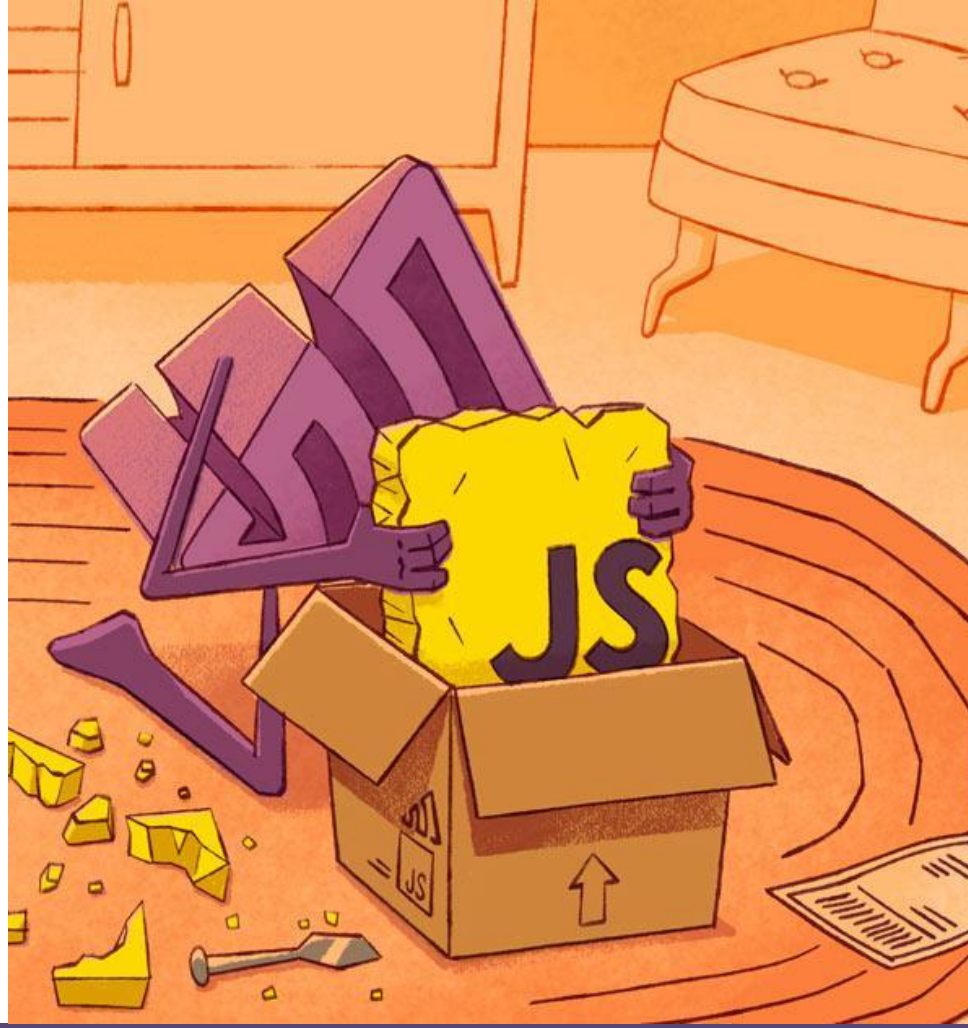
Understanding the Performance Impact of Generated JavaScript

DeveloperWeek 2023



Shana Matthews

- DevRel at Sentry.io
- Fun facts
  - Every version of Windows since 2016 has shipped with code I've written
  - I once delayed a Windows 10 major release

Abhijeet Prasad

- SDK engineer at Sentry.io
- Maintains Sentry JavaScript SDKs
- Cool dude
- Largely responsible for making all this happen

Agenda

1. The background
2. Why & how to minify your JS
3. The results

What is Sentry?

The problem

SDK too big.

The problem

SDK too big.

- V6 JavaScript SDK was 74.47kb (minified, un-gzipped)
- New performance features would increase package size further
- “Package size is massive” issue filed on the SDK

