

# 7. События форм, валидация

[Обработчик событий поля ввода input](#)

[Обработчик событий change](#)

[Применение к чекбоксам/радиокнопкам](#)

[В чем разница между oninput и onchange?](#)

[Отправка формы submit](#)

[Валидация формы validity](#)

[Событие invalid](#)

[Объект validity](#)

## **Практика**

## **Статьи:**

[События: change, input, cut, copy, paste](#)

[change event](#)

[input event](#)

[submit event](#)

[Интерфейсные события](#)

[Формы, элементы управления](#)

[Selection и Range](#)

[Техники валидации форм](#)

[Веб-формы на сайте. Улучшаем опыт взаимодействия](#)

[Тutorial. Список задач с drag & drop](#)

---

## Обработчик событий поля ввода input

**input** срабатывает на каждое изменение значения поля, независимо от того, завершил пользователь ввод или нет.

Если нужно обрабатывать каждое изменение в поле, то это событие является лучшим выбором.

**Пример:** Вывод в консоль того, как пишет пользователь:

```
let textarea = document.querySelector('textarea'); // поле ввода
textarea.oninput = function () {
  console.log(textarea.value);
};
```

или

```
input.addEventListener('input', function () {
  //
});
```

**Используйте обработчик input с осторожностью, так как по мере его выполнения браузер каждый раз перерисовывает страницу.**

---

### Задача “Подсчет зарплаты”:

Используя функцию из прежней задачи, которая *от «грязной» зарплаты (зарплата до вычета налогов) посчитает примерную «чистую» зарплату (которая выдаётся на руки)*, создай интерфейс с полем ввода “грязной” зарплаты и выводом результата “чистой” зарплаты

Напомню: Оформи программу в виде функции calculateSalary с одним параметром — величиной грязной зарплаты. Функция должна возвращать чистую зарплату. Большая точность мне не нужна, просто считаем, что 35% величины грязной зарплаты составляют налоги, а если грязная зарплата больше или равна 100 тысячам, то налоги составляют уже 45%.

---

## Обработчик событий change

**change** срабатывает по окончании изменения элемента. Например, когда пользователь выбирает новое значение из выпадающего списка или переключает чекбокс/радиокнопки.

```
let select = document.querySelector('select'); // выпадающий список
```

```
select.onchange = function () {  
  //  
};
```

или

```
select.addEventListener('change', function () {  
  //  
});
```

## Применение к чекбоксам/радиокнопкам

При выборе чекбокса у него возникает состояние **checked**. Оно имеет булево значение: **true**, если чекбокс включён, и **false**, если нет.

Для того, чтобы отслеживать изменения, на чекбокс добавляется **onchange** с условием **if** на состояние **checked** - если отмечен - делай одно, если не отмечен - другое.

```
checkbox.onchange = function () {  
  if (checkbox.checked) {  
    // если выбран  
  } else {  
    // если не выбран  
  }  
}
```

## В чем разница между oninput и onchange?

- **oninput** срабатывает на каждое изменение значения, независимо от того, завершил пользователь ввод или нет. Например, он сработает на каждое изменение положения ползунка, даже если пользователь продолжает его двигать. И на каждый новый символ в текстовом поле, даже если пользователь продолжает вводить текст.
- **onchange** срабатывает, если значение поля ввода изменилось и пользователь закончил ввод. Например, если пользователь передвинул ползунок и отпустил его. Или ввёл что-то в текстовое поле и убрал из него курсор.

---

**Задача “Расчёт стоимости проекта”:** Создай интерфейс для получения данных этой задачи из предыдущего модуля.

Назови функцию `getPrice`. У неё должно быть два параметра:

- время (в часах), которое нужно потратить на проект;

- булево значение, которое указывает на срочность проекта — `true` для срочного заказа и `false` для обычного.

Для каждого проекта есть фиксированная ставка — 1500 рублей в час. Расчёт стоимости проектов выглядит так: время \* ставка в час.

Есть несколько нюансов. Если проект срочный, то часы уменьшаются в два раза, а ставка за час повышается в 2.5 раз.

А если время проекта больше 150 часов, ставка в час уменьшается на 250 рублей.

В первую очередь проверяй срочность. Функция должна возвращать стоимость проекта.

В интерфейсе должно быть поле ввода времени на проект и чекбокс выбора срочности проекта. При изменении времени/чекбокса ниже сразу же должна отображаться итоговая стоимость проекта.

---

### Задача: сделать сортировку постов по тематикам

Есть новостная страница, на которой публикуются новости разных тематик:

IT, Котики, Игры, Коронавирус

Сортировку делай при выборе соответствующей тематики из выпадающего списка, также должен быть вариант “Все новости”.

