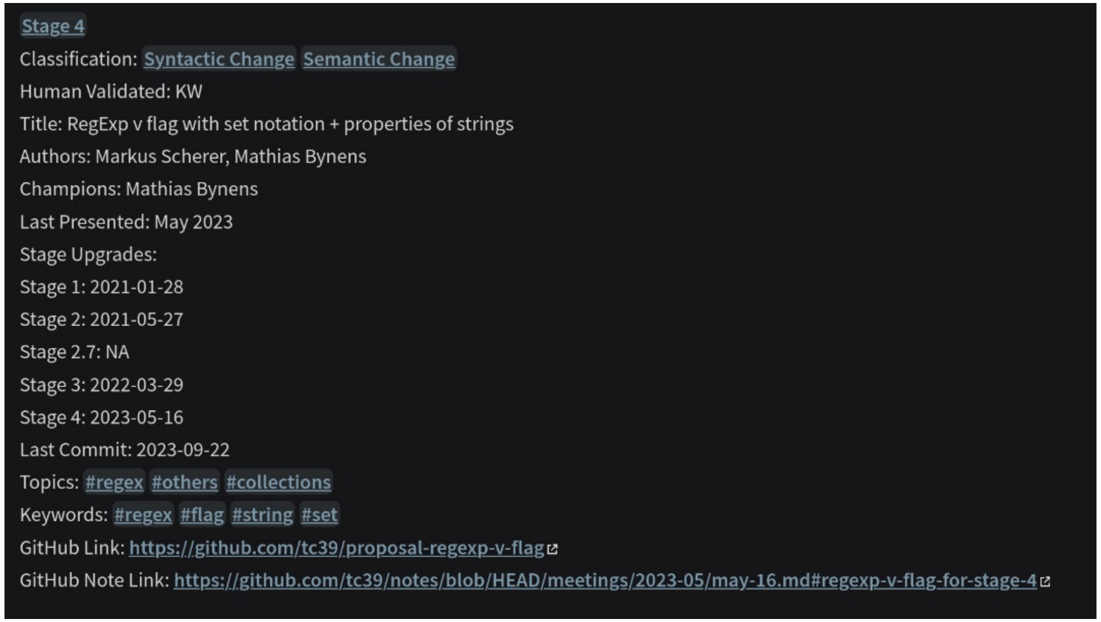
ECMAScript Observations

About this project

- Collect data on and classify TC39 proposals
  - Snapshot from Feb 2025 on data from TC39/proposals repositories
  - Stages, classification, topics, keywords
  - Make a graph linking related proposals



Stage 4

Classification: Syntactic Change Semantic Change

Human Validated: KW

Title: RegExp v flag with set notation + properties of strings

Authors: Markus Scherer, Mathias Bynens

Champions: Mathias Bynens

Last Presented: May 2023

Stage Upgrades:

Stage 1: 2021-01-28

Stage 2: 2021-05-27

Stage 2.7: NA

Stage 3: 2022-03-29

Stage 4: 2023-05-16

Last Commit: 2023-09-22

Topics: #regex #others #collections

Keywords: #regex #flag #string #set

GitHub Link: https://github.com/tc39/proposal-regexp-v-flag

GitHub Note Link: https://github.com/tc39/notes/blob/HEAD/meetings/2023-05/may-16.md#regexp-v-flag-for-stage-4

Classifications

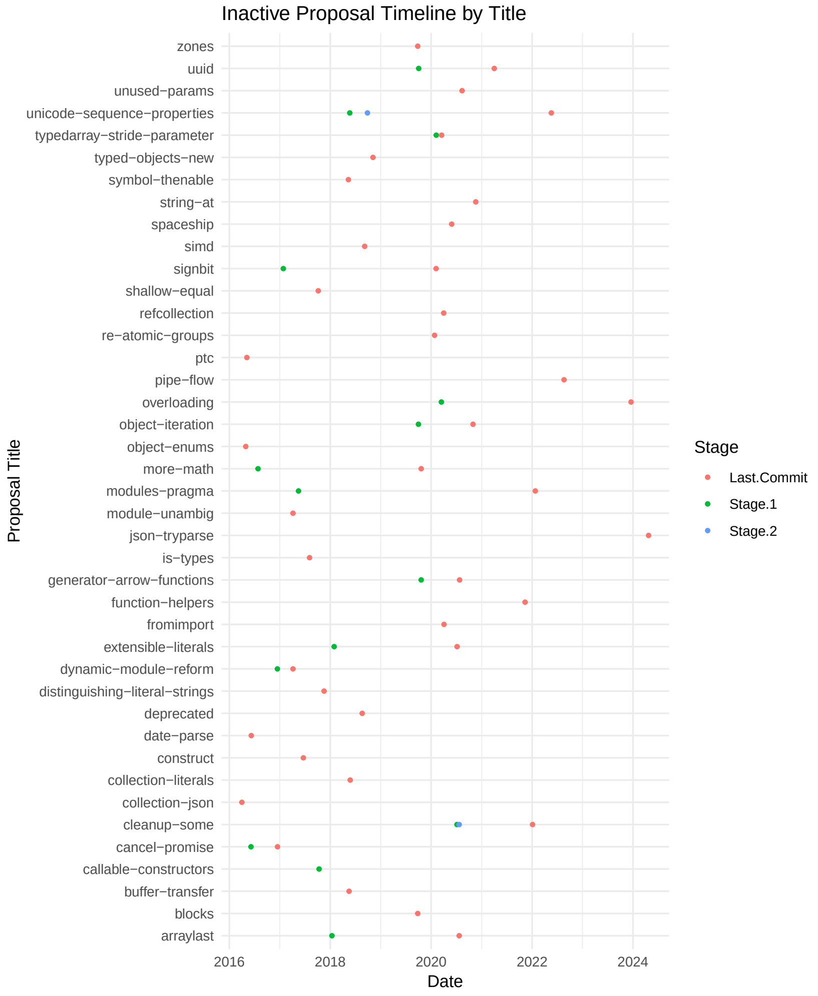
- API Change
  - Modifies or introduces new functions, objects, or methods in the standard library. These changes do not affect the syntax of the language but add new functionality to existing features.
- Semantic Change
  - Changes the meaning of the JavaScript code even if the syntax remains the same. These changes can alter the behavior of existing JavaScript programs in subtle or breaking ways. Usually involves modifying execution rules rather than introducing new syntax.
- Syntactic Change:
  - Introduces new syntax or modifies existing syntax rules. Usually involves new keywords, operators, or expressions. These changes often require updates to parsers and affect how JavaScript code is written.

How?

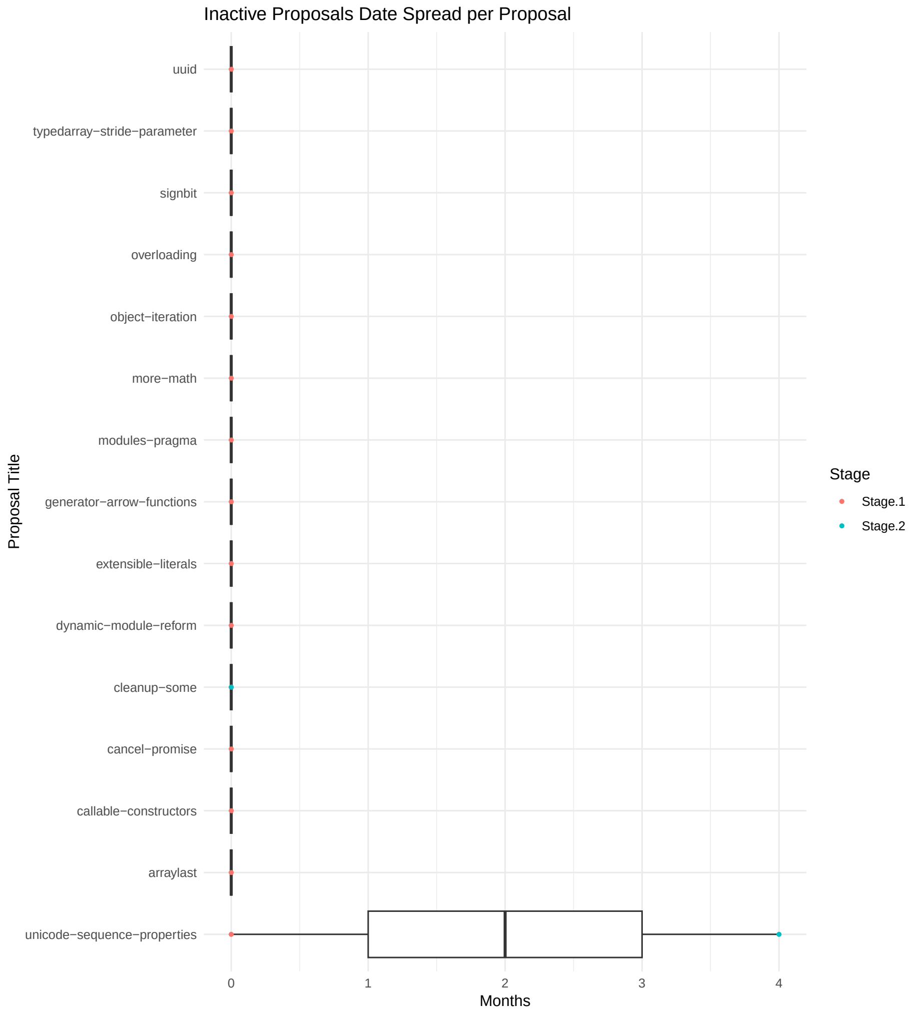
- Data: Obsidian, Python, GPT
  - Retrieve data via Github API – TC39 Datasets?
  - Parse the data, create .md files and save in obsidian
  - GPT assistance – Classifications, Stage bumps from commit messages, keywords
  - Manually verified and curated
  - Data analysis done in R and Rstudio
- Website: Quartz, hosting on Vercel
  - Quartz: Open source static page generator with Obsidian compatibilty
  - Demonstration

