**10/12/18 (Dia-cero)**

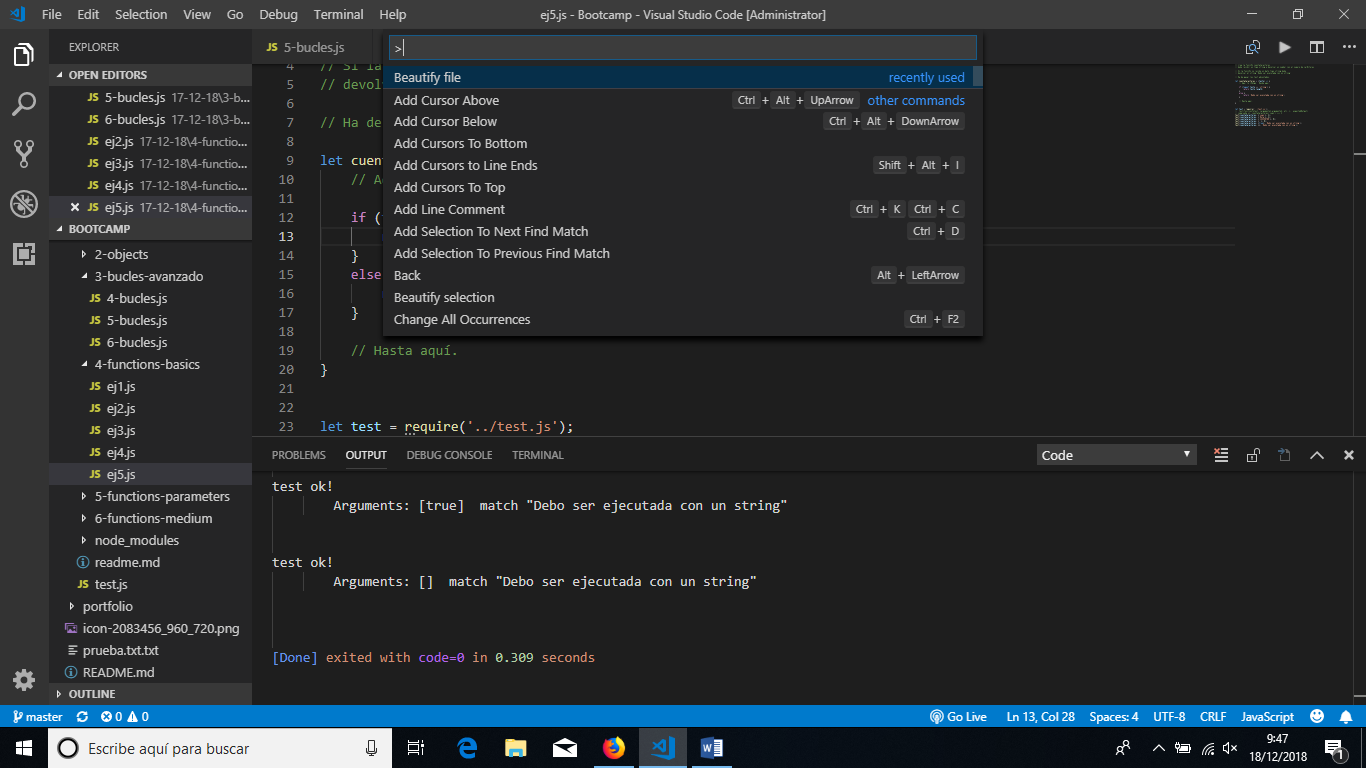
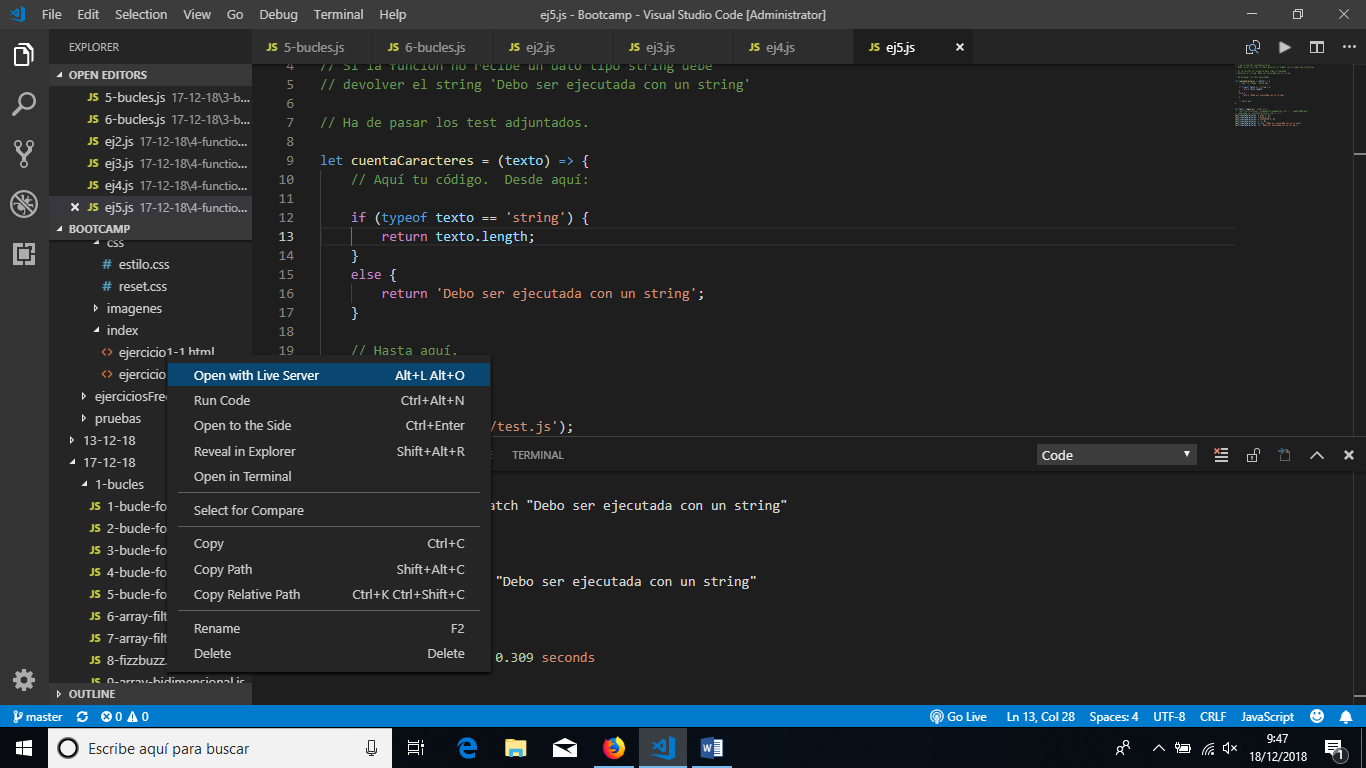
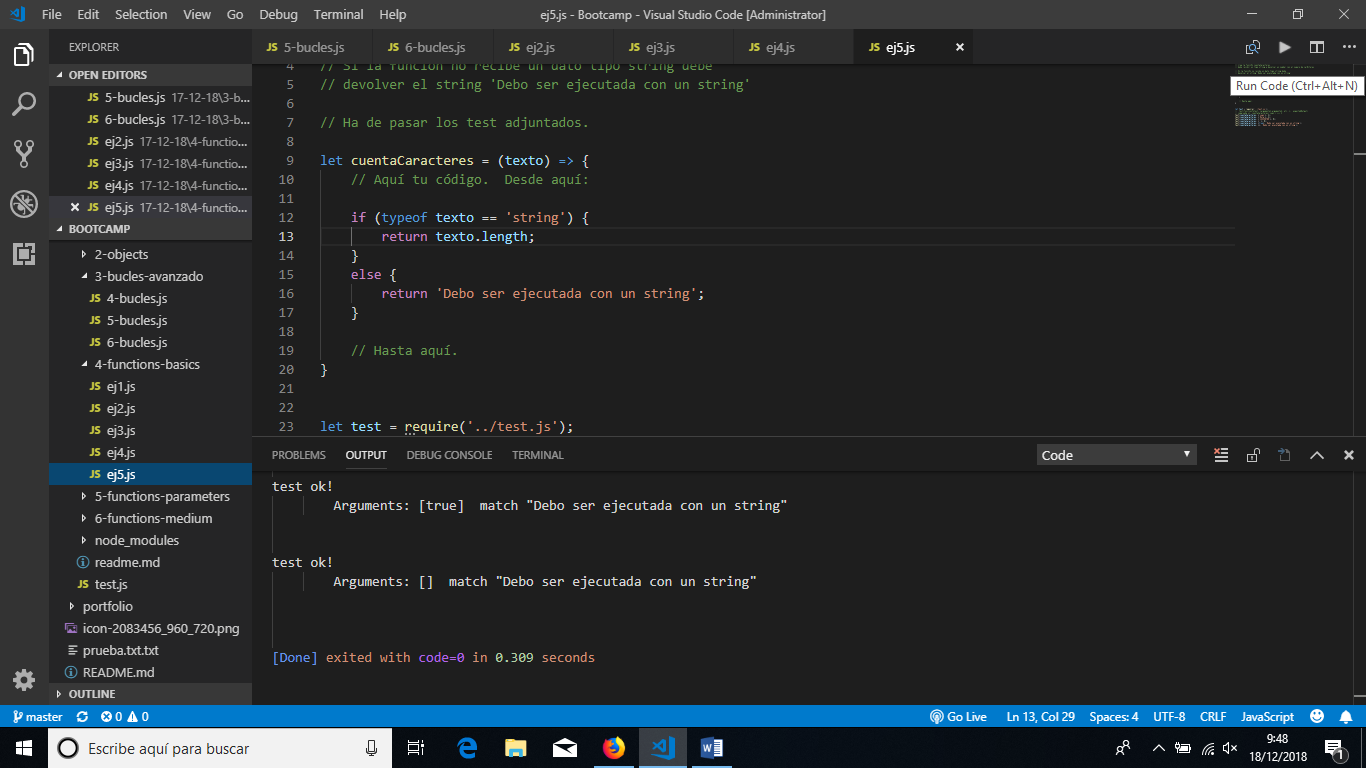
Herramientas:

* Rescue time
* Toggl

Burndown: gráfica que mide cuánto queda para terminar el proyecto.

**Descargar visual studio code.**

Para Visual studio Code:

* **Beautify (Pulsando F1 puedes tabular el texto)** 
* **Live server (permite ejecutar los html en un server)** 
* **Code runner (permite ejecutar programas como .js rápidamente)** 

Opcionales (guays pero no los vamos a usar):

* Settings sync
* Wakatime

Plataformas:

* Freecodecamp
* Enki
* Google grasshopper
* Codewars

**11/12/18 (Fundamentos-Git, Fundamentos-HTML)**

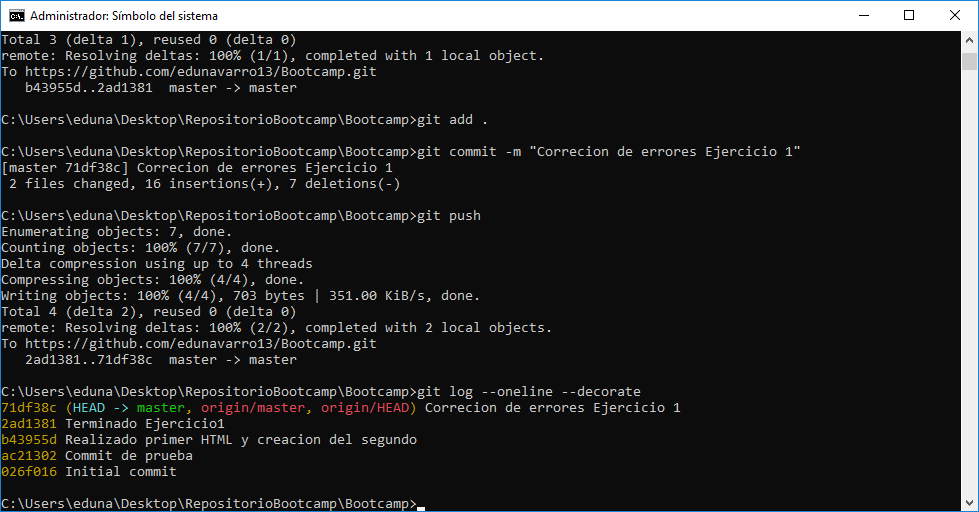
En **img**, **alt=”no carga”** para que si no carga la imagen muestre eso.