The goal

Reduce SDK size by 30% and improve tree shaking

The goal

Reduce SDK size by 30% and improve tree shaking

Measure minified CDN bundle, not gzipped

Track bundle size using the size-limit library

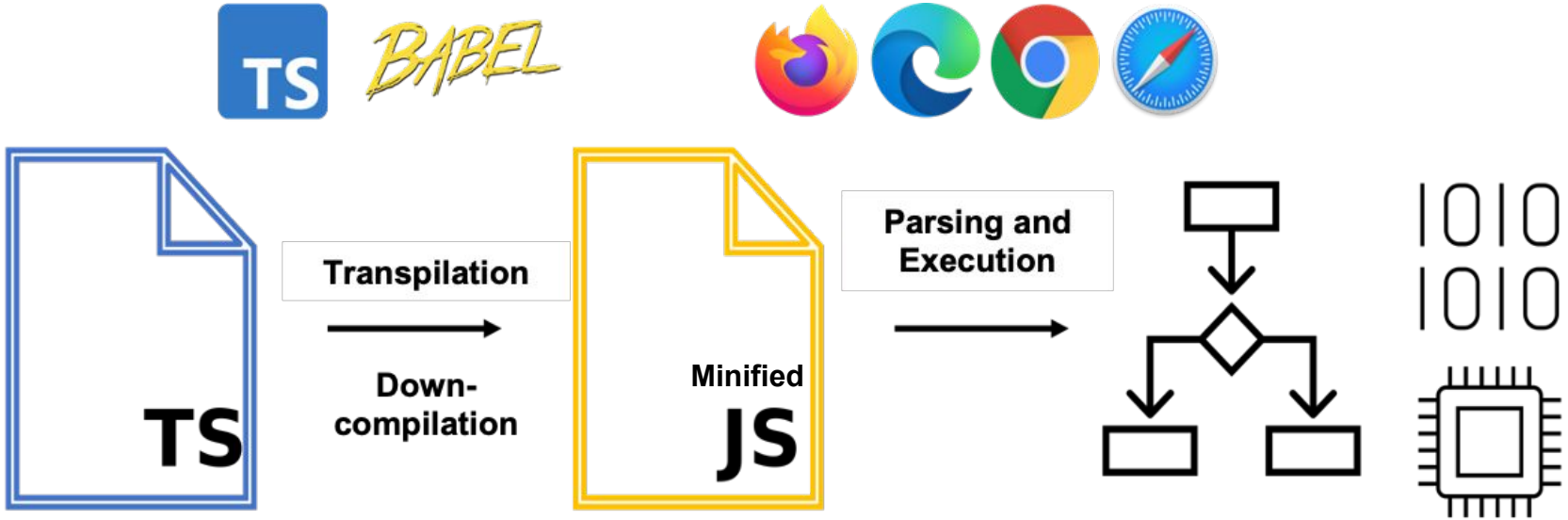
github-actions bot commented on Apr 19 • edited

size-limit report 📦

Path	Size
@sentry/browser - ES5 CDN Bundle (gzipped + minified)	19.86 KB (-1.04% 🔽)
@sentry/browser - ES5 CDN Bundle (minified)	63.09 KB (-2.26% 🔽)
@sentry/browser - ES6 CDN Bundle (gzipped + minified)	18.54 KB (-1.29% 🔽)
@sentry/browser - ES6 CDN Bundle (minified)	56.81 KB (-1.98% 🔽)
@sentry/browser - Webpack (gzipped + minified)	21.77 KB (-6.2% 🔽)
@sentry/browser - Webpack (minified)	74.12 KB (-10.11% 🔽)
@sentry/react - Webpack (gzipped + minified)	21.81 KB (-6.21% 🔽)
@sentry/nextjs Client - Webpack (gzipped + minified)	46.21 KB (-3.72% 🔽)
@sentry/browser + @sentry/tracing - ES5 CDN Bundle (gzipped + minified)	25.67 KB (-1.2% 🔽)
@sentry/browser + @sentry/tracing - ES6 CDN Bundle (gzipped + minified)	24.09 KB (-1.31% 🔽)

☺ 🚀 1

Understanding generated JavaScript



Defintions

Transpilation: the process of converting source code of one language to another language

Down-compilation: the process of converting source code to a more backward-compatible version of that source code.

Prepping for the refactor

- V6 -> v7 major release
- Goals:
  - Improve SDK tree shaking
  - Improve web bundling optimizations
  - No breaking changes to users (public API stays the same)
- **Tree shaking:** the SDK can remove unused code, so devs aren't getting bloat from features they're not using

Prepping for the refactor

Ramped up testing so we could refactor confidently:

- New Playwright browser-based integration tests
- New Node integration tests on our own custom framework (built on Nock library)

Categories of optimizations

1. Optimizations for **down-compiling & transpiling**
2. Optimizations for **minification**

Categories of optimizations

1. **Optimizations for down-compiling & transpiling**
2. Optimizations for minification

1. Down-compiling & transpiling

- Removing usages of **optional chaining**
- Using **const enums** or **string constants** instead of TypeScript enums

1. Down-compiling & transpiling

- **Removing usages of optional chaining**
- Using const enums or string constants instead of TypeScript enums

Optional chaining

```
// optional chaining
if (hey?.me) {
    console.log('me');
}
```

