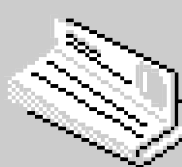
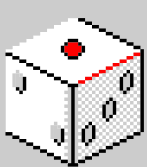
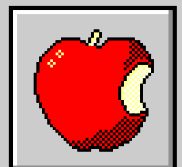


JS funcional

Grupo 5

Nicolás Barría
Sebastián Carrasco
Juan Vargas



11:11 PM

Párrafos y frases



1 Hola. Este es un texto de prueba para eso. Nada mas. Nada menos.
2
3 Este es otro parrafo solo.
4
5 Este parrafo tiene 6 frases. Esta es la segunda frase.
6 Esta es la tercera frase. Esta es la cuarta frase. Esta es
7 la quintafrase. Y finalmente esta es la sexta frase.

Cada frase debe comenzar con n espacios en blanco



```
1  const addSpacesBeforePhrase = (text, n) =>
2    _.replace(text, /(?<=\S)\. [^\S\r\n]+(?=[A-Z])/g, _.padEnd(".", n + 1));
```

Cada párrafo debe estar separado por n líneas



```
1  const spacesBeforeParagraph = (text, n) =>
2    _.replace(
3      text,
4      /(?!<=\S)\.(\r\n|\r|\n){2,}(?=[A-Z])/g,
5      _.padEnd(".", n + 2, "\n")
6    );
```

El ancho del texto debe ser a lo más n



```
1  const maxLineLength = (n, line) => {
2    if (line.length <= n) {
3      return line;
4    }
5    const splitIndex = _.findLastIndex(line, (char, index) => char === ' ' && index <= n);
6    return `${line.slice(0, splitIndex)}\n${maxLineLength(n, line.slice(splitIndex + 1, line.length))}`;
7  };
8
9  const curriedMaxLineLength = _.curry(maxLineLength);
10
11 const maxLengthPerLine = (text, n) => {
12   const NLineLength = curriedMaxLineLength(n);
13   const lines = _.split(text, '\n');
14   const newLines = _.map(lines, NLineLength);
15   return _.join(newLines, '\n');
16 };
```



El ancho del texto debe ser a lo más n

```
1  const maxLineLengthGen = (f) => ((n, line) => {
2    if (line.length <= n) {
3      return line;
4    }
5    const splitIndex = _.findLastIndex(line, (char, index) => char === ' ' && index <= n);
6    return `${line.slice(0, splitIndex)}\n${f(n, line.slice(splitIndex + 1, line.length))}`;
7  });
8
9  const Y = (f) => ((x) => x(x))((x) => f((y, z) => x(x)(y, z)));
10 const maxLineLength = Y(maxLineLengthGen);
11
12 const curriedMaxLineLength = _.curry(maxLineLength);
13
14 const maxLengthPerLine = (text, n) => {
15   const NLineLength = curriedMaxLineLength(n);
16   const lines = _.split(text, '\n');
17   const newLines = _.map(lines, NLineLength);
18   return _.join(newLines, '\n');
19 };
```

Cada párrafo debe tener n espacios de sangría



```
1  const addIndentBeforeParagraph = (text, n) =>
2    _.repeat(" ", n) +
3    _.trim(_.replace(text, /(?!<=.\n\n)\s*(?=[A-Z])/g, _.padEnd(" ", n)), " ");
```

Ignorar párrafos que tienen más/menos de n frases



```
1  const filterLargerSizePhrases = (n, paragraph) => _.size(paragraph.split(/(?<=\. )/g)) > n;
2  const filterSmallerSizePhrases = (n, paragraph) => _.size(paragraph.split(/(?<=\. )/g)) < n;
3
4  const curriedGreater = _.curry(filterLargerSizePhrases);
5  const curriedLesser = _.curry(filterSmallerSizePhrases);
6
7  const filterParagraphs = (text, filter) => {
8    const paragraphs = _.split(text, /(?<=\n\n+)(?=\s*[A-Z])/g);
9    const paragraphsFiltered = _.filter(paragraphs, filter);
10   return _.join(paragraphsFiltered, '');
11 };
12
13 // Ejemplos de uso
14 const filteredText1 = filterParagraphs(text, curriedGreater(n));
15 const filteredText2 = filterParagraphs(text, curriedLesser(n));
```


Cada frase debe aparecer en párrafo aparte



```
1  const eachPhraseInAParagraph = (text) =>
2  _.replace(text, /(?<=\S\.)( | \s)(?=[A-Z])/g, '\n\n');
```

Solo las primeras n frases de cada párrafo



```
1  const onlyFirstPhrases = (text, n) => {
2    const paragraphs = _.split(text, '\n\n');
3    const phrasesPerParagraph = _.map(paragraphs, (paragraph) => _.split(paragraph, /(?<=\. )/g));
4    const firstPhrases = _.map(phrasesPerParagraph, (phrases) => _.take(phrases, n));
5    const firstPhrasesParagraphs = _.map(firstPhrases, (phrases) => _.join(phrases, ' '));
6    const finalText = _.join(firstPhrasesParagraphs, '\n\n');
7    return finalText;
8  };
```

