Markdown to HTML Transpiler

IIC3585 - Grupo 7

Contenido

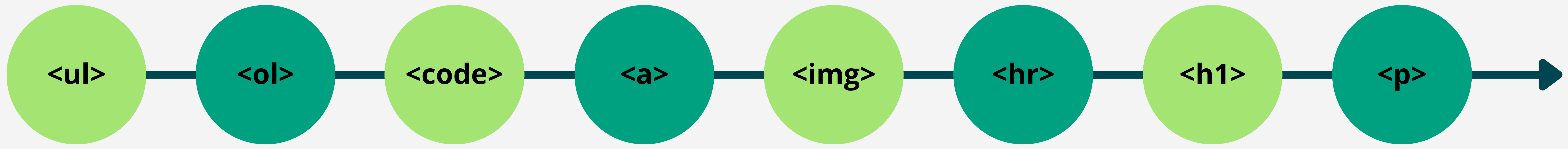
- Estructura Principal: Pipe
- Regex
- Currying
- Funciones destacadas

Pipe

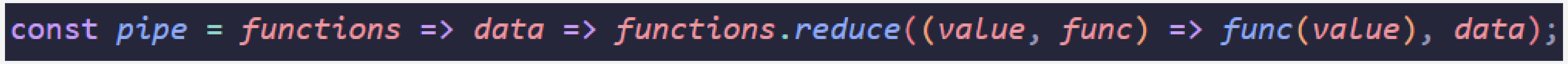
Estructura Principal

Pipe

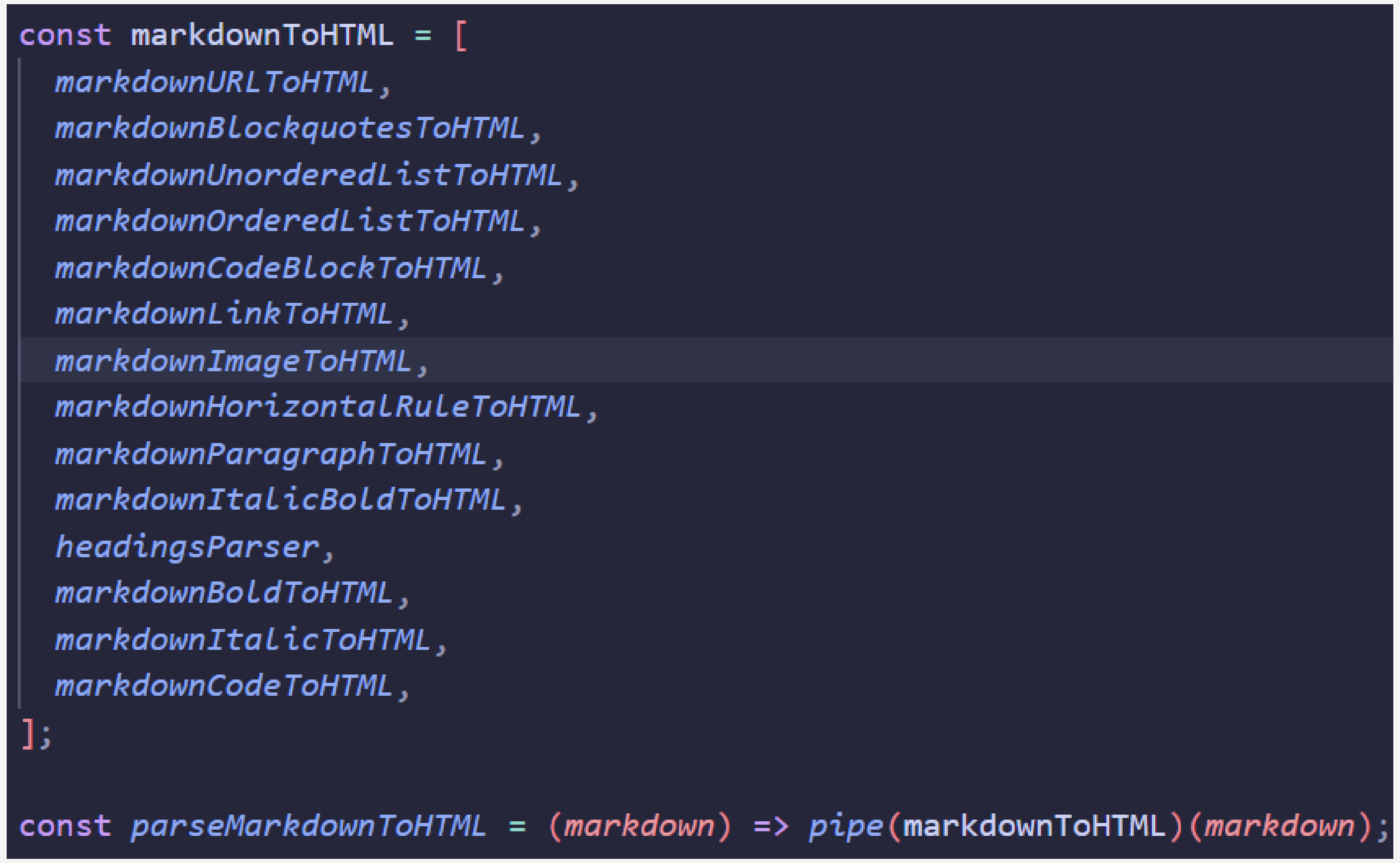
Funciones para cada elemento encadenadas, que cambian el contenido de Markdown a HTML