HTML5:

* Header
* Section
* Article
* Aside
* Footer

Comandos git:

* git log --oneline –decorate

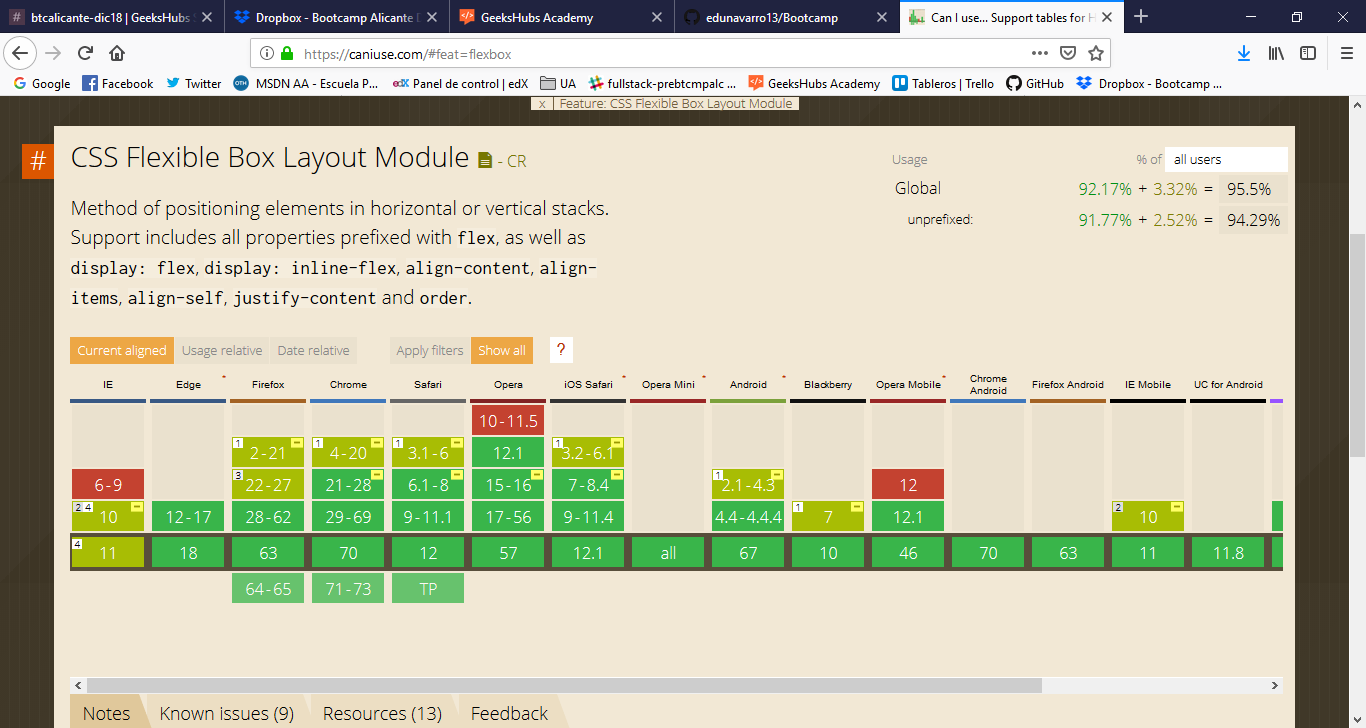


**Ejercicio 1:**

* Negrita y cursiva
* Lista ordenada y desordenada
* Link interno y externo
* Formulario (un par de input, 1 textarea (escríbeme lo que quieras) y desplegable de opciones)
* Todo lo de HTML5

**12/12/18**

Página **Caniuse** para saber estilos de css que podemos usar en determinados navegadores.



Para ver css distintos:

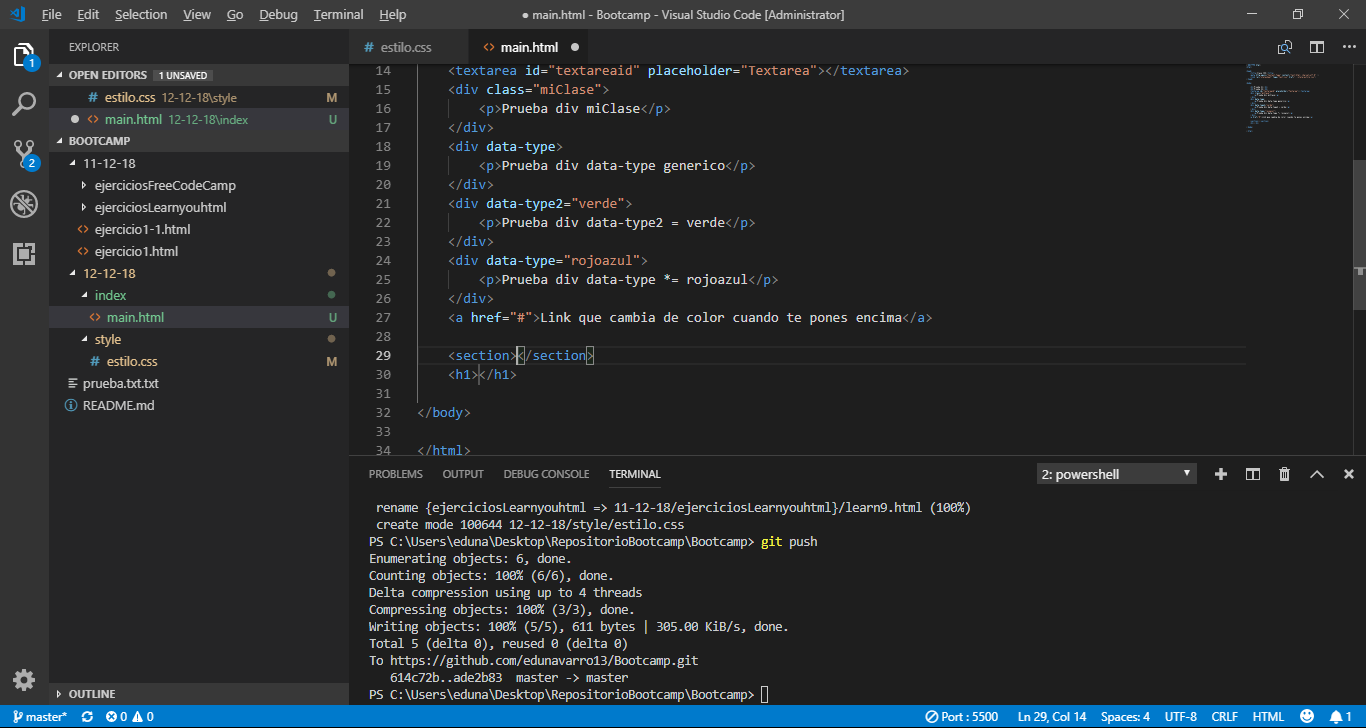
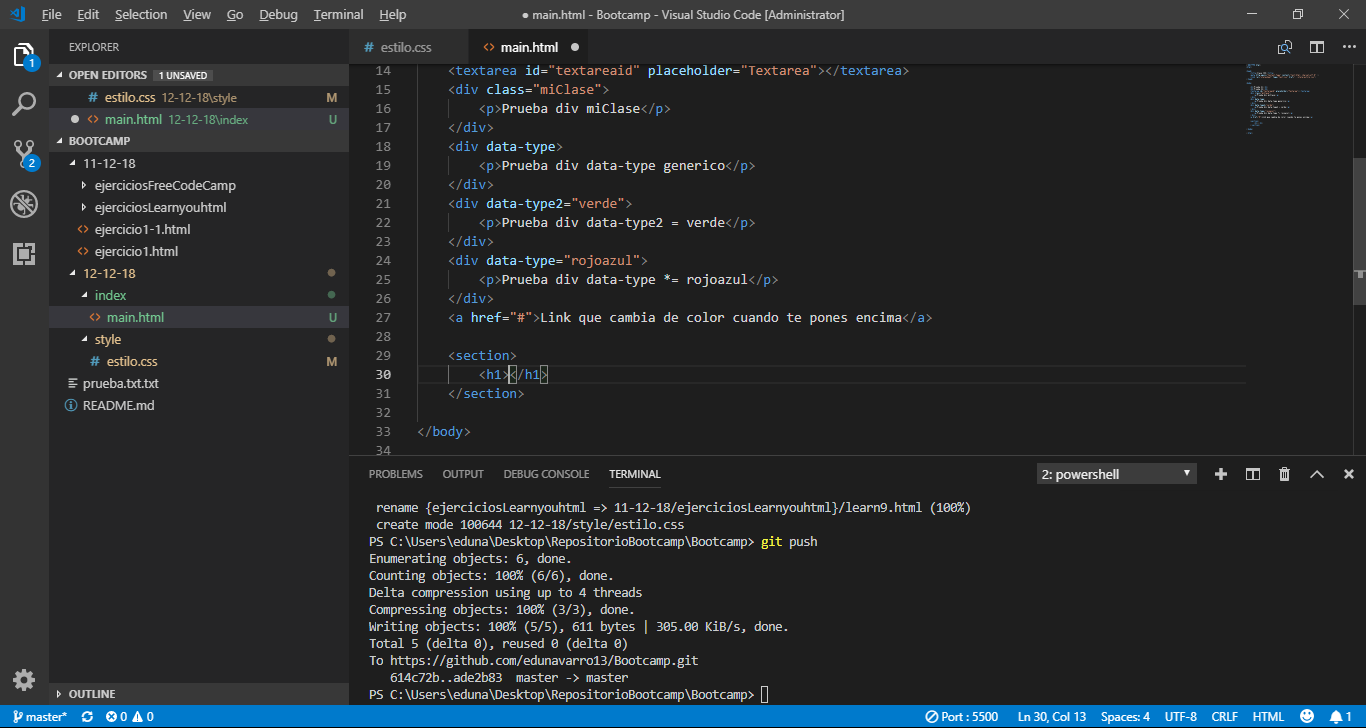
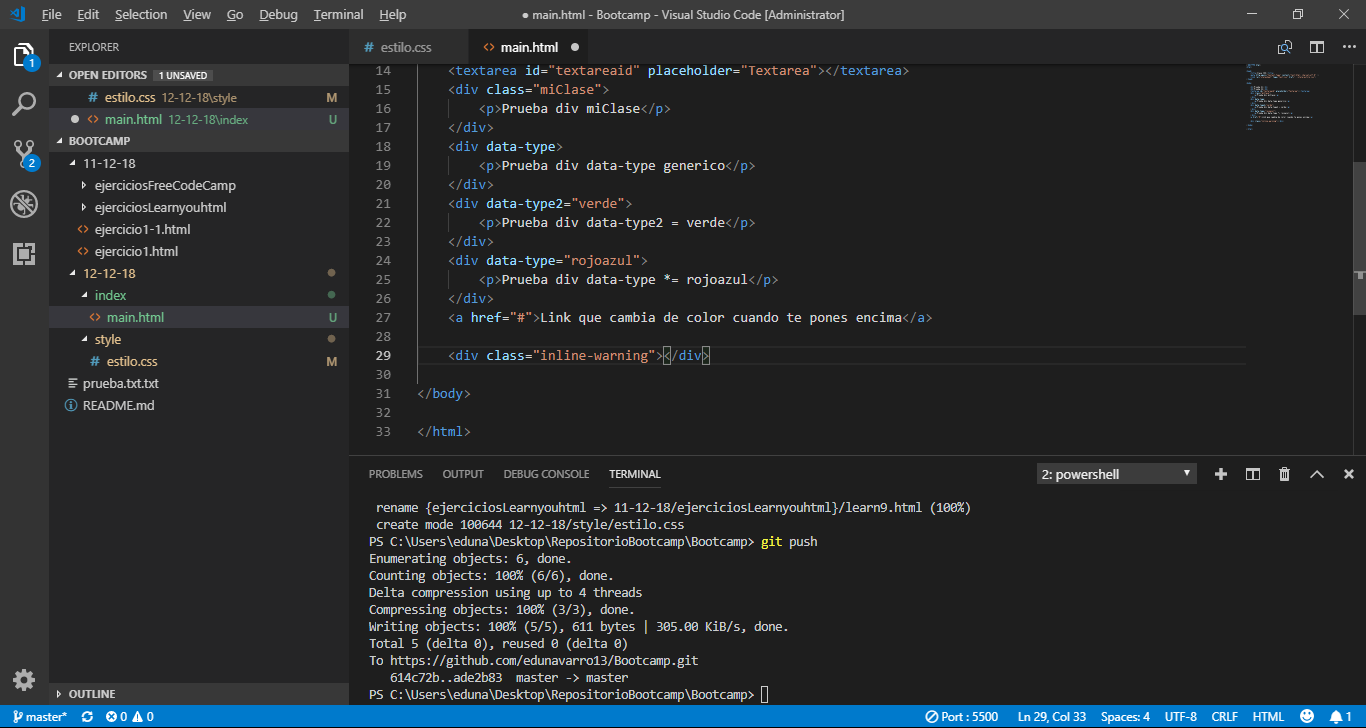
* Codepen
* Codedrops
* <https://developer.mozilla.org/es/docs/Web/CSS/Referencia_CSS>

