Conocimientos generales

1. Código de retorno que devuelve una API cuando la respuesta es exitosa:

   a) 500
   b) 200
   c) 504
   d) 250

2. Código de respuesta HTTP que regresa si un cliente tiene prohibido el acceso:

   a) 403
   b) 402
   c) 404
   d) 400

3. Esta cabecera indica el tipo que el cliente es capaz de entender como respuesta del servidor:

   a) Accept
   b) Content-Type
   c) Access-Control
   d) Authorization

4. Selecciona los elementos necesarios para crear un objeto JSON:

   a) Cadenas
   b) Números
   c) Valores
   d) Llaves

5. ¿Cuál es la forma correcta de definir una lista de objetos de ciudades en formato JSON:

   a)
   ```
   {"ciudades":[
   {"nombre": "Guadalajara", "sigla": "GDL"},
   {"nombre": "Monterrey", "sigla": "MTY"},
   {"nombre": "Guanajuato", "sigla": "GTO"}]}
   ```

   b) No se pueden manejar lista de objetos en JSON.

   c)

```
{"ciudades":
{"nombre": "Guadalajara", "sigla": "GDL"},
{"nombre": "Monterrey", "sigla": "MTY"},
{"nombre": "Guanajuato", "sigla": "GTO"}}
```

d)

```
{[{"nombre": "Guadalajara", "sigla": "GDL"},
{"nombre": "Monterrey", "sigla": "MTY"},
{"nombre": "Guanajuato", "sigla": "GTO"}]}
```

Javascript

1. ¿Cuál es el valor de street?

```
const employee = {
 firstName: 'Pedro',
 age: 30,
}
const street = employee.address?.street;
```

   a) null
   b) undefined
   c) No se puede asignar un valor a street por ser un const
   d) TypeError: Cannot read properties of undefined (reading 'street')

2. ¿Cuál es el valor de street?

```
const employee = {
 firstName: 'Pedro',
 age: 30,
}
const street = employee.address.street;
```

   a) null
   b) TypeError: Cannot read properties of undefined (reading 'street')
   c) No se puede asignar un valor a street por ser un const
   d) undefined

3. Actualmente, para user un el método await es necesario declarar una funcion como asincrona de la siguiente manera:

```
async function miFuncion() {
   //contenido
}
```

   a) falso
   b) verdadero

4. ¿Qué valor obtiene la constante list al ejecutar la siguiente línea de código?

```
const list = document.querySelector('#container').querySelectorAll('div.showlist > p');
```

   a) Una lista de elementos de la clase showlist y que contiene solamente elementos <p>

b) Una lista de elementos <p> que se encuentra dentro de un div de clase showlist que a su vez está dentro de un elemento con el id 'container'
c) Un elemento cuyo id es container y que contiene el elemento de clase showlist que a su vez tiene al elemento párrafo incluido
d) Se obtiene un error debido a que a las constantes no se les puede asignar valores que provengan del DOM

5. ¿Cuál es el resultado de la ejecución del siguiente código?

```
var first = [ 1, 2, 3, 4, 5 ];
var second = [ 1, 2, 3, 4 ];
var isEqual = first.length === second.length && first.every((value, index) => value === second[index])
console.log(isEqual);
```

a) false
b) [1,2,3,4,5,1,2,3,4,5]
c) [1,2,3,4,5]
d) true

6. ¿Qué imprime el siguiente código?

```
let condicion1, resultado, condicion2;
condicion1 = 2>8;
condicion2 = 8>2;
resultado = condicion1 && condicion2;
console.log(resultado);
```

a) 16
b) false
c) true
d) undefined

7. Al ejecutar el siguiente código ¿cuál será la impresión en la consola?

```
const promise1 = () => {
 return new Promise((resolve, reject) => { setTimeout(() => resolve('promise1 fullfilled'), 2000); });
};
const promise2 = () => {
 return new Promise((resolve, reject) => { setTimeout(() => resolve('promise2 fullfilled'), 1000); });
};
promise1().then((result) => console.log(result));
promise2().then((result) => console.log(result));
```

a) [‘promise1 fullfilled’, ‘promise2 fullfilled’]
b) promise1 fullfilled
promise2 fullfilled
c) [‘promise2 fullfilled’, ‘promise1 fullfilled’]
d) promise2 fullfilled
promise1 fullfilled

8. ¿Cuál es el valor de foundNumber?

```
const array = [21, 13, 34, 26, 3, 5];
const foundNumber = array.find(number => number > 20);
```

a) [21, 34, 26]
b) 21
c) [‘21’, ’34’, ‘26’]
d) La función find no forma parte del prototipo de arreglos

9. Partiendo el siguiente arreglo aNum=[5,4,3,2,1,0] ¿Qué método regresa aNum=[0,1,2,3,4,5]?

a) aNum.revert()
b) aNum.flip()
c) aNum.return()
d) aNum.invert()