Pipe

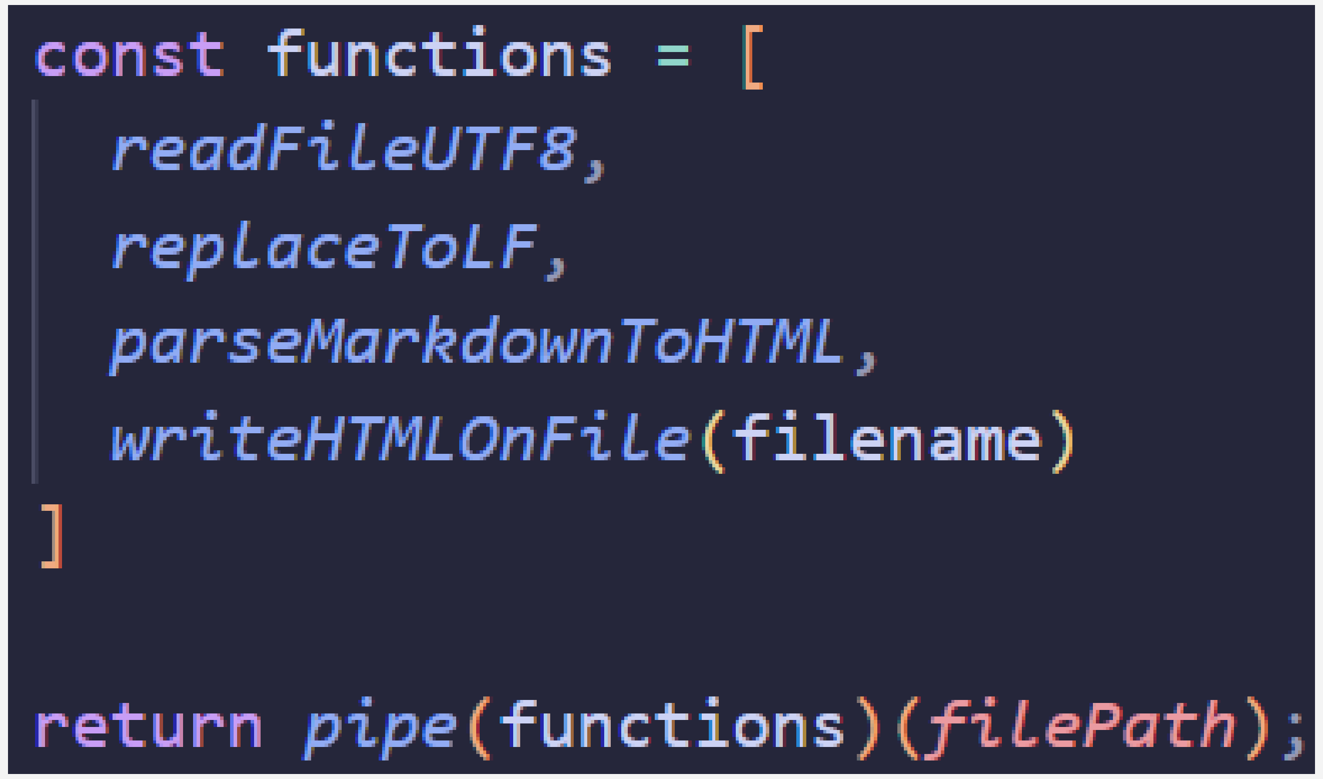


```
const pipe = functions => data => functions.reduce((value, func) => func(value), data);
```



```
const markdownToHTML = [
  markdownURLToHTML,
  markdownBlockquotesToHTML,
  markdownUnorderedListToHTML,
  markdownOrderedListToHTML,
  markdownCodeBlockToHTML,
  markdownLinkToHTML,
  markdownImageToHTML,
  markdownHorizontalRuleToHTML,
  markdownParagraphToHTML,
  markdownItalicBoldToHTML,
  headingsParser,
  markdownBoldToHTML,
  markdownItalicToHTML,
  markdownCodeToHTML,
];

const parseMarkdownToHTML = (markdown) => pipe(markdownToHTML)(markdown);
```

