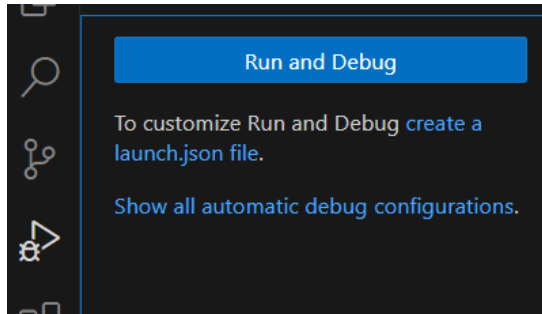


DANIEL CARRIL MIRANDA __ DWEC __ TEMA3 __ EXAMEN

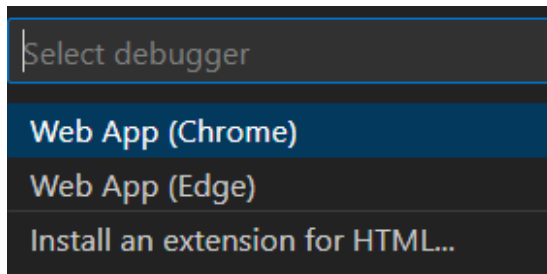
Ahora voy a hacer un ejemplo de debuggeo.

Para empezar, tenemos que estar dentro de Visual Studio Code con nuestro proyecto abierto.

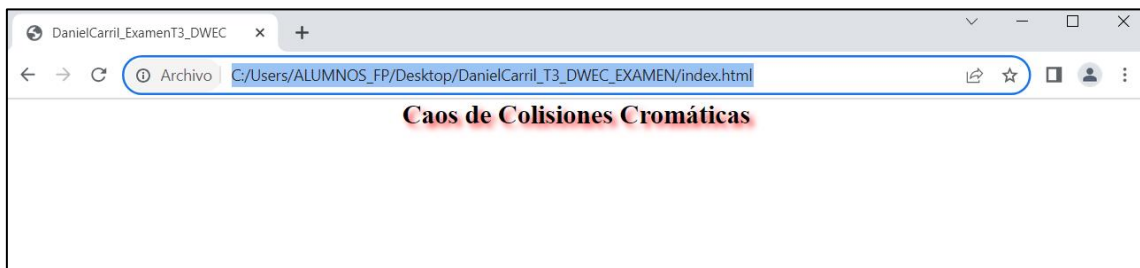
Nos vamos a la pestaña de “Run and Debug” y clicamos en el botón grande azul.



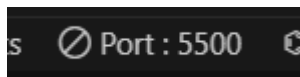
Y seleccionamos Chrome que es nuestro navegador de Google



Se nos abre así y no se ve nada más



Se nos crea un archivo .json en el que tenemos que cambiar el puerto 8080 a 5500, coincidiendo con el puerto local en el que VSCode despliega por defecto los proyectos:



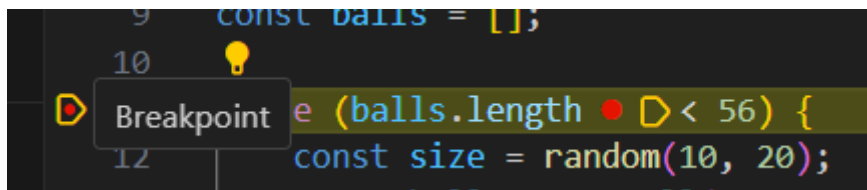
```
index.js launch.json U style.css class.js canvas.js
.vscode > launch.json > [ ] configurations > { } 0 > url
1 {
2     // Use IntelliSense to learn about possible attributes.
3     // Hover to view descriptions of existing attributes.
4     // For more information, visit: https://go.microsoft.com/fwlink/?
5     "version": "0.2.0",
6     "configurations": [
7         {
8             "type": "chrome",
9             "request": "launch",
10            "name": "Launch Chrome against localhost",
11            "url": "http://localhost:5500",
12            "webRoot": "${workspaceFolder}"
13        }
14    ]
15 }
```

Después de guardar este archivo modificado, ya podemos empezar a debuguear y comprobar cómo va funcionando el programa.