Observations

Inactive

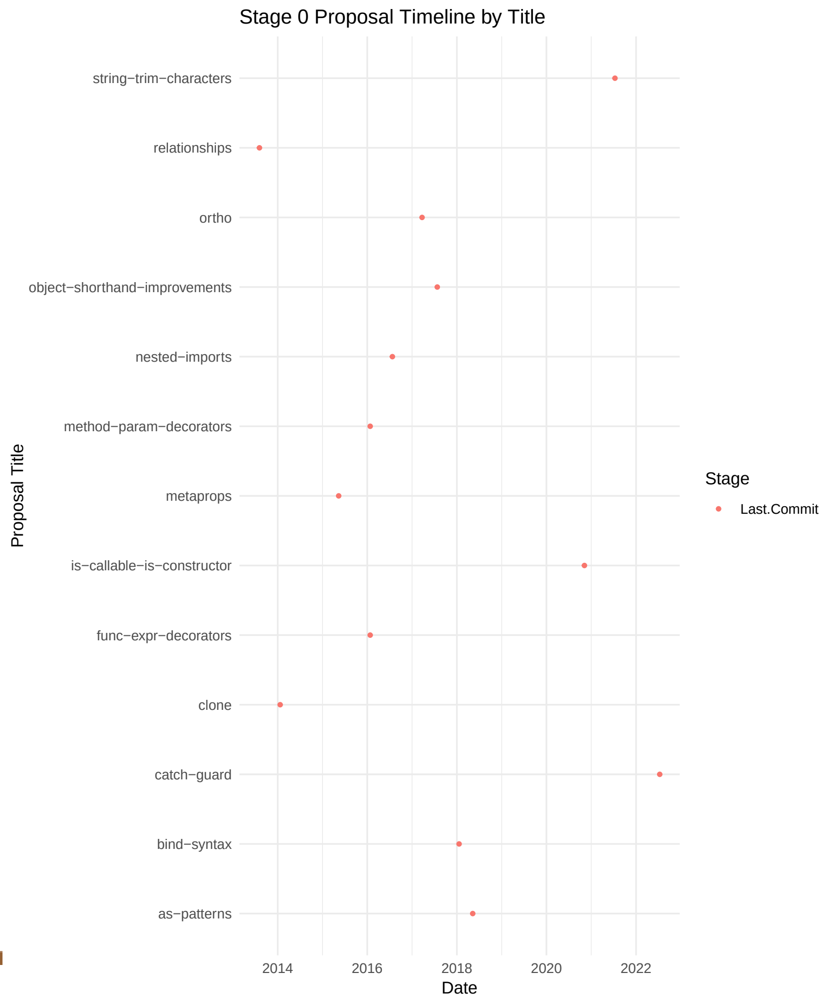


Inactive Proposal Timeline by Title
Proposal Title
Date
2016
2018
2020
2022
2024
Stage
Last.Commit
Stage.1
Stage.2
zones
uuid
unused−params
unicode−sequence−properties
typedarray−stride−parameter
typed−objects−new
symbol−thenable
string−at
spaceship
simd
signbit
shallow−equal
refcollection
re−atomic−groups
ptc
pipe−flow
overloading
object−iteration
object−enums
more−math
modules−pragma
module−unambig
json−tryparse
is−types
generator−arrow−functions
function−helpers
fromimport
extensible−literals
dynamic−module−reform
distinguishing−literal−strings
deprecated
date−parse
construct
collection−literals
collection−json
cleanup−some
cancel−promise
callable−constructors
buffer−transfer
blocks
arraylast

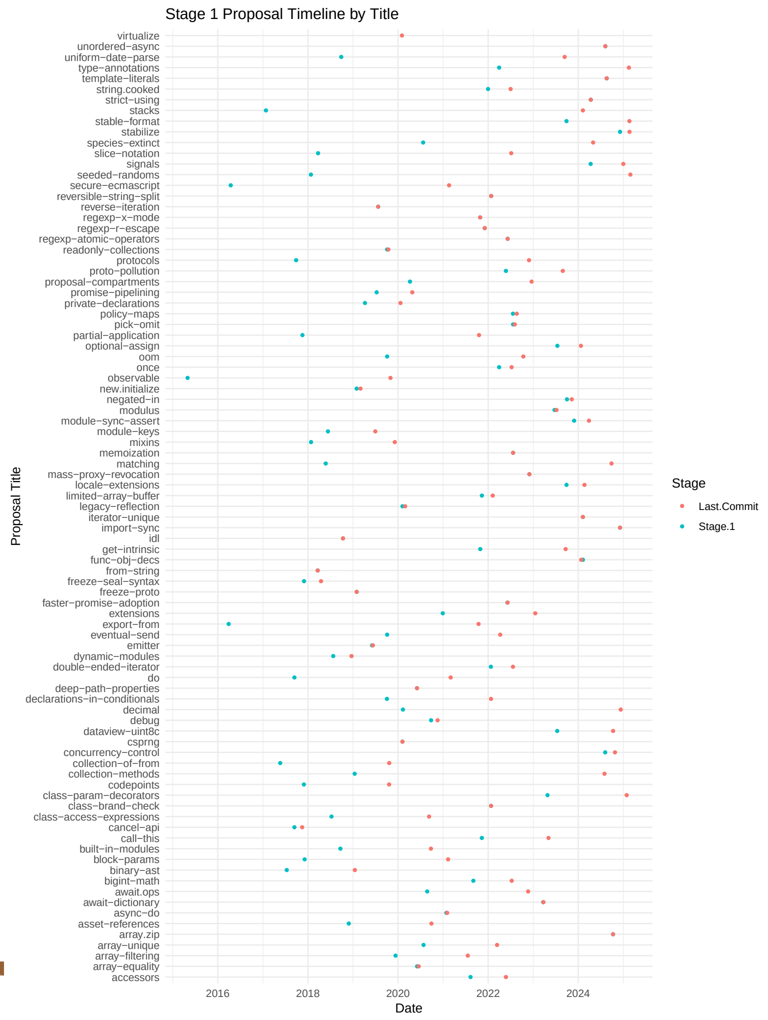


Inactive Proposals Date Spread per Proposal
Proposal Title
Months
0
1
2
3
4
Stage
Stage.1
Stage.2
uuid
typedarray−stride−parameter
signbit
overloading
object−iteration
more−math
modules−pragma
generator−arrow−functions
extensible−literals
dynamic−module−reform
cleanup−some
cancel−promise
callable−constructors
arraylast
unicode−sequence−properties

