Herramientas HTML y CSS II

PEC 1

Moisés Segura Cedrés.

Enlaces

Github: https://github.com/msegurace/html-css-ii-pec1

Página web: https://master--rainbow-tiramisu-d0d617.netlify.app/

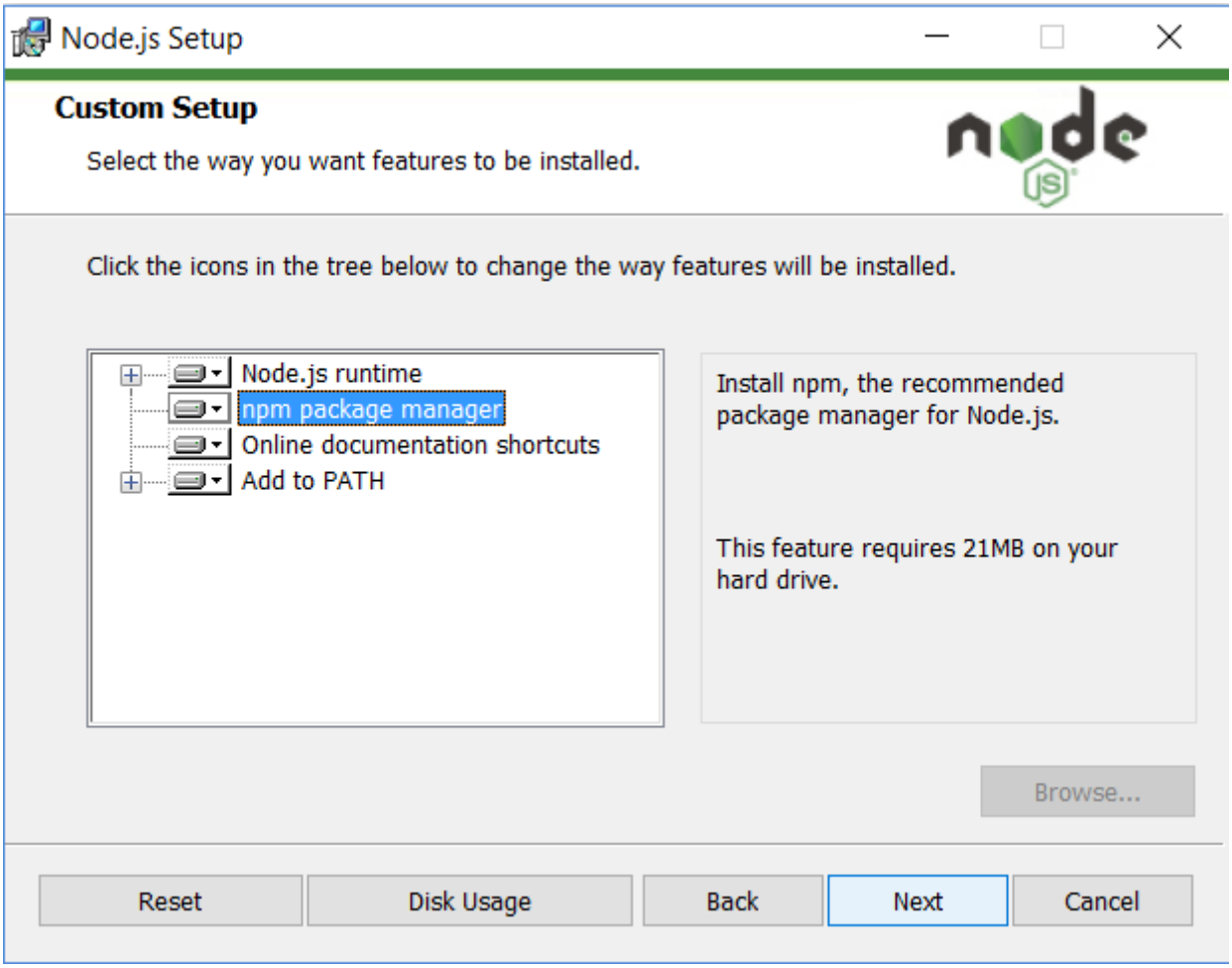
Instalación de la plataforma

Descargo e instalo el Visual Studio Code en Windows desde https://code.visualstudio.com/ ya que voy a utilizar la consola PowerShell como terminal.

Instalación del gestor de paquetes npm y de NodeJs

He seguido la guía de la página https://radixweb.com/blog/installing-npm-and-nodejs-on-windows-and-mac

1. Descarga del instalador desde la página oficial de NodeJs (https://nodejs.org/en/download/)
2. Ejecuto el instalador y en las opciones escojo npm package manager en vez del que viene por defecto.