Optional chaining

```
// down-compiled to ES6
if (hey !== null && hey !== void 0 && hey.me) {
  console.log('me');
}
```

Optional chaining

```
// boolean short circuit
if (hey && hey.me) {
  console.log('me');
}
```

example PRs

1. Down-compiling & transpiling

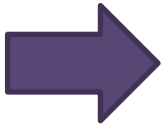
- Removing usages of optional chaining
- **Using const enums or string constants instead of TypeScript enums**

TypeScript enums to…

```
// TS enum
/** SyncPromise internal states */
enum States {
  /** Pending */
  PENDING,
  /** Resolved / OK */
  RESOLVED,
  /** Rejected / Error */
  REJECTED,
}
```

TypeScript enums to const enums

```ts
// TS enum
/** SyncPromise internal states */
enum States {
  /** Pending */
  PENDING,
  /** Resolved / OK */
  RESOLVED,
  /** Rejected / Error */
  REJECTED,
}
```



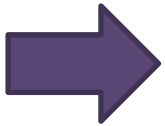
```ts
// generated JS
/** SyncPromise internal states */
var States;
(function (States) {
  /** Pending */
  States[States["PENDING"] = 0] = "PENDING";
  /** Resolved / OK */
  States[States["RESOLVED"] = 1] = "RESOLVED";
  /** Rejected / Error */
  States[States["REJECTED"] = 2] = "REJECTED";
})(States || (States = {}));
```

TypeScript enums to const enums

```
// const enum
/** SyncPromise internal states */
const enum States {
  Pending,
  Resolved,
  Rejected,
}
```

TypeScript enums to string constants

```ts
// string enum
export enum Severity {
  /** JSDoc */
  Fatal = 'fatal',
  /** JSDoc */
  Error = 'error',
  /** JSDoc */
  Warning = 'warning',
  /** JSDoc */
  Log = 'log',
  /** JSDoc */
  Info = 'info',
  /** JSDoc */
  Debug = 'debug',
  /** JSDoc */
  Critical = 'critical',
}
```



```ts
// generated output
export var Severity;
(function (Severity) {
  /** JSDoc */
  Severity["Fatal"] = "fatal";
  /** JSDoc */
  Severity["Error"] = "error";
  /** JSDoc */
  Severity["Warning"] = "warning";
  /** JSDoc */
  Severity["Log"] = "log";
  /** JSDoc */
  Severity["Info"] = "info";
  /** JSDoc */
  Severity["Debug"] = "debug";
  /** JSDoc */
  Severity["Critical"] = "critical";
})(Severity || (Severity = {}));
```

TypeScript enums to string constants

```
// string enum
export enum Severity {
  /** JSDoc */
  Fatal = 'fatal',
  /** JSDoc */
  Error = 'error',
  /** JSDoc */
  Warning = 'warning',
  /** JSDoc */
  Log = 'log',
  /** JSDoc */
  Info = 'info',
  /** JSDoc */
  Debug = 'debug',
  /** JSDoc */
  Critical = 'critical',
}
```

TypeScript enums to string constants

```ts
export const SeverityLevels = ['fatal', 'error', 'warning', 'log', 'info', 'debug', 'critical'] as const;
export type SeverityLevel = typeof SeverityLevels[number];
```

example PR

Categories of optimizations

1. Optimizations for down-compiling & transpiling
2. **Optimizations for minification**

2. Optimizations for minification

Minification = making your JavaScript assets as small as possible

- Remove whitespace, comments, unnecessary tokens
- Shorten variable, function names

Sentry uses terser.

```
// An example JS function
export function theBestFunction(arg1, arg2) {
  const bestObject = {
    key: arg1,
    veryVeryLongKey: {
      nestedKey: arg2,
    }
  }
  return bestObject;
}
```

```
export function
theBestFunction(e,n){return{key:e,veryVeryLongKey:{nestedKey:n}}}
```

2. Optimizations for minification

- Using **try-catch blocks** to simplify code requiring nested object access
- Local **aliases for object keys** to improve minification
- Converting **classes to objects and functions**

2. Optimizations for minification

- **Using try-catch blocks to simplify code requiring nested object access**
- Local aliases for object keys to improve minification
- Converting classes to objects and functions

Nested object access

```
// An example JS function
export function theBestFunction(arg1, arg2) {
  const bestObject = {
    key: arg1,
    veryVeryLongKey: {
      nestedKey: arg2,
    }
  }
  return bestObject;
}
```

```
export function
theBestFunction(e,n){return{key:e,veryVeryLongKey:{nestedKey:n}}}
```