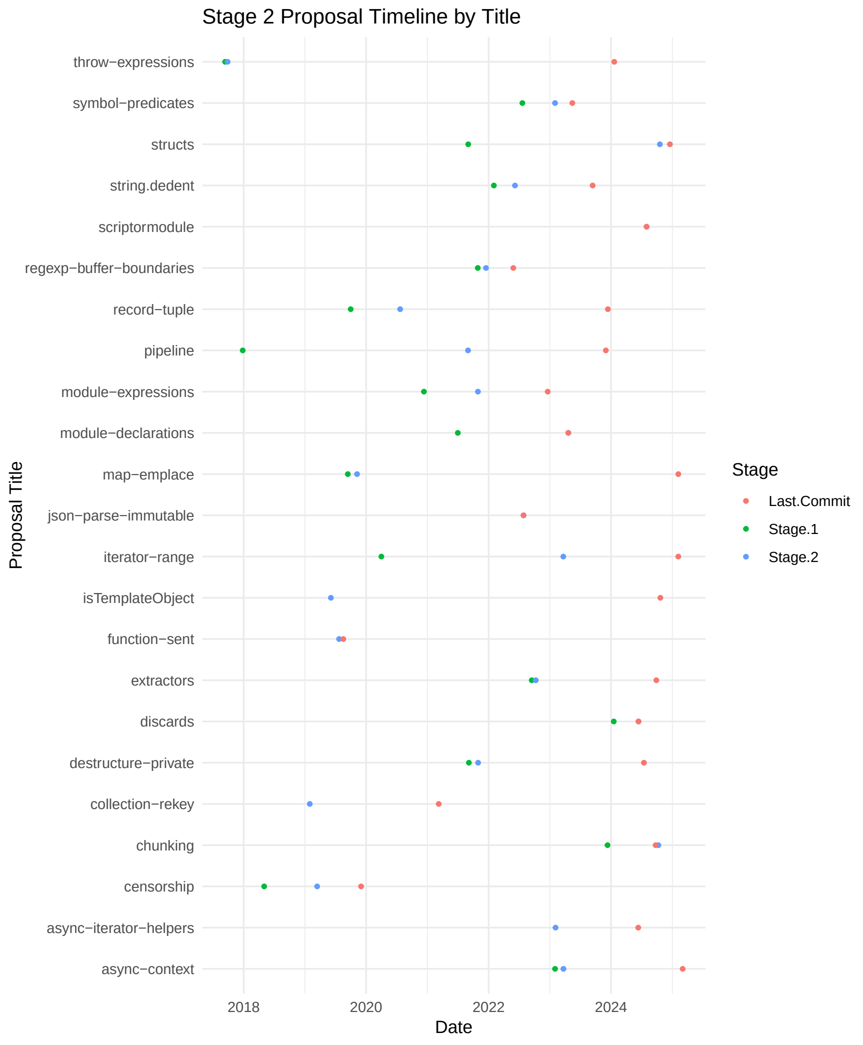
Stage 0



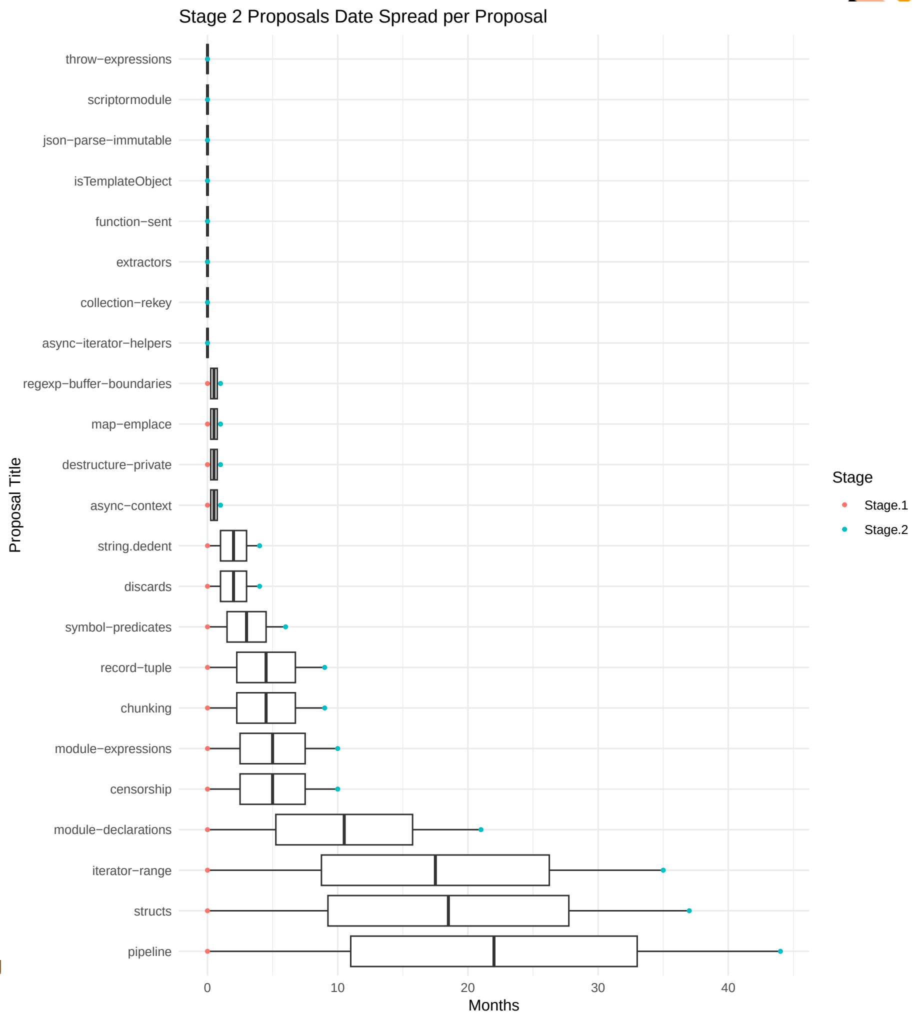
Stage 1



Stage 2

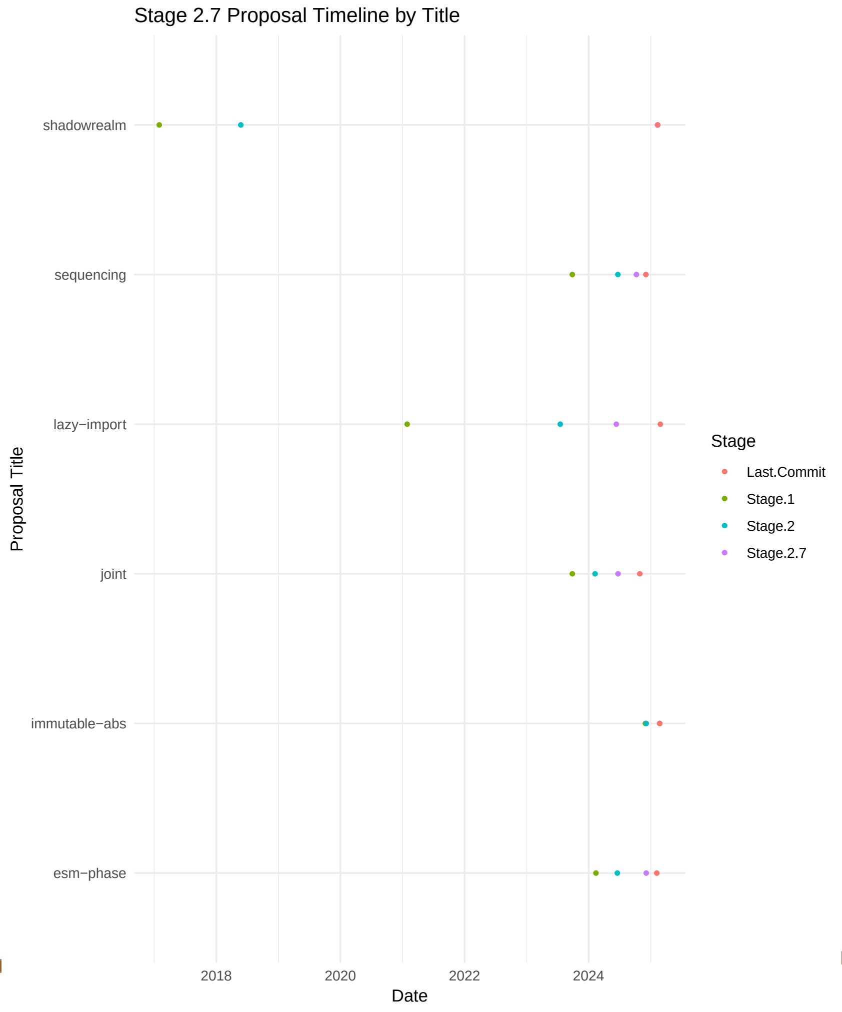


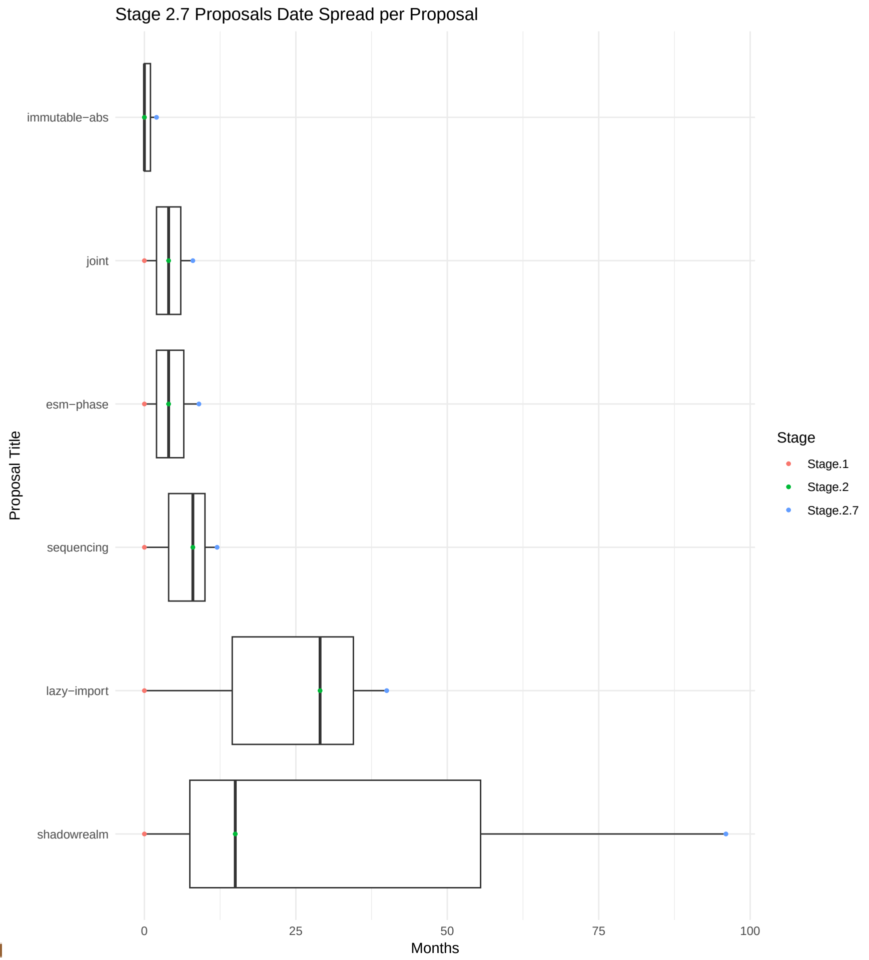
Stage 2 Proposal Timeline by Title
throw−expressions
symbol−predicates
structs
string.dedent
scriptormodule
regexp−buffer−boundaries
record−tuple
pipeline
module−expressions
module−declarations
map−emplace
json−parse−immutable
iterator−range
isTemplateObject
function−sent
extractors
discards
destructure−private
collection−rekey
chunking
censorship
async−iterator−helpers
async−context
Proposal Title
2018
2020
2022
2024
Date
Stage
Last.Commit
Stage.1
Stage.2



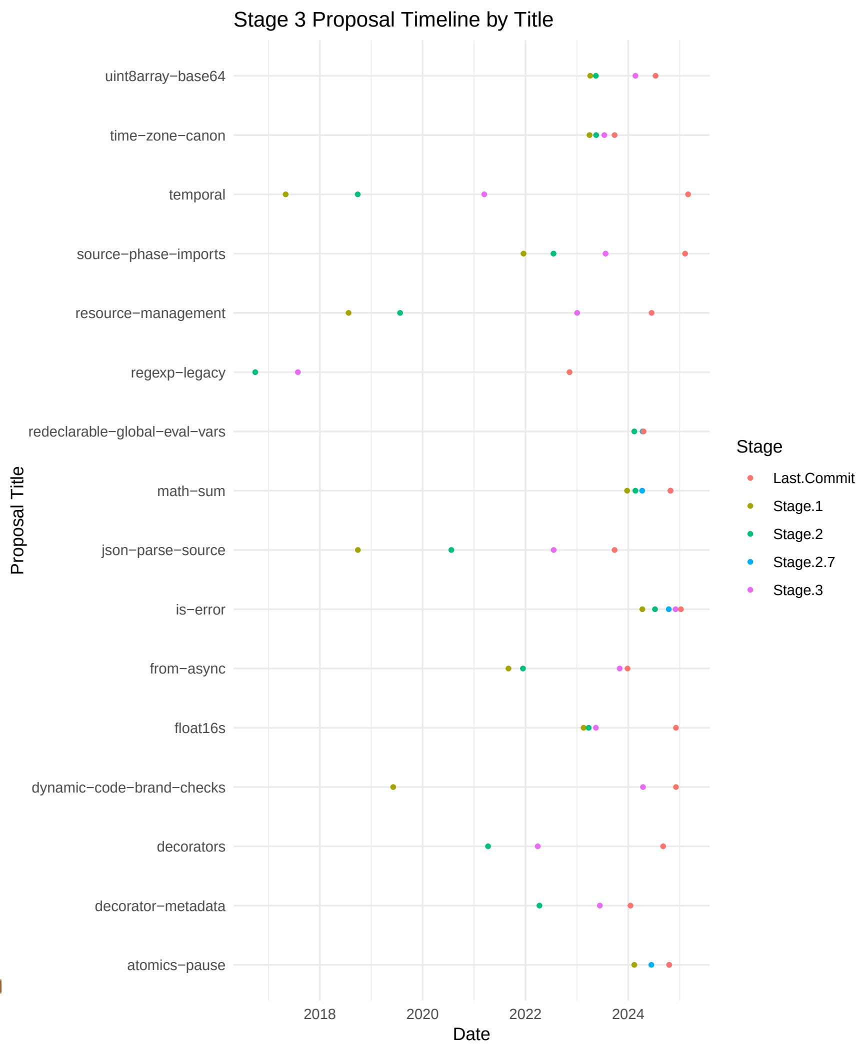
Stage 2 Proposals Date Spread per Proposal
throw−expressions
scriptormodule
json−parse−immutable
isTemplateObject
function−sent
extractors
collection−rekey
async−iterator−helpers
regexp−buffer−boundaries
map−emplace
destructure−private
async−context
string.dedent
discards
symbol−predicates
record−tuple
chunking
module−expressions
censorship
module−declarations
iterator−range
structs
pipeline
Proposal Title
0
10
20
30
40
Months
Stage
Stage.1
Stage.2

Stage 2.7

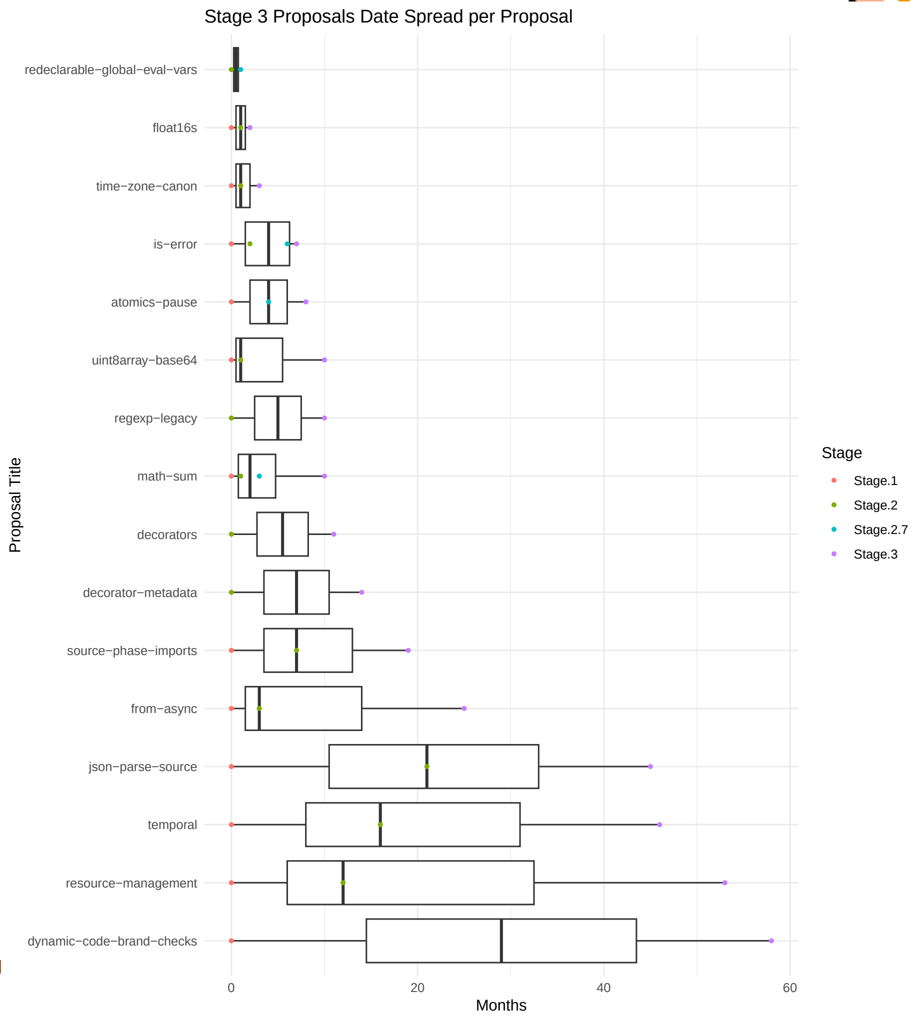

Stage 2.7 Proposal Timeline by Title
Proposal Title
shadowrealm
sequencing
lazy−import
joint
immutable−abs
esm−phase
2018
2020
2022
2024
Date
Stage
Last.Commit
Stage.1
Stage.2
Stage.2.7