Чтобы отслеживать категорию, для каждой новости добавь пользовательский атрибут **data-category** со значением категории. Лишние статьи можно скрывать при помощи класса **hidden**.

[Пример](#)

---

Получить значение `data-` атрибута можно при помощи свойства **dataset**, после которого указывают имя атрибута без префикса `data-`:

```
<p data-user-name="Piter">
```

```
let element = document.querySelector('p');  
console.log(element.dataset.userName);
```

## Отправка формы submit

При отправке формы срабатывает событие **submit**, оно обычно используется для проверки (валидации) формы перед её отправкой на сервер или для предотвращения отправки и обработки её с помощью JavaScript.

**Есть два основных способа отправить форму:**

1. нажать **кнопку** `<input type="submit">` или `<button type="submit">`.
2. нажать **Enter**, находясь на каком-нибудь поле.

Оба действия генерируют событие **submit** на форме. Обработчик может проверить данные, и если есть ошибки, показать их и вызвать `event.preventDefault()`, тогда форма не будет отправлена на сервер.

```
element.onsubmit = function (evt) {  
    evt.preventDefault();  
    // ...  
};
```

или

```
element.addEventListener('submit', function (evt) {  
    evt.preventDefault();  
    // ...  
});
```

---

### Задача “Список дел”

Создай простой список дел, пункты которого добавляются с помощью поля формы и образуют список. Новую задачу добавляй в конец списка.

[Пример](#)

---

# Валидация формы validity

Валидация:

- проверка на правильность введенных пользователем данных (текст, число),
- ограничение значений (минимум, максимум, шаг изменения),
- реакция на изменение значений (появление, блокирование полей),
- запрет на отправку данных формы пока форма не валидна.

Валидация форм - один из способов взаимодействия с пользователем. Валидация на клиенте позволяет избежать необходимости отправлять запрос на сервер, проверять там его на правильность и только после этого показывать ошибки

**Валидация формы:**

[html] проверка на правильность формата введенных данных  
[html] ограничение значений  
**[js] реакция на изменение значений**  
[html] запрет на отправку до тех пор пока форма не валидна

Большую часть валидации можно сделать при помощи атрибутов HTML-тегов.

В **CSS** существует четыре специальных псевдокласса, применимых к полям формы:

- :valid (валидное поле),
- :invalid (невалидное),
- :required (обязательное),
- :optional (необязательное).

Проблема: стили применяются до того, как пользователь начнёт работу с формой. Поля, обязательные для заполнения, сразу подсвечиваются как :invalid, а необязательные — как :valid.

## Событие invalid

наступает в момент, когда при попытке отправки формы значение неправильное

```
userNameField.addEventListener('invalid', function (evt) {  
  // что сделать с невалидным полем  
});
```

Например, можно убрать стандартное окно с ошибкой и добавить свой элемент по шаблону.

В каждом поле содержится специальный **объект validity**, включающий в себя список булевых значений, характеризующих ту или иную проверку на валидность

## Объект validity

- свойства JS-объектов форм, описывающие валидность каждого из полей. В них содержится информация, какая именно ошибка была допущена при вводе значения и сообщение об ошибке, которое покажет браузер

### ValidityState флаги (true/false):

- badInput — введено неправильное значение
- customError — задано кастомное сообщение об ошибке
- patternMismatch — не соответствует паттерну
- rangeOverflow — больше значения max
- rangeUnderflow — меньше значения min
- stepMismatch — значение не попадает в step
- tooLong — больше максимальной длины
- tooShort — слишком короткое значение
- typeMismatch — не совпадает тип
- valid — валидно ли поле
- valueMissing — нет значения

Если поле не валидно, то можно указать **.setCustomValidity** - сообщение, описывающее проблему. Например, произвольное сообщение о количестве символов в имени пользователя:

```
<form action="#">
  <input type="text" class="user-name" minlength="2" required>
  <input type="submit" class="submit-btn">
</form>
```

```
let userName = document.querySelector(".user-name");
userName.addEventListener("invalid", function (evt) {
  if (userName.validity.tooShort) {
    userName.setCustomValidity("Нужно больше символов");
  } else if (userName.validity.valueMissing) {
    userName.setCustomValidity("Поле обязательное!");
  } else {
    userName.setCustomValidity("");
  }
});
```

```
userName.addEventListener("input", function (evt) {
  let target = evt.target;
  if (target.value.length < 2) {
    target.setCustomValidity("Нужно больше символов");
  } else {
    target.setCustomValidity("");
  }
});
```

});

---

**Дополнительно:** проверка формы на валидность

<https://htmlacademy.ru/blog/boost/frontend/form-validation-techniques>

---



# Практика

## Задача “Страница для слабовидящих”

Для страницы с текстовым наполнением создай интерфейс, позволяющий менять при смене значений ползунка размер шрифта страницы (от 14px до 32px), выбрать фоновый цвет страницы и цвет текста из выпадающих списков.

