APUNTES WEB

HTML

Filling the <body> using "blocks":

- **<header> </header>**: Provides a header (intro) inside the body.
- **<footer> </footer>**: Provides a footer at the end of the document.
- ** **: Defines a division or a section in a document. It is used to group block elements so as to be able to format them with CSS.
- **<canvas> </canvas>**: Draws graphics in a document. It is only a container for graphics. JavaScript must be used to actually draw the graphics through the API provided by canvas.
- ** **: Defines a paragraph. Browsers add margins automatically, but they can be modified with CSS.

Some interesting inline elements:

- ** **: It is similar to but it groups inline elements instead. It lets refer to either a part of a text, a paragraph or a document.
- ****: Creates a holding space for the referenced image (through the **required** "src" attribute) in a document. The image is actually linked, not inserted.
- **<a> **: Creates a link to a resource, a location in the own document, or any other URL.

- *Escribir un título y un párrafo*

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

*
 haría lo mismo que /n dentro de un párrafo

*<h1> es el primer título, para subtítulos h2, h3, h4…

* si pones <hr> entre dos párrafos crea una línea separatoria

- *Insertar imagen*

```
<img src="url" width="500" height="600" alt="alternatetext">
```

- *Dentro de un párrafo puedes escribir esto para poner texto en negrita, cursiva, etc*

Formatting elements were designed to display special types of text:

- <b> - Bold text
- - Important text
- <i> - Italic text
- - Emphasized text
- <mark> - Marked text
- <small> - Smaller text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

- *Crear un link (https://www.w3schools.com/html/html_links.asp)*

```
<a href="url">link text</a>
```

Como cambiar el color del link dependiendo de si has hecho clic, estás encima, etc:
https://www.w3schools.com/html/html_links_colors.asp

- *Listas https://www.w3schools.com/html/html_lists.asp*

Con puntos (no ordenadas)

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Con números (ordenadas)

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

- *Si quieres crear un elemento bloque: DIV.*
  Si quieres crear un elemento en línea: SPAN

- *Insertar código Javascript* `<script>`
- *Animaciones(HTML)*

```
<!DOCTYPE html>
<html lang="en">
    <head>
        ...
        <link rel="stylesheet" href="css/animate.css">
    </head>
    <body>
        ...
        <img src="images/boat.png" alt="boat" id="boat">
    </body>
</html>
```

- *Animaciones(CSS)*

```
/* Animation code */
@keyframes navigate {
    0%  {left: 0%; transform:rotate(0deg); background-color: red;}
    20% {left: 30%; transform:rotate(30deg); background-color: yellow;}
    40% {left: 60%; transform:rotate(60deg); background-color: green;}
    50% {left: 80%; transform:rotate(90deg); background-color: darkblue;}
    60% {left: 60%; transform:rotate(60deg); background-color: green;}
    80% {left: 30%; transform:rotate(30deg); background-color: yellow;}
    100% {left: 0%; transform:rotate(0deg); background-color: red;}
}

#boat {
    margin-top: 5em;
    margin-left: 3em;
    width: 120px;
    height: 120px;
    position: absolute;
    animation-name: navigate;
    animation-delay: 2s;
    animation-duration: 6s;
    animation-iteration-count: 3;
    animation-timing-function: linear;
}
```

- *Inputs https://www.w3schools.com/html/html_form_input_types.asp*

Los más importantes:

Radio buttons:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_radio

```
<input type="radio" id="male" name="gender" value="male">
```

Range (un slider):

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_range

```
<input type="range" id="vol" name="vol" min="0" max="50">
```

Botón

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_input_button

```
<button type="button" onclick="alert('Hello world!')">Click Me!</button>
```

Formas de cambiar un estilo:

1. *Directamente en el párrafo, div, etc (Inline)*

```
...
<p style="font-size: 20px">My paragraph</p>
...
```

2. *En el head (embedded)*

```
<!DOCTYPE html>
<html>
<head>
    ...
    <style type="text/css">
        p { font-size: 20px }
    </style>
    ...
</head>
```

3. *En un archivo externo (la mejor)*

```
<link rel="stylesheet" href="css/css_filename1.css">
```

