Você pode usar o **delete** quando você quer remover um elemento do array, mas você não quer que os índices sejam alterados.

Exemplo:

```
const new_words = ['Deliberately', 'Nervousness', 'Anxiety']; // Array literal
new_words[2] = 'Hyperventilate';
delete new_words[2];
console.log(new_words);
```

Output: `[ 'Deliberately', 'Nervousness', <1 empty item> ]`

```
[ 'Deliberately', 'Nervousness', <1 empty item> ]
```

O array new_words eu posso criar de outra maneira não muito utilizada utilizando o construtor do array:

```
const nomes = new Array('Deliberately', 'Nervousness', 'Anxiety');
```

Ela funciona do mesmo jeito que a versão anterior, porém, a versão anterior é mais intuitiva e curta, então eu vou sempre usá-la e isso também funciona com qualquer tipo de dado tanto da forma literal quanto da forma construtora dos tipos de dados que eu precisar.

Arrays são dados por referência então eu sei que eu não posso fazer:

```
const new_words = ['Deliberately', 'Nervousness', 'Anxiety'];
const novo = new_words;
```

O que vai acontecer aqui não é o esperado. Então esse valor **novo** foi passado como referência tudo o que eu fizer em new_words reflete em **novo** e tudo o que eu fizer em **novo** reflete em new_words.

Por exemplo:

```
const new_words = ['Deliberately', 'Nervousness', 'Anxiety'];
const novo = new_words;

novo.pop();
console.log(new_words);
```

Output: `[ 'Deliberately', 'Nervousness' ]`

Então se eu fizer qualquer alteração em qualquer um dos dois os dois apontam para o mesmo valor. Os valores de referência podem ser copiados a gente utilizar o operador chamado **spread**, ele é o mesmo operador do **rest** só muda o contexto aonde eu estou utilizando ele, por exemplo, quando eu quero pegar o resto do Array eu utilizo o **rest** (...), quando eu quero espalhar um Array, pegar os dados que estão dentro do Array e espalhar eles eu vou utilizar também o **rest** que é o mesmo operador do **spread**, então por exemplo se eu quero criar, copiar todos os elementos e espalhar tudo dentro de um outro Array, eu uso o **spread**:

```
const novo = [...new_words];
```

Agora o que eu fizer em new_words não reflete em **novo**, o que eu faço em **novo** não reflete em new_words porque eu fiz uma cópia de new_words pra dentro de **novo**.

metódo termina com ();
atributo não tem ();

pop() remove o último elemento do Array e eu posso pegar o elemento removido colocando dentro de uma variável:

```
const new_words = ['Deliberately', 'Nervousness', 'Anxiety'];
const removed = new_words.pop();
console.log(new_words);
console.log(removed);
```

shift() remove o primeiro elemento:

```
const new_words = ['Deliberately', 'Nervousness', 'Anxiety'];
const removed = new_words.shift();
console.log(new_words);
console.log(removed);
```

É a mesma coisa a diferença é que o shift() vai deslocar todos os outros índices do Array.

Uma coisa que eu vou fazer muito é adicionar elementos no Array, eu posso adicionar um elemento no final do Array assim:

```
new_words.push('Symptom');
```

Mas também similar ao **push** eu também posso adicionar se eu quiser (não é muito comum), eu posso querer adicionar elemento no começo do meu Array porque o

shift e o **unshift** vão deslocar todos os índices do Array, para um Array pequeno não faz tanta diferença mas para um Array muito grande pode trazer um problema de perfomance pro meu programa.

```
new_words.unshift('Symptom');
```

Existe o método **splice** que faz tudo o que esses métodos fazem e um pouquinho mais. A gente pode fatiar o nosso Array usando o método **slice**:

```
const new_words = ['Deliberately', 'Nervousness', 'Anxiety', 'Symptom',
'Prejudice'];
const novo = new_words.slice(1, 3);
console.log(novo);
```

O que é muito útil no slice é que eu posso trabalhar com números negativos pra caso você queira por exemplo remover algum índice:

```
const novo = new_words.slice(0, -1);
```

Eu posso converter uma string um array:

```
const name = 'Leonardo Mancilha Machado';
const names = name.split(' ');
console.log(names);
```

Para converter um Array em uma string usando o método **join**:

```
const names = [ 'Leonardo', 'Mancilha', 'Machado' ];
const name = names.join(' ');
console.log(name);
```