Nested object access in our codebase

```
// example of this in
// packages/core/src/integrations/inboundfilters.ts
try {
  return (
    (event &&
      event.exception &&
      event.exception.values &&
      event.exception.values[0] &&
      event.exception.values[0].type === 'SentryError') ||
      false
  );
  } catch (_oO) {
  return false;
}
```

Simplify with a try catch

```
try {
  // @ts-ignore can't be a sentry error if undefined
  // eslint-disable-next-line @typescript-eslint/no-unsafe-member-access
  return event.exception.values[0].type === 'SentryError';
} catch (e) {
  // ignore
}
```

example PR

2. Optimizations for minification

- Using try-catch blocks to simplify code requiring nested object access
- **Local aliases for object keys to improve minification**
- Converting classes to objects and functions

Aliasing object keys to local variables

```
// alias to local variables
function enhanceEventBefore(event: Event, url: any, line: any, column:
  any): Event {
  event.exception = event.exception || {};
  event.exception.values = event.exception.values || [];
  event.exception.values[0] = event.exception.values[0] || {};
  event.exception.values[0].stacktrace = event.exception.values[0].stacktrace || {};
  event.exception.values[0].stacktrace.frames = event.exception.values[0].stacktrace.frames || [];
  // ...
}
```

Aliasing object keys to local variables

```
function enhanceEventAfter(event: Event, url: any, line: any, column:any):Event {
  // event.exception
  const e = (event.exception = event.exception || {});
  // event.exception.values
  const ev = (e.values = e.values || []);
  // event.exception.values[0]
  const ev0 = (ev[0] = ev[0] || {});
  // event.exception.values[0].stacktrace
  const ev0s = (ev0.stacktrace = ev0.stacktrace || {});
  // event.exception.values[0].stacktrace.frames
  const ev0sf = (ev0s.frames = ev0s.frames || []);
  // ...
}
```

After minification

```
// no local aliases
// 352 bytes
function enhanceEventBefore(n, t, e, i) {
  n.exception = n.exception || {}, n.exception.values =
n.exception.values || [], n.exception.values[0] =
n.exception.values[0] || {}, n.exception.values[0].stacktrace =
n.exception.values[0].stacktrace || {},
  n.exception.values[0].stacktrace.frames =
n.exception.values[0].stacktrace.frames || []
}
```

```
// with local aliases
// 232 bytes
function enhanceEventAfter(n, t, e, i) {
  const a = n.exception = n.exception || {},
    c = a.values = a.values || [],
    h = c[0] = c[0] || {},
    o = h.stacktrace = h.stacktrace || {};
    o.frames = o.frames || []
}
```

