Задача C. Шаблонизатор 3000

Однажды, при работе над очередной задачей Дэн подумал, что он очень много времени тратит на написание html-кода. Как любой уважающий себя фронтенд-инженер Дэн решил написать свой HTML-шаблонизатор, чтобы ускорить процесс.

Он придумал имя для новой библиотеки, создал git-репозиторий и начал с документации в README.md:

$ cat README.md

Templater - новый гибкий html-шаблонизатор на чистом JavaScript.

Доступны такие html-теги: `<div></div>`, `<span></span>`, `<br>`, `<p></p>`

Для использования шаблонизатора необходимо сначала создать новый экземпляра. Далее можно добавлять в шаблон теги вызывая методы с соответствующими названиями:

```
const myTemplate = Templater()
myTemplate.div()  // добавляет в шаблон <div></div>
myTemplate.span()  // добавляет в шаблон <span></span>
myTemplate.br()  // добавляет в шаблон <br>
myTemplate.p()  // добавляет в шаблон <p></p>
```

Для вывода html-кода необходимо вызвать метод `toString`:

```
const myTemplate = Templater()
myTemplate.div('hello')
console.log(myTemplate.toString())
// <div>hello</div>
```

Поддерживается вложенность содержимого и чейнинг вызовов.

```
const template = Templater().div(
  Templater().p('Hello'),
  Templater().p('World')
)
console.log(template.toString())
// <div><p>Hello</p><p>World</p></div>
```

Внимание! Метод `br()` не поддерживает вложенное содержимое и вызовет исключение с текстом ошибки `Nested content is not allowed`:

```
console.log(Templater().br('error').toString())
// Uncaught Error: Nested content is not allowed
```

Шаблонизатор поддерживает атрибуты тегов. Для добавление атрибутов необходимо передать в метод тега объект состоящий из имен в ключах объекта и значений. Атрибуты выводятся в лексикографическом порядке:

```
console.log(Templater().div({id: "header", class: "m-4 float-right"}).toString())
// <div class="m-4 float-right" id="header"></div>
```

Важно! Если одновременно используются атрибуты и вложенный контент, то объект атрибутов должен передаваться **последним**!

```
console.log(Templater().div('World of Templates', {id: "card"}).toString())
// <div id="card">World of Templates</div>
```

Задание

Для того чтобы облегчить себе работу Дэн сразу написал несколько тестов, которые запускал с помощью Jest:

$ cat templater.spec.js

```
const Templater = require('./templater')

describe('Templater', () => {
  it('create instance', () => {
    expect(typeof Templater).toBe('function')
    expect(() => { Templater() }).not.toThrow()
  })
  it('has proper methods', () => {
    const template = Templater()
    expect(typeof template.div).toBe('function')
    expect(typeof template.br).toBe('function')
  })
  it('renders html', () => {
    expect(Templater().span().toString()).toBe('<span></span>')
    expect(Templater().br().toString()).toBe('<br>')
  })
  it('supports nesting, chaining', () => {
    expect(
      Templater().span('Hello').span('World').toString()
    ).toBe('<span>Hello</span><span>World</span>')
    expect(
      Templater().p(
        Templater().span('nested'),
        Templater().span('span')
      ).toString()
    ).toBe('<p><span>nested</span><span>span</span></p>')
    expect(() => {
      Templater().br('some content')
    }).toThrow()
  })
  it('tags has attributes', () => {
    expect(
      Templater().div('Yeah!', { id: "header", class: "awesome" }).toString()
    ).toBe('<div class="awesome" id="header">Yeah!</div>')
  })
})
```

Но в какой-то момент Дэн переключился на другую важную задачу и, наверное, забыл закомитить основной файл с имплементацией шаблонизатора. А в git'е осталась только заготовка в `templater.js`:

$ cat Templater.js

```
// TODO тут будет имплементация Templater

module.exports = function() {
   return new Templater()
}
```

Ваша задача: реализовать логику шаблонизатора и прислать нам своё решение.

Как отправить решение?

Ваше решения должно представлять собой модуль на языке JavaScript. Модуль должен экспортировать шаблонизатор с использованием синтаксиса CommonJS (как в примере выше). Для отправки решения вам нужно выбрать в системе задачу и отправить исходный файл с кодом. Он будет проверен системой на серии тестов. Тест считается пройденным, если программа вывела правильный ответ и уложилась в ограничения по времени работы и используемой памяти. За каждый пройденный тест начисляется один балл. Результаты первых 10 тестов вы сможете увидеть в системе. Первый тест всегда из условия задачи. Общий результат по задаче определяется по решению, набравшему максимальное количество баллов. Количество попыток не ограничено.