parser.js



```
const functions = [
  readFileUTF8,
  replaceToLF,
  parseMarkdownToHTML,
  writeHTMLOnFile(filename)
]

return pipe(functions)(filePath);
```

transpiler.js

Regex

Reconocimiento de patrones

Syntax

- Delimitar inicio

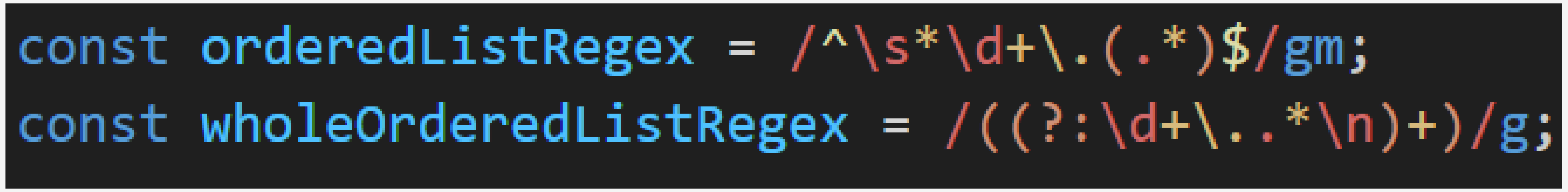
```
/\d+\.
```

- Grupo de captura

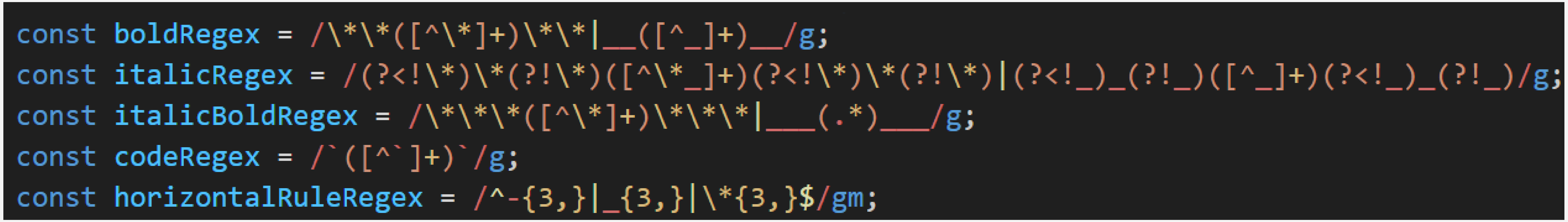
```
(.*)
```

- Delimitar cierre

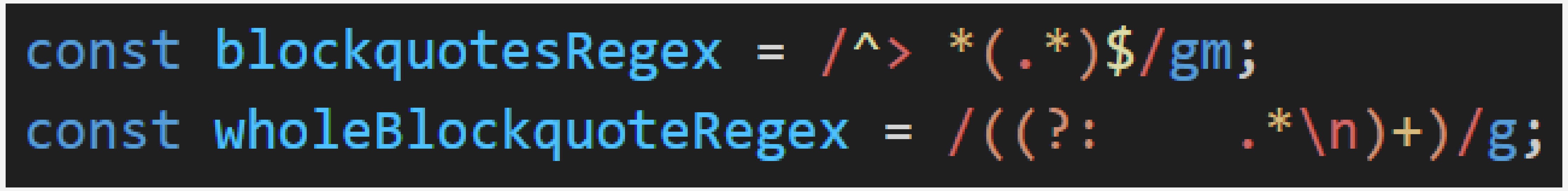
```
_/
```



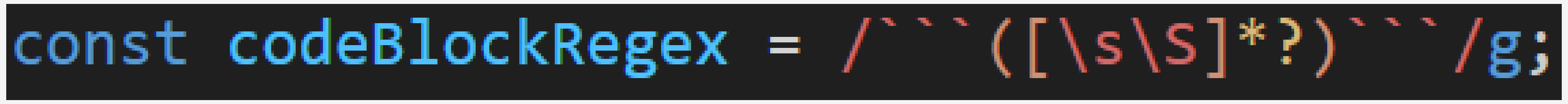