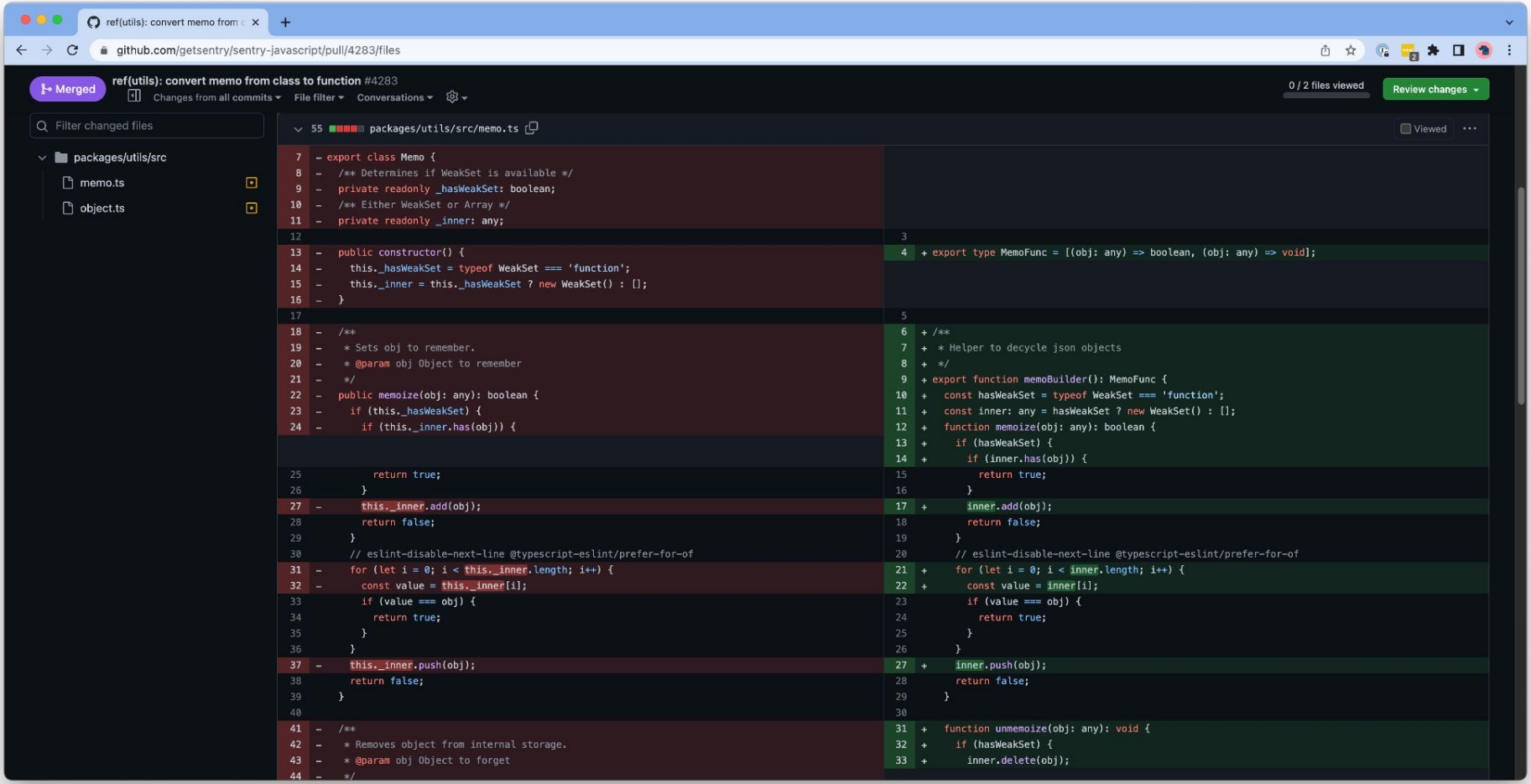
2. Optimizations for minification

- Using try-catch blocks to simplify code requiring nested object access
- Local aliases for object keys to improve minification
- **Converting classes to objects and functions**

Converting classes to objects and functions

```ts
export class API {
  ...
  /** Create a new instance of API */
  public constructor(dsn: DsnLike, metadata: SdkMetadata = {}, tunnel?: string) {
    this.dsn = dsn;
    this._dsnObject = new Dsn(dsn);
    this.metadata = metadata;
    this._tunnel = tunnel;
  }
  ...
  /**
  \* Returns the store endpoint URL with auth in the query string.
  \*
  \* Sending auth as part of the query string and not as custom HTTP headers avoids CORS preflight requests.*/
  public getStoreEndpointWithUrlEncodedAuth(): string {
    return `${this.getStoreEndpoint()}?${this._encodedAuth()}`;
  }
  ...
  /** Returns a URL-encoded string with auth config suitable for a query string. */
  private _encodedAuth(): string {
    const dsn = this.getDsn();
    const auth = {
      // We send only the minimum set of required information. See
      // <https://github.com/getsentry/sentry-javascript/issues/2572>.
      sentry_key: dsn.publicKey,
      sentry_version: SENTRY_API_VERSION,
    };
    return urlEncode(auth);
  }
}
```

```
export class API {
  constructor(t, e = {}, n) {
    this.dsn = t, this.t = new Dsn(t), this.metadata = e, this.i = n
  }
  ...
  getStoreEndpointWithUrlEncodedAuth() {
    return `${this.getStoreEndpoint()}?${this.h()}`
  }
  ...
  h() {
    const t = {
      sentry_key: this.getDsn().publicKey,
      sentry_version: SENTRY_API_VERSION
    };
    return c(t)
  }
}
```