Stage 2.7 Proposals Date Spread per Proposal
Proposal Title
immutable−abs
joint
esm−phase
sequencing
lazy−import
shadowrealm
0
25
50
75
100
Months
Stage
Stage.1
Stage.2
Stage.2.7

Stage 3

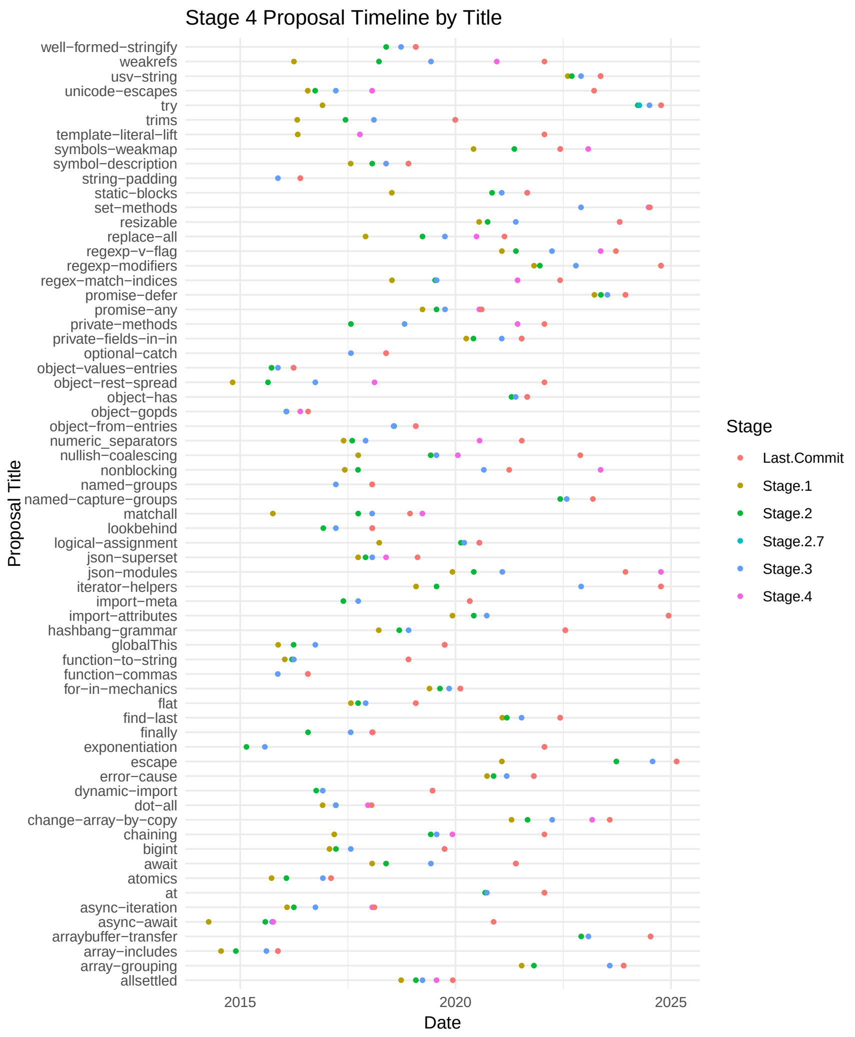


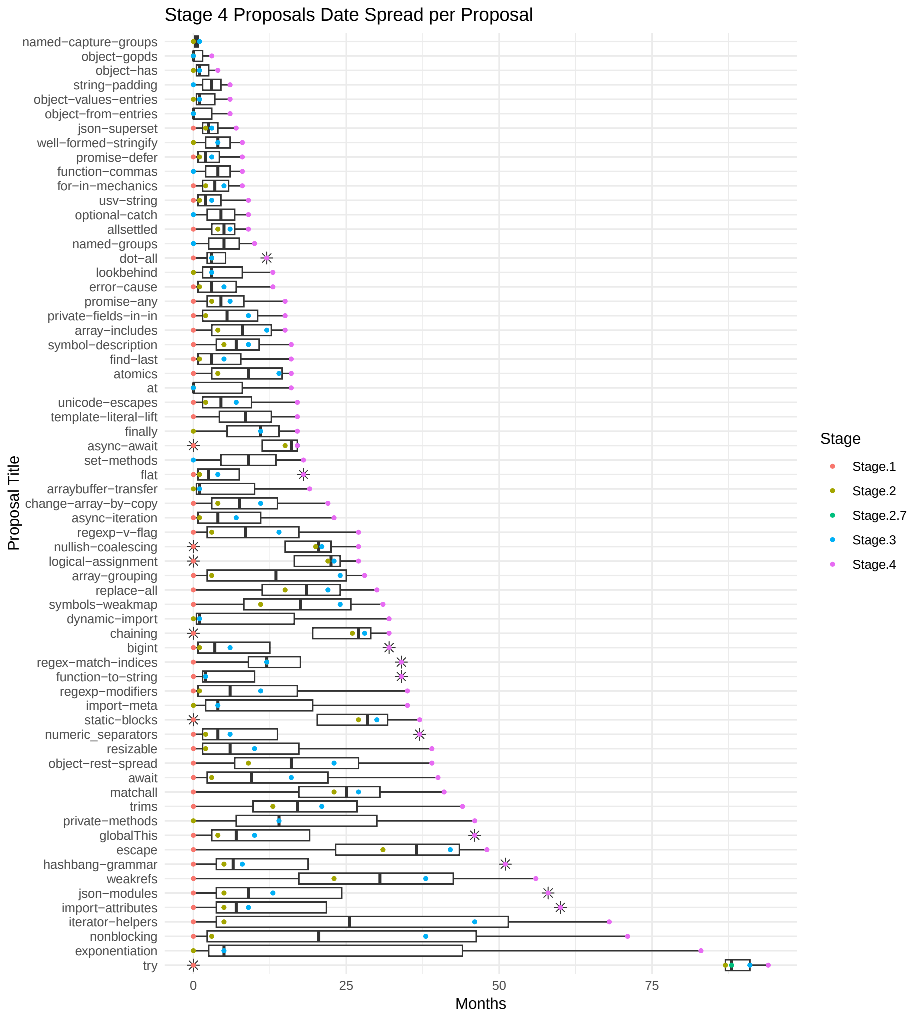
Stage 3 Proposal Timeline by Title
uint8array−base64
time−zone−canon
temporal
source−phase−imports
resource−management
regexp−legacy
redeclarable−global−eval−vars
math−sum
json−parse−source
is−error
from−async
float16s
dynamic−code−brand−checks
decorators
decorator−metadata
atomics−pause
Proposal Title
2018
2020
2022
2024
Date
Stage
Last.Commit
Stage.1
Stage.2
Stage.2.7
Stage.3

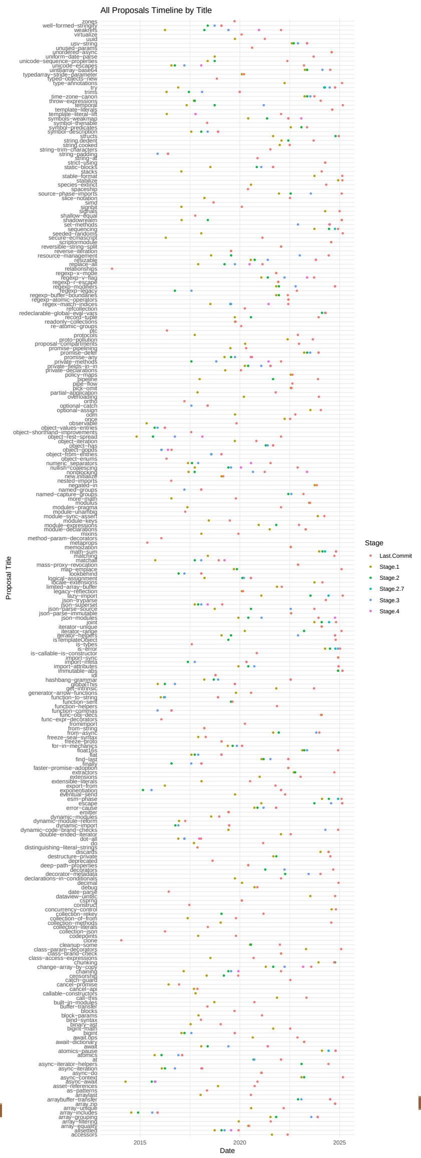


Stage 3 Proposals Date Spread per Proposal
redeclarable−global−eval−vars
float16s
time−zone−canon
is−error
atomics−pause
uint8array−base64
regexp−legacy
math−sum
decorators
decorator−metadata
source−phase−imports
from−async
json−parse−source
temporal
resource−management
dynamic−code−brand−checks
Proposal Title
0
20
40
60
Months
Stage
Stage.1
Stage.2
Stage.2.7
Stage.3

