



CityJS™
BERLIN
2023



Christopher Ehrlich
Frontend Developer @ Axiom.co

The new toys JavaScript MIGHT get



Christopher Ehrlich  

@ccccjjjjeeee



how is your time split between js/ts and other languages?

(for the sake of this poll, "other languages" means java/golang/etc, but not html/jsx/sql/etc)

- ☐ **100% js/ts**
- ☐ mixed, more js/ts
- ☐ mixed, more other
- ☐ 100% other



Christopher Ehrlich



@ccccjjjjeeee

how is your time split between js/ts and other languages?

(for the sake of this poll, "other languages" means java/golang/etc, but not html/jsx/sql/etc)

100% js/ts

69%

mixed, more js/ts

21.1%

mixed, more other

5.6%

100% other

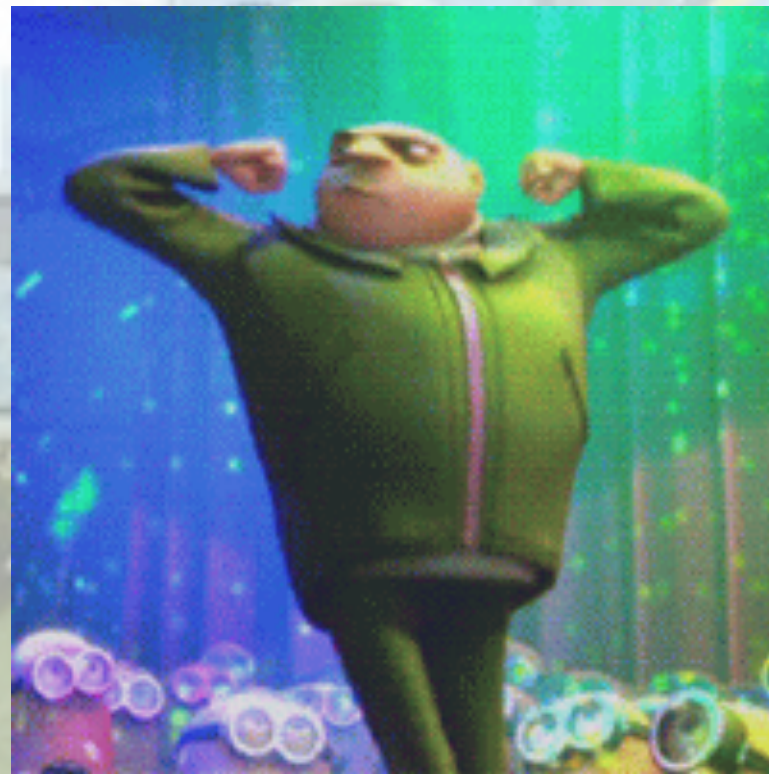
4.2%

WE'RE STUCK WITH JAVASCRIPT

`</> htmx`



OCaml



GruLang







HI, I'M CHRIS
@ccccjjjeeeee



CREATE THE APP



- ◆ Will you be using TypeScript or JavaScript?
TypeScript
- ◆ Will you be using Tailwind CSS for styling?
Yes
- ◆ Would you like to use tRPC?
Yes
- ◆ What authentication provider would you like to use?
NextAuth.js
- ◆ What database ORM would you like to use?
Drizzle
- ◆ **EXPERIMENTAL** Would you like to use Next.js App Router?
● Yes / ○ No

1.

WE'VE ALL WRITTEN THIS BEFORE


```
const feedbackCategory =  
    analyzeCustomerFeedback(feedback);
```

```
let action;
```

```
if (feedbackCategory === 'positive') {  
    action = askForReview(feedback);  
} else if (feedbackCategory === 'negative') {  
    action = offerCustomerSupport(feedback);  
} else {  
    action = handleGeneralFeedback(feedback);  
}
```



```
const feedbackCategory =  
    analyzeCustomerFeedback(feedback);  
  
let action;  
  
switch (feedbackCategory) {  
    case 'positive':  
        action = askForReview(feedback);  
        break;  
    case 'negative':  
        action = offerCustomerSupport(feedback);  
        break;  
    default:  
        action = handleGeneralFeedback(feedback);  
}
```



```
const feedbackCategory =  
  analyzeCustomerFeedback(feedback);
```

```
const actions = {  
  'positive': askForReview,  
  'negative': offerCustomerSupport,  
  'default': handleGeneralFeedback,  
};
```

```
const action = (actions[feedbackCategory]  
  ?? actions['default'])(feedback);
```



```
function determineAction(feedbackCategory, feedback) {  
  if (feedbackCategory === 'positive') {  
    return askForReview(feedback);  
  } else if (feedbackCategory === 'negative') {  
    return offerCustomerSupport(feedback);  
  } else {  
    return handleGeneralFeedback(feedback);  
  }  
}
```

```
const feedbackCategory =  
  analyzeCustomerFeedback(feedback);
```

```
const action =  
  determineAction(feedbackCategory, feedback);
```