Para convertir en comentario todo los que seleccionas en Visual Studio Code:

* Ctrl+Z Ctrl+C

Suele usarse como unidad de medida css los **vh o vw** más que los **px**, ya que es responsive, cambia según el tamaño del navegador, dispositivo, etc…

Atajos de visual studio code:

* **+** 🡪 **section+h1** igual a 
* **>** 🡪 **section>h1** igual a 
* **.** 🡪**.inline-waring** igual a

**13/12/18**

Extensión para Chrome:

* **Muzli**

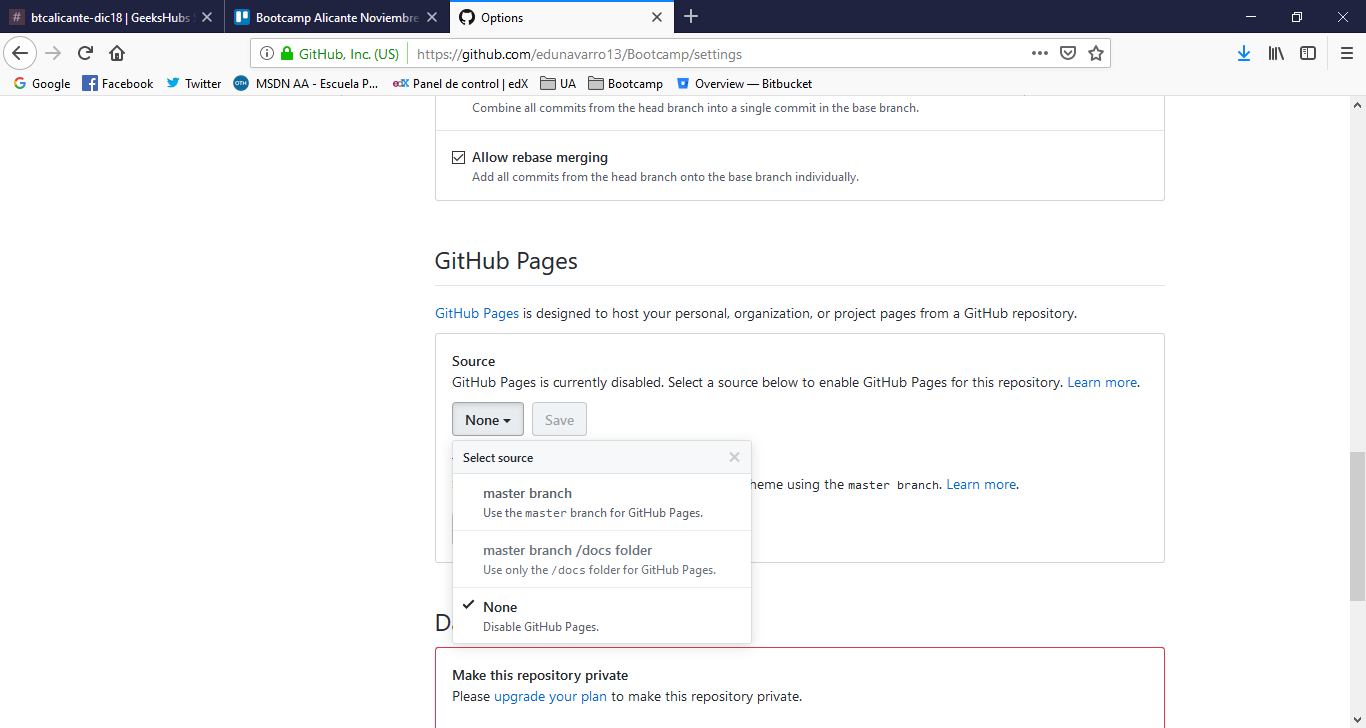
Página **awwwards** para ver las páginas con mejor formato visual CSS.

Gif con varias cosas.

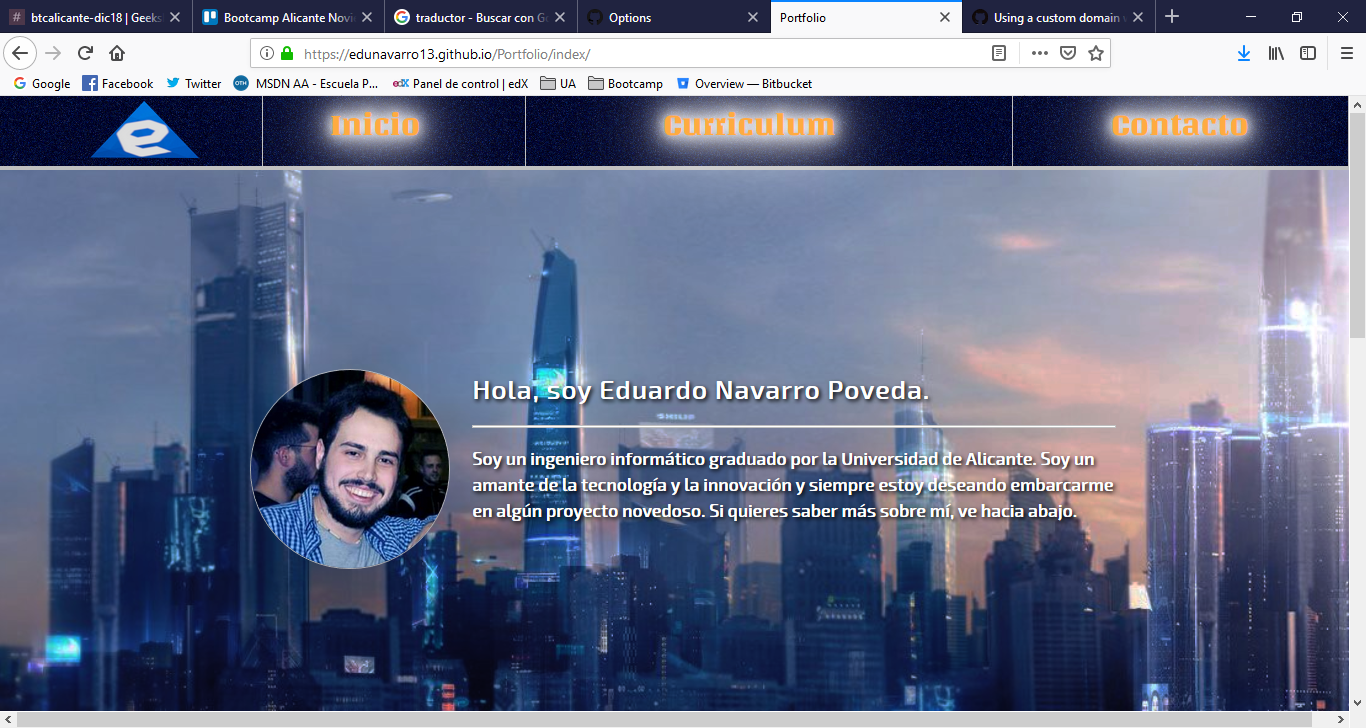
Páginas con diferentes estilos por mi diversidad de gustos.

**17/12/18**

Para Github Pages en Source cambias de None a la rama master y pulsas en Save.



Tras eso, ya estará en, por ejemplo, <https://edunavarro13.github.io/Portfolio/index/>



Javascript: Negación unaria 🡪 si tienes un string que es un número y lo quieres convertir a int, debes ponerle un + (o – si quieres que sea negativo); si lo quieres convertir a booleano, debes ponerle !! (! Si quieres que sea el negativo) :

* Ejemplo:**+”3”** 🡪 **3**
* Ejemplo 2: **-“3”** 🡪 **-3**
* Ejemplo 3:**!”3”** 🡪 **false**
* Ejemplo 4: **!!“3”** 🡪 **true**
* **If(0)** 🡪 **false** (cualquier número distinto devuelve true)
* **If(“”)** 🡪 **false** (cualquier string distinto devuelve true)

Estrictamente igual es que los datos han de ser iguales y sus tipos también (en javascript con == aunque los datos no sean del mismo tipo, puede dar igual)

* Ejemplo:**2 == ‘2’** 🡪 **true**
* Ejemplo 2: **2 === ‘2’** 🡪 **false**

Con objetos no funciona igual, aunque devuelvan lo mismo son objetos distintos (son distintos punteros, habría que usar JSON.stringify() para convertirlo en un string):

* Ejemplo:**a = {x:1}; b = {x:1}; a == b** 🡪 **false**
* Ejemplo 2: **a = {x:1}; b = {x:1}; a === b** 🡪 **false**
* Ejemplo 3: **a = {x:1}; b = {x:1}; JSON.stringify(a) === JSON.stringify(b)** 🡪 **true**

**18/12/18**

En una función si envías menos argumentos de los que hay, los rellena con undefined:

* **function log (a, b, c) {…}**
* **log(20) 🡪 a = 20; b = undefined; c = undefined**

Si pones más argumentos de los que hay, los obvia pero los guarda en arguments:

* **function log (a, b, c) {…}**
* **log(20, 5, 6, 7) 🡪 a = 20; b = 5; c = 6; arguments {‘0’: 20; ‘1’: 5; ‘2’: 6; ‘3’: 7 }**

Para ordenar con sort:

* Números con **-** 🡪 **a – b** (ordenará ascendentemente); **b – a** (ordenará descendentemente)
* Strings con **>** o **<** 🡪 **a > b** (ordenará ascendentemente según las letras); **a < b** (ordenará descendentemente según las letras);

**19/12/18**

Con ENMAScript6 aparece un nuevo tipo de comilla: la comilla **`** (la que está al lado de la P)

