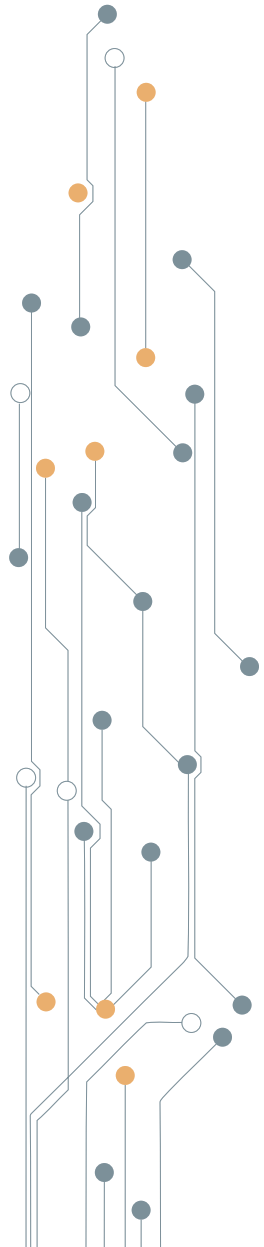
Eventos y su manejo mediante funciones

Telefónica
EDUCACIÓN DIGITAL

Índice



1. Introducción

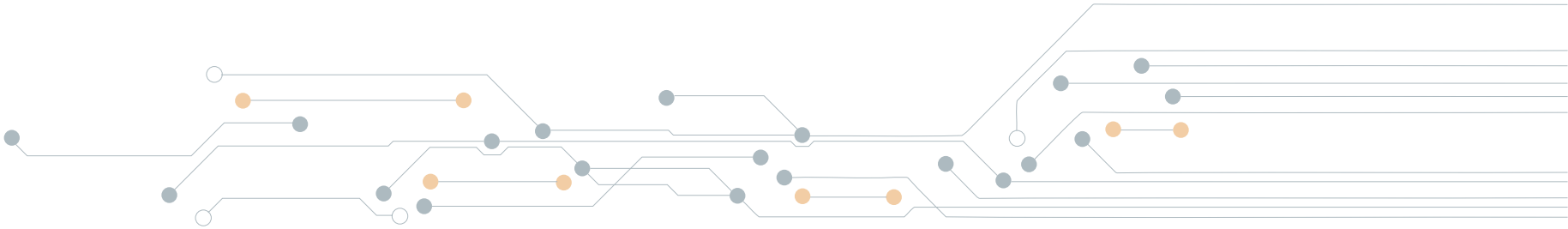
Hasta el momento, todos los códigos que hemos visto siguen ejecutándose sentencia a sentencia, sin interactuar con el usuario.

Estos códigos son poco útiles en programas de JavaScript normales ya que se espera y desea una interacción con los clientes que estén usando esa página web. Así, clickar con el botón del ratón en ciertos apartados, mover el mismo ratón, o teclear pueden ser eventos que se produzcan dentro de nuestra aplicación y se traduzcan en funciones que se tienen que ejecutar en ciertos momentos en JavaScript.

El propio lenguaje nos aporta una serie de eventos para medir la interacción del usuario con nuestra aplicación. Estas funciones creadas en JavaScript son denominadas "manejadores de eventos" o "event handlers".

Hay tres tipos de manejadores de eventos:

- Como atributos de las etiquetas HTML.
- Como funciones en código JavaScript externo.
- "Semánticos".



Evento	Descripción	Elementos para los que está definido
Onblur	Deseleccionar el elemento	*<button>, <input>, <label>, <select>,<textarea>, <body>*
onchange	Deseleccionar un elemento que se ha modificado	*<input>, <select>, <textarea>*
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	*<button>, <input>, <label>, <select>, <textarea>, <body>*
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y *<body>*
onkeypress	Pulsar una tecla	Elementos de formulario y *<body>*
onkeyup	Soltar una tecla pulsada	Elementos de formulario y *<body>*
onload	La página se ha cargado completamente	*<body>*

Evento	Descripción	Elementos para los que está definido
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón "sale" del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover encima terminado	El ratón "entra" en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
onreset	Inicializar el formulario (borrar todos sus datos)	*<form>*
onresize	Se ha modificado el tamaño de la ventana del navegador	*<body>*
onselect	Seleccionar un texto	*<input>, <textarea>*
onsubmit	Enviar el formulario	*<form>*
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	*<body>*

1.1 | Manejadores de eventos como atributos HTML

Se incluye en un atributo del propio elemento HTML.

