1. What is the difference between cookies, localStorage and sessionStorage? Which is worth using if we need to store a lot of user data for a long time?

Local Storage

- Stores data indefinitely.

- It can be cleared only with JavaScript or by clearing the browser cache.

- Stores up to 5 MB of data, the largest of the three storage options.

- Not supported by older browsers like IE 7 and below.

Session Storage

- Stores data as long as the current session lasts. When the user closes the browser, the data is no longer available.

- Uses a top-level browser context, so each browser tab stores unique data.

- The amount of data is larger than in Cookie.

- Not supported by older browsers such as IE 7 and below.

Cookie

- Stores data that can be sent to the server via headers.

- Local and session storage are only available on the client side, and cookies can be sent to the server.

- The retention period is set when the cookie is created.

- The volume of data does not exceed 4 Kbytes.

✔️2. How to add and retrieve a value from a web store?

`setItem()` - adds a key-value pair to the local web repository

`getItem()` - retrieves the value from the key

you can access the cookie directly from the browser, using the document.cookie property

✔️3. Think of at least 3 other situations besides those suggested in the lesson, what might you need to save user data for and what are they?

Store notes, save avatar, theme (dark, light)

✔️4. How to do validation of credit card number?

Using a regular expression:

/^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6(?:011|5[0-9][0-9])[0-9]{12}|3[47][0-9]{13}|3(?:0[0-5]|[68][0-9])[0-9]{11}|(?:2131|1800|35\d{3})\d{11})$/

✔️5. How do I make an input that only accepts numbers (at least 2 ways)?

1) <input type="number" pattern="\d+(\.\d\*)?">

2) HTML:

<input type="text">

<button class="btn">click me</button>

JS:

const inputField = document.querySelector('input');

const btn = document.querySelector('btn');

btn.addEventListener('click', () => {

if (inputField.value.trim().length === 0 || Number.isNaN(Number(inputField.value)) {

inputField.value = '';

alert('Numbers only!');

} else {

inputField.value = '';

alert('Hi');

}

})

```

✔️6. Find a regular expression for the name in Russian

/^[А-ЯЁ][а-яё]\*([-][А-ЯЁ][а-яё]\*)?\s[А-ЯЁ][а-яё]\*\s[А-ЯЁ][а-яё]\*$/

✔️7. Write the most beautiful and clear form of what a non-invalid field should look like, so that the user can understand what to type into it and what is wrong if the input is incorrect (html+css tags)

```

input:valid { border-color: green; }

input:invalid { border-color: red; }

✔️8. What are the disadvantages of the standard way of setting validation via HTML5?

Styles are applied before the user interacts with the form. Fields that must be filled in are immediately highlighted to us as :invalid, and optional fields are highlighted as :valid. This means that the user, even without starting to fill out the form, can immediately receive negative feedback. It's not nice at all and it's not clear to the user what they want from him.

Another disadvantage of this method is its limitation. We can only check the fact of completion, but not the correctness of the data entered.