

# The Introduction of CodeSlide

## README

### # CodeSlide

[](<https://www.npmjs.com/package/codeslide-cli?activeTab=readme>)

### ## Features

- CodeSlide makes a slideshow for code snippets
- Its applications:
  - [CodeSlide CLI](./applications/cli/)

### ## Dependencies

- It uses [esbuild](<https://github.com/evanw/esbuild>) as module bundler
- It uses [Commander.js](<https://github.com/tj/commander.js>) as CLI framework
- It uses [Eta](<https://github.com/eta-dev/eta>) as HTML template engine
- It uses [Highlight.js](<https://github.com/highlightjs/highlight.js>) as syntax highlighter
- It uses [Node Fetch](<https://github.com/node-fetch/node-fetch>) as resource fetcher
- It uses [Puppeteer](<https://github.com/puppeteer/puppeteer>) as PDF printer
- It uses [Zod](<https://github.com/colinhacks/zod>) as JSON schema validator

### ## Documents

- See [**\*\*Reference\*\***](./docs/REFERENCE.md) for more information

### ## Creator

- [AsherJingkongChen](<https://github.com/AsherJingkongChen>)

# The essentials

## Render the HTML template and CSS to a slideshow

```
import HorizontalStylesheet from './slides.horizontal.css';
import VerticalStylesheet from './slides.vertical.css';
import Template from './slides.html';

const Stylesheet = {
  horizontal: HorizontalStylesheet,
  vertical: VerticalStylesheet,
};

export { Stylesheet, Template };

import { z } from 'zod';
import { isFormat } from './format';
import { isLang } from './lang';
import { isLayout } from './layout';
import { isPagesize } from './pagesize';

export type Printer = z.infer<typeof Printer>;

export const Printer = z.object(
  {
    fontFamily: z
      .string()
      .default('')
      .transform((arg) => `
${arg ? `${arg}, ` : ''}ui-monospace, SFMono-Regular, \
SF Mono, Menlo, Consolas, Liberation Mono, monospace`
      ),
    fontSize: z
      .string()
      .default('large'),
    fontWeight: z
      .string()
```

```

        .default('normal'),
format: z
    .string()
    .refine(isFormat)
    .default('html'),
layout: z
    .string()
    .refine(isLayout)
    .default('horizontal'),
pagesize: z
    .string()
    .refine(isPagesize)
    .default('a4'),
slides: z
    .array(
        z.object({
            code: z
                .string()
                .default(''),
            lang: z
                .string()
                .refine(isLang)
                .optional(),
            title: z
                .string()
                .default(''),
        })
        .strict()
    )
    .default([]),
styles: z
    .array(z.string())
    .default([])
    .transform((arg) => [
        'https://cdnjs.cloudflare.com/ajax/libs/highlight.js/'
+
        '11.8.0/styles/github-dark-dimmed.min.css',
        ...arg,
    ]),
    })
    .transform((arg) => {

```

```
    if (
      arg.layout === 'horizontal' &&
      arg.format === 'pdf'
    ) {
      arg.layout = 'vertical';
    }
    return arg;
  });
```

```
import { render as renderEta } from 'eta';
import { Stylesheet, Template } from './slides';
import { Printer } from './printer';
```

```
export const render = (printer: Printer): string => renderEta(
  Template,
  {
    layout: printer.layout,
    slides: printer.slides,
    style: `
<style>
${
  [
    Stylesheet[printer.layout],
    ...printer.styles,
    `code { font-family: ${printer.fontFamily}; }`,
    `#slides { font-size: ${printer.fontSize}; }`,
    `#slides { font-weight: ${printer.fontWeight}; }`,
  ].join('\n')
}
</style>`,
  },
  {
    autoTrim: false,
    tags: ['{%', '%}'],
  }
);
```

```
export * from './format';
export * from './lang';
export * from './layout';
```

```
export * from './pagesize';
export * from './printer';
```