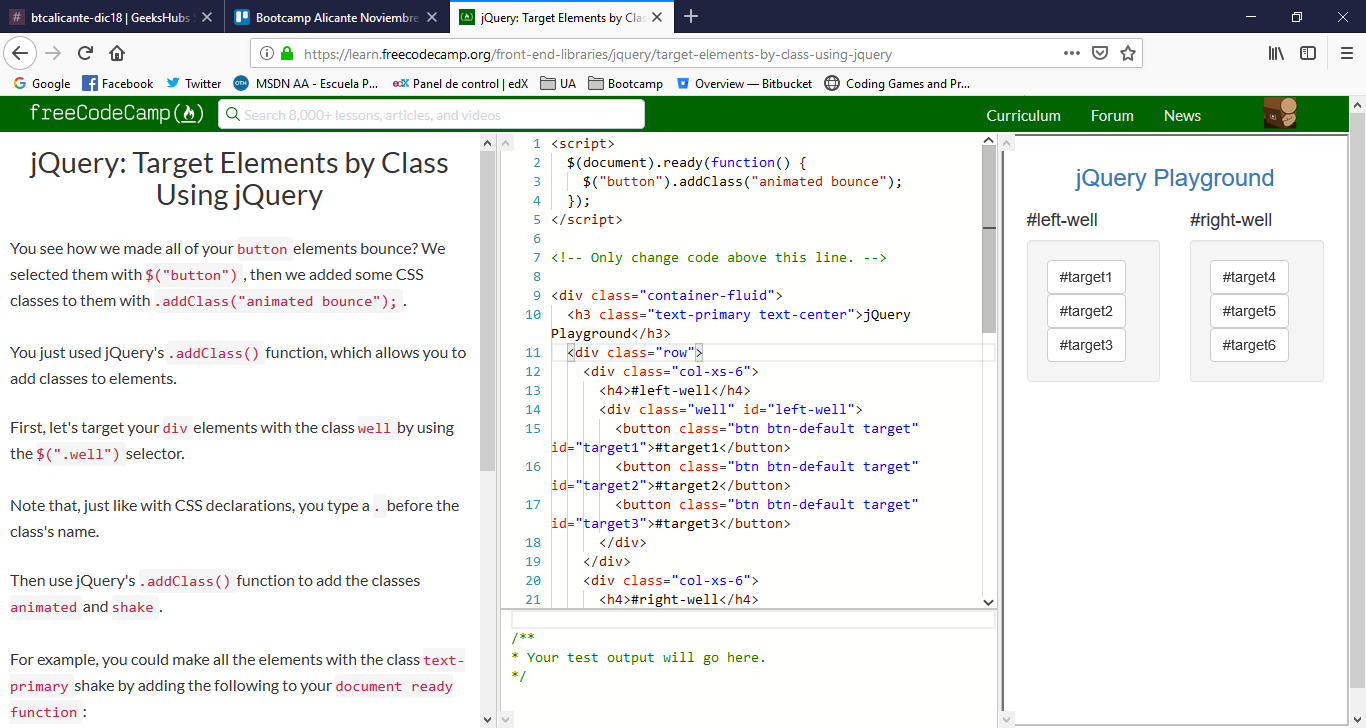
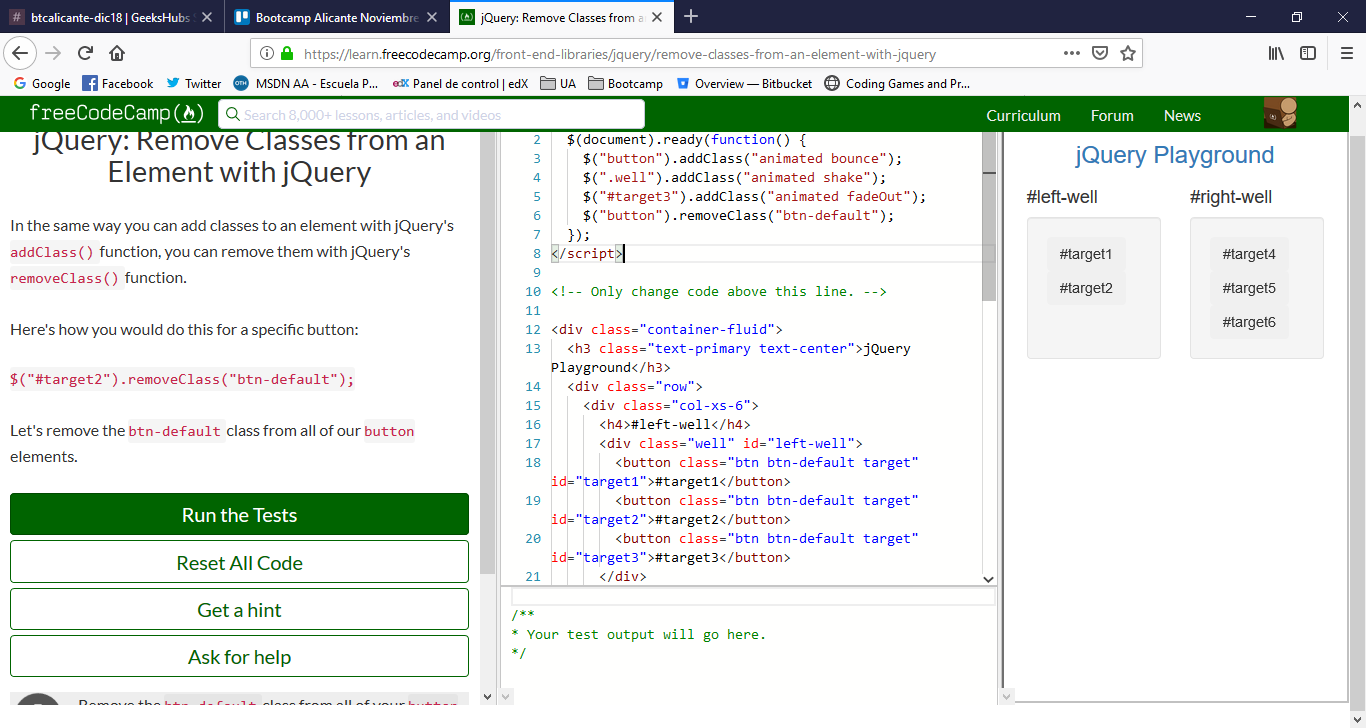
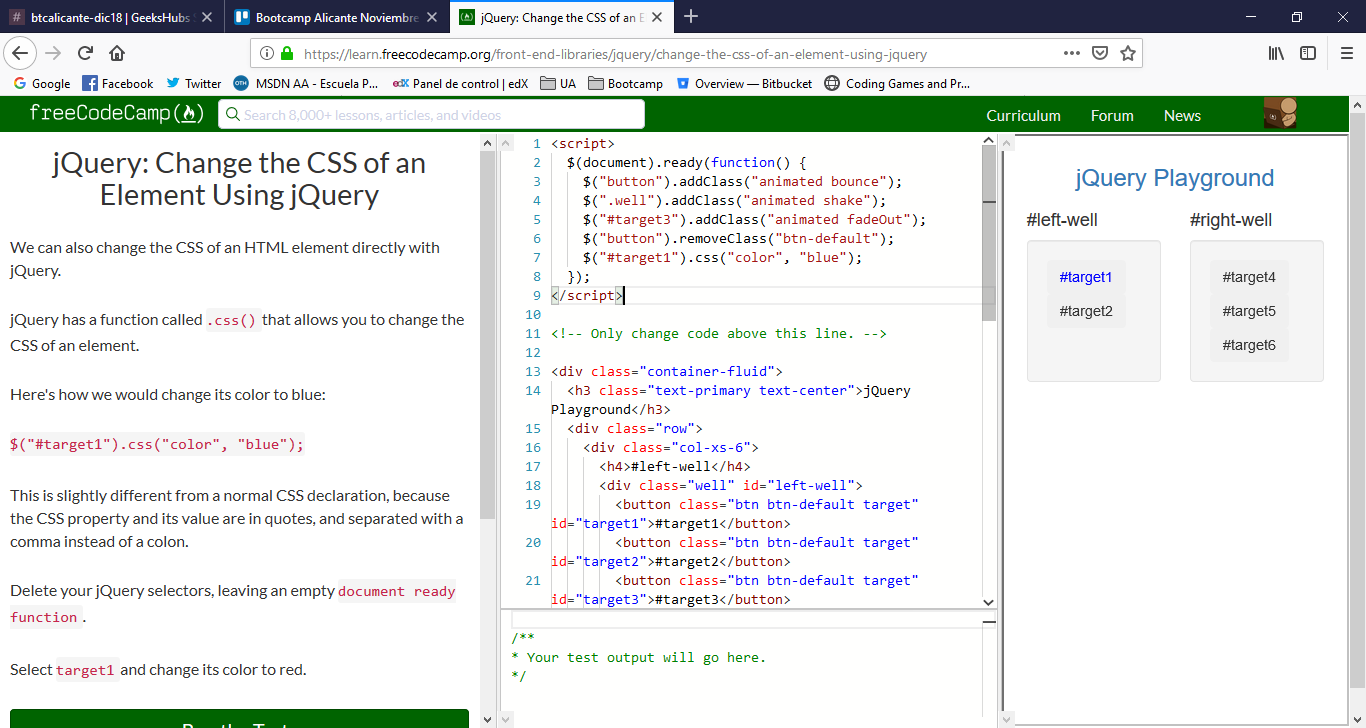
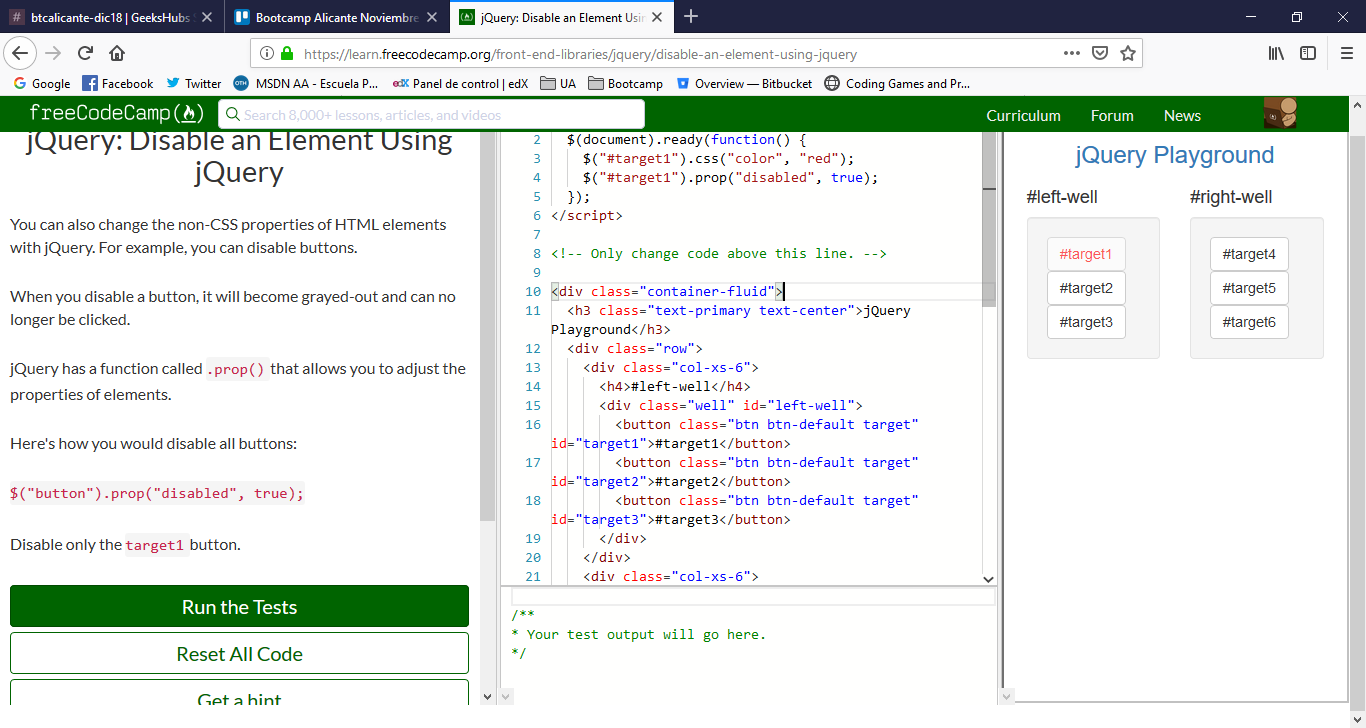