3. Compruebo que están bien instalados y su versión.

```
PS C:\Users\MOISES\Documents\UOC\Herramientas HTML y CSS II\PEC1> node -v
v19.8.1
PS C:\Users\MOISES\Documents\UOC\Herramientas HTML y CSS II\PEC1> npm -v
9.6.4
```

Instalación del Boilerplate de la UOC:

1. Creo una carpeta PEC1 en mi ordenador.
2. Accedo a https://github.com/uoc-advanced-html-css/uoc-boilerplate

3. En una consola PowerShell de mi equipo ejecuto:

   git clone https://github.com/uoc-advanced-html-css/uoc-boilerplate

4. Ejecuto npm install para que instale las dependencias en node_modules e instale el module bundle Parcel
5. Para utilizar el plugin de posthtml y poder utilizar <include> dentro de mi HTML creo el archivo .psthtmlrc con el siguiente contenido:

```
{
 "plugins": {
  "posthtml-include": {},
  "posthtml-expressions": {
   "locals": {
    "active": "OK"
   }
  }
 }
}
```

6. Añado al script main.js el siguiente contenido:

```
const { readFileSync } = require('fs')

const posthtml = require('posthtml')
const include = require('posthtml-include')

const html = readFileSync('index.html')

posthtml([ include({ encoding: 'utf8' }) ])
  .process(html)
  .then((result) => console.log(result.html))
```

7. Pruebo la instalación base funciona con npm run dev

Despliegue del proyecto con Netlify

1. Creo una cuenta gratuita a partir de mi cuenta de github.com
2. Añado el proyecto de github como origen para hacer el despliegue.
3. Ejecuto el despliegue en Netlify y obtengo la url:

https://master--rainbow-tiramisu-d0d617.netlify.app/

Elección de guías de estilo

Me he centrado en “CSS Guidelines” de Harry Roberts ya que propone una herramienta para los equipos que:

- construyen y mantienen productos durante un tiempo razonablemente largo;
- tienen desarrolladores con diferentes habilidades y especialidades;
- tienen un número variable de desarrolladores que trabajan en el proyecto en un momento determinado;
- hay una constante contratación de personal nuevo;
- tienen una serie de código base en el que los desarrolladores entran y salen constantemente.

En mi día a día laboral tengo que lidiar con gran cantidad de código “heredado” que no ha seguido nunca una guía conllevando esto una pérdida importante de horas/hombre en sólo tratar de entenderlo. El autor propone una serie de prácticas y principios muy sencillos de aplicar y que, en un corto periodo de tiempo, supondría un mejora significativa a cualquier organización y proyecto.

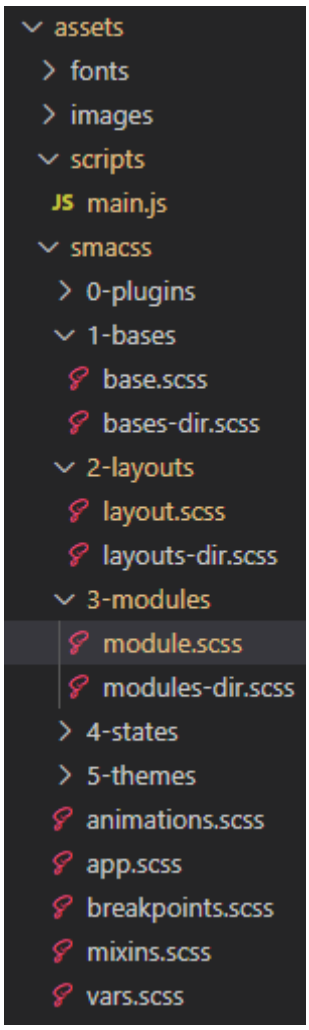
En este caso se encarga un sitio web de una sola página, para construir este sitio se ha generado una gran cantidad de código SCSS que, si no se siguiesen estas guías de estilo y al presuponer que el sitio seguirá creciendo, sería mucho más difícil de mantener, reutilizar, etc.

Como sintaxis, siguiendo las recomendaciones del autor, aplico a mi código:

- dos espacios de indentación
- columnas de 80 caracteres.
- CSS multilínea
- Uso de espacios en blanco.

He configurado todas estas opciones en Visual Studio Code para que, por ejemplo, cuando se pulse un tabulador introduzca los dos espacios.

He aplicado al sitio web una guía de estilo basado en el uso SMASS debido a su organización en carpetas que hace más fácil leer y mantener el código, me permite su reutilización reduciendo el tamaño del proyecto y aumentando la velocidad de carga.
Me he apoyado en la herramienta SMACSS_IT (https://github.com/cbedroid/smacss-it) para generar la estructura de carpetas y ficheros quedando así:



- En base incluyo todlo lo que tenga que ver con tamaños globales, estilo de vínculos predenterminados, fuentes, etc.

```scss
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

*,
*::before,
*::after {
  box-sizing: inherit;
}

html {
  font-family: Montserrat, sans-serif;
  font-size: 10px;
```

- En layouts pondré todo lo referente a la estructura de la página como headers, footers, etc.

```scss
header {
  background-color: $color-texto;
  text-align: center;
  top: 0;
  display: flex;
  flex-direction: row;
  max-width: $layout-with;
  margin: 0 auto;
}

header > div {
  position: relative;
  max-width: $layout-with;
  display: flex;
}

header a img {
  flex: 1;
  max-width: 180px;
}

header h1 {
  flex: 1;
  padding-top: 65px;
  padding-left: 150px;
}

footer {
  background-color: $color-texto-fondo-oscuro;
  color: $color-texto;
  line-height: 1.5;
  display: table;
  overflow: hidden;
```