## The HTML template

```
<!DOCTYPE HTML>
<html class="hljs">
<head>
  <meta
    name="viewport"
    charset="utf-8"
    content="width=device-width, initial-scale=1, user-
scalable=no">
    {%~ it.style %}
</head>
<body class="hljs">
  <div id="slides">
{%_ for (const [index, slide] of it.slides.entries()) { %}
  <div class="slide" id="_{%~ index %}" hidden>
{%_ if (slide.title) { %}
    <div class="title
{%_ if (index !== 0 && it.layout === 'vertical') { %}
      {%_ ~ ' bordered' %}
{%_ } _%}
    ">
      <pre>{%_ _%}
        <code class="language-plaintext hljs">
          {%_ = slide.title _%}
        </code>{%_ _%}
      </pre>
    </div>
{%_ } %}
{%_ if (slide.code) { %}
  <div class="code">
    <pre>{%_ _%}
      <code class="
{%_ if (slide.lang) { %}
        {%_ ~ `language-${slide.lang}` %}
{%_ } _%}
```

```

">
    {%_ = slide.code _%}
  </code><br>{%_ _%}
</pre>
</div>
{%_ } %}
</div>
{%_ } %}
</div>
<script type="module">
document.addEventListener('DOMContentLoaded', () => {
  hljs.highlightAll();
  for (const slide of
document.getElementsByClassName('slide')) {
    slide.hidden = false;
  }
}, { once: true });

```

```

hljs.registerLanguage('armasm', armasm);
hljs.registerLanguage('c', c);
hljs.registerLanguage('clojure', clojure);
hljs.registerLanguage('cmake', cmake);
hljs.registerLanguage('coffeescript', coffeescript);
hljs.registerLanguage('cpp', cpp);
hljs.registerLanguage('csharp', csharp);
hljs.registerLanguage('css', css);
hljs.registerLanguage('dart', dart);
hljs.registerLanguage('diff', diff);
hljs.registerLanguage('elixir', elixir);
hljs.registerLanguage('erlang', erlang);
hljs.registerLanguage('go', go);
hljs.registerLanguage('graphql', graphql);
hljs.registerLanguage('groovy', groovy);
hljs.registerLanguage('haskell', haskell);
hljs.registerLanguage('ini', ini);
hljs.registerLanguage('java', java);
hljs.registerLanguage('javascript', javascript);
hljs.registerLanguage('json', json);
hljs.registerLanguage('julia', julia);
hljs.registerLanguage('kotlin', kotlin);
hljs.registerLanguage('less', less);

```

```
hljs.registerLanguage('lisp', lisp);
hljs.registerLanguage('lua', lua);
hljs.registerLanguage('makefile', makefile);
hljs.registerLanguage('markdown', markdown);
hljs.registerLanguage('objectivec', objectivec);
hljs.registerLanguage('perl', perl);
hljs.registerLanguage('php', php);
hljs.registerLanguage('plaintext', plaintext);
hljs.registerLanguage('python', python);
hljs.registerLanguage('r', r);
hljs.registerLanguage('ruby', ruby);
hljs.registerLanguage('rust', rust);
hljs.registerLanguage('scala', scala);
hljs.registerLanguage('scss', scss);
hljs.registerLanguage('shell', shell);
hljs.registerLanguage('sql', sql);
hljs.registerLanguage('swift', swift);
hljs.registerLanguage('typescript', typescript);
hljs.registerLanguage('vbnet', vbnet);
hljs.registerLanguage('xml', xml);
hljs.registerLanguage('yaml', yaml);
```

```
/* Import dependencies from CDN */
```

```
import hljs from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/core.min.js';
import armasm from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/armasm.min.js';
import c from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/c.min.js';
import clojure from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/clojure.min.js';
import cmake from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/cmake.min.js';
import coffeescript from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/coffeescript.min.js';
import cpp from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/cpp.min.js';
```

```
import csharp from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/csharp.min.js';
import css from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/css.min.js';
import dart from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/dart.min.js';
import diff from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/diff.min.js';
import elixir from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/elixir.min.js';
import erlang from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/erlang.min.js';
import go from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/go.min.js';
import graphql from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/graphql.min.js';
import groovy from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/groovy.min.js';
import haskell from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/haskell.min.js';
import ini from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/ini.min.js';
import java from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/java.min.js';
import javascript from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/javascript.min.js';
import json from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/json.min.js';
import julia from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/julia.min.js';
import kotlin from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/kotlin.min.js';
```



```
import less from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/less.min.js';
import lisp from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/lisp.min.js';
import lua from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/lua.min.js';
import makefile from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/makefile.min.js';
import markdown from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/markdown.min.js';
import objectivec from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/objectivec.min.js';
import perl from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/perl.min.js';
import php from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/php.min.js';
import plaintext from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/plaintext.min.js';
import python from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/python.min.js';
import r from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/r.min.js';
import ruby from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/ruby.min.js';
import rust from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/rust.min.js';
import scala from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/scala.min.js';
import scss from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/scss.min.js';
import shell from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/shell.min.js';
import sql from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-release@11.8.0/build/es/languages/sql.min.js';
```

```
import swift from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/swift.min.js';
import typescript from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/typescript.min.js';
import vbnet from
'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/vbnet.min.js';
import xml from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/xml.min.js';
import yaml from 'https://cdn.jsdelivr.net/gh/highlightjs/cdn-
release@11.8.0/build/es/languages/yaml.min.js';
</script>
</body>
</html>
```