THEY'RE ALL TERRIBLE




```
$feedbackCategory = analyzeCustomerFeedback($feedback);
```

```
$action = match ($feedbackCategory) {  
    'positive' => askForReview($feedback),  
    'negative' => offerCustomerSupport($feedback),  
    default => handleGeneralFeedback($feedback),  
};
```


A vibrant, futuristic cityscape under a bright blue sky. In the foreground, a man in a black t-shirt and shorts walks a small dog on a leash along a curved white path. To his right, a sleek, silver, aerodynamic car is partially submerged in a pool of water. The background features a variety of futuristic buildings, including a tall, curved skyscraper on the left and a central tower with multiple circular observation decks. Numerous flying cars of various shapes are seen in the sky. The overall scene is a blend of nature (green grass and trees) and advanced technology.

THE WORLD IF JAVASCRIPT HAD THIS

ES

TC39

PROPOSAL STAGES

0: Looking for a champion

1: Idea

2: Draft

3: Details are worked out

4: Success

WH: Part of the spec says that's allowed; part of the spec says that's not allowed. What is the intended behavior?

JSC: I'll deal with issues like that after the meeting, whenever you want: happy to hash out after the meeting. The *intent* is to allow them in functions. Although I wouldn't say that might be good *style*.

TAB: The percent is just a variable binding that applies just within the context of the things. So you can lexically bind over and should be able to lexically bind over that in any way that you could a normal variable.

WH: Okay, so it's not like `this` where you can't use it inside nested functions? Thank you. All of these syntax problems are solvable and I see no real showstoppers here. `%` would work as far as the syntax is concerned.

TAB: Yeah, and I think that also then would address your second topic (`x = 3%4; a |> %== y; b |> x+%== y;`). Both of those your other one also appears to be about the `%==` operator attempt at parsing.

RW: So value piped into a—just to be clear, this is not a modulo operator. It's the remainder operator; let's use the right words—value piped into a remainder operation that ballad. I'm going over there because that's where it is.

Source: ECMA, TC39 Meeting Notes - <https://github.com/robpalme/notes/>


```
const feedbackCategory =  
    analyzeCustomerFeedback(feedback);
```

```
const action = match (feedbackCategory) {  
    when "positive": askForReview(feedback);  
    when "negative": offerCustomerSupport(feedback);  
    default: handleGeneralFeedback(feedback);  
}
```

```
match (res) {  
  when ({ status: 200, body, ...rest }):  
    handleData(body, rest);  
  
  default: throwSomething();  
}
```



```
match (res) {  
  when ({ status, url })  
    if (300 <= status && status < 400):  
      handleRedirect(url)  
  
  default: throwSomething();  
}
```

```
match (res) {  
  when ({ status: 500 }): do {  
    retry(req);  
    this.hasRetried = true;  
  }  
  
  default: throwSomething();  
}
```


2.

WHEN ALL YOU HAVE IS A BASH SHELL

```
import isEqual from 'lodash/isEqual';
```



We want to count these


```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
lib/dash/routes/query/QueryPage.tsx:import isEqual from 'lodash/isEqual';  
lib/dash/routes/settings/Settings.tsx:import words from 'lodash/words';  
lib/dash/stores/OrgStore.ts:import cloneDeep from 'lodash/cloneDeep';  
(...)
```



```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.*\/\1/p'
```

```
lib/dash/routes/query/QueryPage.tsx:import isEqual from 'lodash/isEqual';  
lib/dash/routes/settings/Settings.tsx:import words from 'lodash/words';  
lib/dash/stores/OrgStore.ts:import cloneDeep from 'lodash/cloneDeep';  
(...)
```

```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.*/>\1/p'
```

```
isEqual
```

```
words
```

```
cloneDeep
```

```
(...)
```



```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.* /\1/p'
```

```
| sort
```

```
isEqual
```

```
words
```

```
cloneDeep
```

```
(...)
```

```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.* /\1/p'
```

```
| sort
```

```
capitalize  
capitalize  
capitalize  
(...)
```



```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.*/>\1/p'
```

```
| sort
```

```
| uniq -c
```

```
capitalize
```

```
capitalize
```

```
capitalize
```

```
(...)
```

```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.*/>\1/p'
```

```
| sort
```

```
| uniq -c
```

```
4 capitalize
```

```
19 cloneDeep
```

```
10 debounce
```

```
(...)
```



```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.*/>\1/p'
```

```
| sort
```

```
| uniq -c
```

```
| sort -nr
```

```
4 capitalize
```

```
19 cloneDeep
```

```
10 debounce
```

```
(...)
```

```
import isEqual from 'lodash/isEqual';
```

```
rg "from 'lodash"
```

```
| sed -nE 's/.*import ([^ ]+) from.*/>\1/p'
```

```
| sort
```

```
| uniq -c
```

```
| sort -nr
```

```
21 isEqual
```

```
19 cloneDeep
```

```
10 debounce
```

```
(...)
```

```
const products = [  
  { name: "Laptop", category: "electronics", price: 1000, discount: 15 },  
  { name: "Shirt", category: "clothing", price: 50, discount: 10 },  
  { name: "Microphone", category: "electronics", price: 150, discount: 5 },  
];  
  
const discountedElectronics = products  
  .filter(isElectronics)  
  .map(applyDiscount)  
  .sort(compareByPrice);
```



```
function processRequest(request) {  
    const body = extractRequestBody(request);  
    const data = convertJsonToData(body);  
    const itemId = extractItemId(data);  
    const item = retrieveItemById(itemId);  
    const response = prepareResponseWithItem(item);  
    return response;  
}
```

```
function processRequest(request) {  
    const body = extractRequestBody(request);  
    const data = convertJsonToData(body);  
    if (someCondition) data.foo = "bar";  
  
    const itemId = extractItemId(data);  
    const item = retrieveItemById(itemId);  
    const response = prepareResponseWithItem(item);  
    return response;  
}
```