- En modules pondré todo lo que se incluya en los layouts, es decir, si tengo clases dentro de un selector <header>, éstas irán en modules.

```scss
dt {
  font-size: 2.5rem;
  color: $color-principal;
  font-weight: 700;
}

dd {
  color: $color-principal;
  font-size: 1.9rem;
  margin-left: 0;
  padding-bottom: 20px;
}

.mapa {
  width: $layout-with;      You, 13 minutes ago •
}

@media screen and (max-width: 830px) {
  .portada {
    padding-top: 20px;
    display: flex;
    flex-direction: column;
    gap: 31px;
    text-align: center;
  }

  .portada img {
    max-width: 470;
    margin: 0 auto;
  }
```

- En states irán todas aquellas clases que tienen que ver con los estados, por ejemplo si un acordeón está colapsado o no. En este proyecto no se usan.
- En themes irá todo lo referente a temas, en este proyecto no se usan.

Instalación de Stylelint

1. Instalo Stylelint mediante:
   npm init
   `npm init stylelint`

2. Como en el proyecto se utiliza SASS, extiendo la configuración de lint
   `npm install --save-dev stylelint-scss stylelint-config-recommended-scss`
   `npm install --save-dev stylelint stylelint-config-standard-scss`
   +

3. Añado Sytlelint a las rutinas del Boilerplate

```
"scripts": {
  "dev": "npm-run-all clean stylelint parcel:serve",
  "build": "npm-run-all clean stylelint parcel:build",
  "stylelint": "stylelint src/**/*.scss"
```

```
  },
```

4. Configuro styleint para usarlo con SASS siguiendo https://www.npmjs.com/package/stylelint-config-sass-guidelines

```
npm install --save-dev stylelint-config-standard-scss
npm install --save-dev stylelint-config-recommended-scss
```

5. Creo el fichero .stylelintrc.json:

```
{
 "extends": [
  "stylelint-config-standard-scss",
  "stylelint-config-recommended-scss"
 ],
 "rules": {}
}
```

6. Elimino la carpeta styles del proyecto ya que se sustituye por smacss

Configuración de Git.

Ejecuto las siguientes órdenes para integrar el proyecto en Git:

```
git init
git add package.json ./src/*

git config --global user.name "Moisés Segura"
git config --global user.email "msegurace@uoc.edu"
git commit -m "initial commit"
git branch -M main
git remote add origin
https://github.com/msegurace/herr_html_css_pec1.git
git push -u origin main
git add ./src/*
git commit -m "Integración contínua"
git push -u origin main
```

1. Como dependencia externa he incluído “parcel-plugin-imagemin” (https://www.npmjs.com/package/parcel-plugin-imagemin) que optimizará el tamaño de las imágenes. Esto se hace necesario cuando, como en este caso, el número de imágenes es bastante grande ya que no hace falta ir una a una cambiándoles el tamaño.

   Para instalar el paquete:

```
npm install parcel-plugin-imagemin --save-dev
```

   El tamaño de las imágenes en la distribución ha bajado significativamente haciendo más ágil la carga del sitio.

Decisión de diseño:

He seguido una aproximación mobile first ya que normalmente es más difícil maquetar en dispositivos con pantalla pequeña.

Página responsive:

Para que la web sea responsive he creado tres breakpoints:

- 480px para los dispositivos pequeños como Smartphones.

  ```
  @media screen and (max-width: 480px) {
  ```

- Entre 481 y 830 para dispositivos tipo Tablets:

  ```
  @media screen  and (max-width: 830px) and (min-width: 481px)
  ```

- Superiores a 830 para el resto de dispositivos.

En las imágenes estoy usando Lazy loading para hacer más rápida la carga, igual en este sitio no tiene mucho sentido por ser pequeño, pero es una buena práctica utilizar esta aproximación con los componentes que no tienen que estar visibles a primera vista y su carga puede ser diferida.

He utilizado “flex” para distribuir los elementos según el tamaño de pantalla, ejemplos es la galería de fotos en el que las fotos se distribuyen según el tamaño de las mismas y de la pantalla:

```
.galeria-fotos {
  text-align: center;
  display: flex;
  flex-flow: wrap;
  flex-direction: row;
  max-width: $layout-with;
  margin: 0 auto;
  justify-content: space-around;
}

.galeria-fotos figure {
  padding-top: 20px;
}

.galeria-fotos figure figcaption {
  text-align: center;
}
```

En las imágenes utilizo “Resolution switching (size)” para cargar la imagen según el tamaño de pantalla:

```
<figure>
```

```
    <picture>
      <img srcset="./assets/images/1.jpg, ./assets/images/1g.jpg" sizes="(max-
width: 600px) 150px, 300px"
       src="./assets/images/1g.jpg" alt="Foto 1 de Las Canteras" />
    </picture>
    <figcaption>Foto 1</figcaption>
  </figure>
```