What's new coming up in ECMAScript 2022?

Chetan Gawai
Software Engineer @Saeloun

Chetan Gawai

Senior Software Engineer @ Saeloun
JavaScript and ReactJs Enthusiast | Blogger
Website | Twitter | Linkedin | Github

Contents

Backstory of ECMAScript

- **ECMA International**

  Organization dedicated to standardization of information and communication systems

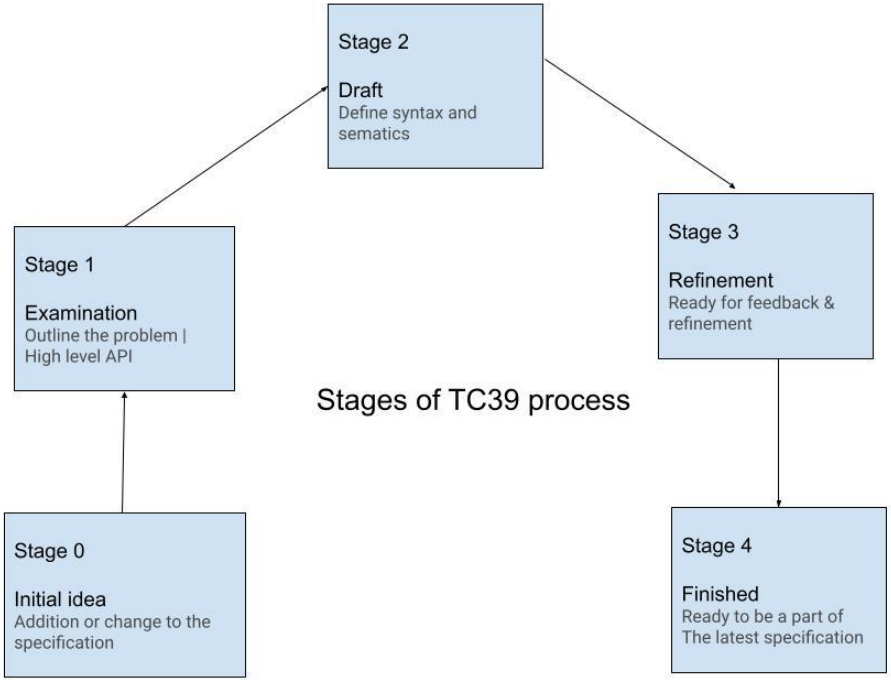
- **TC39 committee**

  Committee at ECMA international which looks into the evolution of JavaScript

- **ECMAScript**

  A set of rules on how a language should work .
  These rules are used by browsers to developer their engines.

The TC39 Process



New features coming up in ECMAScript 2022

- Class Fields (Private instance methods and accessors, Class Public Instance Fields & Private Instance Fields, Static class fields and private static methods)
- Ergonomic brand checks for Private Fields
- Class Static Block
- Top-level await
- RegExp Match Indices
- Object.hasOwn()
- Addition of .at() method in Array, String, TypedArray
- Error Cause

Class field declarations

- Classes were introduced in ECMAScript 2015 using constructor method for initialization
- ECMAScript 2022 adds the new class fields syntax allowing class fields to be initialized on the top level of class
- Simplifies the class definition making the code look pretty and readable
- Example

Private instance fields, accessors, methods

- JavaScript lacked making class fields and methods private since inception. Though people followed convention of using `_` for making the fields and methods private, they were still fully public.
- ECMAScript 2022 introduced prefixing fields, methods, accessors using `#` to make them private.
- Private fields, accessors, methods are not accessible in subclass
- Limitations of private fields:

  - Should be declared upfront in the field declaration

  - Cannot be deleted

- Example

Ergonomic brand checks for private fields

- Accessing undeclared public fields => ‘undefined’
  Accessing undeclared private fields => throws error
- To check if an object has a private fields, try-catch could be used. Too much to write though😔
- ECMAScript 2022 provides **‘in’** to check if object has private fields/methods
- Some people suggested optional chaining but it does not prevent exceptions
- Example

Static class fields & private static methods

- Useful when a field should exist per class not per instance
  Use-cases: Caching, fixed-configuration
- Static public methods were introduced in ES2015
- ECMAScript 2022 adds the remaining
  - Static public fields
  - Static private fields
  - Static private methods
- Example

Class static block

- ECMAScript 2022 adds Class static block feature to evaluate static initialization elegantly
- The static block has access to private fields of class
- Example

Top-level await

- Top level await enables developers to use await keyword outside async function
- It acts like a async function causing other modules who import them to wait before they start evaluating
- Use-cases

  - Loading modules dynamically

```
const strings = await import(`/i18n/${navigator.language}`);
```

  - Resource initialization

```
const connection = await dbConnector();
```

  - Dependency fallback

```
let translations;
try {
      translations = await import('https://app.fr.json');
    } catch {
      translations = await import('https://fallback.en.json');
    }
```

- Example

RegExp Match Indices

- Regular expression is used for matching text with pattern
- RegExp.exec and String.matchAll return matches and the indices of the match but not end indices
- ECMAScript 2022 adds a new flag ‘/d’ to provide start and end indices of the matched string
- Example

Object.hasOwn(object, property)

- JavaScript has `**Object.prototype.hasOwnProperty**` to check if object has a particular property.
- But it does not work with all objects - ‘Object.create(null)’
- ECMAScript introduces Object.hasOwn(object, property) to safely check for object properties
- Example

Addition of .at() method in Array, String, TypedArray

- JavaScript is missing the ability to do negative indexing
- ECMAScript 2022 adds `.at(index)` method to access the elements of array from the end by specifying negative index
- Example

Error cause

- Error() constructor is used to report errors occurring at runtime
- ECMAScript 2022 provides a ‘cause’ property to be added to the `Error()` constructor allowing errors to be chained
- Example

References

- Code snippets - https://github.com/chetangawai/ecmascript_2022_snippets/
- ECMAScript finished proposals - https://github.com/tc39/proposals/blob/main/finished-proposals.md
- TC39 process - https://tc39.es/process-document/

Questions?

Thank you!