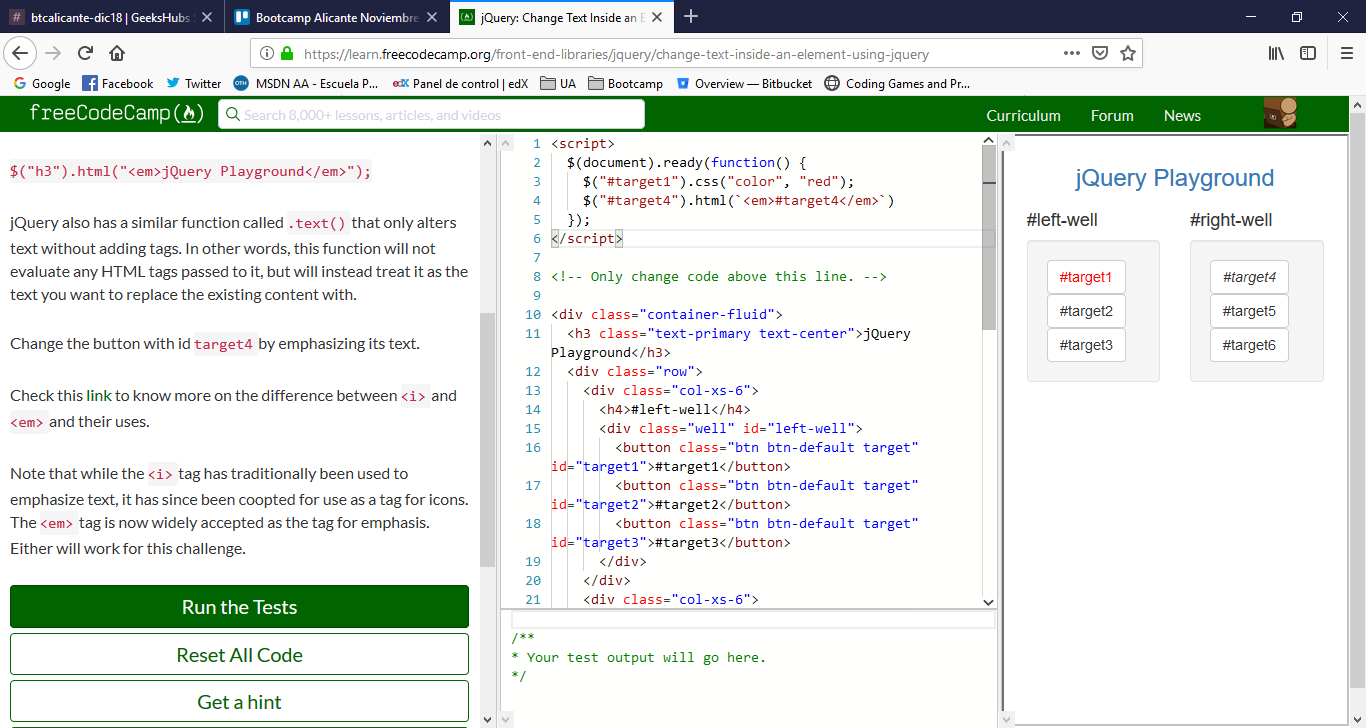
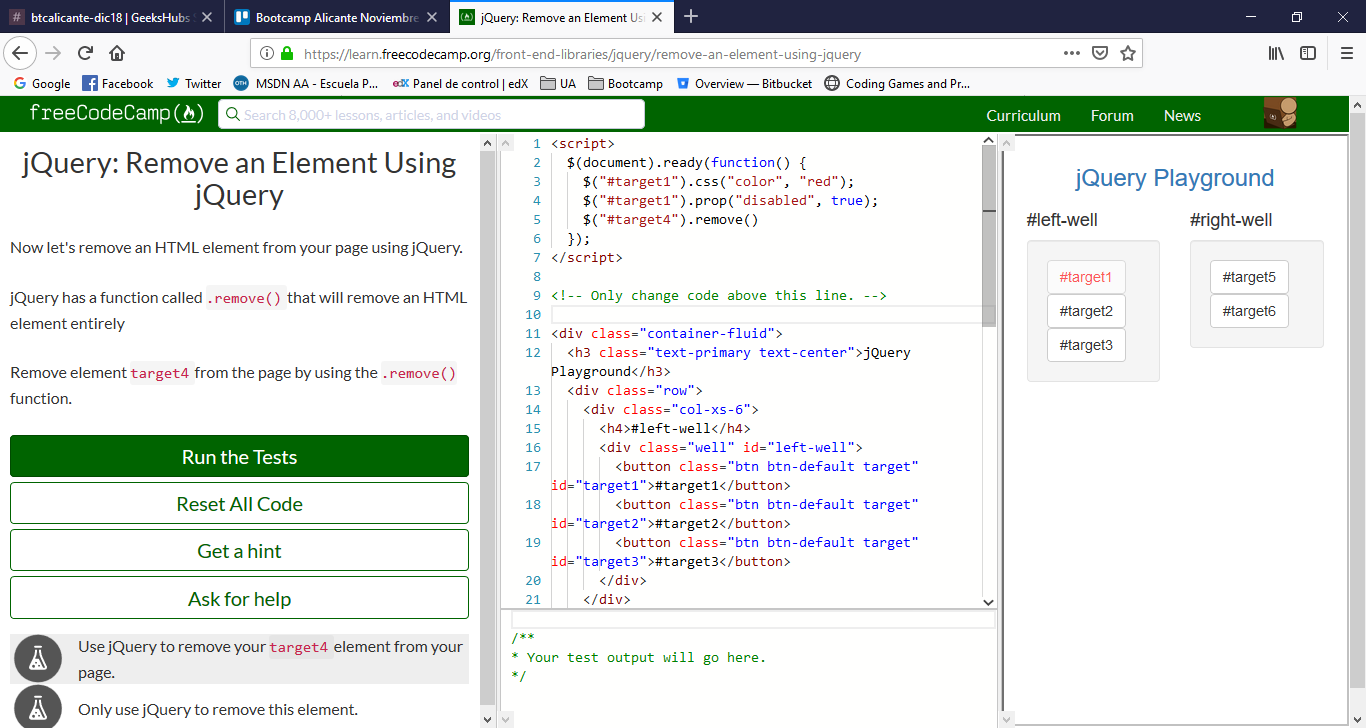
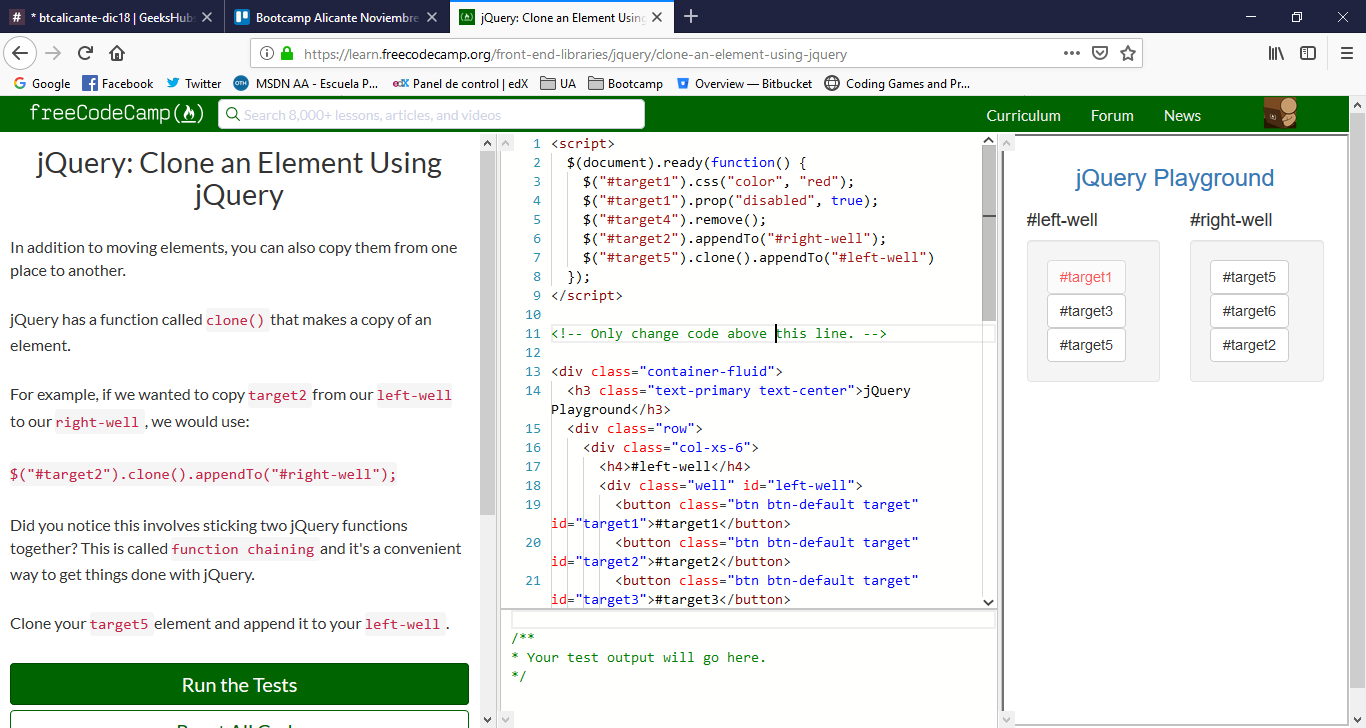
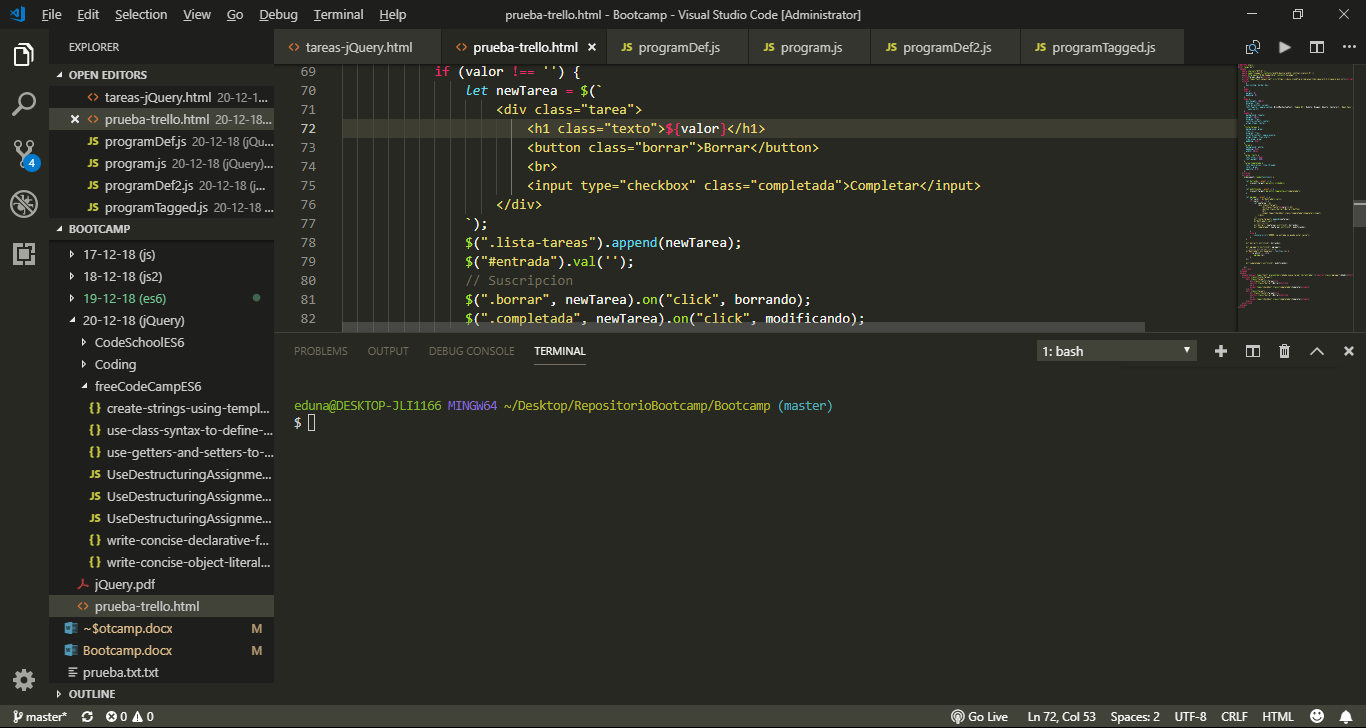
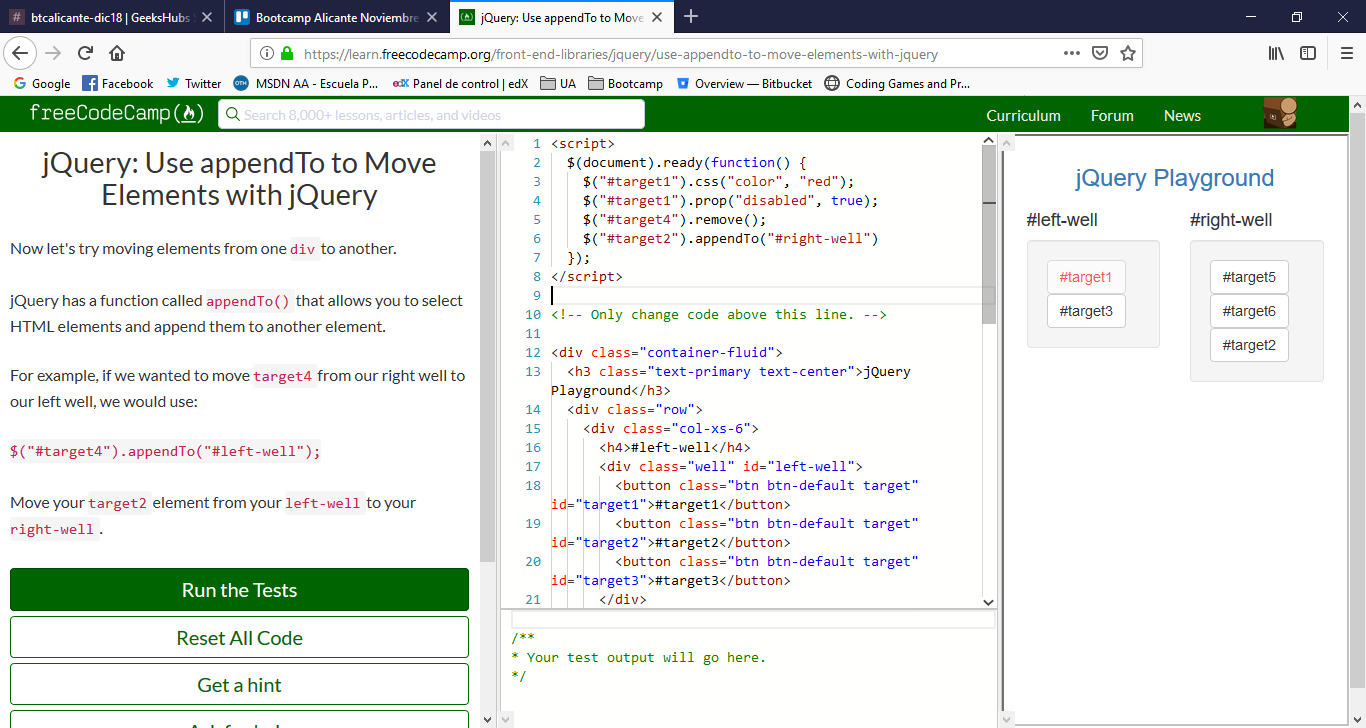