Solo las primeras n frases de cada párrafo



```
1  const pipe = (functions) => (data) => functions.reduce((value, func) => func(value), data);
2
3  const onlyFirstPhrases = (text, n) => {
4    const operations = [
5      (txt) => _.split(txt, '\n\n'),
6      (paragraphs) => _.map(paragraphs, (paragraph) => _.split(paragraph, /(?<=\. )/g)),
7      (phrasesPerParagraph) => _.map(phrasesPerParagraph, (phrases) => _.take(phrases, n)),
8      (firstPhrases) => _.map(firstPhrases, (phrases) => _.join(phrases, ' ')),
9      (firstPhrasesParagraphs) => _.join(firstPhrasesParagraphs, '\n\n'),
10   ];
11   const pipeline = pipe(operations);
12   return pipeline(text, n);
13  };
```

Solo las primeras n frases de cada párrafo



```
1  const pipe = (functions) => (data) => functions.reduce((value, func) => func(value), data);
2
3  const takeFirstPhrases = (n, phrases) => _.take(phrases, n);
4  const curriedTakeFirstPhrases = _.curry(takeFirstPhrases);
5
6  const onlyFirstPhrases = (text, n) => {
7    const takeFirstNPhrases = curriedTakeFirstPhrases(n);
8
9    const operations = [
10      (txt) => _.split(txt, '\n\n'),
11      (paragraphs) => _.map(paragraphs, (paragraph) => _.split(paragraph, /(?<=\. )/g)),
12      (phrasesPerParagraph) => _.map(phrasesPerParagraph, takeFirstNPhrases),
13      (firstPhrases) => _.map(firstPhrases, (phrases) => _.join(phrases, ' ')),
14      (firstPhrasesParagraphs) => _.join(firstPhrasesParagraphs, '\n\n'),
15    ];
16    const pipeline = pipe(operations);
17    return pipeline(text);
18  };
```

Ejemplo Simple

Input —→ `addIndentBeforeParagraph(text, 5);` —→ **Output**

```
1 Hola. Este es un texto de
2 prueba para eso. Nada mas.
3 Nada menos.
4
5 Este es otro parrafo
6 solo.
7
8 Este parrafo tiene 6 frases.
9 Esta es la segunda frase. Esta
10 es la tercera frase. Esta es la
11 cuarta frase. Esta es la quinta
12 frase. Y finalmente esta
13 es la sexta frase.
```

```
1      Hola. Este es un texto de
2      prueba para eso. Nada mas.
3      Nada menos.
4
5      Este es otro parrafo
6      solo.
7
8      Este parrafo tiene 6 frases.
9      Esta es la segunda frase. Esta
10     es la tercera frase. Esta es la
11     cuarta frase. Esta es la quinta
12     frase. Y finalmente esta
13     es la sexta frase.
```

Ejemplo Doble

Input —→ `text = maxLengthPerLine(text, 20);` —→ `text = maxLengthPerLine(text, 13);`

```
1  Hola. Este es un texto de
2  prueba para eso. Nada mas.
3  Nada menos.
4
5  Este es otro parrafo
6  solo.
7
8  Este parrafo tiene 6 frases.
9  Esta es la segunda frase. Esta
10 es la tercera frase. Esta es la
11 cuarta frase. Esta es la quinta
12 frase. Y finalmente esta
13 es la sexta frase.
```

Output

```
1  Hola. Este es
2  un
3  texto de
4  prueba para
5  eso. Nada
6  mas. Nada
7  menos.
8
9  Este es otro
10 parrafo
11 solo.
12
13 Este parrafo
14 tiene 6
15 frases. Esta
16 es la
17 segunda
18 frase. Esta
19 es la tercera
20 frase.
21 Esta es la
22 cuarta
23 frase. Esta
24 es la
25 quinta frase.
26 Y
27 finalmente
28 esta es
29 la sexta
30 frase.
```


Ejemplo Múltiple

Input —→ `text = onlyFirstPhrases(text, 3);` —→ **Output**

`text = eachPhraseInAParagraph(text);`

`text = addIndentBeforeParagraph(text, 2);`

```
1  Hola. Este es un texto de
2  prueba para eso. Nada mas.
3  Nada menos.
4
5  Este es otro parrafo
6  solo.
7
8  Este parrafo tiene 6 frases.
9  Esta es la segunda frase. Esta
10 es la tercera frase. Esta es la
11 cuarta frase. Esta es la quinta
12 frase. Y finalmente esta
13 es la sexta frase.
```

```
1  Hola.
2
3  Este es un texto de
4  prueba para eso.
5
6  Nada mas.
7
8  Este es otro parrafo
9  solo.
10
11 Este parrafo tiene 6 frases.
12
13 Esta es la segunda frase.
14
15 Esta
16 es la tercera frase.
```