

TEMA 9: JQuery

a. JSON

1. ¿Qué es JSON?

- JSON son las siglas en inglés de **JavaScript Object Notation**.
- Se trata de un formato para guardar e intercambiar información que cualquier persona pueda leer.
- Los archivos json contienen solo texto y usan la extensión **.json**.

2. ¿Para qué se utiliza un archivo JSON?

- JSON es un formato que almacena información estructurada y se utiliza principalmente para transferir datos entre un servidor y un cliente.
- El archivo es básicamente una alternativa más simple y liviana al XML (Lenguaje de marcado extenso, por sus siglas en inglés) que cuenta con funciones similares.
- Los desarrolladores usan JSON para trabajar con AJAX (JavaScript asíncrono y XML, por sus siglas en inglés). Estos formatos funcionan bien juntos para lograr la carga asincrónica de los datos almacenados, lo que significa que un sitio web puede actualizar su información sin actualizar la página.

3. Sintaxis JSON

- Hay dos elementos centrales en un objeto JSON: claves (Keys) y valores (Values).
 - Las **Keys** deben ser cadenas de caracteres (strings). Como su nombre en español lo indica, estas contienen una secuencia de caracteres rodeados de comillas.
 - Los **Values** son un tipo de datos JSON válido. Puede tener la forma de un array, objeto, string, booleano, número o nulo.
- Un objeto JSON comienza y termina con llaves **{}**. Puede tener dos o más pares de claves/valor dentro, con una coma para separarlos.
- Así mismo, cada **key** es seguida por **dos puntos** : para distinguirla del valor.

3. Sintaxis JSON

- Ejemplo:

```
{  
  "ciudad": "New York",  
  "pais": "EEUU"  
}
```

- Aquí tenemos dos pares de clave/valor: **ciudad** y **pais** son las claves; **Nueva York** y **EEUU** son los valores.

3. Sintaxis JSON. Tipos de valores

1) Array

- Un array es una colección ordenada de valores.
- Está rodeado de **corchetes** `[]` y cada valor dentro está separado por una coma.
- Un valor de un array puede contener objetos JSON, lo que significa que utiliza el mismo concepto de par clave/valor. Por ejemplo:

3. Sintaxis JSON. Tipos de valores

```
{  
  "estudiantes": [  
    {  
      "primerNombre": "Tom",  
      "Apellido": "Jackson"  
    },  
    {  
      "primerNombre": "Linda",  
      "Apellido": "Garner"  
    },  
    {  
      "primerNombre": "Adam",  
      "Apellido": "Cooper"  
    }  
  ]  
}
```

3. Sintaxis JSON. Tipos de valores

2) Objeto

- Un objeto contiene una clave y un valor. Hay dos puntos después de cada clave y una coma después de cada valor, que también distingue a cada objeto. Ambos están entre comillas.
- El objeto, como valor, debe seguir la misma regla que un objeto común. Ejemplo:

3. Sintaxis JSON. Tipos de valores

```
{  
  "empleados": {  
    "nombre": "Tom",  
    "apellido": "Jackson"  
  }  
}
```

- Aquí, **empleados** es la clave, mientras que todo lo que está dentro de las llaves es el objeto.

3. Sintaxis JSON. Tipos de valores

3) Strings

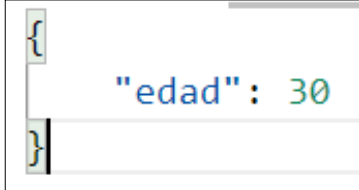
- Un string es una secuencia establecida de cero o más caracteres.
- Está encerrado entre dos comillas dobles.
- Este ejemplo muestra que **Tom** es un string ya que es un conjunto de caracteres dentro de una comilla doble.

```
{  
  "Primer Nombre": "Tom"  
}
```

3. Sintaxis JSON. Tipos de valores

4) Número

- El número en JSON debe ser un **número entero** o un **punto flotante**, como:

A screenshot of a code editor window. The editor contains a JSON object:

```
{  
  "edad": 30  
}
```

 The opening curly brace is on the first line, followed by a line with the key-value pair `"edad": 30`, and the closing curly brace is on the third line. The text is color-coded: the opening brace is blue, the key `"edad"` is blue, the colon is red, and the value `30` is green. The closing brace is blue.

3. Sintaxis JSON. Tipos de valores

5) Booleano

- Puedes usar **verdadero** o **falso** como valor, de la siguiente manera:

```
{  
  "Casado": "false"  
}
```

6) Nulo

- Es para mostrar que no hay información.

```
{  
  "Tipo de sangre": "null"  
}
```

4. Datos JSON almacenados

- Hay dos formas de almacenar datos JSON: objeto y vector.
- El primero se ve así:

```
{  
  "nombre": "Tom",  
  "apellido": "Jackson",  
  "género": "masculino"  
}
```

- Las llaves indican que es un objeto JSON. Implica tres pares clave/valor que están separados por comas.