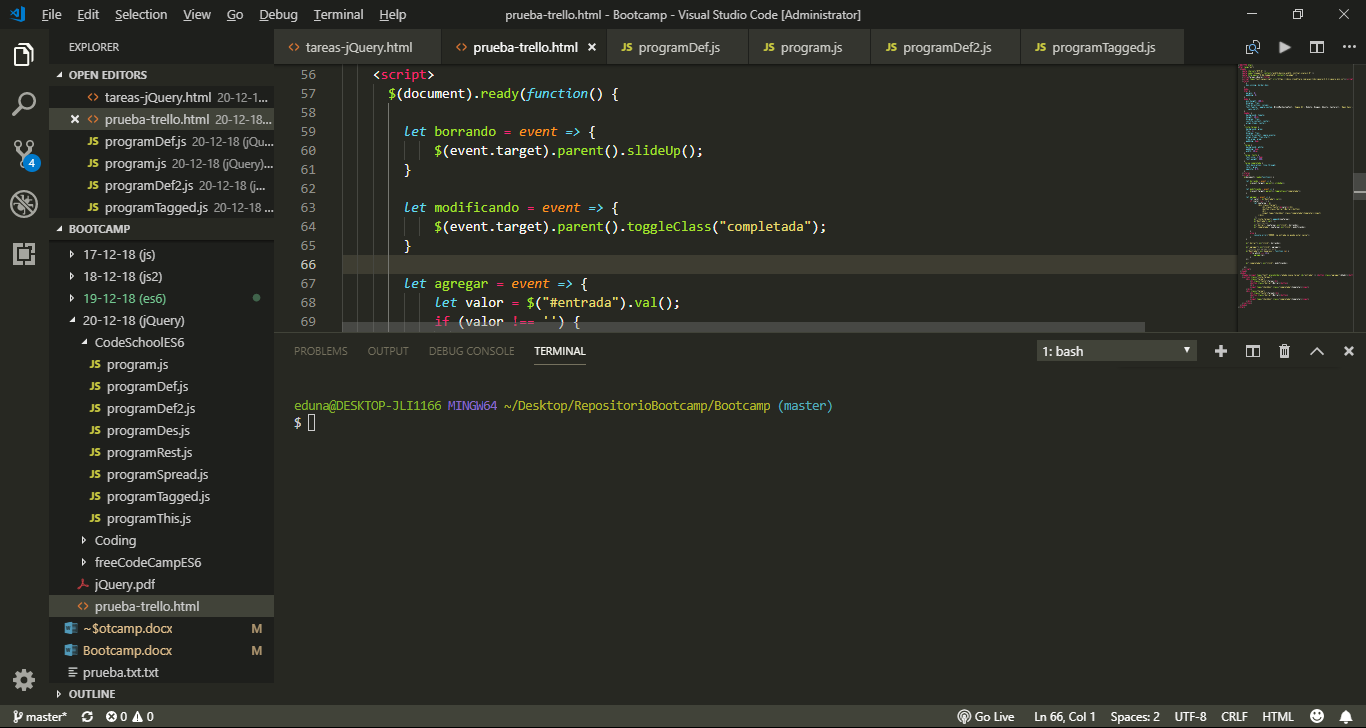
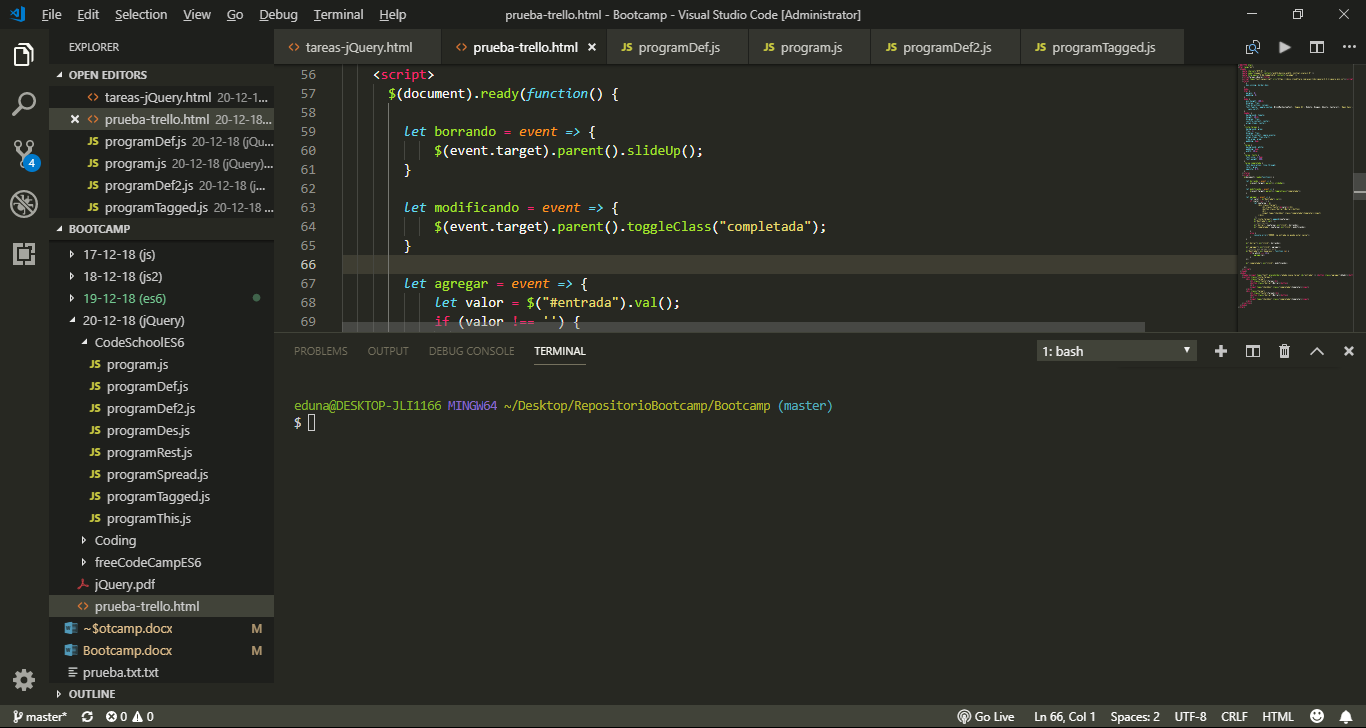
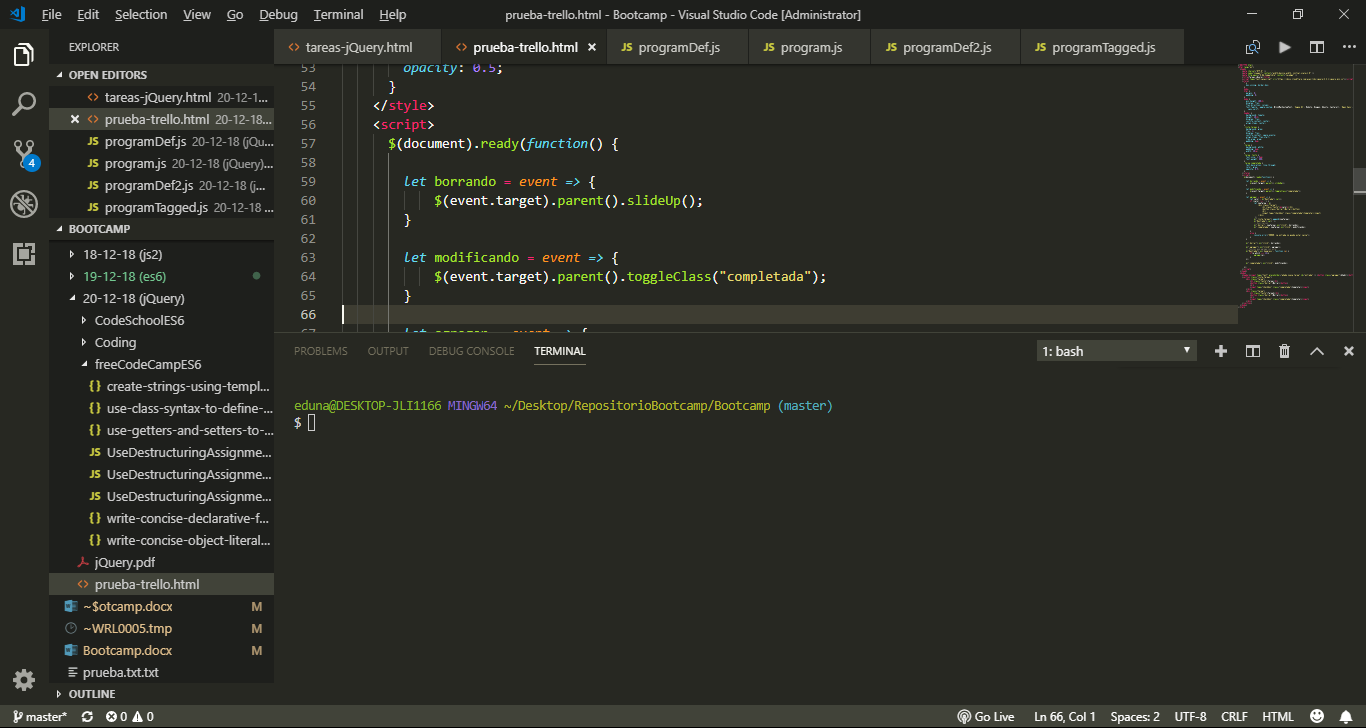
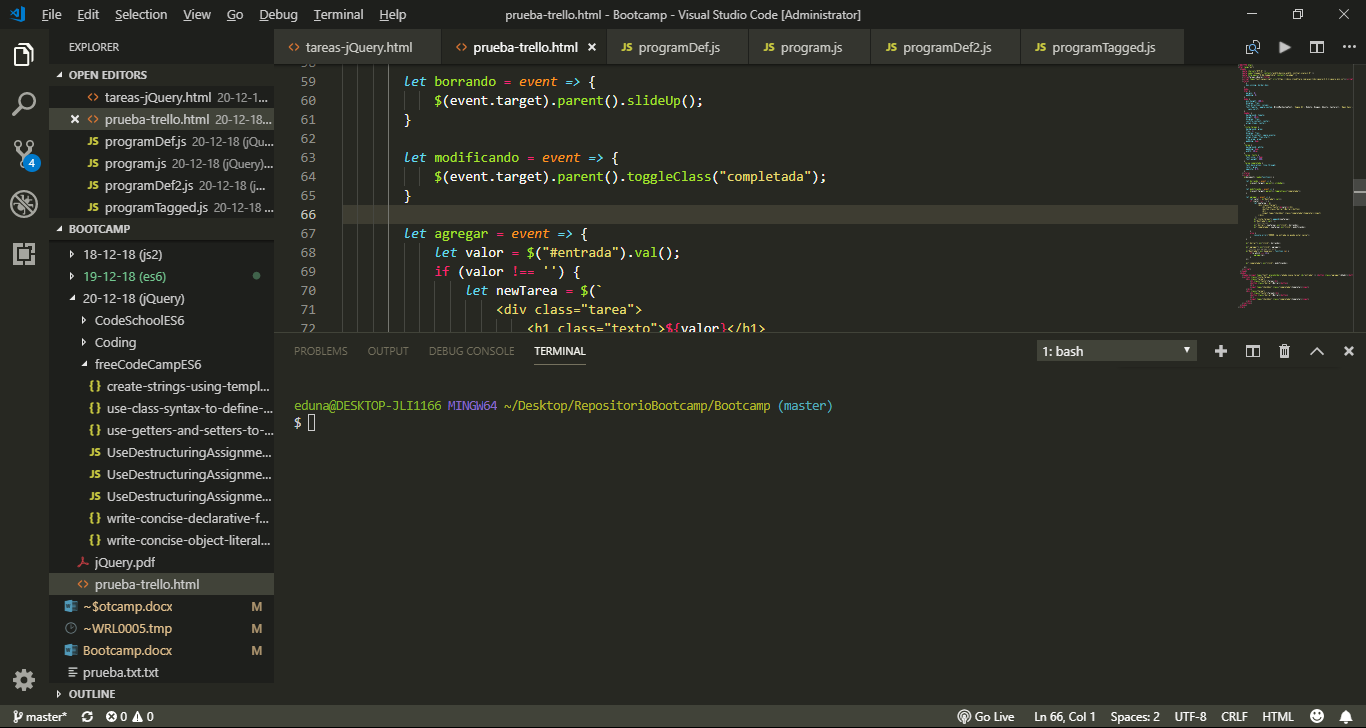