```
const orderedListRegex = /^\s*\d+\.(.*)$/gm;
const wholeOrderedListRegex = /((?:\d+\..*\n)+)/g;
```



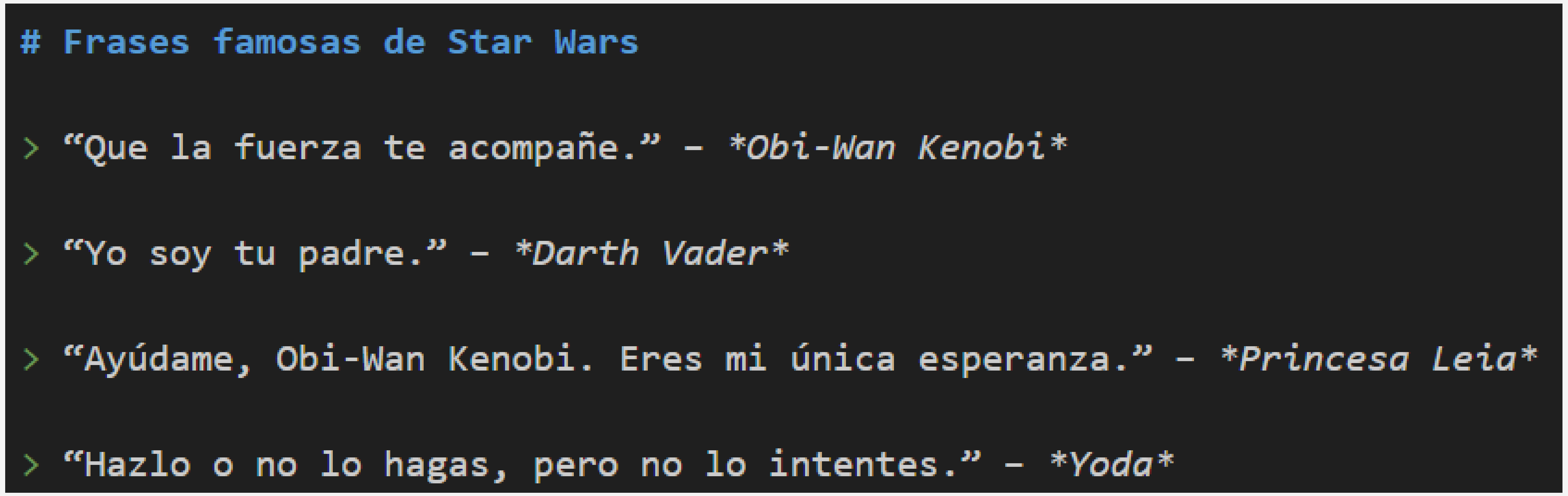
```
const boldRegex = /\*\*([^\*]+)\*\*|__([^_]+)__/g;
const italicRegex = /(?<!\*)\*(?!\*)([^\*_]+)(?<!\*)\*(?!\*)|(?<!_)_(?!_)([^_]+)(?<!_)_(?!_)/g;
const italicBoldRegex = /\*\*\*([^\*]+)\*\*\*|___(.*)___/g;
const codeRegex = /`([^`]+)`/g;
const horizontalRuleRegex = /^-{3,}|_{3,}|\*{3,}$/gm;
```