ref(utils): convert memo from c
github.com/getsentry/sentry-javascript/pull/4283/files
Merged
ref(utils): convert memo from class to function #4283
Changes from all commits
File filter
Conversations
0 / 2 files viewed
Review changes
Filter changed files
packages/utils/src
memo.ts
object.ts
55 packages/utils/src/memo.ts
Viewed
- export class Memo {
- /** Determines if WeakSet is available */
- private readonly _hasWeakSet: boolean;
- /** Either WeakSet or Array */
- private readonly _inner: any;
- public constructor() {
- this._hasWeakSet = typeof WeakSet === 'function';
- this._inner = this._hasWeakSet ? new WeakSet() : [];
- }
- /**
- * Sets obj to remember.
- * @param obj Object to remember
- */
- public memoize(obj: any): boolean {
- if (this._hasWeakSet) {
- if (this._inner.has(obj)) {
return true;
}
- this._inner.add(obj);
return false;
}
// eslint-disable-next-line @typescript-eslint/prefer-for-of
- for (let i = 0; i < this._inner.length; i++) {
- const value = this._inner[i];
if (value === obj) {
return true;
}
}
- this._inner.push(obj);
return false;
}
- /**
- * Removes object from internal storage.
- * @param obj Object to forget
- */
+ export type MemoFunc = [(obj: any) => boolean, (obj: any) => void];
+ /**
+ * Helper to decycle json objects
+ */
+ export function memoBuilder(): MemoFunc {
+ const hasWeakSet = typeof WeakSet === 'function';
+ const inner: any = hasWeakSet ? new WeakSet() : [];
+ function memoize(obj: any): boolean {
+ if (hasWeakSet) {
+ if (inner.has(obj)) {
return true;
}
+ inner.add(obj);
return false;
}
// eslint-disable-next-line @typescript-eslint/prefer-for-of
+ for (let i = 0; i < inner.length; i++) {
+ const value = inner[i];
if (value === obj) {
return true;
}
}
+ inner.push(obj);
return false;
}
+ function unmemoize(obj: any): void {
+ if (hasWeakSet) {
+ inner.delete(obj);

Converting classes to objects and functions

Example 1: convert Memo class to function
Example 2: convert Logger class to functions

Let’s review!

Techniques to optimize your JS

1. Optimizations for **down-compiling & transpiling**
    - Removing usages of **optional chaining**
    - Using **const enums** or **string constants** instead of TypeScript enums

2. Optimizations for **minification**
    - Using **try-catch blocks** to simplify code requiring nested object access
    - Local **aliases for object keys** to improve minification
    - Converting **classes to objects and functions**

Our results

Version	Minified unzipped size
version 6.16.1	74.47kb
version 7.3.1	52.67kb
	29% decrease

Our results

29% bundle size decrease out of the box

Plus tree-shaking related improvements, like…
Next.js **30kb** reduction in run-time JS

Finally closed: Package size is massive #2707

“We have been very impressed with the new Sentry JS SDK. Not only is the bundle size significantly smaller out of the box, but we were able to reduce it further through tree shaking.”

Shu Ding
Software Engineer, Vercel

Thank you 💖

Read more in our blog post:

bit.ly/generated-javascript-blog

GitHub discussion for the changes:

bit.ly/js-v7-gh-discussion

Sentry JavaScript docs:

bit.ly/sentry-js-docs

Check out our Sandbox:

bit.ly/sentry-sandbox

Title

Author

Title

1 **Topic 1**
Optional subheading

2 **Topic 2**
Optional subheading

3 **Topic 3**
Optional subheading

Section Header

Alternate Section Header

Title Goes Here

Text Column 1

- Bullet Point 1
- Bullet Point 2
- Bullet Point 3

Text Column 2

- Bullet Point 1
- Bullet Point 2
- Bullet Point 3

Text Column 2

1. Bullet Point 1
2. Bullet Point 2
3. Bullet Point 3

Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna nec. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Numbers list

1. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna nec.

2. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna nec.

3. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna nec.

4. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna nec.

Three column numbers

Consectetur nec labore

45K

Lorem ipsum dolor sit amet ut labore et dolore magna

Adipiscing elit sed

690K

Tempor incididunt ut labore et dolore magna aliqua

Sed eiusmod

100K

Do eiusmod tempor incididunt ut labore et dolore magna

Three column icons

Caption

Caption

Caption

Title and Image

Title Text and Image

Body text

85%

Caption here

We get a piece of mind from Sentry. In the case of a real issue, we immediately know what to fix or roll back.

Valentino Volonghi
CTO, Adroll

Looking for Images to use?

Try the DAM (Digital Asset Manager)

https://www.notion.so/sentry/4b467386cb9b4ee8b19b7bf65bc175ae?v=36e9101612c446dab8d3b372d0832e42