Tipos de referencias

1. ***Id**. Sólo se debe usar UNA VEZ en el mismo documento*
2. ***Class**. Se puede usar varias veces*

```
<body>
    ...
    <p id="text1">My text</p>
    ...
</body>
```

Un mismo elemento puede pertenecer a la vez a una clase y a un id.

```
<div id="gamestartscreen" class="gamelayer">
  <img src="images/play.png" alt="Play Game" onclick="myGame.showCanvas()">
</div>
<div id="endingscreen" class="gamelayer">
  <img src="images/playagainbutton.png" alt="Play Again" onclick="myGame.restart()">
</div>
```

CSS

https://www.w3schools.com/css/css_intro.asp

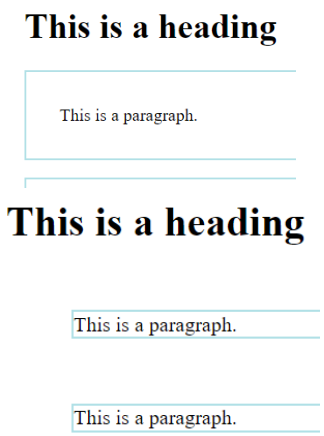
Tipos de referencias

Id. *Con un #*
Class. *Con un .*

```
#gamecontainer {
  width:800px;
  height:600px;
  border: 1px solid black;
}

#endingscreen {
  background: url(../images/end.png);
  border: 1px solid black;
}

.gamelayer {
  width:800px;
  height:600px;
  position:absolute;
  display:none;
}

#gamecanvas {
  background: lightblue;
}
...
```

Algunas funciones

- *background-color property defines the background color.*
- `color` *property defines the text color*
- `font-family` *property defines the font to be used*
- `font-size` *property defines the text size*
- `text-align` *property defines the horizontal text alignment*
- `border:2px solid Violet` *para crear un borde alrededor.*
- `padding` *property defines a padding (space) between the text and the border.*
- `margin` *property defines a margin (space) outside the border.*
- *Si en un div escribes en el style* `display: none` *oculta la etiqueta div*



JAVASCRIPT

Primero cargamos un canvas en el HTML

```
<canvas id="myCanvas" width="200" height="100" style="border:1px
solid #000000;">
</canvas>
```

Y en el archivo de javascript es donde escribimos las instrucciones para dibujar

```
let canvas = document.getElementById("testcanvas"); let ctx =
canvas.getContext("2d");
```

Las variables se escriben como `let` *si no son constantes y* `const` *si son constantes.*
Para obtener el elemento que queremos editar en JS, escribimos esta función con el id que hayamos escrito en el elemento en concreto.

document.getElementById("myCanvas")

Algunos ejemplos de dibujo en Canvas:
https://www.w3schools.com/html/html5_canvas.asp
ARC es para hacer un círculo
Cada vez que vayamos a empezar a dibujar hay que escribir

```
let canvas = document.getElementById('myCanvas');
        let ctx = canvas.getContext('2d');
ctx.beginPath(); //Para empezar a dibujar
...
ctx.fill();//Lo rellena de un color
ctx.stroke(); //Dibuja el contorno
```

Here are some JS functions for drawing some primitives on a canvas:

Function	Purpose
`fillRect(x, y, width, height)`	Draws a filled rectangle
`strokeRect(x, y, width, height)`	Draws a rectangular outline
`clearRect(x, y, width, height)`	Clears the specified rectangular area and makes it fully transparent
`beginPath()`	Starts recording a new shape
`closePath()`	Closes the path by drawing a line from the current drawing point to the starting point
`fill()` `stroke()`	Fills or draws an outline of the recorded shape
`moveTo(x, y)`	Moves the drawing point to (x, y)
`lineTo(x, y)`	Draws a line from the current drawing point to (x, y)
`arc(x, y, radius, startAngle, endAngle, anticlockwise)`	Draws an arc at (x, y) with specified radius; we set anticlockwise to false for clockwise direction
`strokeText(text,x,y)`	Draws an outline of the text at (x, y)
`fillText(text,x,y)`	Fills out the text at (x, y)
`fillStyle()`	Sets the default colour for all future fill operations
`strokeStyle()`	Sets the default colour for all future stroke operations
`drawImage(image, x, y)`	Draws the image on the canvas at (x, y)
`drawImage(image, x, y, width, height)`	Scales the image to the specified width and height and then draws it at (x, y)
`drawImage(image, sourceX, sourceY, sourceWidth, sourceHeight, x, y, width, height)`	Clips a rectangle from the image (sourceX, sourceY, sourceWidth, sourceHeight), scales it to the specified width and height, and draws it on the canvas at (x, y)