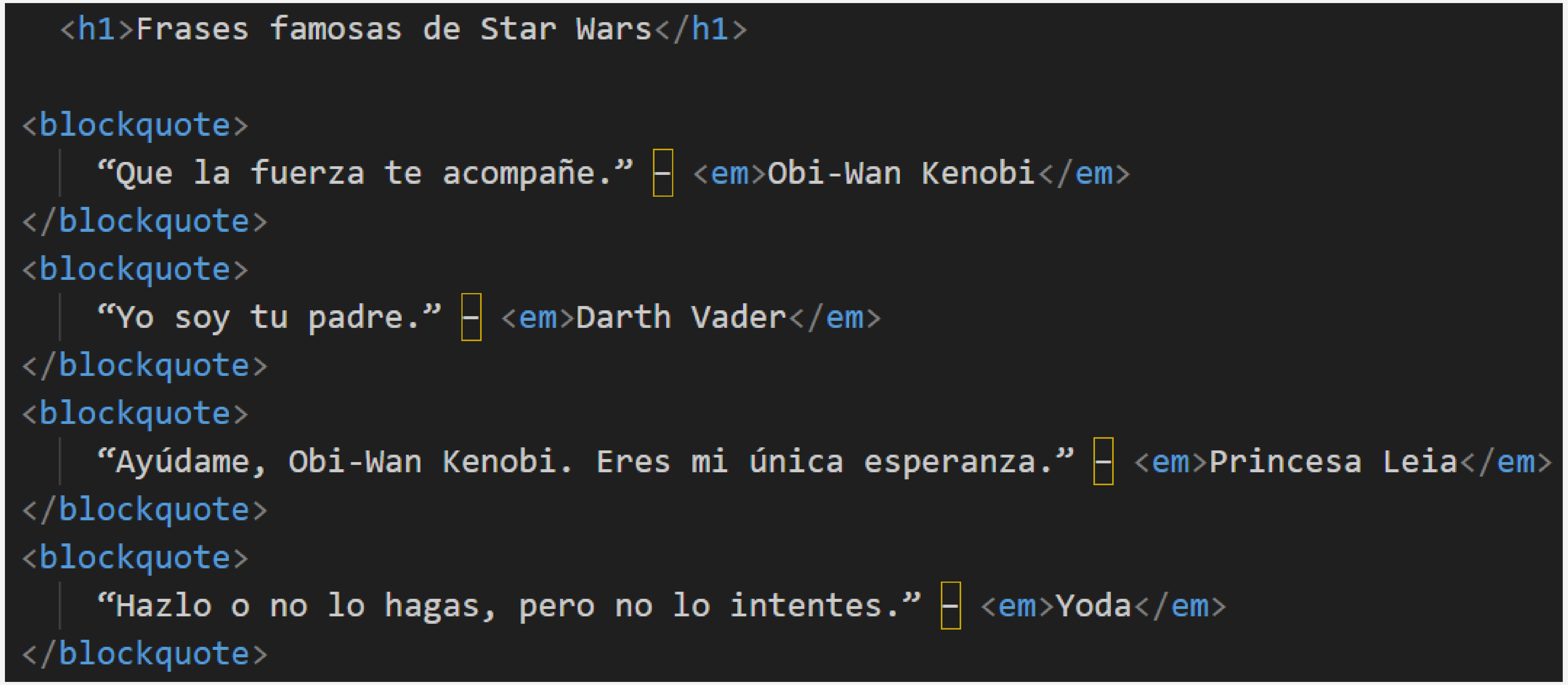
```
const blockquotesRegex = /^> *(.*)$/gm;
const wholeBlockquoteRegex = /((?:     .*\n)+)/g;
```



```
const codeBlockRegex = /```([\s\S]*?)```/g;
```



```
# Frases famosas de Star Wars

> “Que la fuerza te acompañe.” – *Obi-Wan Kenobi*

> “Yo soy tu padre.” – *Darth Vader*

> “Ayúdame, Obi-Wan Kenobi. Eres mi única esperanza.” – *Princesa Leia*

> “Hazlo o no lo hagas, pero no lo intentes.” – *Yoda*
```



```
  <h1>Frases famosas de Star Wars</h1>

<blockquote>
    “Que la fuerza te acompañe.” – <em>Obi-Wan Kenobi</em>
</blockquote>
<blockquote>
    “Yo soy tu padre.” – <em>Darth Vader</em>
</blockquote>
<blockquote>
    “Ayúdame, Obi-Wan Kenobi. Eres mi única esperanza.” – <em>Princesa Leia</em>
</blockquote>
<blockquote>
    “Hazlo o no lo hagas, pero no lo intentes.” – <em>Yoda</em>
</blockquote>
```