Stage 4

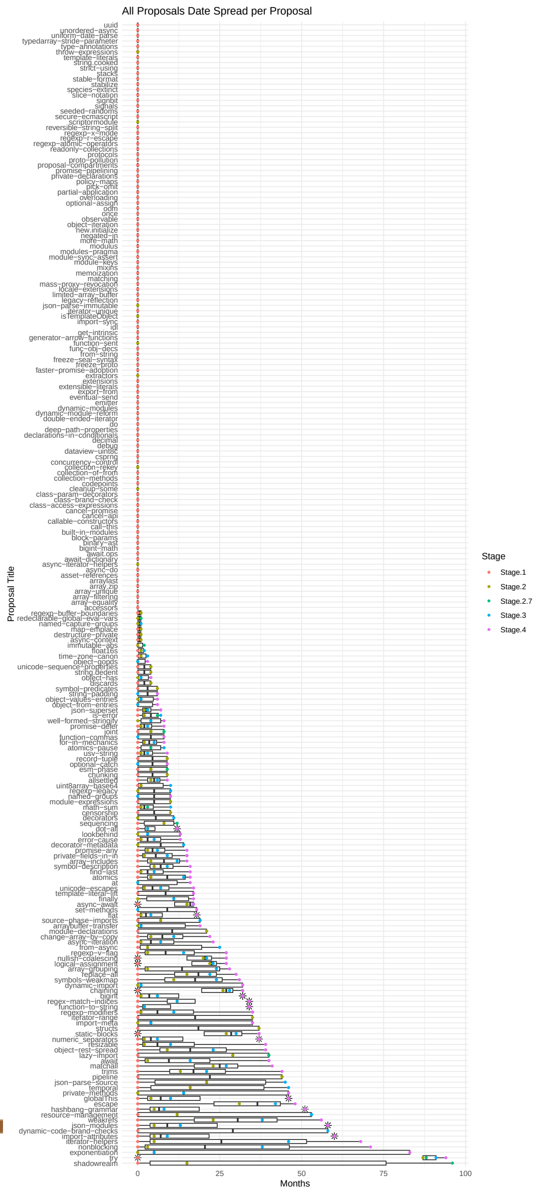

Stage 4 Proposal Timeline by Title
Proposal Title
Date
2015
2020
2025
well−formed−stringify
weakrefs
usv−string
unicode−escapes
try
trims
template−literal−lift
symbols−weakmap
symbol−description
string−padding
static−blocks
set−methods
resizable
replace−all
regexp−v−flag
regexp−modifiers
regex−match−indices
promise−defer
promise−any
private−methods
private−fields−in−in
optional−catch
object−values−entries
object−rest−spread
object−has
object−gopds
object−from−entries
numeric_separators
nullish−coalescing
nonblocking
named−groups
named−capture−groups
matchall
lookbehind
logical−assignment
json−superset
json−modules
iterator−helpers
import−meta
import−attributes
hashbang−grammar
globalThis
function−to−string
function−commas
for−in−mechanics
flat
find−last
finally
exponentiation
escape
error−cause
dynamic−import
dot−all
change−array−by−copy
chaining
bigint
await
atomics
at
async−iteration
async−await
arraybuffer−transfer
array−includes
array−grouping
allsettled
Stage
Last.Commit
Stage.1
Stage.2
Stage.2.7
Stage.3
Stage.4


Stage 4 Proposals Date Spread per Proposal
Proposal Title
Months
0
25
50
75
named−capture−groups
object−gopds
object−has
string−padding
object−values−entries
object−from−entries
json−superset
well−formed−stringify
promise−defer
function−commas
for−in−mechanics
usv−string
optional−catch
allsettled
named−groups
dot−all
lookbehind
error−cause
promise−any
private−fields−in−in
array−includes
symbol−description
find−last
atomics
at
unicode−escapes
template−literal−lift
finally
async−await
set−methods
flat
arraybuffer−transfer
change−array−by−copy
async−iteration
regexp−v−flag
nullish−coalescing
logical−assignment
array−grouping
replace−all
symbols−weakmap
dynamic−import
chaining
bigint
regex−match−indices
function−to−string
regexp−modifiers
import−meta
static−blocks
numeric_separators
resizable
object−rest−spread
await
matchall
trims
private−methods
globalThis
escape
hashbang−grammar
weakrefs
json−modules
import−attributes
iterator−helpers
nonblocking
exponentiation
try
Stage
Stage.1
Stage.2
Stage.2.7
Stage.3
Stage.4



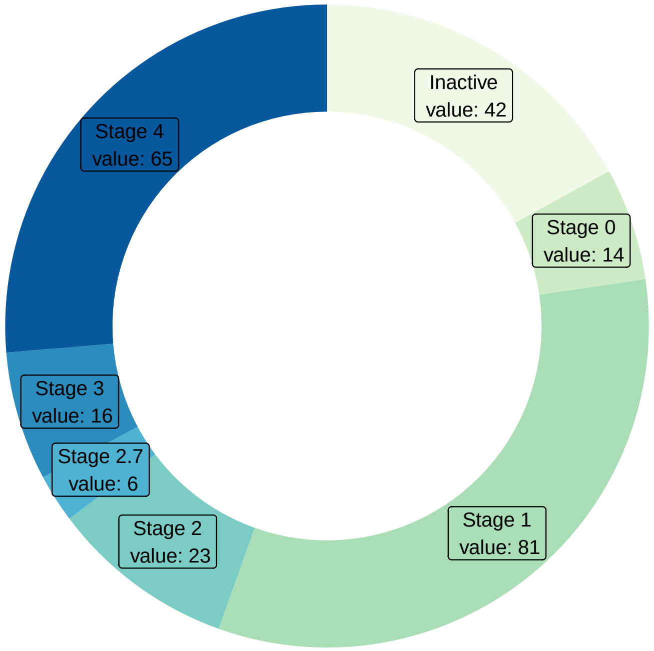
All together

What data can be extracted?
- Classifications
- Stage distribution
- Average duration per stage

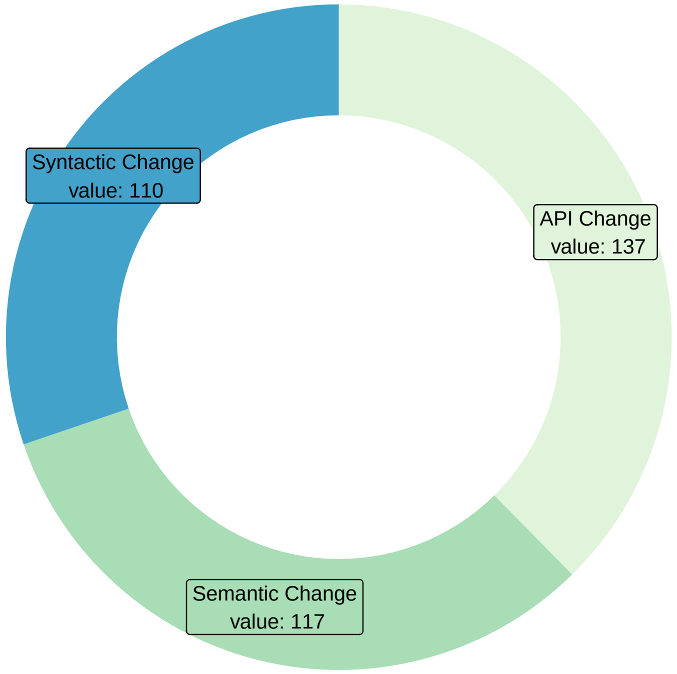


For starters

- Total number of proposals: 257



- Per Classification:

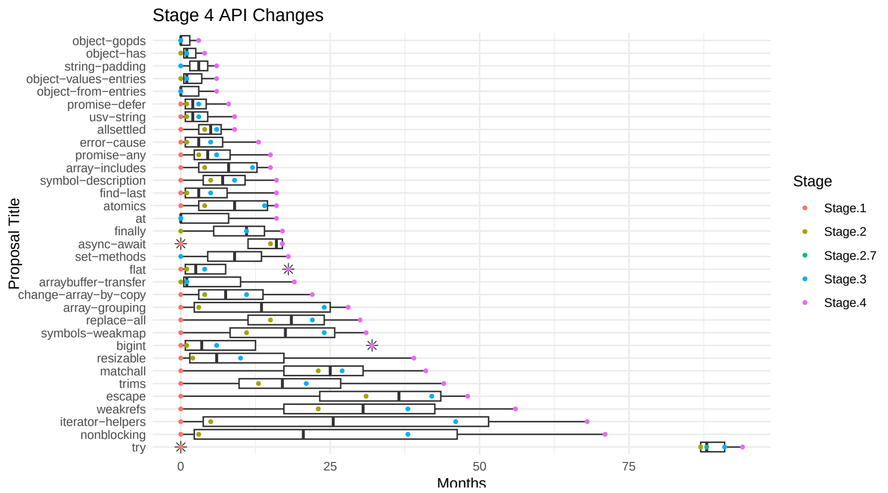


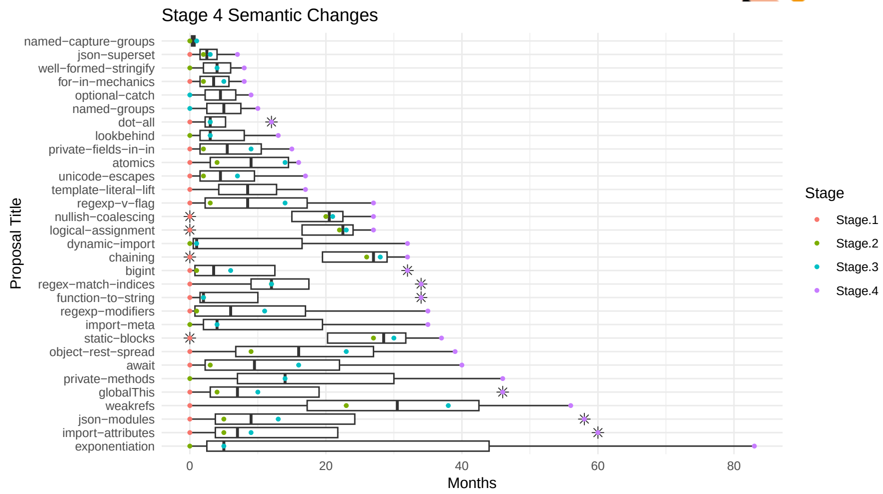
Note: Proposals can overlap classifications

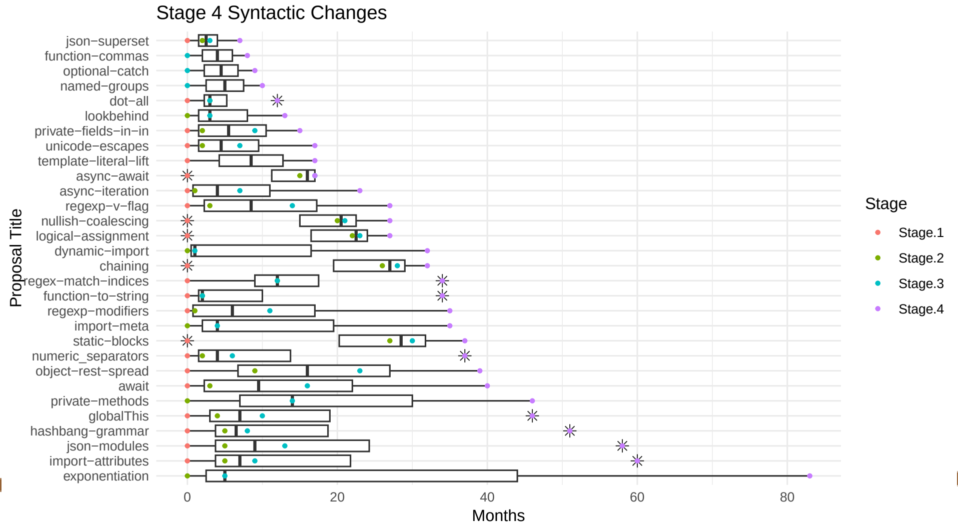
Lets look at Stage 4

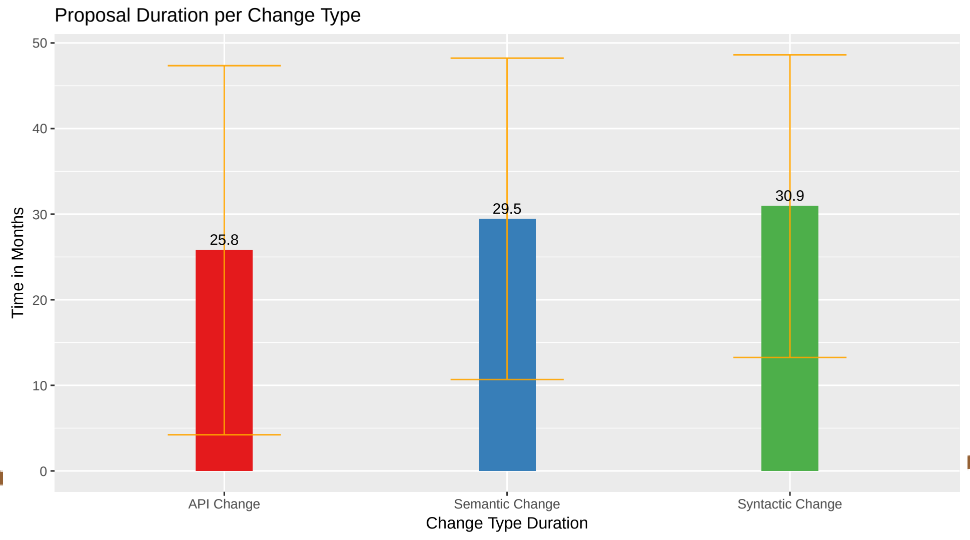
- Most complete data set
- Data gets skewed by the earlier stages

