Quiz - week 3

1. Which of the following statements is false about the DOM?
    a. The DOM is language-agnostic
    b. Properties and methods of window object have to be always called by referencing window explicitly (correct answer)
    c. Browsers expose an API that you can use to interact with the DOM
2. Which of the following statements is true about the History API?
    a. History API can change the contents of the address bar (correct answer)
    b. history.go() only takes positive numbers as values
    c. We can only go backwards and forwards using history API, we cannot push new URLs to the browser state
3. Which of the following statements are true about the Location API?
    a. We can retrieve query params (parameters in the URL preceded by a ?) using Location.search (correct answer)
    b. The Location API cannot retrieve port numbers from the URL
    c. We cannot reload a page using the Location API, we have to use the History API to achieve this
4. Which of the following statements is true?

    a. There is no way to find out what operating system or browser a user is using, with browser API’s
    b. Temporary URL’s created using URL.createObjectURL() using cannot be destroyed from memory
    c. The Navigator API acts as a container object for various browser API’s (correct answer)
5. Why do you think locationStorage and sessionStorage have the same set of properties and methods?
    a. This is just coincidence - that is how the ECMAscript team happened to design the API’s
    b. Both locationStorage and sessionStorage return the same Storage object (correct answer)
    c. Locationstorage and sessionStorage are one and the same, they are just different names for the same API
6. Which of the following statements if false about cookies, web storage API’s and indexedDB
    a. IndexedDB is an asynchronous API and cookies/web storage API’s are synchronous
    b. All these storage mechanisms are domain-specific, your data can only be accessed against the website you stored it for
    c. You can only have one IndexedDB instance per website (correct answer)
7. Which of the following options would you choose if you wanted to perform smooth animations on a modern browser running continuously on a loop?

    a. setTimeOut with 1000/60 value in timer running inside an infinite loop
    b. setInterval with 1000/60 value in milliseconds
    c. requestAnimationFrame (correct answer)

8. Which of the following statements is true about the Fetch API?
    a. Fetch API can only perform GET requests
    b. We cannot use the Fetch API to catch any errors that may occur when we make API calls, this is a major limitation of this API
    c. The Fetch API lets you access the headers and HTTP status codes from the response object available inside the then clause (correct answer)
9. Which of the following are valid reasons for choosing the localStorage API over the indexedDB?
    a. We are only storing key value pairs of the string data type (correct answer)
    b. localStorage API is more secure than the indexedDB
    c. localStorage has a higher storage limit than the indexedDB
10. Which of the following is not true about iterators in javascript?
    a. Iterators can only be used in array traversal sequences (correct answer)
    b. Iterators need to expose a next method
    c. Iterators can be used to express sequences of unlimited size

11. Given the following piece of code, what input parameters do you have to pass to make it generate odd numbers from 1 to 100?
    a. Start = 0, end = 100, step = 2
    b. Start = 1; end = 100, step = 2 (correct answer)
    c. Start = 1; end = 99, step = 1

```
function* makeRangeIterator(start = ?, end = ?, step =
?) {
    let iterationCount = 0;
    for (let i = start; i < end; i += step) {
        iterationCount++;
        yield i;
    }
    return iterationCount;
}

it = makeRangeIterator();

for (const itItem of it) {

console.log(itItem);

}
```

12. Which of the following statements is true about async/await?
    a. Async/await can only be used with arrow functions
    b. You can have multiple await's inside an async functions (correct answer)

c. Async/await cannot be used together with promises

13. What will be the 3rd message we see in the output after we run this code?

```
console.log('this is the start');

setTimeout(function cb() {

   console.log('Callback 1: this is a msg from call
back');

  }, 0);

  console.log('this is just a message');

  setTimeout(function cb1() {

    console.log('Callback 2: this is a msg from call
back');

  }, 0);

  console.log('this is the end');
```

a. this the end (correct answer)
b. this is just a message
c. Callback 1: this is a msg from call back

14. Which of the following is false about promises?

a. Promises can be in any of the three following states - pending, resolved, rejected
b. The only way we can catch rejected error messages is from the catch block (correct answer)
c. We can use finally to run code irrespective of the promise being rejected or accepted

15. What is the output of the following?

```
function job() {

return new Promise(function(resolve, reject) {

        reject();

    });

}

let promise = job();

promise.then(function() {

    console.log('Success 1');

}).then(function() {

    console.log('Success 2');

}).then(function() {

    console.log('Success 3');
}).catch(function() {

    console.log('Error 1');

});
```

    Success 1, Success 2, Success 3, Error 1

    Error 1 (correct answer)

    Error 1, Success 1, Success 2, Success 3

16. What is the output of the following code?

```
function job() {

return new Promise(function(resolve, reject) {

        reject();

    });

}

let promise = job();

promise.then(function() {

    console.log('Success 1');

}).then(function() {

    console.log('Success 2');
}).then(function() {

    console.log('Success 3');

}).catch(function() {

    console.log('Error 1');

}).finally(function() {

console.log(‘yay!’);

});
```

a. Success 1, Success 2, Success 3, Error 1, yay!

b. Error 1, yay! (correct answer)

c. Error 1, yay!, Success 1, Success 2, Success 3

17. In the following piece of code, we wrote “resolve” instead of “reject” in the promise code by mistake. What do you think will happen?

```
const promise = new Promise(function(resolve, reject) {
    setTimeout(function() {
        resolve('Rejected!');
    }, 1000);
})

promise.catch(function(value) {
    console.log(value);
});
```

a. Nothing will happen, there is no .then clause to handle this
b. The .catch clause will be able to handle this error we made
c. Promise will be in pending state (correct answer)

18. Which of the following would you use if you wanted a list of all resolved or rejected promises in an array?

a. Promise.all
b. Promise.allSettled (correct answer)
c. Promise.race

19. What will be the output of the following?

```
 const promise = new Promise(function(resolve, reject)
{
    resolve('Promise has been resolved!');
});

promise.then((value) => console.log(value));
console.log('I am not the promise');
```

a. I am not the promise
b. I am not the promise, Promise has been resolved!(correct answer)
c. Promise has been resolved!, I am not the promise

20. XHR objects have 4 states defined in ready state, depending on what state the request made is in. Which

of the following will we check for when we want to run a callback after the request is received?

a. 3 (LOADING): the response begins to download
b. 1 (OPENED): the request starts
c. 4 (DONE): the response has been downloaded (correct answer)