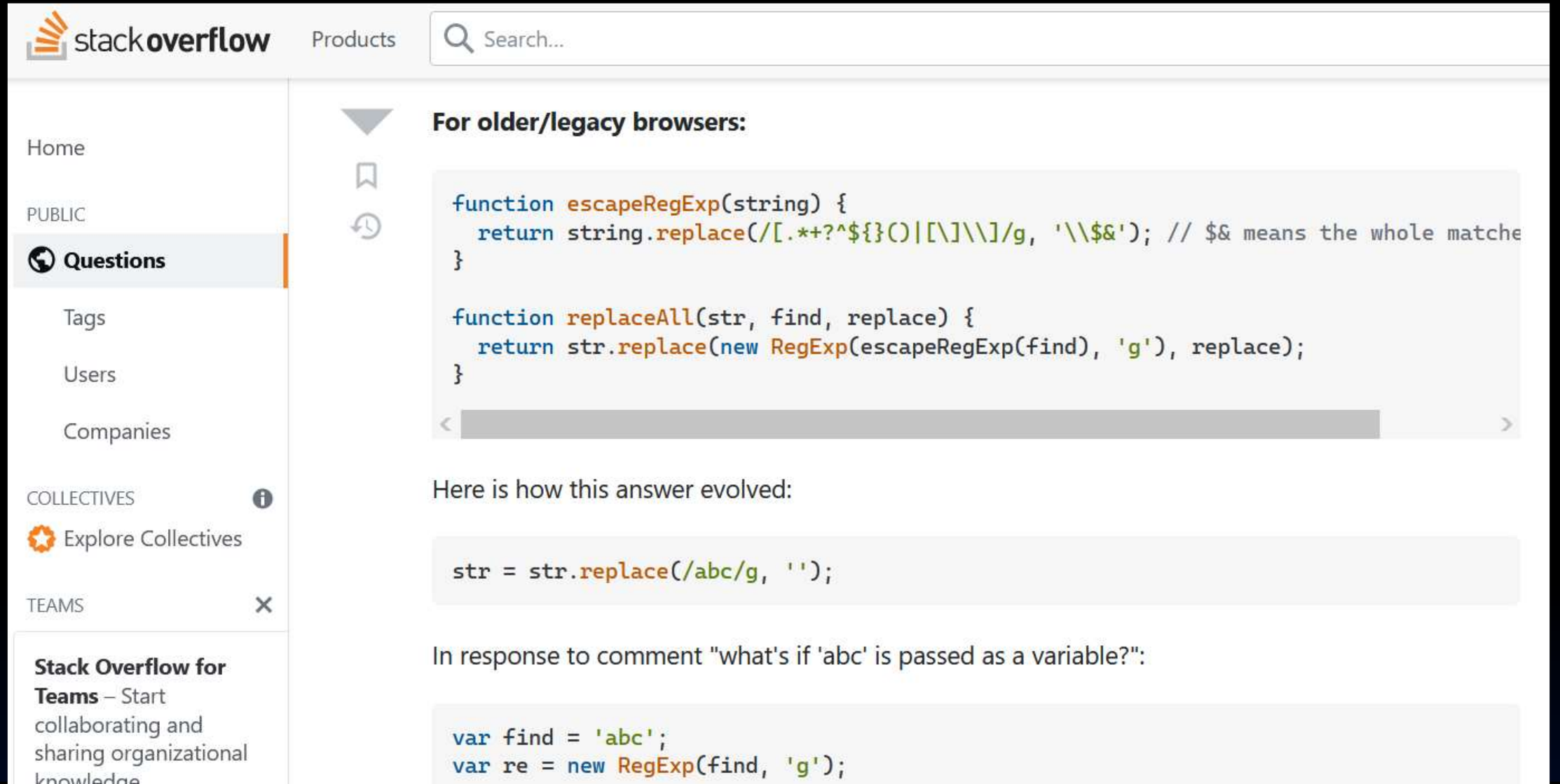
```
const million = 1000000;
const readableMiliion = 1_000_000;
const decimal = 0.220720;
const readableDecimal = 0.220_720;
```

elasticpath

replaceAll

“How do I replace all occurrences of a string in JavaScript?”