Stage 4: Average Duration per Change





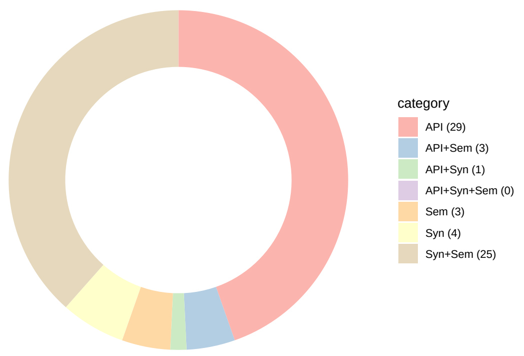


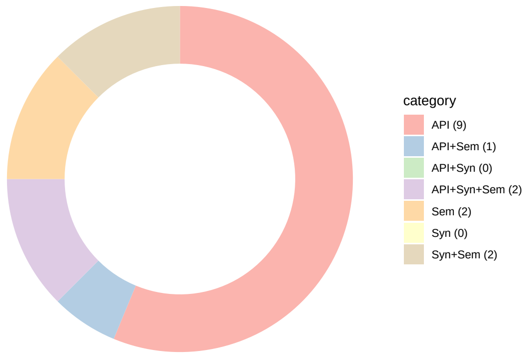


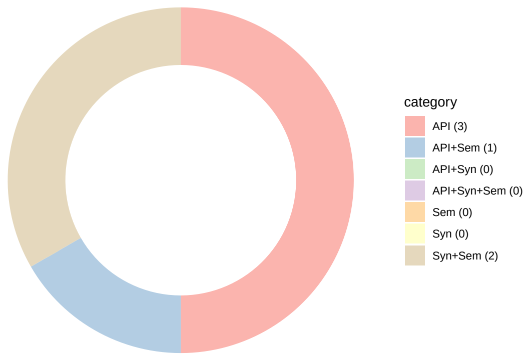
Lets look at more granular classifications

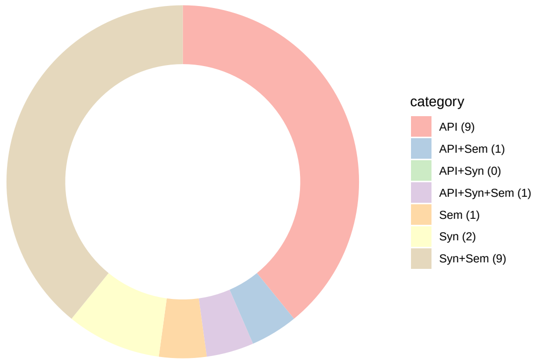
- API only
- Semantic only
- Syntactic only
- API and Semantic
- API and Syntactic
- Semantic and Syntactic
- API and Semantic and Syntactic