## The CSS

```
/*! CodeSlide slides.horizontal.css */
body {
  margin: 0;
  -webkit-print-color-adjust: exact;
  print-color-adjust: exact;
  overflow: hidden;
  overscroll-behavior: none;
  scrollbar-width: none;
}
body::-webkit-scrollbar {
  display: none;
}
pre {
  margin: 0;
  white-space: pre-wrap;
  word-break: break-word;
}
#slides {
  display: flex;
  flex-direction: row;
```

```
position: absolute; /* fix height on mobile */
width: 100vw;
height: 100vh;
overflow-x: scroll;
scroll-behavior: smooth;
scroll-snap-type: x mandatory;
}
.slide {
  display: flex;
  flex-direction: column;
  min-width: 100vw;
  height: 100vh;
  overflow-y: scroll;
  scroll-snap-align: start;
  scroll-snap-stop: always;
  scrollbar-width: none;
}
.slide::-webkit-scrollbar {
  display: none;
}
.slide > .title {
  font-size: larger;
  font-weight: bolder;
}
@page {
  margin: 0;
  size: auto;
}
@media print {
  #slides {
    width: auto;
    height: auto;
  }
}
```

---

**Let's see some applications!**

# CodeSlide CLI: A Node.js Command Line Interface

```
import { program } from 'commander';
import { version, homepage, name } from '../package.json';
import { run } from './run';
```

```
program
  .name(name)
  .description(`\n
Example: ${name} -o ./output.html
```

Make a slideshow (HTML/PDF file) for code snippets  
with CLI options

Go to home page for more information: \${homepage}

```
` )
  .version(version, '-v, --version',
    'Check the version number.'
  )
  .helpOption('-h, --help',
    'Check all options and their description.'
  )
  .option('-o, --output [local_path]',
    'The "output file path" of slideshow. ' +
    'If not set, it writes the output to stdout.'
  )
  .option('--font-family [string]',
    'The font family of "displayed texts". ' +
    'Default is "ui-monospace, SFMono-Regular, ' +
    'SF Mono, Menlo, Consolas, Liberation Mono, monospace".'
  )
  .option('--font-size [string]',
    'The font size of "displayed texts". ' +
    'Default is "large".'
  )
  .option('--font-weight [string]',
    'The font weight of "displayed texts". ' +
    'Default is "normal".'
  )
  )
```

```

.option('--format [html | pdf]',
  'The "output file format" of slideshow. ' +
  'Default is "html".'
)
.option('--layout [horizontal | vertical]',
  'The "layout" of slideshow. ' +
  'Default is "horizontal".'
)
.option('--pagesize [letter | legal | tabloid | ledger | a0
| a1 | a2 | a3 | a4 | a5 | a6]',
  'The page size of slideshow "in PDF format". ' +
  'Default is "a4".'
)
.option('--slides [slide...]',
  'The "contents" to show. ' +
  'An array of slides, each slide is a pair of title and
path (URL). ' +
  'Example: --slides "Intro" "./README.md" "Program"
"./index.js"; ' +
  'There are 2 slides where the first is titled as "Intro" '
+
  'and shows the content from "./README.md".'
)
.option('--styles [path...]',
  'The "display styles" of slideshow. ' +
  'An array of paths (URLs) of CSS files. ' +
  'You may need this if: ' +
  '1. To load custom font family ' +
  '2. To load custom syntax highlighting theme ' +
  '3. To change the background '
)
.action(run)
.parseAsync();