elasticpath



String.prototype.replaceAll()

elasticpath

```
const p = 'The quick brown fox jumps over the lazy dog.';

console.log(p.replaceAll(' ', '_'));
//The_quick_brown_fox_jumps_over_the_lazy_dog.
```

elasticpath

Logical Assignment Operators

Logical OR assignment (||=)

elasticpath

```
// Better than
x = x || y

// Equivalent to
x || (x = y)

// Can now be written as
x ||= y
```

Logical AND assignement (&&=)

elasticpath

```
// Better than
x = x && y

// Equivalent to
x && (x = y)

// Can now be written as
x &&= y
```

Nullish coalescing assignment (??=)

elasticpath

```
// Better than
x = x ?? y

// Equivalent to
x ?? (x = y)

// Can now be written as
x ??= y
```

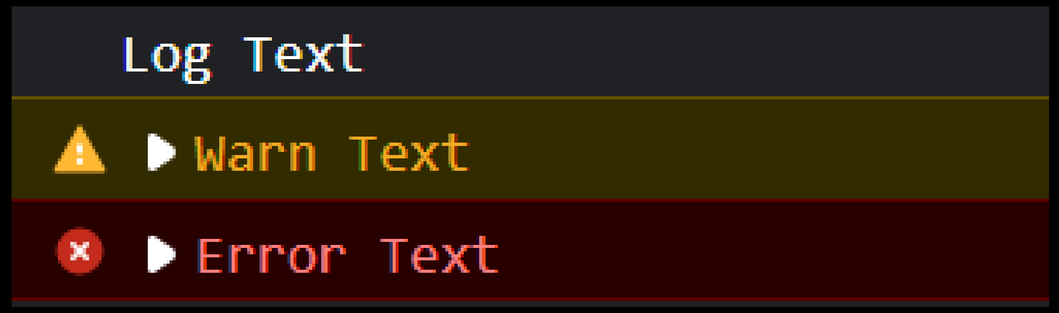
elasticpath

Console.*

Log, Warn, Error

elasticpath

```
console.log('Log Text');
console.warn('Warn Text');
console.error('Error Text');
```



Table

elasticpath

```
const transactions = [{
    id: "7cb1-e041b126-f3b8",
    seller: "WAL0412",
    buyer: "WAL3023",
    price: 203450,
    time: 1539688433
},
{
    id: "1d4c-31f8f14b-1571",
    seller: "WAL0452",
    buyer: "WAL3023",
    price: 348299,
    time: 1539688433
},
{
    id: "b12c-b3adf58f-809f",
    seller: "WAL0012",
    buyer: "WAL2025",
    price: 59240,
    time: 1539688433
}];
```

```
console.log(transactions);
console.table(transactions);
```

▶ (3) [{…}, {…}, {…}] console.html:31

console.html:32

(index)	id	seller	buyer	price	time
0	'7cb1-e041b12…	'WAL0412'	'WAL3023'	203450	1539688433
1	'1d4c-31f8f14…	'WAL0452'	'WAL3023'	348299	1539688433
2	'b12c-b3adf58…	'WAL0012'	'WAL2025'	59240	1539688433

▶ Array(3)

Time

elasticpath

```
const slowFunction = number => {
    console.time('slowFunction');
    // something slow or complex
    console.timeEnd('slowFunction');
}
console.time();

for (i = 0; i < 10; ++i) {
    slowFunction(i);
} console.timeEnd();
```

```
slowFunction: 0.002685546875 ms
slowFunction: 0.002197265625 ms
slowFunction: 0.001708984375 ms
slowFunction: 0.001953125 ms
slowFunction: 0.001220703125 ms
slowFunction: 0.000732421875 ms
slowFunction: 0.0009765625 ms
slowFunction: 0.001220703125 ms
slowFunction: 0.0009765625 ms
slowFunction: 0.001953125 ms
default: 0.329833984375 ms
```

Group

elasticpath

```
console.group('OutsideLoop');
console.log('Hi There!');
console.group('Inside Loop');
console.log('1');
console.log('2');
console.log('3');
console.groupEnd();
console.log('Bye!');
console.groupEnd();
```