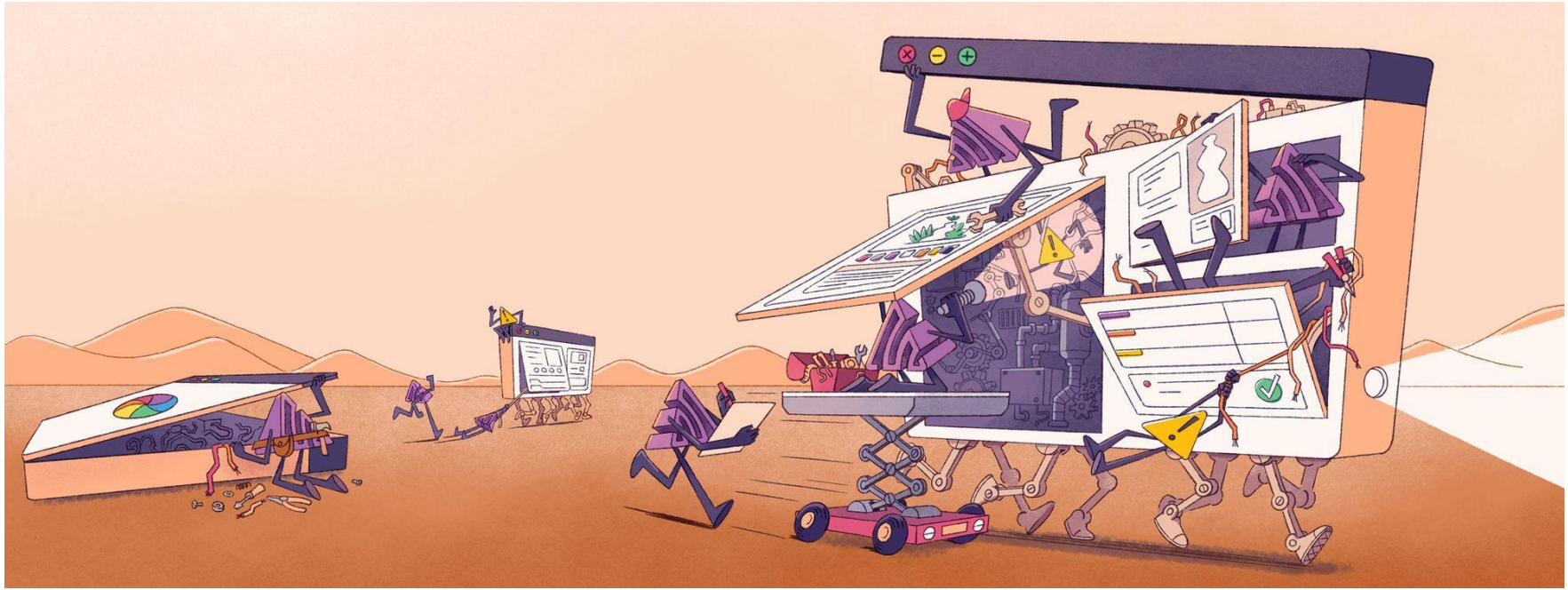
Understanding the Performance Impact of Generated JavaScript

DeveloperWeek 2023

What is Mobile Application Monitoring?

- The process of tracking the performance and health of an application **in production**.
- Tracking the performance of individual components or services.
- Monitoring for errors and other issues.
- **Goal:** identify and resolve problems with the application in a timely manner.
- Improve the UX and prevent issues in the app.

Key features of App Monitoring platforms



Error tracking

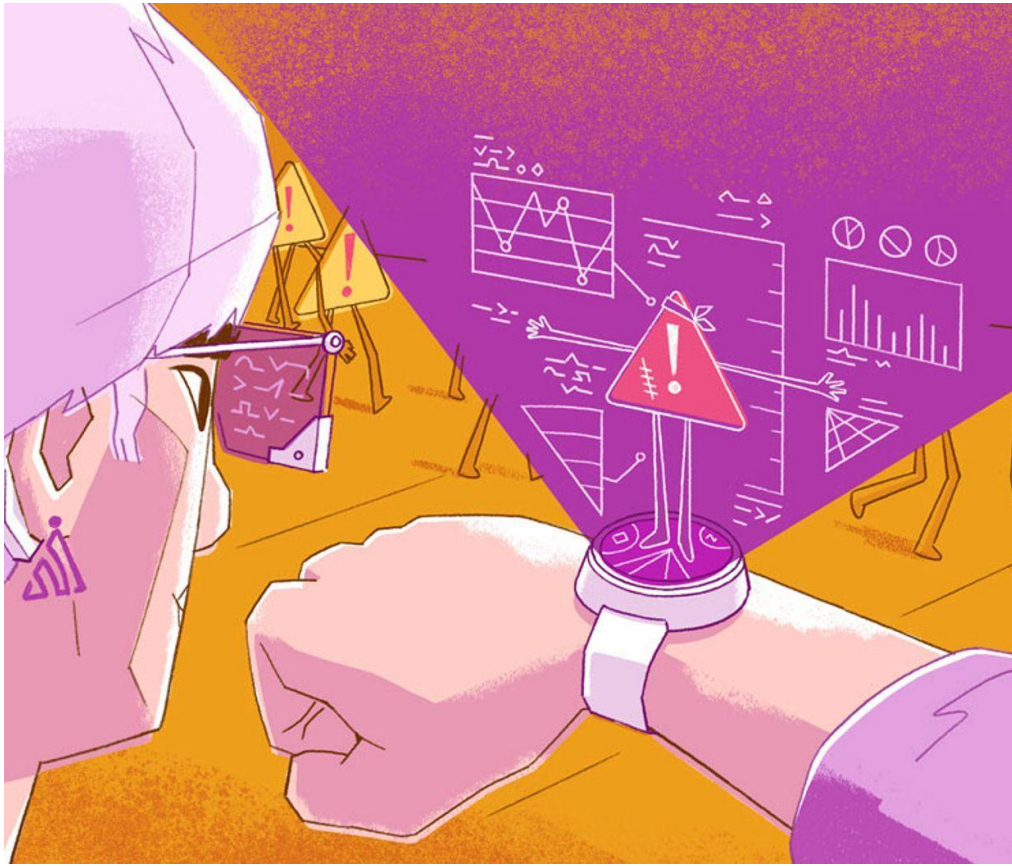
- **Monitoring for errors** within the application on every phone that's installed on in this world.
- **Providing detailed** information about these errors, including their cause and all sorts of hardware and software details.