Frases famosas de Star Wars

“Que la fuerza te acompañe.” – *Obi-Wan Kenobi*

“Yo soy tu padre.” – *Darth Vader*

“Ayúdame, Obi-Wan Kenobi. Eres mi única esperanza.” – *Princesa Leia*

“Hazlo o no lo hagas, pero no lo intentes.” – *Yoda*

Currying

Serie de funciones de 1 argumento

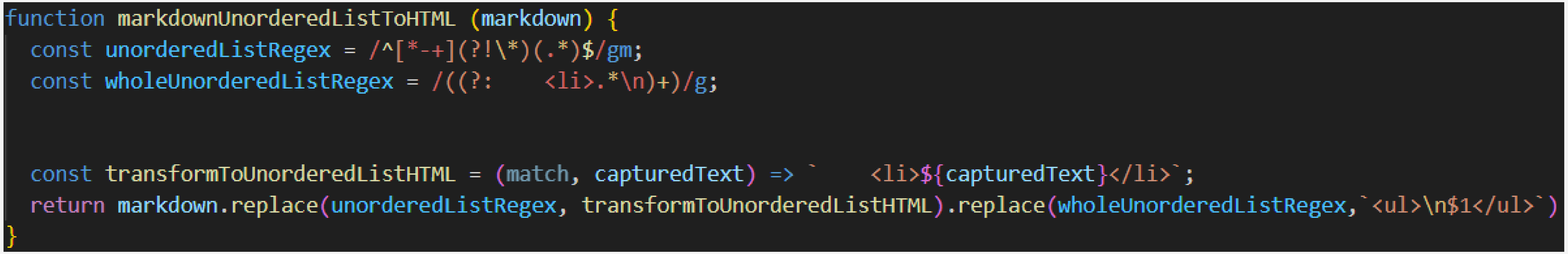
```
const replaceMarkdown = regex => replacement => markdown => markdown.replace(regex, replacement);
```

```
const markdownBoldToHTML = replaceMarkdown(boldRegex)('<strong>$1$2</strong>');
const markdownItalicToHTML = replaceMarkdown(italicRegex)('<em>$1$2</em>');
const markdownItalicBoldToHTML = replaceMarkdown(italicBoldRegex)('<strong><em>$1$2</em></strong>');
const markdownCodeToHTML = replaceMarkdown(codeRegex)('<code>$1</code>');
const markdownHorizontalRuleToHTML = replaceMarkdown(horizontalRuleRegex)('<hr>');
```

Funciones destacadas

Funciones

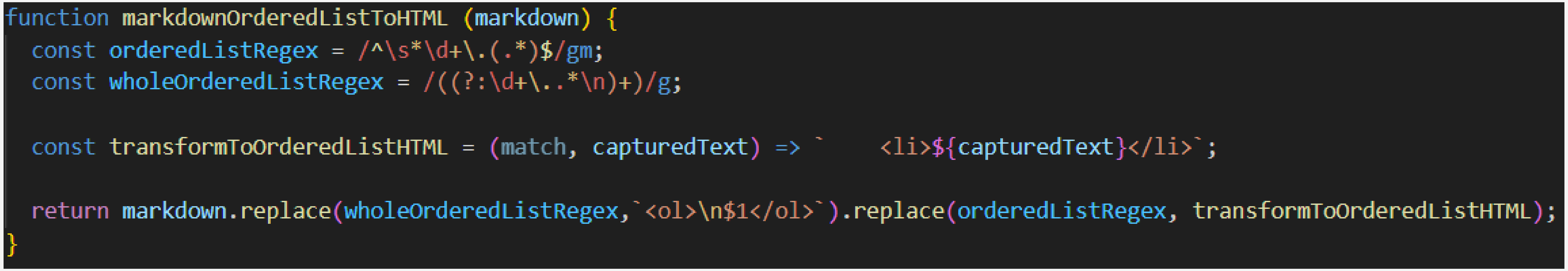
Listas no ordenadas



```
function markdownUnorderedListToHTML (markdown) {
  const unorderedListRegex = /^[*-+](?!\*)(.*)$/gm;
  const wholeUnorderedListRegex = /((?:    <li>.*\n)+)/g;


  const transformToUnorderedListHTML = (match, capturedText) => `    <li>${capturedText}</li>`;
  return markdown.replace(unorderedListRegex, transformToUnorderedListHTML).replace(wholeUnorderedListRegex,`<ul>\n$1</ul>`)
}
```