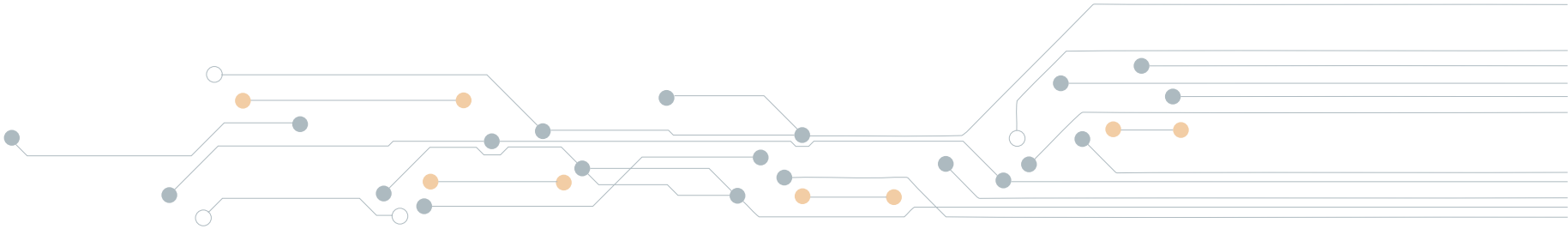
```
<input type="button" value="Pinchame" onclick="alert('Gracias');" />
```

En este ejemplo, cuando nuestro cliente pinche en el botón se ejecutará todo el contenido que hay dentro del atributo "onclick" que es nuestro manejador de eventos.

Este método es quizás el menos práctico ya que impide la reutilización del código JavaScript (que solo se encuentra asignado a ese atributo).

También se pueden poner varios manejadores en una misma etiqueta, pero en diferentes atributos que son los manejadores, claro.

```
<div onclick="alert('Has pulsado');"
onmouseover="alert('Has pasado con el ratón');">
```



1.2 | Manejadores de eventos y variable this

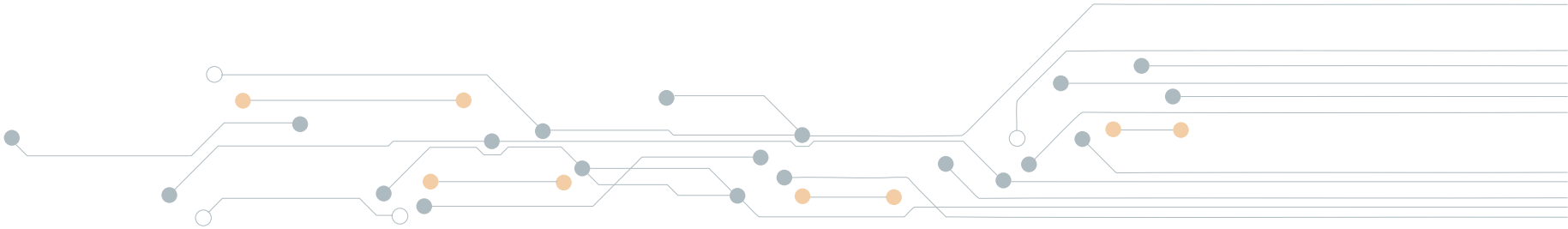
La variable "this" es especial en JavaScript. Se usa para que tome como valor el invocador del evento dentro del manejador. Es decir, qué componente nos ha invocado a la función.

```
<div id="contenido" style="width:180px; height:80px; border:thin solid silver"
onmouseover="this.style.borderColor='green';" onmouseout="this.style.borderColor='red';">
Sección de contenidos...</div>
```

Contenido

Contenido

En este código, los manejadores de eventos "onmouseout" y el "onmouseover" (que cambian el color del borde de nuestro elemento) utilizan la variable "this" para hacer referencia al elemento que nos ha invocado al evento, este es, el div con id "contenido". De esta manera, aunque cambie el id del elemento nuestro código seguirá en funcionamiento.