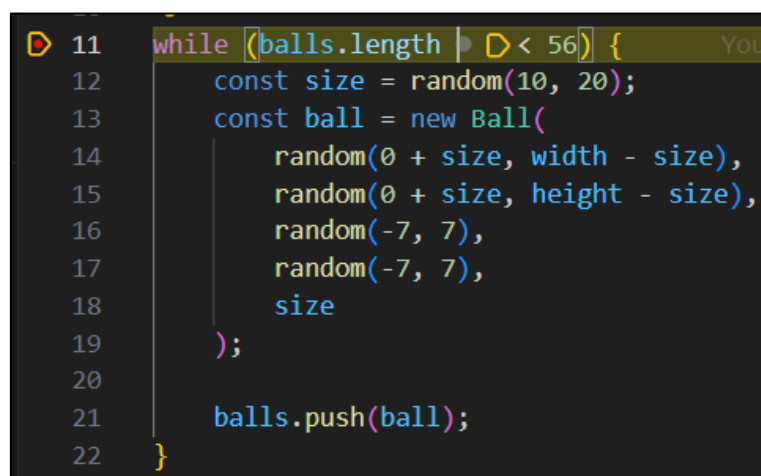
Nos aparece en la esquina de arriba a la derecha la barra de acciones:



Lo primero que hacemos es colocar un Breakpoint en cualquier lugar del programa para que comience a debuguearlo a partir de ahí:




Vamos a hacer el ejercicio 3 poniendo el breakpoint en el bucle donde comienza a generar las bolas. Entonces lo pongo justo aquí:




En la pestaña de la izquierda podemos ir viendo los objetos que hay, las variables y sus valores. En este primer momento no hay bolas creadas:

```
✓ VARIABLES
  ✓ Module
    > Ball: class Ball {
    > balls: (0) []
    > ctx: CanvasRenderingContext2D {ca...
      height: 714
    > loop: f loop() {
    > random: f random(min, max) {
      this: undefined
      width: 1039
    > Global
  ✓ WATCH
```



Si nosotros pulsáramos el botón de  , iría dando grandes saltos en el código, saltando los procesos intermedios y recorrería sólo las funciones principales, con lo que se vería



rápidamente cómo se crean las bolas. Si le damos a  saltaría todos los pasos y no mostraría ninguno hasta generar la nueva bola, es decir, por cada vez que le demos genera una nueva bola:


```
✓ VARIABLES
  ✓ Module
    > Ball: class Ball {
    > balls: (1) [Ball]
    > ctx: CanvasRenderingContext2D...
      height: 738
    > loop: f loop() {
    > random: f random(min, max) {
      this: undefined
      width: 768
    > Global
  ✓ WATCH

scripts > Js index.js > ...
3 import { width } from './canvas.js';
4 import { height } from './canvas.js';
5 import { Ball } from './class.js';
6 import { random } from './canvas.js';
7 import { ctx } from './canvas.js';
8
9 const balls = [];
10
11 while (balls.length < 56) {
12   const size = random(10, 20);
13   const ball = new Ball(
14     random(0 + size, width - size),
15     random(0 + size, height - size),
16     random(-7, 7),
17     random(-7, 7),
18     size
19   );
20
21   balls.push(ball);
22 }
```

Vamos a ver las propiedades de esta bola desplegando:

```
▼ balls: (1) [Ball]
  ▼ 0: Ball {x: 597, y: 658, velX: -1, velY: 6, color: 'rgb(202,71,196)', ...}
    color: 'rgb(202,71,196)'
    size: 10
    velX: -1
    velY: 6
    x: 597
    y: 658
  > [[Prototype]]: Object
```



Para crear las siguientes bolas voy a darle a  para ver exactamente todos los procesos y funciones que se recorren a la hora de crear la bola. Esto recorre todos los archivos .js necesarios y muestra cómo utiliza las funciones como `random()` que son necesarias en ese bucle.

Ahí se ven las primeras bolas creadas y los valores de sus atributos. Son diferentes porque utilizan valores randoms.

```
▼ balls: (3) [Ball, Ball, Ball]
  > 0: Ball {x: 597, y: 658, velX: -1, velY: 6, color: 'rgb(202,71,196)', ...}
  > 1: Ball {x: 34, y: 355, velX: 7, velY: 0, color: 'rgb(129,88,55)', ...}
  > 2: Ball {x: 546, y: 562, velX: -2, velY: 3, color: 'rgb(57,67,180)', ...}
    length: 3
  > [[Prototype]]: Array(0)
  > [[Prototype]]: Object
```

Para ver todos los commits creados en el repositorio local de git usamos el comando **git log –online** (salir con q)