```
//
Libraries
            const fs = require('fs');
            const jsdom = require('jsdom');
            const { assert } = require('chai');

            // HTML
            const srcHtml = fs.readFileSync('./src/index.html');
            const doc = jsdom.jsdom(srcHtml);

            // Tests
            describe('The webpage', () => {

              /**
               * HEADER
               */
              describe('header', () => {
                it('should exist @header', () => {
                  const header = doc.querySelector('.header');
                  assert.isOk(header, 'We need a `.header` element.');
                });

                it('should have a non-empty title @h1', () => {
                  const h1 = doc.querySelector('.header h1');
                  assert.isOk(h1, 'We need an `h1` element inside `.header`.');
                  assert.isOk(h1.textContent, 'Our header\'s `h1` element cannot be
            empty.');
                });

                it('should have a non-empty description @h2', () => {
                  const h2 = doc.querySelector('.header h2');
                  assert.isOk(h2, 'We need an `h2` element inside `.header`.');
                  assert.isOk(h2.textContent, 'Our header\'s `h2` element cannot be
            empty.');
                });
              });


              /**
               * TAGLINE
               */
              describe('tagline', () => {
                it('should exist @tagline', () => {
```

```
      const tagline = doc.querySelector('.tagline');
      assert.isOk(tagline, 'We need a `.tagline` element.');
    });

    it('should have a non-empty h3 tag @tagline-content', () => {
      const h3 = doc.querySelector('.tagline h3');
      assert.isOk(h3, 'We need an `h3` element inside `.tagline`.');
      assert.isOk(h3.textContent, 'Our tagline\'s `h3` element cannot be
empty.');
    });

    it('should have a descriptive paragraph @tagline-content', () => {
      const p = doc.querySelector('.tagline p');
      assert.isOk(p, 'We need a `p` element inside `.tagline`.');
      assert.isOk(p.textContent, 'Our tagline\'s `p` element cannot be
empty.');
    });
  });


  /**
   * SKILLS
   */
  describe('skills', () => {
    it('should exist @skills', () => {
      const skills = doc.querySelector('.skills');
      assert.isOk(skills, 'We need a `.skills` element.');
    });

    it('should have a non-empty h3 tag @skills-content', () => {
      const h3 = doc.querySelector('.skills h3');
      assert.isOk(h3, 'We need an `h3` element inside `.skills`.');
      assert.isOk(h3.textContent, 'Our skills\' `h3` element cannot be
empty.');
    });

    it('should have a descriptive paragraph @skills-content', () => {
      const p = doc.querySelector('.skills p');
      assert.isOk(p, 'We need a `p` element inside `.skills`.');
      assert.isOk(p.textContent, 'Our skills\' `p` element cannot be empty.');
    });

    it('should have an unordered list of your skills @skills-list', () => {
      const ul = doc.querySelector('.skills ul');
      assert.isOk(ul, 'We need a `ul` element inside `.skills`.');
    });

    it('should have at least 3 skills @skills-list', () => {
      const skillItems = doc.querySelectorAll('.skills ul li');
      assert.isAtLeast(skillItems.length, 3, 'We need at least 3 `li` elements
inside the skills\' `ul`.');
    });

    it('should have one skill that contains HTML @skills-list', () => {
      const skillItems = Array.from(doc.querySelectorAll('.skills ul li'));
      const htmlRegex = /html/i;

      const skillsWithHtml = skillItems
        .map(li => li.textContent)
        .filter(skill => htmlRegex.test(skill));

      assert.equal(skillsWithHtml.length, 1, 'HTML needs be part of one of your
skills.');
    });
  });


  /**
   * CONTACT
   */
  describe('contact', () => {
    it('should exist @contact', () => {
      const contact = doc.querySelector('.contact');
      assert.isOk(contact, 'We need a `.contact` element.');
    });

    it('should have a non-empty h3 tag @contact-content', () => {
      const h3 = doc.querySelector('.contact h3');
      assert.isOk(h3, 'We need an `h3` element inside `.contact`.');
      assert.isOk(h3.textContent, 'Our contact\'s `h3` element cannot be
empty.');
    });

    it('should have a descriptive paragraph @contact-content', () => {
      const p = doc.querySelector('.contact p');
      assert.isOk(p, 'We need a `p` element inside `.contact`.');
      assert.isOk(p.textContent, 'Our contact\'s `p` element cannot be
empty.');
    });

    it('should have a link with an href within the paragraph @contact-link', ()
=> {
      const a = doc.querySelector('.contact p a');
      assert.isOk(a, 'We need a `a` element our inside contact\'s `p`
element.');
      assert.isOk(a.textContent, 'Our contact\'s `a` element cannot be
empty.');
      assert.isOk(a.getAttribute('href'), 'Our contact\'s `a` element needs a
non-empty `href` attribute.');
    });
  });

});
```