

Enlace de la tarea: <https://tonylu11.github.io/DAWEC/index.html>

1. En una línea, define qué es jQuery.

Es un framework de JavaScript que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales.

2. Identifica la última versión jQuery

La última versión de JQuery es la 3.1.1

3. Indica las diferencias entre la versión DEVELOPMENT, PRODUCTION y SLIM.

- 1. Development:** Código sin comprimir, se podrá leer la implementación de las funciones del framework, que puede ser interesante en la etapa de desarrollo.
- 2. Production:** Adecuada para páginas web en producción puesto que está minimizada y ocupa menos espacio.
- 3. Slim:** Versión minimalista de JQuery, con algunas funciones.

4. Indica la línea donde introduces todo el código de la librería jQuery.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Antonio Luque Bravo</title>
    <script src="js/jquery-3.1.1.js" type="text/javascript"></script>
```

Se incluye como cualquier otro archivo js.

5. Indica qué es el jQuery CDN.

CDN significa **Content Delivery Network**, que significa **red de entrega de contenido** y no es otra cosa que un servicio que nos permite incluir las librerías de código de JQuery desde los servidores de algunas importantes empresas.

6. Indica brevemente y con tus palabras las ventajas del CDN de jQuery.

El CDN hace que el JQuery sea ligero y aumente la velocidad de carga de nuestra página web.

7. Indica la línea donde introduces las últimas versiones de al menos dos jQuery CDN.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <title>Antonio Luque Bravo</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
    <script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-1.9.0.js"></script>
```

8. Indica cómo jquery ejecuta un código cuando el árbol DOM está totalmente cargado.

Indica el equivalente en JavaScript.

JQuery: \$(document).ready(function () {});

JavaScript: function init y window.addEventListener ('DOMContentLoaded', init);

9. Función \$ o función jQuery. Indica brevemente los argumentos que puedes enviarle.

Añádele a la explicación un breve código de ejemplo (distinto al del manual)

10. Indica cómo puedes reemplazar el clásico \$(document).ready(){...} con jQuery

Se puede reemplazar con las funciones que se llaman a sí mismas, como por ejemplo (\$ (function () {}));:

```
$(function () {
  let documento = $("p");
  documento.css("background-color", "#ff8833");
});
```

11. En una línea, explica qué hace el método `each()` de jQuery. Explica qué es la iteración implícita.

Sirve para ejecutar la función que se le pasa por parámetro por todos y cada uno de los elementos

La **Iteración Implícita** significa que no tenemos que estar programando bucles de código para buscar todos los elementos en el DOM que cumplen con un criterio.

12. Indica el argumento que ha de enviársele al método `each()`.

Recibe una función que se ejecuta por cada elemento que encuentre.

13. Englobado en el contexto del `each`:

1. Explica la utilidad de la palabra reservada `this`.

Significa el elemento que está siendo iterado en ese momento del bucle.

2. Indica cómo se utiliza el índice de la iteración.

Se utiliza para indicar el número que corresponde al elemento que está siendo iterado.

3. Explica la utilidad de `return false`.

Para que salga del bucle `each` y no continúe.

4. Indica la diferencia entre `return true` y no ponerlo. Explícalo mediante un trozo de código.

```
$("#div").each(function(i){
    elemento = $(this);
    if(elemento.html() == "white")
        return true;
    if(elemento.html() == "nada")
        return false;
    elemento.css("color", elemento.html());
});
```

Aquí indicamos que si el elemento contiene 'white' no hará nada.

14. Indica las diferencias y semejanzas entre el método `size()` y la propiedad `length`.

Indica las ventajas e inconvenientes de utilizar uno u otra.

`size()`: Indica el número de elementos que recoge el **objeto JQuery**

`length`: Obtiene el número de elementos de la **página**

Ambos cuentan los elementos, pero uno obtiene el número de elementos de la página entera y otro el número de elementos del objeto JQuery.

15. Indica qué hace el método `data()`.

Sirve tanto para guardar como consultar un elemento.

- Si recibe un solo parámetro devuelve el valor que haya en el dato cuyo nombre se pasa por parámetro.
- Si recibe dos parámetros almacena un dato, cuyo nombre recibe en el primer parámetro, con el valor que recibe en el segundo parámetro.

16. Tipos de datos admitidos por `data()`

Cualquier tipo de elementos del DOM.

17. Indica qué significa que `data()` almacena valores por referencia.

No se copia el objeto, sino que se asigna por referencia. Esto quiere decir que no se harían copias independientes del objeto a guardar, sino que permanecería tal cual y lo que se asignaría como dato es una referencia a ese único objeto.

18. Cuántos objetos se crean si `data()` opera sobre un conjunto de elementos.

Se guarda un dato por cada elemento del objeto JQuery seleccionado: En caso que en el objeto JQuery sobre el que estemos almacenando cosas con `data()` haya referencias a varios elementos de la página, el dato se almacena en todos los elementos.

19. Indica qué hace el método `removeData()`. ¿Está sobrescrito?

Este método sirve para eliminar un dato de un elemento y su funcionamiento es tan simple como enviar por parámetro el dato que se quiere eliminar del elemento.

20. Identifica con tu propio código (en una línea a ser posible) los distintos tipos de selectores. Indica cómo se recogen en una variable.

```
$(document).ready(function() {  
    $("#boton").click(function(evento) {  
        let selectorId = $("#camposelector").prop("value");  
        let selectorClase = $(".clasecamposelector").prop("value");  
        let selectorElemento = $("p").prop("value");  
        let selectorVariasClases = $(".clase1.clase2").prop("value");  
    });  
});
```