10. ¿Qué muesta el alert en el siguiente código?

```
message = 'hello';
alert(typeof message);
```

a) string
b) ‘hello’
c) hello
d) ‘typeof message’

LitElement

1. Para el siguiente componente ¿Con cuál de las opciones la propiedad name obtendrá el valor undefined?

```
import { html, LitElement } from 'lit';
export class MyWebcomponent extends LitElement {
 static get properties() {
   return {
     name: {
       type: String,
       attribute: 'fullname'
     }
   };
 }
}
```

   a) <my-webcomponent .name=${“Some Name”}></ my-webcomponent>
   b) <my-webcomponent name=“Some Name”></ my-webcomponent>
   c) <my-webcomponent fullname=${“Some Name”}></ my-webcomponent>
   d) <my-webcomponent fullname=“Some Name”></ my-webcomponent>

2. ¿Cuál de los siguientes NO es un beneficio de trabajar con el shadow DOM?

   a) DOM scoping
   b) Encapsulation
   c) Style scoping
   d) Composition

3. Para el siguiente componente ¿Cuál de las opciones es la correcta para hacer el binding?

```
import { html, LitElement } from 'lit';
export class MyWebcomponent extends LitElement {
 static get properties() {
   return {
     persons: { type: Array }
   };
 }
}
```

   a) <my-webcomponent persons=${JSON.parse(‘[“Ana”, “Juan”]’)}></ my-webcomponent>
   b) <my-webcomponent persons=‘[“Ana”, “Juan”]’></ my-webcomponent>
   c) <my-webcomponent persons=${[‘Ana’, ‘Juan’]}></ my-webcomponent>
   d) <my-webcomponent persons=”[‘Ana’, ‘Juan’]”></ my-webcomponent>

4. ¿Para qué sirve el valor nothing en LitElement?

   a) Para asignar null a una variable
   b) Es el estado en el ciclo de vida, donde no hace nada y espera a hacer la primera carga del template
   c) Para no renderizar nada
   d) No existe esa definición dentro de LitElement

5. Si se requiere reordenar una lista grande o modificarla ya sea agregando o quitando elementos ¿Qué se recomienda usar?

   a) map
   b) render
   c) sort
   d) for

6. ¿Cuál es el resultado de Array colors?

```
import {LitElement, html} from 'lit';
class MyElement extends LitElement {
static properties = {
 colors: { type: Array },
};
constructor() {
 super();
 this.colors = ['Rojo', 'Verde', 'Negro', 'Azul'];
}
render() {
 return html`
   <ul>
     ${this.colors.map((color) =>
       html`<li style="color: ${color}">${color}</li>`
     )}
   </ul>
 `;}
}
customElements.define('my-element', MyElement);
```

   a) Rojo, Verde, Negro, Azul
   b) 1:Rojo, 2:Verde, 3:Negro, 4:Azul
   c) Rojo, Verde, Negro, Azul
   d) Red, Green, Negro, Blue

7. Dado el siguiente código, ¿Cuál es la correcta descripción?

```
const event = new CustomEvent('my-event', {detail: {message: 'Something happened'}});

this.dispatchEvent(event);
```

a) Se crea un objeto con el contenido {detail: {message: 'Sometinh happened'}}
b) Se dispara el evento 'my-event' con el contenido {detail: {message: 'Somethin happened'}}
c) Se guarda en la constante 'my-event' el contenido 'Sometinh happened'
d) Se dispara el evento 'my-event' con el mensaje 'Sometinh happened'

8. ¿Qué es un slot?

a) Es un elemento HTML que no es recomendable usar.
b) Es un elemento HTML que nos permite definir marcadores de posicion en tu plantilla que pueden rellenar con cualquier fragmento de marcado cuando el elemento es usado.
c) Es un elemento HTML que nos permite renderizar arrays.
d) Es un elemento HTML que nos permite renderizar templates condicionales.

9. ¿Cuál es la secuencia del ciclo de vida de un componente en su primer renderizado?

a) firstUpdated -> constructor -> connectedCallback -> update -> render -> updated
b) constructor -> connectedCallback -> updated -> firstUpdated -> update -> render
c) constructor -> connectedCallback -> update -> render -> updated -> firstUpdated
d) constructor -> connectedCallback -> firstUpdated -> update -> render -> updated

10. ¿Con cuál o cuáles de los siguientes casos de renderizado condicional no hay error?

a)
```
return html`${ this.condition ? html`<label>Condition true</label>`: nothing }`;
```

b)
```
return html`${ this.condition ? html`<label>Condition true</label>`: null }`;
```

c)
```
return html`${ this.condition ? html`<label>Condition true</label>`: '' }`;
```

d)
```
return html`${ this.condition ? html`<label>Condition true</label>`: empty }`;
```