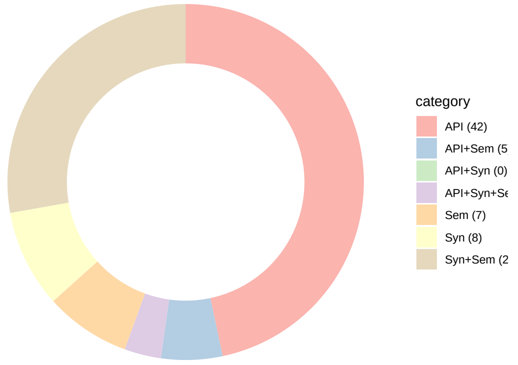
Specific Classifications

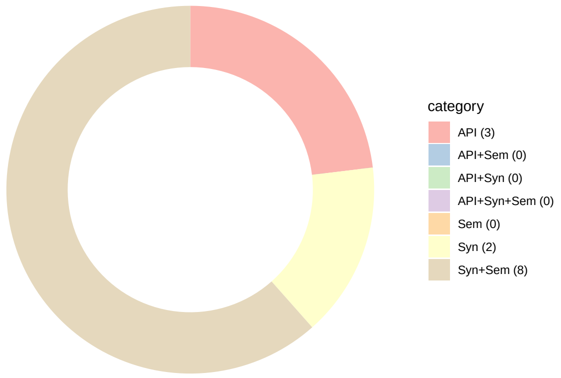

Specific Classification Distribution at Stage 4


Specific Classification Distribution at Stage 3

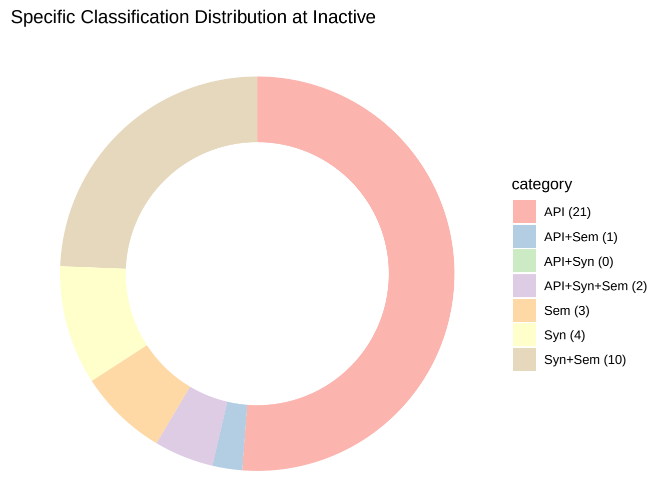

Specific Classification Distribution at Stage 2.7


Specific Classification Distribution at Stage 2

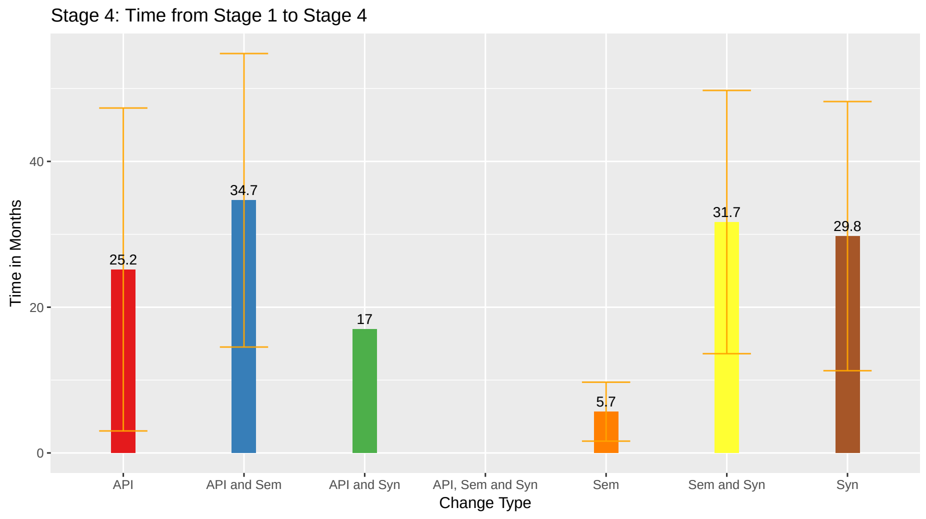

Specific Classification Distribution at Stage 1

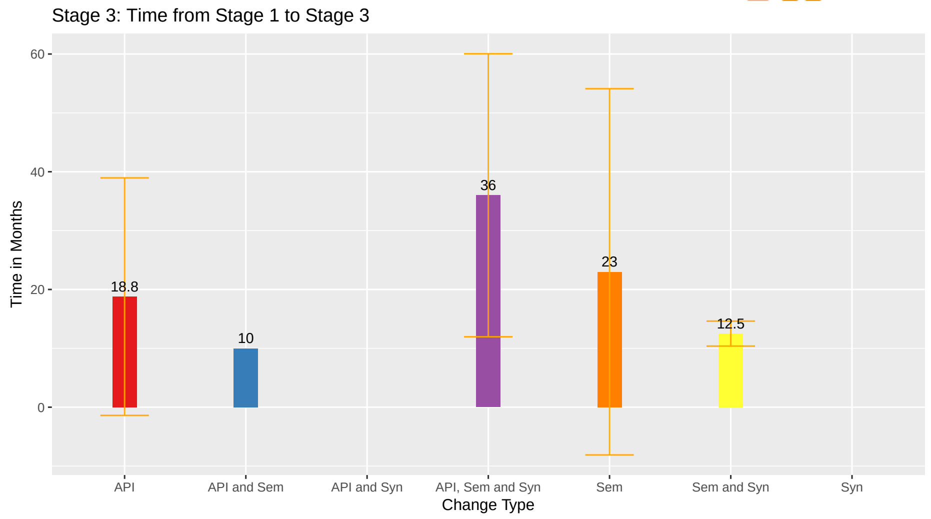

Specific Classification Distribution at Stage 0

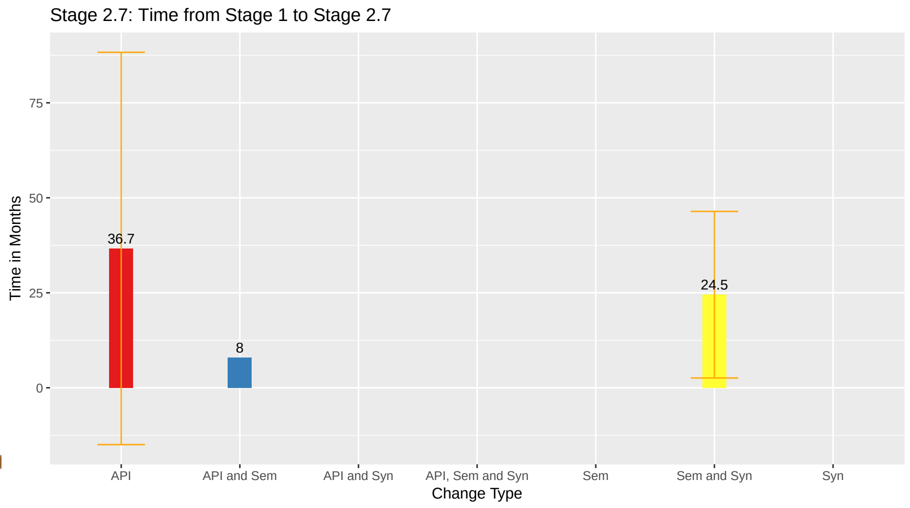
Continuing with Inactive

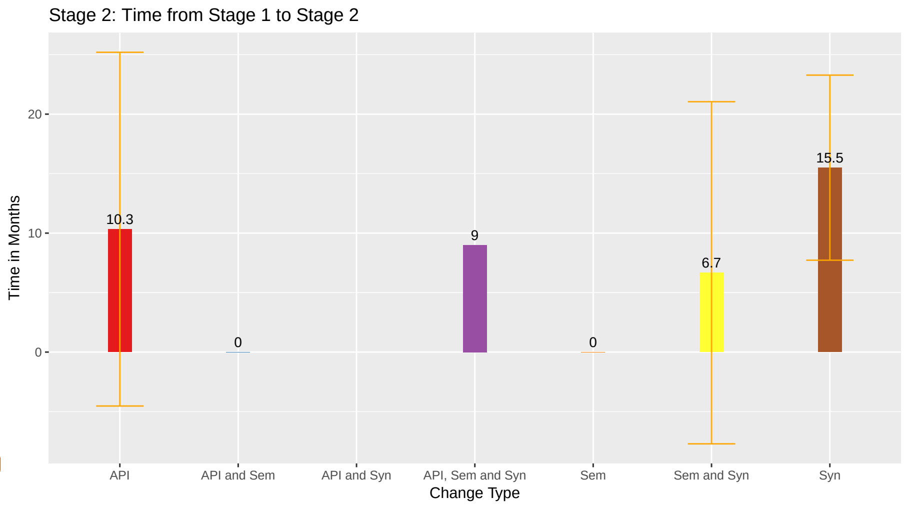


Time From Stage 1

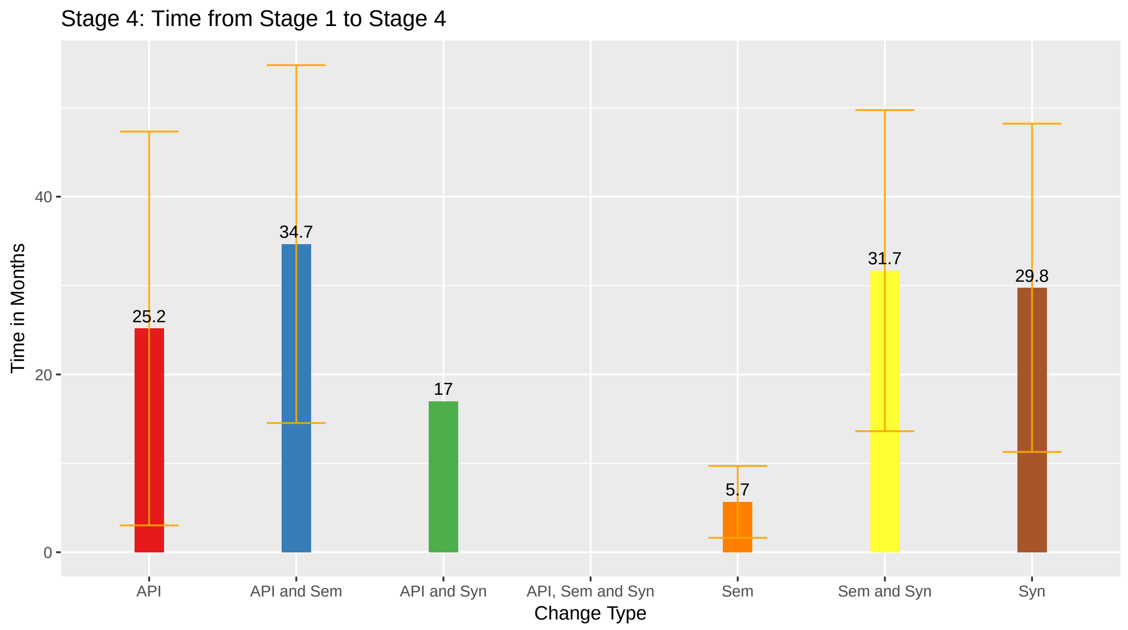


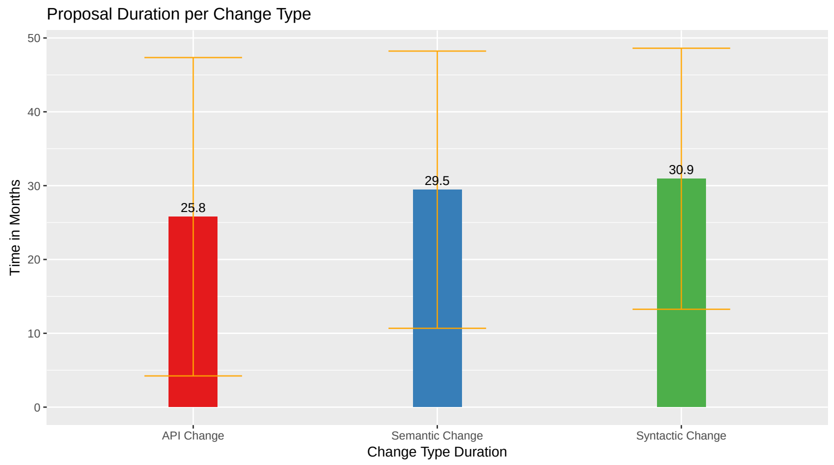




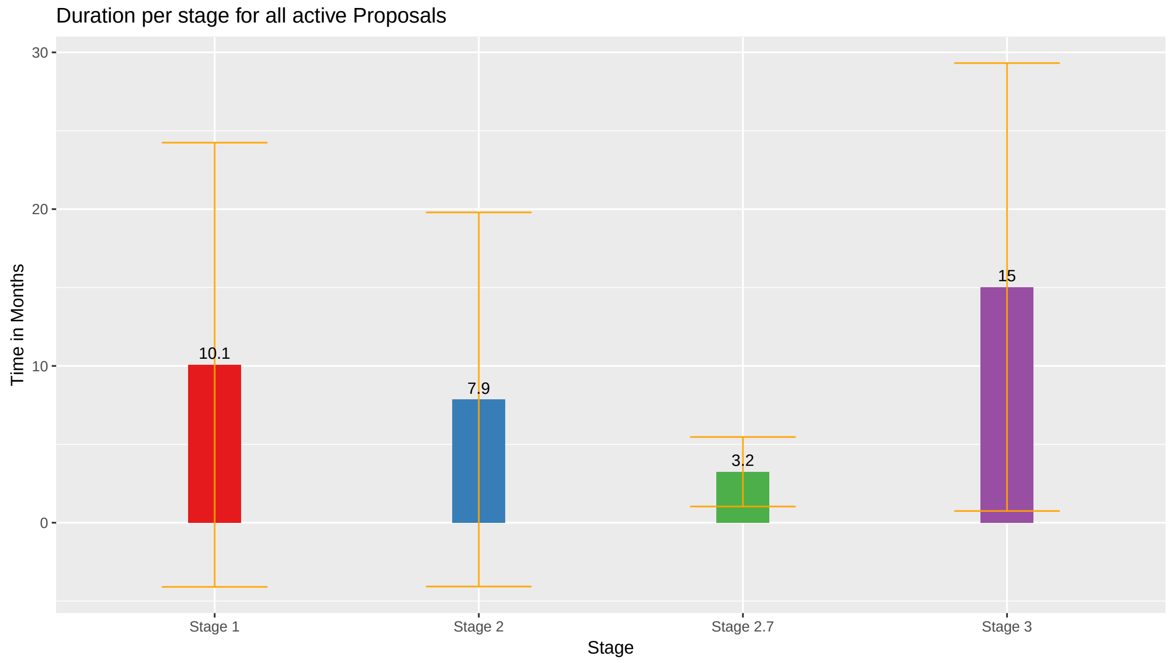


Comparison granular vs overlapping classification





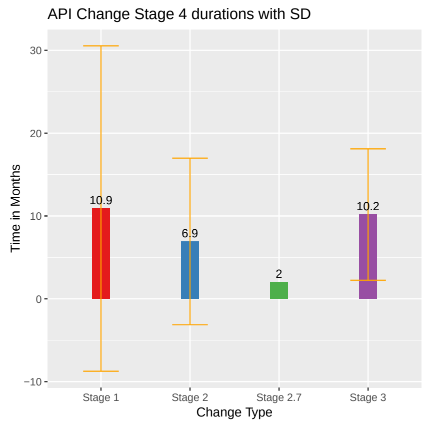
Duration per Stage

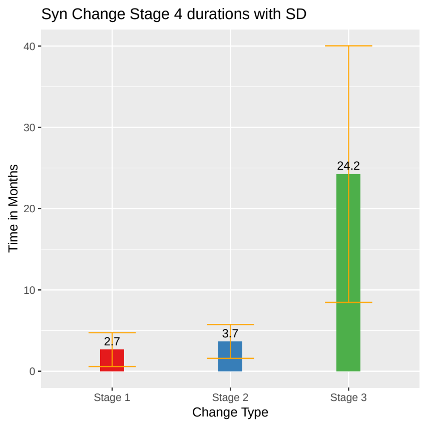


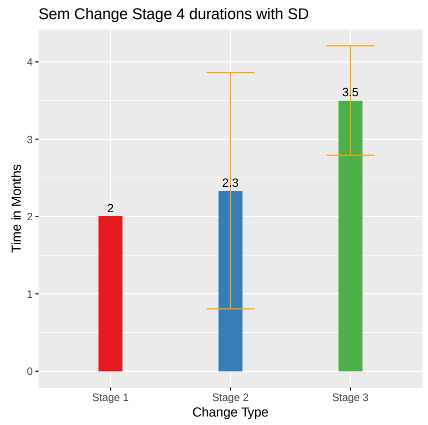
Observations:

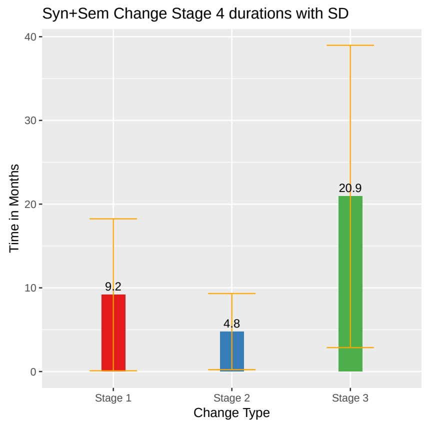
- Length Stage 3 → Stage 1 → Stage 2 → Stage 2.7
- Large SD
- Stage 2.7 is the smallest group

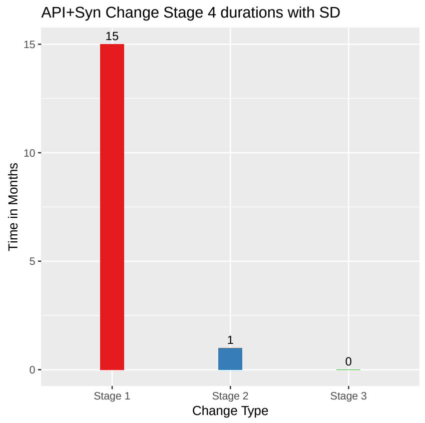
Durations per stage for Stage 4 per classification