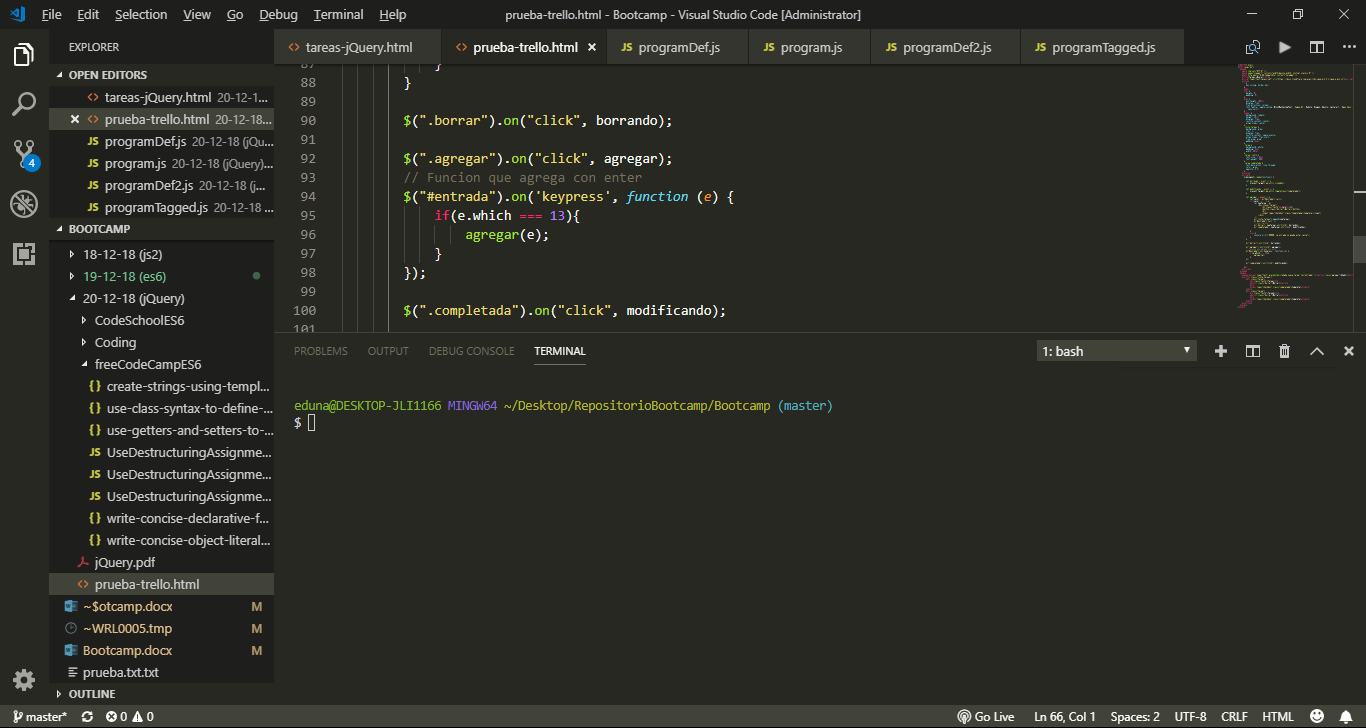
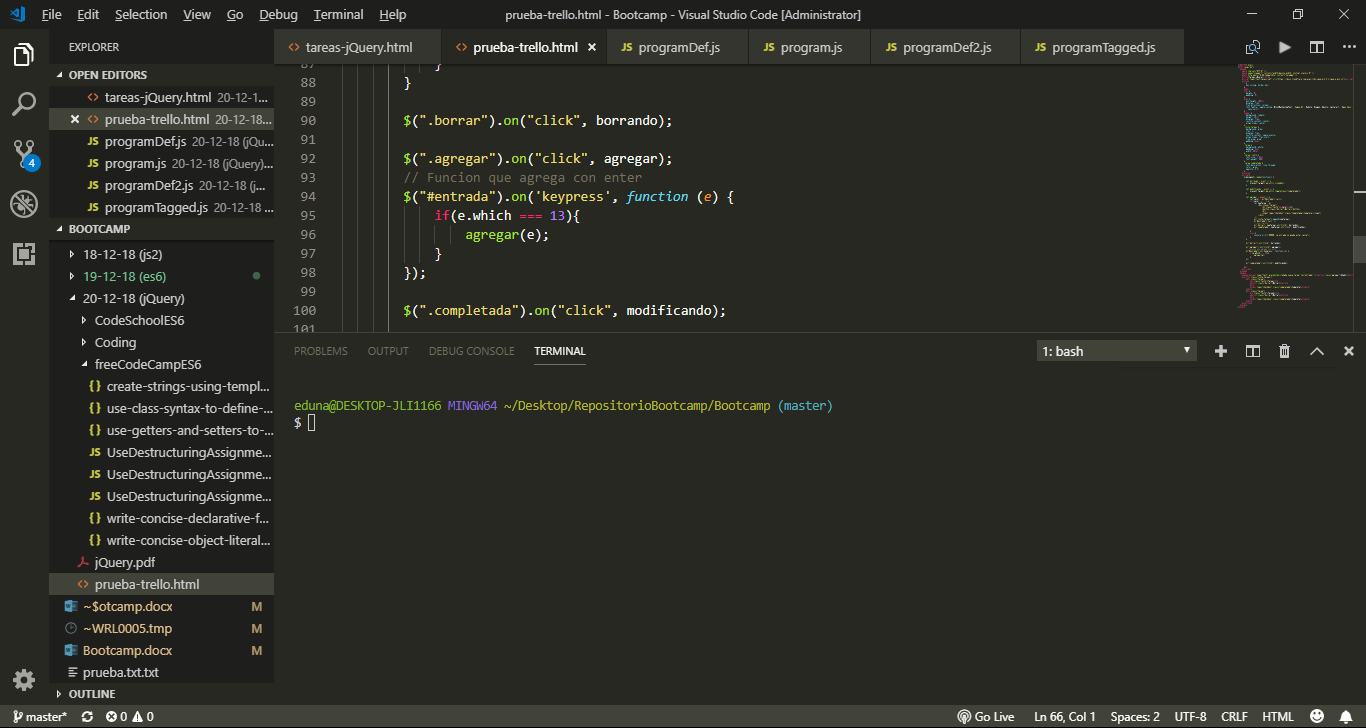
Esto permite crear una cadena multilínea dándole al **Enter** y no teniendo que poner **\n**