**MUTATING DATA IS THE NUMBER
ONE SOURCE OF BUGS IN
JAVASCRIPT APPLICATIONS***

*** I MADE THAT UP, IT'S PROBABLY USEEFFECT**


```
function processRequest(request) {  
    return prepareResponseWithItem(  
        retrieveItemById(  
            extractItemId(  
                convertJsonToData(  
                    extractRequestBody(  
                        request  
                    )  
                )  
            )  
        )  
    );  
}
```

```
def process_get_item_request(request) do
  request
  |> extract_request_body()
  |> convert_json_to_data()
  |> extract_item_id()
  |> retrieve_item_by_id()
  |> prepare_response_with_item()
end
```

```
fun processGetItemRequest(request: Request): Response {  
    return extractRequestBody(request).let { requestBody ->  
        convertJsonToData(requestBody)  
    }.let { data ->  
        extractItemId(data)  
    }.let { itemId ->  
        retrieveItemById(itemId)  
    }.let { item ->  
        prepareResponseWithItem(item)  
    }  
}
```


return request

```
|> req => extractRequestBody(req)
|> body => convertJsonToData(body)
|> data => extractItemId(data)
|> id => retrieveItemById(id)
|> item => prepareResponseWithItem(item);
```

TS

3.

THERE'S NOT ALWAYS A HAPPY END


```
class ValidationError extends Error {};  
  
function getRequestData(req) {  
  const { data, isValid } = doSomeValidation(req.body);  
  if (!isValid) {  
    throw new ValidationError(`Invalid JSON: ${req.body}`);  
  }  
  return data;  
}  
  
// some route handler  
try {  
  const data = getRequestDate(req);  
  const result = await doSomething(data);  
  return res.status(200).send(result);  
} catch (e) {  
  if (e instanceof ValidationError) {  
    res.status(400).send({ type: "validation", message: e.message });  
  } else {  
    res.status(500).send({ type: "unknown", message: e.message });  
  }  
}
```

```
static void methodThatThrows() throws MyCheckedException {  
    throw new MyCheckedException("Custom Exception!");  
}
```

```
static void catches() {  
    try {  
        methodThatThrowsException();  
    } catch (MyCheckedException e) {  
        System.out.println("Caught custom exception!");  
    }  
}
```

```
static void doesntCatch() throws MyCheckedException {  
    methodThatThrowsException();  
}
```

```
static void callBuggyFunction() {  
    try {  
        someBuggyFunction();  
    } catch (MyCheckedException e) {  
        System.out.println("Caught MyCheckedException");  
    } catch (SomeOtherException e) {  
        System.out.println("Caught SomeOtherException");  
    }  
}
```


JSON.PARSE CAN THROW

```
class MyError extends Error { ... }
```

```
function fn(...) throws MyError { ... }
```

```
// From a library that's annotated
function doSomething1(): void throws DocumentedException;

// From a library that's not annotated.
function doSomething2(): void;

function fn() {
    try {
        doSomething1();
        doSomething2();
    } catch (e) {
        // if we assume doSomething2 doesn't throw,
        // e is DocumentedException

        // if we assume doSomething2 throws, e is unknown
    }
}
```

```
function callSomeCallbacks(  
    f: () => void,  
    g: () => void,  
): void {  
    // ...  
}
```



```
function callSomeCallbacks(  
  f: () => void,  
  g: () => void,  
): void  
  does not rethrow from f but does rethrow from g  
  except if it's a RangeError  
{  
  // ...  
}
```



RyanCavanaugh closed this as not planned on Apr 19

Temporal API

Array.findLast (ES2023)

Better Timezones

Records and Tuples

Set.isSubsetOf, intersection, union, etc

ShadowRealm 🦇🎃💀 (seriously!)

A bunch of stuff for library and runtime authors

THANK YOU FOR LISTENING



@ccccjjjeeee