readme.md for @push.rocks/smartdiff

A powerful, cross-platform text diffing library for TypeScript/JavaScript with built-in visualization for CLI and browser. □□

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

Install

```
npm install @push.rocks/smartdiff
# or
pnpm add @push.rocks/smartdiff
```

Features □

- □□ **Character-level diffs** — See exactly which characters changed
- □□ **Word-level diffs** — Perfect for document comparisons
- □□ **Line-level diffs** — Ideal for code and file comparisons
- □□ **CLI output** — ANSI-colored terminal output
- □□ **HTML output** — Styled spans for browser rendering
- □□ **Compact storage format** — Efficient diff serialization for storage/sync
- □□ **Git-style unified diffs** — Standard patch format
- □□ **Built-in CSS** — Ready-to-use styles for HTML output

Usage

Quick Start

```
import {
  createDiff,
  applyPatch,
  formatLineDiffForConsole,
  formatWordDiffAsHtml,
} from '@push.rocks/smartdiff';

// Create and apply compact diffs
const patch = createDiff('hello world', 'hello beautiful world');
const result = applyPatch('hello world', patch);
// result: 'hello beautiful world'

// Visualize changes in terminal
console.log(formatLineDiffForConsole(oldCode, newCode));

// Generate HTML for browser display
const html = formatWordDiffAsHtml(oldText, newText);
```

□□ Compact Diff Storage

For storing diffs efficiently (version control, sync, undo/redo):

`createDiff(original, revision)`

Creates a compact JSON-encoded diff optimized for storage.

```
import { createDiff } from '@push.rocks/smartdiff';

const diff = createDiff('Hello World', 'Hello smart World');
console.log(diff);
// Output: [[0,6],[1,"smart "],[0,5]]
```

Format:

- `[0, n]` — Keep `n` characters from original
- `[1, "text"]` — Insert `"text"`
- `[-1, n]` — Delete `n` characters

`applyPatch(original, diffString)`

Reconstructs the revised text from original + diff.

```
import { createDiff, applyPatch } from '@push.rocks/smartdiff';

const original = 'hi there';
const revised = 'hi Polly, there is a Sandwich.';

const patch = createDiff(original, revised);
const reconstructed = applyPatch(original, patch);
// reconstructed === revised ✓
```

□□ Character-Level Diff

For precise character-by-character comparison:

`getCharDiff(original, revision)`

Returns an array of diff segments.

```
import { getCharDiff } from '@push.rocks/smartdiff';

const segments = getCharDiff('hello', 'hallo');
// [
//   { type: 'equal', value: 'h' },
//   { type: 'delete', value: 'e' },
//   { type: 'insert', value: 'a' },
//   { type: 'equal', value: 'llo' }
// ]
```

`formatCharDiffForConsole(original, revision)`

Returns ANSI-colored string for terminal output.

```
import { formatCharDiffForConsole } from '@push.rocks/smartdiff';

console.log(formatCharDiffForConsole('hello world', 'hello beautiful world'));
// Displays: hello [green background]beautiful [/green]world
```

`formatCharDiffAsHtml(original, revision)`

Returns HTML with styled spans.

```
import { formatCharDiffAsHtml } from '@push.rocks/smartdiff';

const html = formatCharDiffAsHtml('hello', 'hallo');
// <span class="smartdiff-equal">h</span>
// <span class="smartdiff-delete">e</span>
// <span class="smartdiff-insert">a</span>
// <span class="smartdiff-equal">llo</span>
```

□□ Word-Level Diff

For document and prose comparison:

`getWordDiff(original, revision)`

```
import { getWordDiff } from '@push.rocks/smartdiff';

const segments = getWordDiff('The quick brown fox', 'The slow brown dog');
// [
//   { type: 'equal', value: 'The ' },
//   { type: 'delete', value: 'quick' },
//   { type: 'insert', value: 'slow' },
```

```
//   { type: 'equal', value: ' brown ' },
//   { type: 'delete', value: 'fox' },
//   { type: 'insert', value: 'dog' }
// ]
```

`formatWordDiffForConsole(original, revision)`

```
import { formatWordDiffForConsole } from '@push.rocks/smartdiff';

console.log(formatWordDiffForConsole('Hello World', 'Hello Beautiful World'));
// Displays: Hello [green]Beautiful [/green]World
```

`formatWordDiffAsHtml(original, revision)`

```
import { formatWordDiffAsHtml } from '@push.rocks/smartdiff';

const html = formatWordDiffAsHtml('one two three', 'one TWO three');
// <span class="smartdiff-equal">one </span>
// <span class="smartdiff-delete">two</span>
// <span class="smartdiff-insert">TWO</span>
// <span class="smartdiff-equal"> three</span>
```

□□ Line-Level Diff

For code and file comparison:

`getLineDiff(original, revision)`

```
import { getLineDiff } from '@push.rocks/smartdiff';

const lineDiff = getLineDiff('line1\nline2\nline3', 'line1\nmodified\nline3');
// [
//   { lineNumber: 1, type: 'equal', value: 'line1' },
```

```
//   { lineNumber: 2, type: 'delete', value: 'line2' },
//   { lineNumber: -1, type: 'insert', value: 'modified' },
//   { lineNumber: 3, type: 'equal', value: 'line3' }
// ]
```

`formatLineDiffForConsole(original, revision)`

Produces unified diff-style output with colors:

```
import { formatLineDiffForConsole } from '@push.rocks/smartdiff';

const oldCode = `function hello() {
  console.log("hi");
}`;

const newCode = `function hello() {
  console.log("hello world");
}`;

console.log(formatLineDiffForConsole(oldCode, newCode));
```

