**Вопросы 💎**

1. **Чем отличаются куки, localStorage и sessionStorage? Что стоит использовать, если нам нужно надолго сохранить много пользовательских данных?**

localStorage – хранит данные бессрочно, объем хранения довольно большой, нужно очищать вручную или с помощью JS;

sessionStorage отличается тем, что данный тип работает сессионно – то есть во время открытой вкладки браузера, и прекращает работу, когда браузер закрывается. Для долгосрочного хранения не подходит.

Cookies Хранит данные, которые можно передавать на сервер через заголовки, срок хранения устанавливается при разработке, объем данных небольшой. Подходит, если данные нужно передавать на сервер.

Если нужно хранить много данных и долго, то подойдет **localStorage.**

1. **Как добавить и получить значение из веб-хранилища?**

Добавить с помощью setItem()

парой ключ (key) - значение (value). Названия ключей мы создаем сами, обязательно на английском языке, но значение может быть каким угодно, и русским, и английским.

И чтобы получить конкретный ключ, то getItem().

Чтобы посмотреть, что есть вообще, в консоль ввести  localStorage

1. **Придумайте еще минимум 3 ситуации помимо предложенных в уроке, для чего может быть нужно сохранять данные пользователя и какие?**

Для ускорения работы сайта, когда он будет загружаться экономичнее и быстрее;

В том числе, чтобы пользователь мог, например, работать с сайтом без подключения к интернету в автономном режиме;

Персонализация настроек сайта, к примеру, выбор языка

***Пример из урока: запомнить, что пользователь уже залогинился, что у него лежит в корзине или в каком разделе сайта он сейчас находится***

1. **Как сделать валидацию номера кредитной карты?**

Ответ здесь **https://github.com/catitgirl/card-validation**

1. **Как сделать input, который будет принимать только числа (минимум 2 способа)?**
   1. Через html с помощью указания типа полей – просто и не требует участия js

<input *type*="number">

* 1. С помощью регулярного выражения

document.querySelector('.ss').addEventListener('input',

function(*e*){

*this*.value = *this*.value.replace(/[^\d.]/g, '');

    }

)

1. **Найдите регулярное выражение для ФИО на русском языке**

'/^(?=.{1,40}$)[а-яёА-ЯЁ]+(?:[-' ][а-яёА-ЯЁ]+)\*$/';

/\*Разбор регулярного выражения:

^ - начало строки

(?=.{1,40}$) - длина от 1 до 40, кроме символов разрыва строки в строке

[а-яёА-ЯЁ]+ - 1 и более русских символов

(?: - начало повторной группы, не выполняющей захват, 0 или более раз, соответствующие последовательностям

[-' ] - символ ', -, или пробел

[а-яёА-ЯЁ]+ - 1 и более русских символов

)\* - конец группировки

$ - конец строки

1. **Как должно выглядеть невалидное поле, чтобы пользователю было понятно, что в него вводить? Как показать пользователю, что не так, если ввод некорректен (теги и атрибуты html, псевдоклассы css) ? *вопрос теоретический, код писать не надо***

Мы можем применять здесь как раз встроенные функции html5?а также css.

- использовать <label for=”correct\_input”> использование настоящих лейблов делает интерфейс более понятным,

- используем placeholder. Он является подсказкой того, каким должно быть валидное поле,

- используем focus, opacity в css, чтобы при наведении курсора происходили определенные вещи – подсвечивалось поле, закрашивалось и пр.

- с помощью атрибута required делаем определенные поля обязательными,

- подсвечиваем верно заполненные поля с помощью псевдокласса :valid

- не показываем никаких подсказок, если поле пустое (это бесит).

1. **Какие есть недостатки у стандартного способа задания валидации через HTML5?**

Во-первых, мы не можем задать какой-то другой вид пользовательского интерфейса. Плюс ограниченность этих способов, не всегда можно решить проблему без js (точнее, редко когда можно). В более сложных задачах, встроенного компонента будет мало! Например, валидацию той же банковской карты невозможно реализовать без js. В примере выше я показала не все, что можно сделать. В банковской карте есть де еще другие поля, есть разные банковские карты и пр.