Funciones

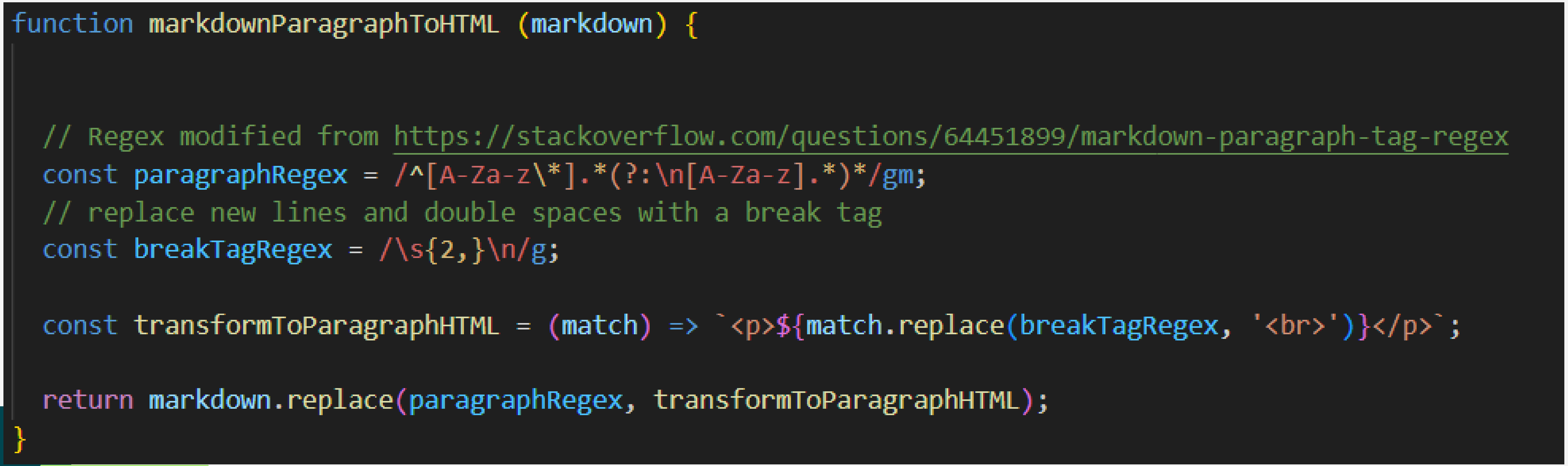
Listas ordenadas



```
function markdownOrderedListToHTML (markdown) {
  const orderedListRegex = /^\s*\d+\.(.*)$/gm;
  const wholeOrderedListRegex = /((?:\d+\..*\n)+)/g;

  const transformToOrderedListHTML = (match, capturedText) => `    <li>${capturedText}</li>`;

  return markdown.replace(wholeOrderedListRegex,`<ol>\n$1</ol>`).replace(orderedListRegex, transformToOrderedListHTML);
}
```