1.3 | Manejadores de eventos como funciones externas

Es mucho mejor para nuestro código y para el reutilizamiento externalizar todas nuestras sentencias JavaScript en funciones externas.

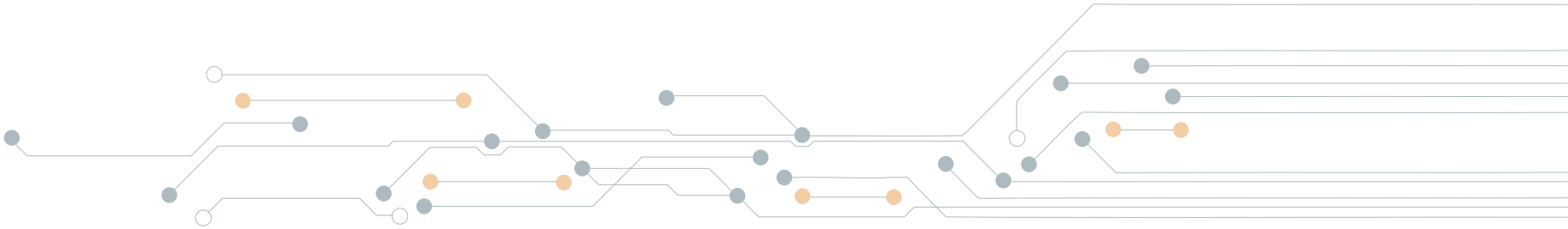
Es mucho mejor para nuestro código y para el reutilizamiento externalizar todas nuestras sentencias JavaScript en funciones externas.

De ese modo, llamamos en nuestros manejadores de eventos a estas funciones externas.

```
function muestra() {alert('Gracias');}
<input type="button" value="Pinchame" onclick="muestra()" />
```

En este caso no podemos utilizar la variable "this" con lo que tenemos que pasar el elemento en cuestión que queremos variar.

```
function resaltar(elemento) {switch(elemento.style.borderColor) {... }}
<div style="width:150px; height:60px; border:thin solid silver"
onmouseover="resaltar(this)" onmouseout="resaltar(this)">
Sección de contenidos...</div>
```



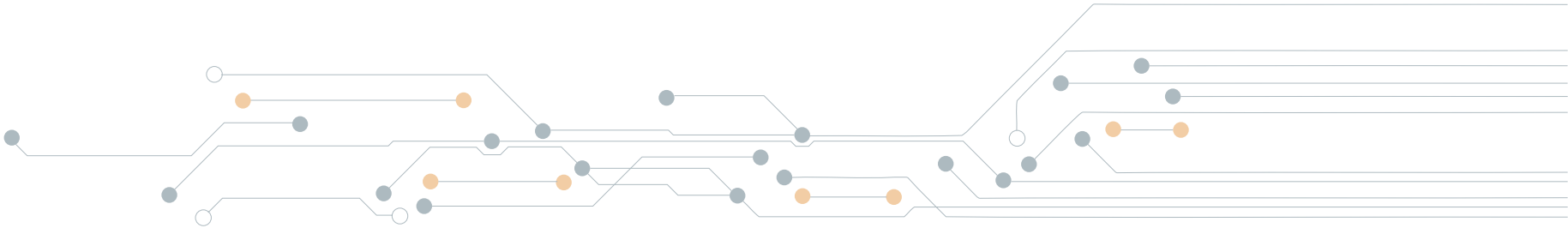
1.4 | Manejadores de eventos semánticos

Esta implementación del manejador se basa en la externalización del código JavaScript, seleccionando el componente al que queremos agregar un manejador de evento y desvinculando completamente nuestro código HTML del código JavaScript.

Esta implementación del manejador se basa en la externalización del código JavaScript, seleccionando el componente al que queremos agregar un manejador de evento y desvinculando completamente nuestro código HTML del código JavaScript.

Es la evolución ideal debido a sus múltiples ventajas entre las que entran la reutilización del código HTML desvinculándola de todo funcionamiento mediante JavaScript, pudiendo reescribir dicho funcionamiento continuamente.

```
// Función externa
function muestra() {alert('Gracias');}
// Asignar la función externa al elemento
document.getElementById("boton").onclick = muestra;
// Elemento XHTML
<input id="boton" type="button" value="Pinchame" />
```



Telefónica

EDUCACIÓN DIGITAL