4. Datos JSON almacenados

- En cada par, tienes las claves (nombre, apellido y género) seguidas de dos puntos para distinguirlos de los valores (Tom, Jackson, masculino).
- Los valores en este ejemplo son strings. Por eso también están entre comillas, similares a las claves.

4. Datos JSON almacenados. Usando arrays

- Otro método para almacenar datos es un array. Ejemplo:

```
{  
  "nombre": "Tom",  
  "apellido": "Jackson",  
  "género": "masculino",  
  "hobby": [  
    "fútbol",  
    "lectura",  
    "natación"  
  ]  
}
```

- Lo que diferencia esto del método anterior es el cuarto par clave/valor. **Hobby** es la clave y hay varios valores (fútbol, lectura, natación) entre corchetes, que representan un array.

5. ¿Cómo utilizar JSON?

- Si analizamos bien la sintaxis de un JSON, nos daremos cuenta de que es muy similar a algo a lo que ya deberíamos conocer:

```
let hombre = {  
  nombre: "Federico",  
  vida: 99,  
};
```

- Nos daremos cuenta de que se trata de un objeto de JavaScript y que no debería ser muy complicado pasar de JSON a JavaScript y viceversa.
- En JavaScript tenemos una serie de métodos que nos facilitan esa tarea, pudiendo trabajar con que contengan JSON y objetos JavaScript de forma indiferente.

6. Convertir JSON a Objeto

- La acción de convertir JSON a objeto JavaScript se le suele denominar parsear.
- Es una acción que analiza un String que contiene un JSON válido y devuelve un objeto JavaScript con dicha información correctamente estructurada.
- Para ello, utilizaremos el método **JSON.parse()**:

```
let str = '{ "nombre": "Juan", "vida": 99 }';  
  
let obj = JSON.parse(str);  
  
obj.nombre; // Juan  
obj.vida; // 99
```

6. Convertir JSON a Objeto

- Como se puede ver, `obj` es un objeto generado a partir del JSON recogido en la variable `str` y podemos consultar sus propiedades y trabajar con ellas sin problemas.

7. Convertir Objeto a JSON

- La acción inversa, convertir un objeto JavaScript a JSON también se puede realizar fácilmente haciendo uso del método **JSON.stringify()**.
- Este método difícil de pronunciar viene a ser algo así como «convertir a texto», y lo podemos utilizar para transformar un objeto de JavaScript a JSON rápidamente:

```
let obj={  
  nombre:"Juan",  
  email:"juan.ferrer@murciaeduca.es"  
};  
  
let strjson=JSON.stringify(obj);  
  
document.write(strjson);
```



```
{"nombre":"Juan","email":"juan.ferrer@murciaeduca.es"}
```

8. Leyendo JSON externo

- Teniendo en cuenta todo lo visto hasta ahora, JSON es un formato ideal para guardar en pequeños archivos de texto que se puedan leer desde JavaScript, pasar a objetos y trabajar con ellos.
- Para hacer esto, existen varias estrategias. La más común es utilizar AJAX para leer este tipo de datos, pero eso lo veremos en el apartado 9e.

9. JSON5

- JSON5 es una extensión de JSON (JavaScript Object Notation) que busca hacer la escritura y lectura de datos estructurados más fácil y legible para los humanos. Fue creado para mejorar la sintaxis de JSON y proporcionar una mayor flexibilidad. Algunas de las características clave de JSON5 incluyen:
 - **Comentarios:** JSON5 permite comentarios tanto de una línea como de varias líneas, lo que facilita la documentación y la comprensión de los datos. Los comentarios se escriben usando el formato de doble barra // para comentarios de una línea y /* */ para comentarios de varias líneas.
 - **Comas finales:** En JSON5, puedes incluir comas al final de las listas y objetos, lo que hace que la edición de datos sea más flexible y evita errores al agregar o eliminar elementos.

9. JSON5

- **Cadenas de caracteres:** Las cadenas de caracteres en JSON5 pueden incluir comillas simples en lugar de comillas dobles, y también se pueden dividir en varias líneas para una mejor legibilidad.
- **Valores literales:** JSON5 admite valores literales como NaN, Infinity, -Infinity, null, true y false en mayúsculas o minúsculas.
- **Nombres de propiedades sin comillas:** En JSON5, puedes omitir las comillas alrededor de los nombres de las propiedades si cumplen con las reglas de identificador de JavaScript.

9. JSON5

- Ejemplo de JSON5:

```
1 {  
2   // Comentario de una línea  
3   nombre: 'Ejemplo',  
4   edad: 30,  
5   activo: true,  
6   intereses: [  
7     'programación',  
8     'datos',  
9     'tecnología'  
10  ]  
11 }
```

9. JSON5

- Es importante destacar que JSON5 es una extensión de JSON y no todos los analizadores de JSON admiten la sintaxis de JSON5.
- Por lo tanto, debes asegurarte de que cualquier aplicación que utilice JSON5 pueda interpretar y procesar los datos correctamente.
- JSON5 es especialmente útil cuando necesitas escribir configuraciones o datos estructurados de una manera más legible y amigable para los humanos.