Funciones

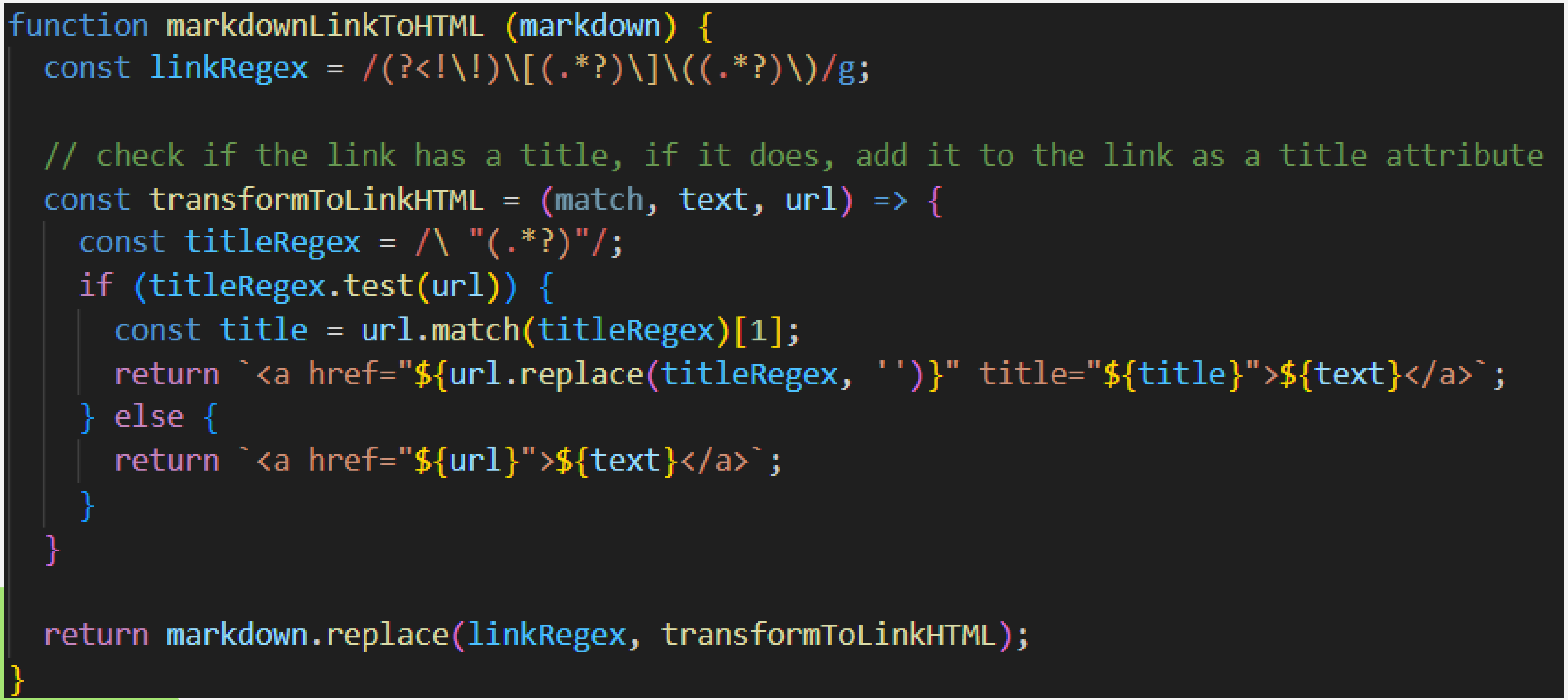
Parrafos



```
function markdownParagraphToHTML (markdown) {


  // Regex modified from https://stackoverflow.com/questions/64451899/markdown-paragraph-tag-regex
  const paragraphRegex = /^[A-Za-z\*].*(?:\n[A-Za-z].*)*/gm;
  // replace new lines and double spaces with a break tag
  const breakTagRegex = /\s{2,}\n/g;

  const transformToParagraphHTML = (match) => `<p>${match.replace(breakTagRegex, '<br>')}</p>`;

  return markdown.replace(paragraphRegex, transformToParagraphHTML);
}
```

Funciones

Links



```
function markdownLinkToHTML (markdown) {
  const linkRegex = /(?<!\!)\[(.*?)\]\((.*?)\)/g;

  // check if the link has a title, if it does, add it to the link as a title attribute
  const transformToLinkHTML = (match, text, url) => {
    const titleRegex = /\ "(.*?)"/;
    if (titleRegex.test(url)) {
      const title = url.match(titleRegex)[1];
      return `<a href="${url.replace(titleRegex, '')}" title="${title}">${text}</a>`;
    } else {
      return `<a href="${url}">${text}</a>`;
    }
  }

  return markdown.replace(linkRegex, transformToLinkHTML);
}
```

Markdown to HTML Transpiler

IIC3585 - Grupo 7

Demo