

Unidad2: Eventos y Formas

Interfaces de Usuario e Interacción

Repaso [eventos]

- Secuencial:
 - Conocemos el flujo y secuencia de ejecución “unívocamente”.
- Eventos:
 - No conocemos la secuencia exacta.
 - Distintos code-paths en base a la situación “particular” que conformen los eventos.

Repaso [eventos]

- HTML

```
<p id="demo" onclick="myFunction()">Click me.</p>
```

```
<script>  
function myFunction() {  
    document.getElementById("demo").innerHTML = "YOU CLICKED ME!";  
}  
</script>
```

- JS

```
<p id="demo">Click me.</p>
```

```
<script>  
document.getElementById("demo").onclick = function() {myFunction()};  
  
function myFunction() {  
    document.getElementById("demo").innerHTML = "YOU CLICKED ME!";  
}  
</script>
```

- JS y EventListener

```
<p id="demo">Click me.</p>
```

```
<script>  
document.getElementById("demo").addEventListener("click", myFunction);  
  
function myFunction() {  
    document.getElementById("demo").innerHTML = "YOU CLICKED ME!";  
}  
</script>
```

Repaso [Selectores]

Clase - Al primero con esta clase

```
document.querySelector('.my-element').addEventListener('click', event => {  
  //handle click  
})
```

Clase - A todos los que tengan

```
document.querySelectorAll('.some-class').forEach(item => {  
  item.addEventListener('click', event => {  
    //handle click  
  })  
})
```

ID

```
document.querySelector('#my-element-id').addEventListener('click', event  
=> {  
  //handle click  
})
```

TP2

[Mouse Down]

- Element del document

```
function mouseDown(e){  
  var eX = e.layerX;  
  var eY = e.layerY;  
}
```

Nota: Ojo con compatibilidad (Chrome, etc)
Deprecated en breve? ☐ offsetX, offsetY

- Página

```
function mouseDown(e){  
  var pX = e.pageX;  
  var pY = e.pageY;  
}
```

- Pantalla

```
function mouseDown(e){  
  var sX = e.screenX;  
  var sY = e.screenY;  
}
```

TP₂ [Ordenando un poco]

Agregar algunas figuras (agregar un random para \leftrightarrow figuras)

Dónde? Arreglo de figuras

Dibujar todas las figuras

Agregar eventos

Repeat 😊

TP2

[Ordenando un poco]

```
const NUM_FIGURES = 4;
const FIGURE_SIZE = 40;
let figures = [];

function addRectangle() {
  let posX = Math.round(Math.random() * canvas.width);
  let posY = Math.round(Math.random() * canvas.height);
  let color = randomRGBA();
  let rect = new Rect(posX, posY, FIGURE_SIZE, FIGURE_SIZE, color, context);
  figures.push(rect);
}

function drawFigures() {
  clearCanvas('#F8F8FF', canvas);
  for (let i = 0; i < figures.length; i++) {
    figures[i].draw();
  }
}

function clearCanvas(color, canvas) {
  context.fillStyle = color;
  context.fillRect(0, 0, canvas.width, canvas.height);
}
```

TP2

[Ordenando un poco]

```
const NUM_FIGURES = 4;
const FIGURE_SIZE = 40;
let figures = [];

function addRectangle() {
  let posX = Math.round(Math.random() * canvas.width);
  let posY = Math.round(Math.random() * canvas.height);
  let color = randomRGBA();
  let rect = new Rect(posX, posY, FIGURE_SIZE, FIGURE_SIZE, color, context);
  figures.push(rect);
}

function drawFigures() {
  clearCanvas('#F8F8FF', canvas);
  for (let i = 0; i < figures.length; i++) {
    figures[i].draw();
  }
}

draw() {
  super.draw();
  this.context.fillRect(this.posX, this.posY, this.width, this.height);
}
```



TP2

[Ej4]

4. Implementar funcionalidad para verificar si un punto se encuentra dentro de una figura. Usar la posición del mouse como punto a verificar. Agregar los métodos correspondientes al diseño del ejercicio anterior.

```
canvas.addEventListener('click')
clickedFigure = findClickedFigure(event.layerX,event.layerY);
if (clickedFigure != null)
    //do stuff
```

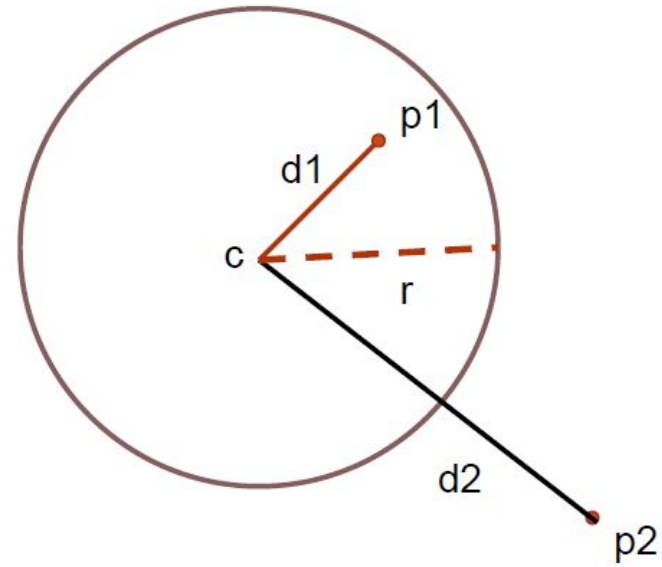
```
function findClickedFigure(x, y) {
    for (index = 0; index < figures.length; index++) {
        element = figures[index];
        if (element.isPointInside(x, y)) {
            return element;
        }
    }
}
```



```
isPointInside(x, y) {
    return !(x < this.posX || x > this.posX + this.width || y < this.posY || y > this.posY + this.height);
}
```

TP2 [Mouse Down]

Círculo:



$$d1 = \sqrt{[(p1.x - c.x)^2 + (p1.y - c.y)^2]}$$



?

```
isPointInside(x, y) {  
  let _x = this.posX - x;  
  let _y = this.posY - y;  
  return Math.sqrt(_x * _x + _y * _y) < this.radius;  
}
```

TP₂

[Mouse Down]

5. Implementar la funcionalidad necesaria para mover las figuras dentro del canvas:
 - a. Mover una figura seleccionada utilizando el mouse (dragear = arrastrar)
 - b. Mover una figura seleccionada utilizando el teclado
- MouseDown:
 - Encontrar la figura.
 - Dar feedback al usuario (highlight por ejemplo)
 - Dibujar las figuras.
 - MouseMove:
 - Colocarle a la figura la posición del mouse
 - ☐ (sólo si tengo una figura “draggéandose” y si... mouseDown
 - Dibujar las figuras.
 - MouseUp: ?