*fillStyle va sin paréntesis

Clases

```
class gameArea {
    constructor(canvas, hero, obstacles) {
        this.canvas = canvas;
        this.hero = hero;
        this.obstacles = obstacles;
        this.context = null;
        this.interval = null;
        this.frameNumber = undefined; }
    initialise() {
        this.canvas.width = GAME_AREA_WIDTH;
        this.canvas.height = GAME_AREA_HEIGHT;
        this.context = this.canvas.getContext("2d");
        let theDiv = document.getElementById("gameplay");
        theDiv.appendChild(this.canvas);
        this.interval = setInterval(updateGame, 1000 / FPS);
        this.frameNumber = 0; }
    render() {
        for (const obstacle of this.obstacles) {
            obstacle.render(this.context); }
        this.hero.render(this.context); }
    clear() {
        this.context.clearRect(0, 0, this.canvas.width, this.canvas.height); }
    addObstacle(obstacle) {
        this.obstacles.push(obstacle); }
    removeObstacle(i) {
        this.obstacles.splice(i, 1); } }
...
let gameArea = new GameArea(document.createElement("canvas"), theSquare, []);
```

Eventos

Event registration:

❶–❸: handler, target, and event.

❶.addEventListener(❷,❸)

Ejemplo:

```
pGround.addEventListener("mouseout",function(event) {
    tableFootballData.isPaused = true;
});
```

Cuando el raton esté fuera del pGround, el método isPaused de la clase tableFootballData se hará true (y se pausará el juego).

Evento especial: el tiempo

```
this.interval = setInterval(updateGame, 1000 / FPS);
```

Evento especial: cuando carga la pantalla

```
window.onload = startGame;
```

Como hacer que cuando haya una imagen la cargue y si no ponga un dibujo de un cuadrado:

```
class SquaredForm {
    constructor(x, y, width, height, color, img = null) {
      ...
 render(ctx) {
        if(this.img == null){
            ctx.fillStyle = this.color;
            ctx.fillRect(this.x, this.y, this.width, this.height);
        }
        else{
            ctx.drawImage(this.img, this.x, this.y);
        }
    }
let spaceship = new Image();
spaceship.src = "img/nave.png";
let theSquare = new SquaredForm(0, GAME_AREA_HEIGHT / 2,
SQUARE_SIZE, SQUARE_SIZE, SQUARE_COLOR, spaceship);
```

Reescalar imágenes

```
spaceship.style.height = '10px';
spaceship.style.width = '10px';
```

Añadir dinámicamente un elemento al HTML

```
let theDiv = document.getElementById("gameplay");
        theDiv.appendChild(this.canvas);
```

Si lo quieres quitar pues removeChild

Como hacer una pantalla de GAME OVER

```
function endGame() {
```

```
  continueGame = false;
    clearInterval(gameArea.interval);
    window.document.removeEventListener("keydown", handlerOne);
    window.document.removeEventListener("keyup", handlerTwo);

    // Hide Game Screen
    let gameScreen = document.getElementById("initialScreen");
    gameScreen.style.display = "none";

    // GAME OVER
    let endScreen = document.getElementById("endScreen");
    let header = document.createElement("header");
    let msg = document.createTextNode("GAME OVER");
    header.append(msg);
    let record = document.createTextNode("You have achived " +
seconds + " seconds. Congratulations!!!");
    endScreen.appendChild(header);
    endScreen.appendChild(record);

    // Show End Screen
    endScreen.style.display = "inline-block";
}
```

Obtener el valor marcado en un radio button

```
// Get the value of the radio button checked in the initial screen
    let radioButtons = document.getElementsByName("level");
    let choice, msg;
    for (const button of radioButtons) {
        if (button.checked) {
            choice = parseInt(button.value);
            break;
        }
    }
```