Output:

```
  function hello() {
-   console.log("hi");
+   console.log("hello world");
  }
```

`formatLineDiffAsHtml(original, revision)`

```
import { formatLineDiffAsHtml } from '@push.rocks/smartdiff';

const html = formatLineDiffAsHtml('a\nb\nc', 'a\nB\nc');
// <div class="smartdiff-lines">
//   <div class="smartdiff-line smartdiff-equal"><span class="smartdiff-prefix">
</span>a</div>
//   <div class="smartdiff-line smartdiff-delete"><span class="smartdiff-prefix">-
```

```
</span>b</div>
//   <div class="smartdiff-line smartdiff-insert"><span class="smartdiff-
prefix">+</span>B</div>
//   <div class="smartdiff-line smartdiff-equal"><span class="smartdiff-prefix">
</span>c</div>
// </div>
```

□□ Unified Diff (Git-style)

`createUnifiedDiff(original, revision, options?)`

Creates a standard unified diff patch.

```
import { createUnifiedDiff } from '@push.rocks/smartdiff';

const patch = createUnifiedDiff(oldContent, newContent, {
  originalFileName: 'file.txt',
  revisedFileName: 'file.txt',
  context: 3, // lines of context
});

console.log(patch);
```

Output:

```
--- file.txt
+++ file.txt
@@ -1,5 +1,5 @@
 line1
-line2
+modified
 line3
```

`formatUnifiedDiffForConsole(original, revision, options?)`

Same as above, but with ANSI colors for terminal display.

□□ CSS Styles for HTML Output

`getDefaultDiffCss()`

Returns ready-to-use CSS for styling HTML diff output.

```
import { getDefaultDiffCss } from '@push.rocks/smartdiff';

const css = getDefaultDiffCss();
// Inject into your page:
// <style>${css}</style>
```

Included styles:

- `.smartdiff-delete` — Red background, strikethrough
- `.smartdiff-insert` — Green background
- `.smartdiff-equal` — Unchanged text
- `.smartdiff-lines` — Container for line diffs
- `.smartdiff-line` — Individual line styling
- `.smartdiff-prefix` — The +/- prefix styling

□□ Real-World Use Cases

Collaborative Editing

Track document versions without storing full copies:

```
import { createDiff, applyPatch } from '@push.rocks/smartdiff';
```

```
const versions = ['Initial draft'];
const patches: string[] = [];

function saveVersion(newText: string) {
  const lastVersion = versions[versions.length - 1];
  patches.push(createDiff(lastVersion, newText));
  versions.push(newText);
}

function getVersion(index: number): string {
  let text = versions[0];
  for (let i = 0; i < index; i++) {
    text = applyPatch(text, patches[i]);
  }
  return text;
}
```

Network Synchronization

Send compact diffs instead of full content:

```
import { createDiff, applyPatch } from '@push.rocks/smartdiff';

// Client
function sendUpdate(localText: string, serverText: string) {
  const diff = createDiff(serverText, localText);
  socket.emit('update', diff); // Much smaller than full text!
}

// Server
socket.on('update', (diff: string) => {
  document = applyPatch(document, diff);
});
```

Undo/Redo System

```
import { createDiff, applyPatch } from '@push.rocks/smartdiff';
```

```
interface HistoryEntry {
  forward: string;
  backward: string;
}

const history: HistoryEntry[] = [];
let currentText = 'Start';

function edit(newText: string) {
  history.push({
    forward: createDiff(currentText, newText),
    backward: createDiff(newText, currentText),
  });
  currentText = newText;
}

function undo() {
  const entry = history.pop();
  if (entry) {
    currentText = applyPatch(currentText, entry.backward);
  }
}
```

Code Review UI

```
import { formatLineDiffAsHtml, getDefaultDiffCss } from '@push.rocks/smartdiff';

function renderCodeReview(oldCode: string, newCode: string) {
  return `
    <style>${getDefaultDiffCss()}</style>
    ${formatLineDiffAsHtml(oldCode, newCode)}
  `;
}
```

□□ TypeScript Types

```
interface IDiffSegment {
  type: 'equal' | 'insert' | 'delete';
  value: string;
}

interface ILineDiff {
  lineNumber: number;
  type: 'equal' | 'insert' | 'delete';
  value: string;
}
```

API Reference

Function	Description
`createDiff(original, revision)`	Create compact diff for storage
`applyPatch(original, diffString)`	Reconstruct text from diff
`getCharDiff(original, revision)`	Character-level diff segments
`getWordDiff(original, revision)`	Word-level diff segments
`getLineDiff(original, revision)`	Line-level diff segments
`formatCharDiffForConsole(...)`	ANSI-colored char diff
`formatWordDiffForConsole(...)`	ANSI-colored word diff
`formatLineDiffForConsole(...)`	ANSI-colored line diff
`formatCharDiffAsHtml(...)`	HTML char diff
`formatWordDiffAsHtml(...)`	HTML word diff
`formatLineDiffAsHtml(...)`	HTML line diff
`createUnifiedDiff(...)`	Git-style unified diff
`formatUnifiedDiffForConsole(...)`	Colored unified diff
`getDefaultDiffCss()`	CSS styles for HTML output

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the LICENSE file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #3
Created 2026-03-28 11:10:34 UTC by foss.global Team
Updated 2026-03-28 12:17:21 UTC by foss.global Team