**20/12/18**

Para el **multicursor** (lo de poder hacer varios cambios a la vez) te has de poner encima de la variable a modificar, pulsar **Ctrl+d** para seleccionarla y los siguientes **Ctrl+d** dividirán el cursor entre las diferentes variables con ese nombre.

**21/12/18**

jQuery ejemplos:

* **addClass(atr)** 🡪 agrega al elemento la clase **atr**
* **removeClass(atr)** 🡪 borra al elemento la clase **atr**
* **css(atr, atr2)** 🡪 modifica el css del elemento
* **prop(atr, atr2)** 🡪 modifica propiedades del elemento que no son css
* **html(atr)** 🡪 cambia el html almacenado en el elemento
* **remove()** 🡪 elimina el elemento del html
* **clone()** 🡪 copia el elemento en donde lo une con append
* **append(elem) ||appendTo(elem)** 🡪 une el elemento al **elem** pasado por parámetro, mientras que **append** es al revés (le une **elem** al final del elemento) 
* **toggleClass(atr)** 🡪 busca en el elemento la clase pasada en **atr** y, si no está la crea y, si está, la borra.
* **parent() || children()** 🡪 afecta al nodo padre del elemento (children al hijo)
* **slideUp()** 🡪 hace que elemento vaya desapareciendo hacia arriba en segundos hasta desaparecer por completo
* **val()** 🡪 obtiene el value del elemento
* **on(“click”, atr)** 🡪 evento que ejecuta cuando hacemos click en el elemento
* **on(“keypress”, atr)** 🡪 evento que ejecuta cuando pulsamos una tecla en el elemento

**07/01/19**

Para instalar typescript:

* **npm i -g typescript**

Se ejecuta **tsc** para compilar el código typescript

Typescript:

1. x: number
2. x :number
3. x:number

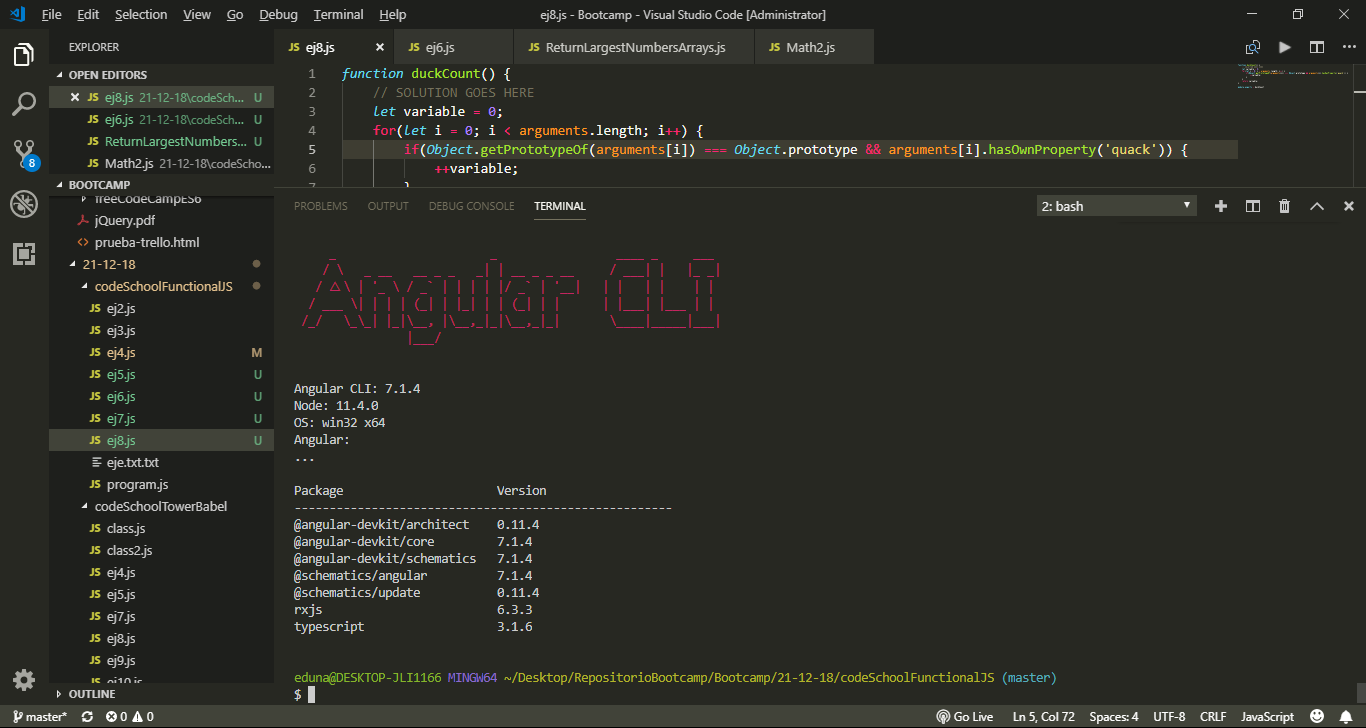
Cualquiera de las 3 funcionan pero se suele usar la primera.

Para instalar angular:

* **npm install -g @angular/cli**

Para ver la versión de angular instalada:

* **ng version**

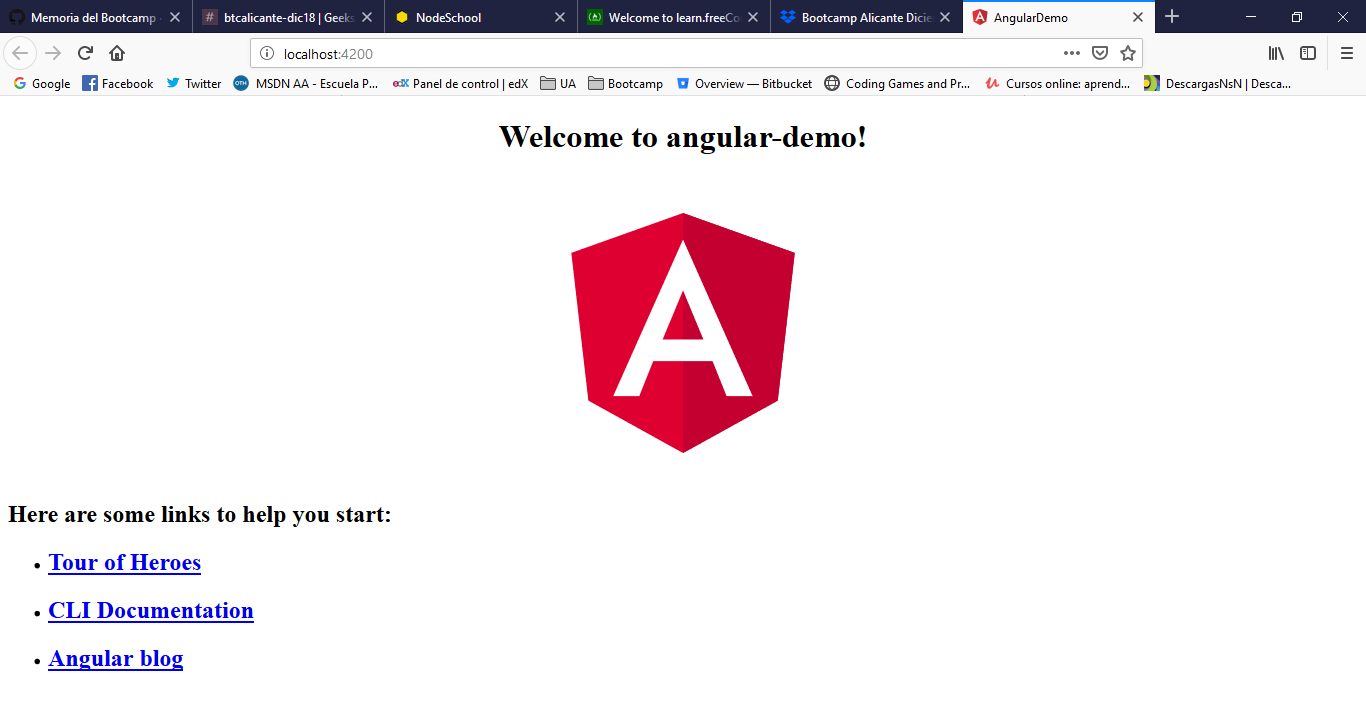


Para crear un proyecto desde el principio:

* **ng new angular-demo**

Para ejecutar el proyecto, vas a la carpeta y ejecutas:

* **ng serve --open**



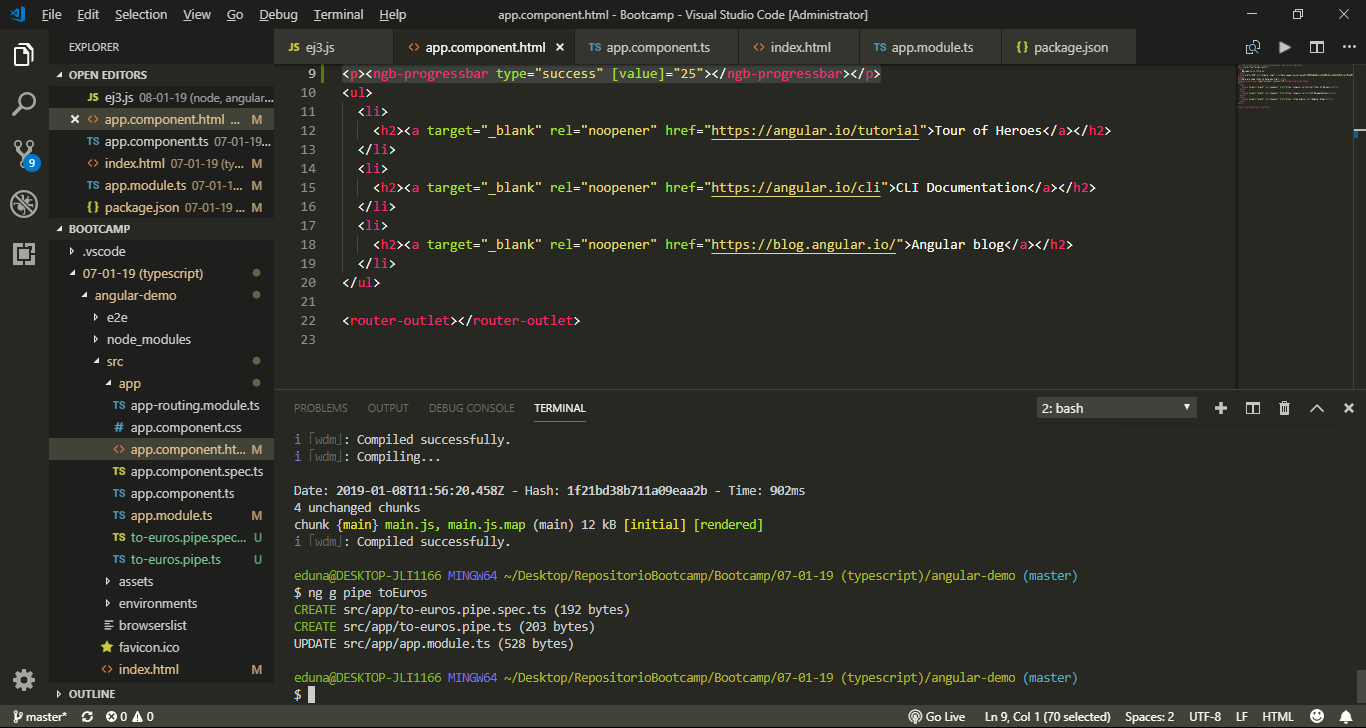
Para obtener los archivos distribuibles se ejecuta:

* **ng build**

**08/01/19**

Para crear un pipe (función que modifica un valor, similar a los functions)

* **ng g pipe toEuros**



Crea esos 2 elementos y modifica el módulo para añadirlo.

No existe el **for** normal en angular, solo el **for of.**

**09/01/19**

Para crear un componente ejecutamos:

* **ng g component headerAngular**