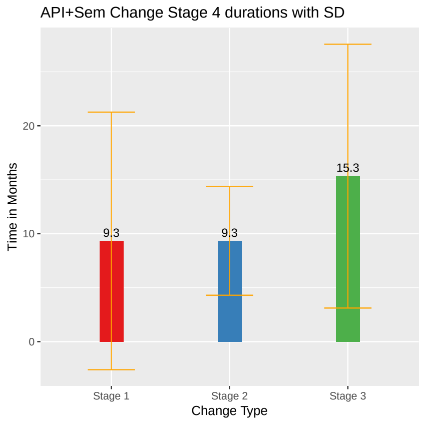




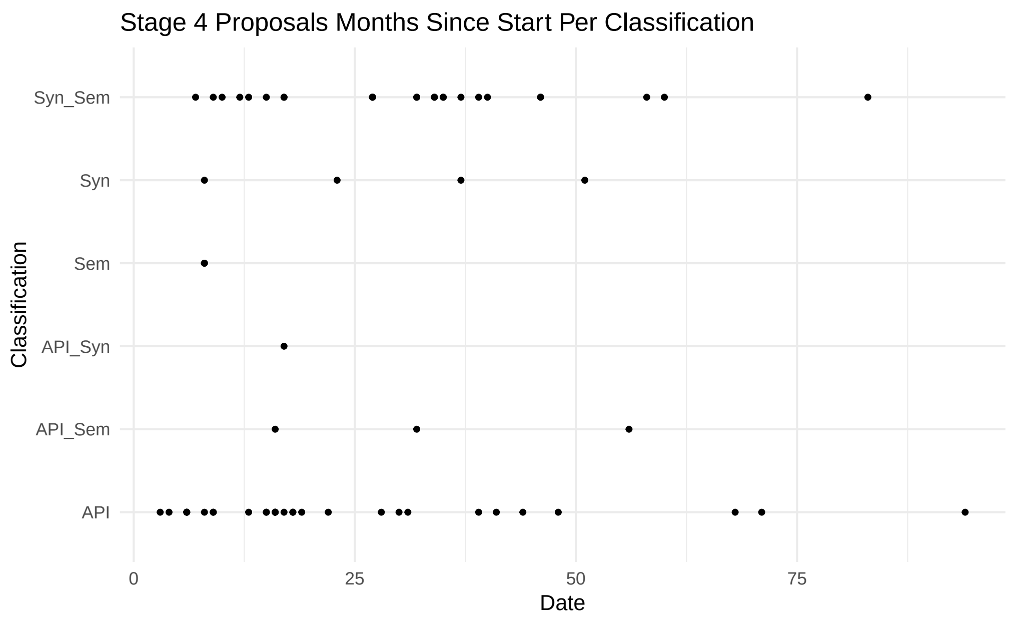


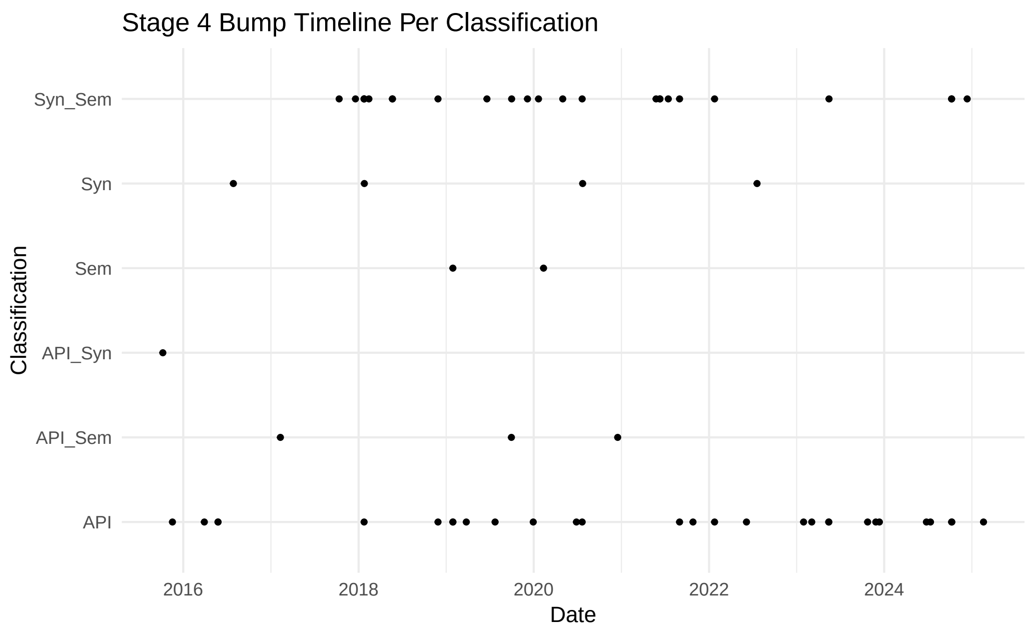






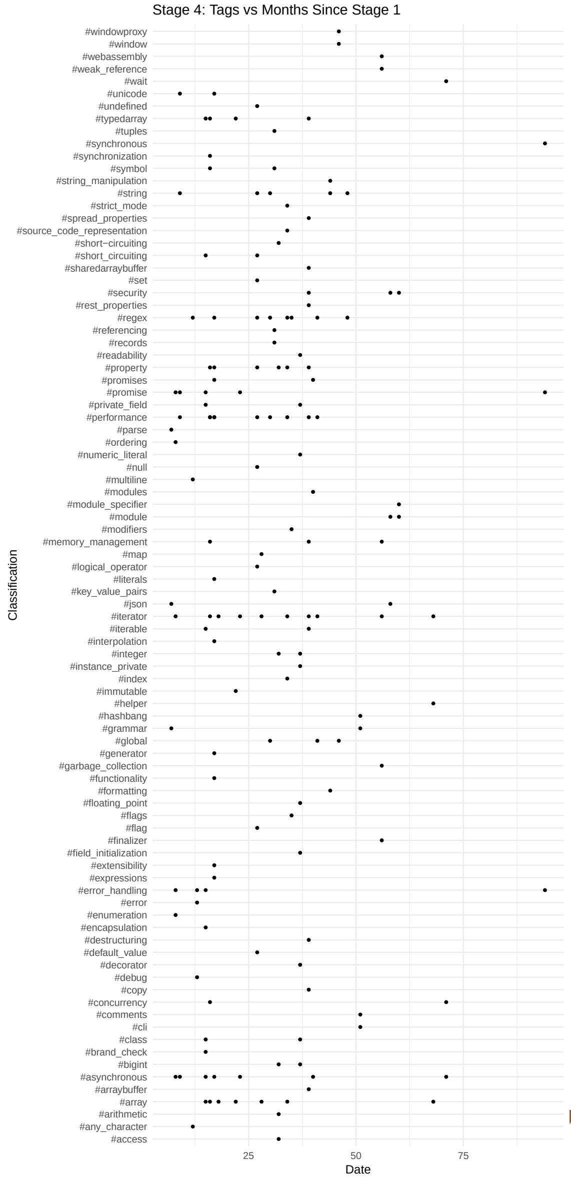
Stage 4 Proposals per Classifications





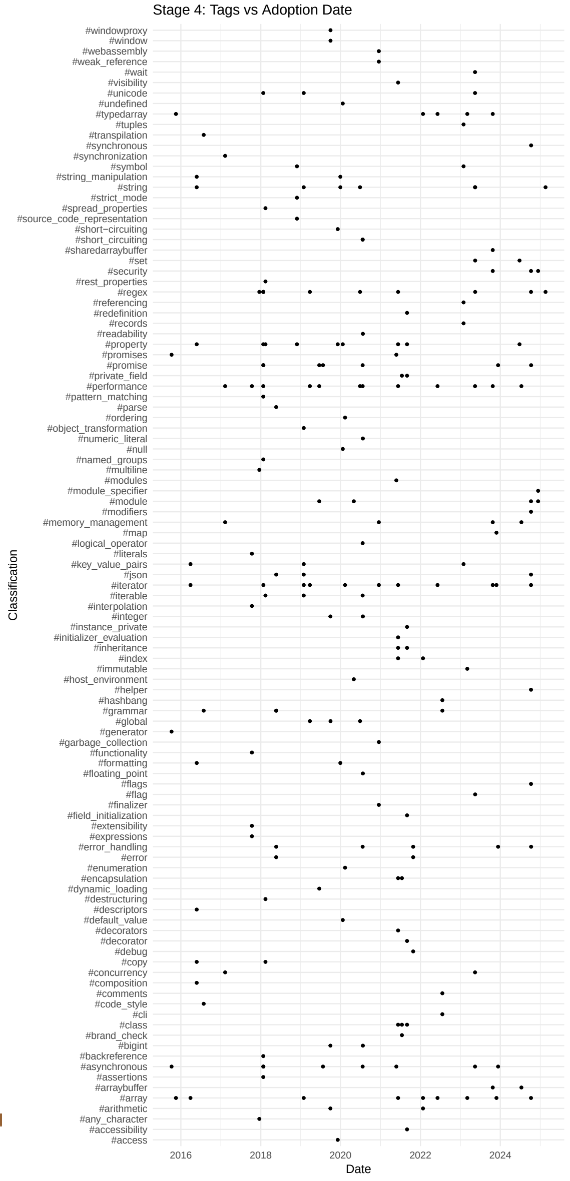
326 Keywords

Rank	Keywords	n
1	#performance	32
2	#iterator	26
3	#asynchronous	22
4	#promise	22
5	#module	21
6	#regex	20
7	#array	19
8	#property	19
9	#class	18
10	#security	18
11	#string	17
12	#error_handling	15
13	#memory_management	15
14	#typedarray	12
15	#concurrency	10
16	#arithmetic	9
17	#destructuring	8
18	#map	8
19	#numeric	8
20	#arraybuffer	7
21	#decorator	7
22	#json	7
23	#math	7
24	#unicode	7
25	#bigint	6
26	#generator	6
27	#global	6
28	#iterable	6
29	#key_value_pairs	6
30	#parse	6
31	#realm	6
32	#symbol	6
33	#date_time	5
34	#encapsulation	5
35	#grammar	5
36	#metadata	5
37	#operator	5
38	#pattern_matching	5
39	#readability	5
40	#resource_management	5
41	#set	5
42	#string_manipulation	5
43	#synchronous	5
44	#wait	5
45	#accessor	4



Keywords Continued

- Too many keywords
- Reduce to 20



Topics

- Topics are broader than keywords
- Keywords are more individual
- Can be refined but this is a starting point

Rank	Topics	Count
1	#others	281
2	#objects	131
3	#async	51
4	#arrays	47
5	#iterators	45
6	#modules	37
7	#numbers	36
8	#performance	32
9	#concurrency	31
10	#collections	25
11	#regex	25
12	#security	23
13	#memory	22
14	#intl	21
15	#functions	12
16	#types	11
17	#realms	9
18	#ergonomics	8
19	#json	6
20	#webassembly	2