```

## CLI options validator

```

import { z } from 'zod';

export type CLIOptions = z.infer<typeof CLIOptions>;

```

```

export const CLIOptions = z.object(
  {
    output: z.string().optional(),
    fontFamily: z.string().optional(),
    fontSize: z.string().optional(),
    fontWeight: z.string().optional(),
    format: z.string().optional(),
    layout: z.string().optional(),
    pagesize: z.string().optional(),
    slides: z.array(z.string()).optional(),
    styles: z.array(z.string()).optional(),
  })
  .strict()
  .superRefine((ref, ctx) => {
    if ((ref.slides?.length ?? 0) % 2 !== 0) {
      ctx.addIssue({
        code: z.ZodIssueCode.custom,
        message: 'The option --slides should has even number
of arguments',
      });
    }
  });

```

## Parse CLI options -> Print to output

```

import { Printer } from '../../../src';
import { CLIOptions } from './options';
import { mayfail } from './tool';

export const parse = (
  options: CLIOptions,
): Printer => {
  options = mayfail(() => CLIOptions.parse(options));

  const slides: Printer['slides'] = [];
  options.slides?.forEach((arg, index) => {
    if (index % 2 === 0) {
      slides.push({ title: arg, code: '' });
    }
  });

```

```
    } else {  
      slides[slides.length - 1].code = arg;  
    }  
  });  
});
```

```
return mayfail(() => Printer.parse({  
  ...options,  
  slides,  
}));  
};
```

```
import { PathOrFileDescriptor, writeFileSync } from 'fs';  
import { launch } from 'puppeteer';  
import { render, Printer } from '../../src';  
import { mayfailAsync } from '../tool';
```

```
export const print = (  
  output: PathOrFileDescriptor,  
  printer: Printer,  
) : Promise<void> => mayfailAsync(async () => {  
  if (printer.format === 'html') {  
    writeFileSync(output, render(printer), 'utf8');  
  } else if (printer.format === 'pdf') {  
    const browser = await mayfailAsync(launch());  
    const page = await mayfailAsync(browser.newPage());  
    await mayfailAsync(page.setContent(render(printer)));  
    const result = await mayfailAsync(  
      page.pdf({  
        printBackground: true,  
        format: printer.pagesize, // is it redundant?  
      })  
    );  
    const closeBrowser = mayfailAsync(browser.close());  
    writeFileSync(output, result, 'base64');  
    await closeBrowser;  
  }  
});
```

```
import { stdout } from 'process';
import { guessLangFromURL } from '../../../src';
import { CLIOptions } from './options';
import { parse } from './parse';
import { print } from './print';
import { getContent, parseURL } from './tool';

export const run = async (
  options: CLIOptions,
): Promise<void> => {
  const printer = parse(options);

  printer.slides = await Promise.all(
    printer.slides.map(async (slide) => {
      if (slide.code) {
        const codeURL = parseURL(slide.code);
        return {
          code: await getContent(codeURL),
          lang: guessLangFromURL(codeURL),
          title: slide.title,
        };
      }
      return slide;
    })
  );

  printer.styles = await Promise.all(
    printer.styles.map((path) => getContent(path))
  );

  return print(options.output ?? stdout.fd, printer);

  // // Not paralleled
  // for (const slide of printer.slides) {
  //   if (slide.code) {
  //     const codeURL = parseURL(slide.code);
  //     slide.code = await getContent(codeURL);
  //     slide.lang = guessLangFromURL(codeURL);
  //   }
  // }
```



```
// for (const [index, path] of printer.styles.entries()) {  
//   printer.styles[index] = await getContent(path);  
// }  
};
```

---

**The End**