|  |  |
| --- | --- |
| <input type="email" placeholder="E-mail">  const EMAIL\_REGEXP = /^[\w-\.]+@[\w-]+\.[a-z]{2,4}$/i;  const input = document.querySelector('input');  function isEmailValid(value) {  return EMAIL\_REGEXP.test(value);  }  function onInput() {  if (isEmailValid(input.value)) {  input.style.borderColor = 'green';  } else {  input.style.borderColor = 'red';  }  }  input.addEventListener('input', onInput);  Валидность адреса электронной почты  Рассмотрим адрес электронной почты (test@mail.ru). Вот его обязательные части:   * Название (test) — один или много символов; * Знак собаки (@); * Доменное имя почтового сервера (mail) — один или много символов; * Точка (.); * Доменное имя первого уровня (ru) от двух до пяти букв.   Составим регулярное выражение для наших требований:  /^[\w-\.]+@[\w-]+\.[a-z]{2,4}$/i  Разберём правило по частям:   1. Регулярное выражение должно открываться и закрываться символами **«/»**. После закрывающегося символа можно указать директиву. В нашем случае такой директивной является **«i»**, которая отключает проверку вводимых букв на регистр. То есть, становится не важно, ввели «test@mail.ru» или «Test@Mail.RU». 2. Знаки **«^» и «$»** обозначают начало и конец проверяемой строки. Если их убрать, то правило вернет положительный результат даже если в начале или конце электронного адреса поставить запрещенные знаки. То есть, при вводе «%:&test@mail.ru#6&» функция проверки вернет положительный результат, так как в строке имеется последовательность символов, удовлетворяющая нашему правилу. Для исключения такой возможности указываем, что правило должно применяться ко всей строке, а не к её части. 3. Блок **«[\w-\.]+»** отвечает за проверку названия ящика. В квадратных скобках указываем разрешенные символы: **«\w»** — все латинские буквы, цифры и знак подчеркивания. Так же рекомендую добавлять знак тире и точку **«-\.»**. **«+»** после квадратных скобок указывает на возможность повторения символов — один или много раз. 4. Далее идет знак собаки и доменное имя почтового сервера — **«@[\w-]+»**. Здесь практически тоже самое что и в предыдущем блоке. Исключаем только из набора символов точку. 5. Осталось прописать правило для проверки наличия точки и корректности доменного имени верхнего уровня (ru,com,info). **«\.[a-z]{2,4}»**. Для обозначения знака точки мы указываем её с обратным слешем **«\.»** Без него она будет восприниматься зарезервированным символом регулярки, который обозначает возможность наличия на её месте любого символа. За точкой должно следовать доменное имя верхнего уровня. Это минимум 2 латинские буквы — **«[a-z]{2,4}»** | function maskPhone(value) {  const element = document.querySelector('.input-contacts')  console.log(element)  const maskOptions = {  mask:value;  };  const mask = IMask(element, maskOptions);  }  maskPhone('+{7}(000)000-00-00')  maskPhone('{f\\acebook.com/}\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  maskPhone('{vk.com/}\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  //Создаем маску для полей ввода  function inputMask() {  const contacts = document.querySelector(".contacts-container").children;  Array.from(contacts).forEach((contact)=> {  const select = contact.querySelector('.select-contacts');  console.log(select)  const input = contact.querySelector('.input-contacts');  console.log(input)  select.addEventListener('change', () => {  const selectValue = select.value;  switch(selectValue) {  case "Телефон":  input.setAttribute('id', 'tel')  console.log(input);  break;  case "Вконтакте":  input.setAttribute('id', 'vk')  console.log(input);  break;  case "Email":  input.setAttribute('id', 'Email')  console.log(input);  break;  case "Facebook":  input.setAttribute('id', 'Facebook')  console.log(input);  break;  }  })  })  } |