


Functional JS

A large, bold, black 'JS' logo is centered on a bright yellow rectangular background. The letters are thick and have a slightly rounded, modern feel.

Supuestos

- El archivo a leer sigue la guía de Markdown.
- No consideramos anidaciones de elementos.

Flujo principal: MD a HTML

```
const convertMarkdownToHtml = _.flow(  
  readFile,  
  parseBolds,  
  parseItalics,  
  parseCode,  
  parseImages,  
  parseLinks,  
  addLineBreaks,  
  _.split('\n'),  
  _.map(_.cond(conditions)),  
  parseLists,  
  _.join('\n'),  
  joinTags,  
  writeFile  
);
```

Condiciones

```
// [function: bool, function to execute if bool is true]
const conditions = [
  [(line) => line === '', () => '\n'],
  [(line) => /^#{1,6} /g.test(line), (line) => parseHeader(line)],
  [(line) => /^[*+-] /g.test(line), (line) => parseUnorderedList(line)],
  [(line) => /^[0-9]+. /g.test(line), (line) => parseOrderedList(line)],
  [(line) => line.startsWith('>'), (line) => parseBlockquote(line)],
  [(line) => line === '---', () => '<hr />'],
  [(line) => line === '<br>', (line) => line],
  [_ .stubTrue, (line) => `<p>${line}</p>`],
];
```

Input y output

```
const fs = require('fs');
const _ = require('lodash/fp');

// Curried function to read file
const readFile = _.curry(fs.readFileSync)(_, 'utf8');

// Curried function to write file
const writeFile = _.curry(fs.writeFileSync)('output.html', _, 'utf8');
```

Uso de regex

```
// Function to parse a text to a tag using regex  
const parseTag = (regex, tag) => (text) =>  
  text.replace(regex, `<${tag}>$1</${tag}>`);
```

Bold, italic y code

```
const parseBoldAsterisks = parseTag(/\*\.*(.*)\*/gm, 'strong');  
const parseBoldUnderscores = parseTag(/\_\.*(.*)\_/gm, 'strong');
```

```
const parseItalicAsterisks = parseTag(/\*\.*(.*)\*/gm, 'em');  
const parseItalicUnderscores = parseTag(/\_\.*(.*)\_/gm, 'em');
```

```
const parseCodeDoubleBackticks = parseTag(/`\.*(.*)`/gm, 'code');  
const parseCodeSingleBackticks = parseTag(/`\.*(.*)`/gm, 'code');
```

Bold, italic y code

```
// Parses markdown bold text to html
const parseBolds = (text) =>
  |_.flow(parseBoldAsterisks, parseBoldUnderscores)(text);

// Parses markdown italic text to html
const parseItalics = (text) =>
  |_.flow(parseItalicAsterisks, parseItalicUnderscores)(text);

// Parses markdown code text to html
const parseCode = (text) =>
  |_.flow(parseCodeDoubleBackticks, parseCodeSingleBackticks)(text);
```


Line break, image y link

```
// Adds line breaks to text
const addLineBreaks = (text) => text.replace(/  $/gm, '<br>');

// Parse markdown images to html
const parseImages = (text) =>
  text.replace(/!\[ (.*) \] \((.*) \)/gm, "<img alt='$1' src='$2' />");

// Parse markdown links to html
const parseLinks = (text) =>
  text.replace(/\[ (.*) \] \((.*) \)/gm, "<a href='$2'>$1</a>");
```

Blockquotes

```
const parseBlockquote = (line) => {  
  const content = line.slice(1).trim();  
  return `<blockquote>${content}</blockquote>`;  
};
```

Headers

```
// Parses markdown headers to html
const parseHeader = (line) => {
  const level = line.split(' ')[0].length;
  const content = line.slice(level).trim();
  return `<h${level}>${content}</h${level}>`;
};
```

Lists

```
// Parses markdown unordered lists to html
const parseUnorderedList = (line) => {
  const content = line.slice(2).trim();
  return `

${content}</ul>`;
};

// Parses markdown ordered lists to html
const parseOrderedList = (line) => {
  const index = line.indexOf('.') + 1;
  const content = line.slice(index).trim();
  return `

${content}</ol>`;
};
```

Lists

```
// Parses de <ul> and <ol> tags to <li> tags
const parseListTags = (line, tag) => {
  const regexStartingTag = new RegExp(`<${tag}>`, 'g');
  const regexEndingTag = new RegExp(`</${tag}>`, 'g');
  return `<${tag}>\n${line
    .replace(regexStartingTag, '  <li>')
    .replace(regexEndingTag, '</li>\n')
    .trimEnd()} \n</${tag}>`;
};
```

```
// Parses de <ul> and <ol> tags to <li> tags
const parseLists = (array) => {
  const joinedArray = array.join('').split('\n');
  return joinedArray.map((line) =>
    line.startsWith('<ul>')
      ? parseListTags(line, 'ul')
      : line.startsWith('<ol>')
        ? parseListTags(line, 'ol')
        : line
  );
};
```

Join tags

```
// Join paragraphs tags
const joinParagraphs = (text) => text.replace(/<\n/p><p>/g, ' ');

// Join blockquotes tags
const joinBlockquotes = (text) =>
  text.replace(/<\n/blockquote><blockquote>/g, '<br>');

// Join tags
const joinTags = (text) => _.flow(joinParagraphs, joinBlockquotes)(text);
```

Functional JS

A large, bold, black 'JS' logo is centered on a bright yellow rectangular background. The letters are thick and have a slightly rounded, modern feel.