Performance Monitoring

- Tracking the performance of **individual components** or **services** within the app.
- **Identify any bottlenecks** or other issues that may be impacting your app's overall performance.

Alerting

- **Send alerts** when certain conditions are met.
- Performance drops below a certain threshold, or when an error occurs.
- Email, Slack, etc…

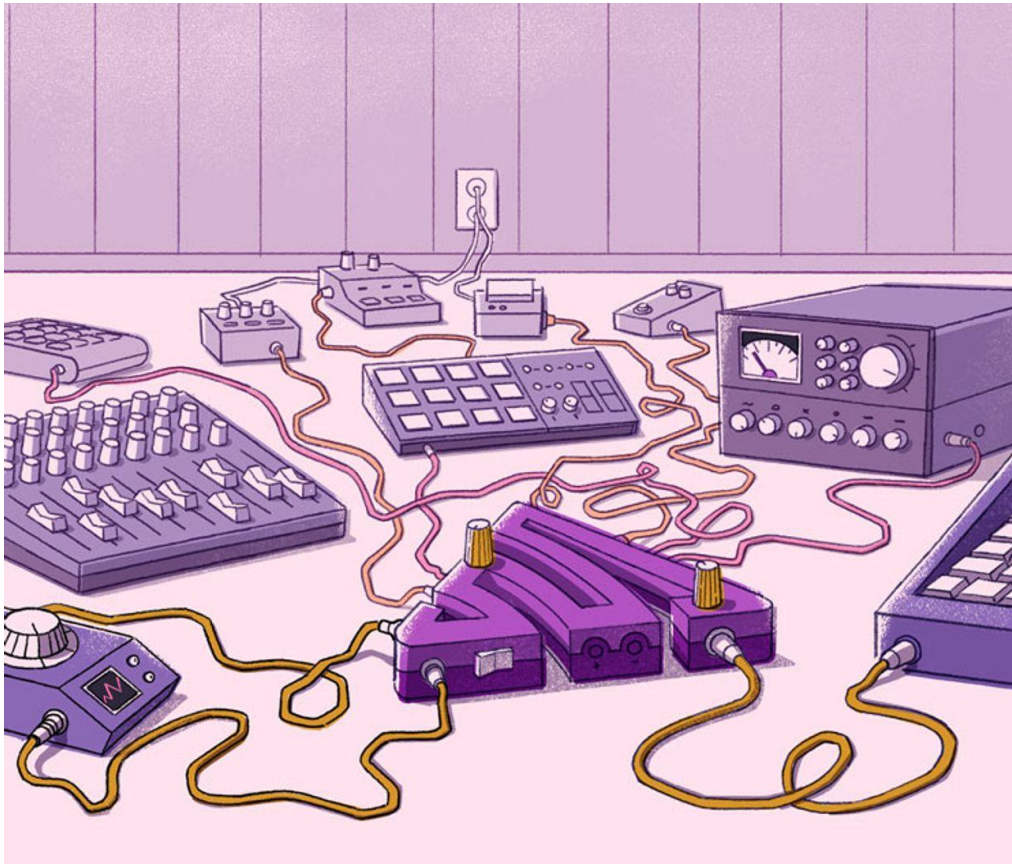


Dashboards & reporting

- Allows you to see the current status of your application.
- Historical data and trends.

Integrations

- **Incident Management Systems** (PagerDuty, TaskCall)
- **Source Control Platforms** (GitHub, GitLab, BitBucket)
- **Project Management Platforms** (JIRA, Asana, Linear)
- **Communication Tools** (Slack, Microsoft Teams)



Demo time!