[Пример](#)

---

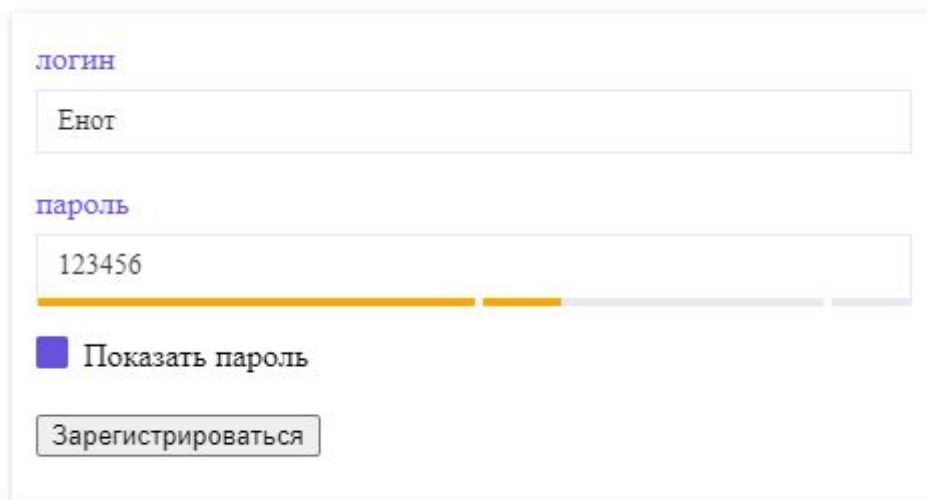
## Проверка пароля

Создай форму регистрации пользователя с возможностью посмотреть пароль при выборе соответствующего чекбокса и проверкой длины пароля:

до 5 символов - слабый пароль

от 5 до 8 символов - нормальный пароль

более 8 символов - хороший пароль



[Пример](#)

---

## “Калькулятор доставки”

Создай интерфейс подсчета стоимости доставки груза. Пользователю нужно ввести данные:

- вес груза (числовое поле)
- расстояние в км (числовое поле)
- нужна ли разгрузка (чекбокс)
- нужна ли доставка до двери (чекбокс)
- хрупкий ли груз (чекбокс)

**Алгоритм подсчета стоимости доставки такой:**

Стоимость доставки 1 кг груза на 1 км = 700 рублей. Если расстояние более 100 км, то 900 рублей.

Максимальное расстояние для доставки - 200 км. Минимальное расстояние - 1 км.

Если груз хрупкий, то к стоимости добавляется еще 3000 рублей. Если нужна разгрузка, то 1500 руб. Если нужна доставка до двери, то 2000 руб

---

### Задача: интерфейс задачи Как ты мне дорог

Создай интерфейс для задачи предыдущего модуля.

Напомню: Напиши функцию `calculateExpenses`, которая вычисляет ежемесячные затраты компании на сотрудника из «чистой» зарплаты работника. Функция должна принимать параметр `netSalary` — это «чистая» зарплата после вычета налогов. В переменную `incomeTax` записан размер НДФЛ в процентах. В переменной `contributions` указан общий размер взносов в процентах. Функция должна возвращать общие затраты компании на сотрудника. Округляй результат вычислений с помощью `Math.round()`. Разберёмся на примере, как устроены затраты работодателя на одного сотрудника. Если «чистая» зарплата сотрудника 87 тысяч, то работодатель тратит 100 тысяч — 87 работнику и 13 в налоговую (НДФЛ 13%). Это «грязная» зарплата. Ещё нужно заплатить различные взносы, это около 30% от «грязной» зарплаты сотрудника. В задаче нам нужно найти все затраты работодателя, в примере выше это 130000.

---

### Задача: интерфейс задачи Деньги к деньгам

Мне нужно посчитать, сколько я заработаю денег на вкладах с разными условиями. Оформи программу, как функцию `calculateDeposit` с четырьмя параметрами:

начальная сумма депозита; процент годовых (число от 0 до 100); срок вклада в месяцах; с капитализацией процентов или нет (флаг с булевым значением).

Функция должна возвращать итоговую сумму депозита, округлённую до рублей с помощью `Math.floor()`. Название параметров используй любые.

Если вклад простой, то процент годовых делится на 12 и умножается на срок вклада, а затем начальная сумма увеличивается на посчитанный процент.

Вклад с капитализацией считается сложнее: каждый месяц к сумме депозита прибавляются накопленный за месяц процент годовых (не забывай делить процент на 12), а процент следующего месяца считается уже от увеличенной суммы депозита.

[Пример](#)

---

## Прогулка кота

Создай интерфейс для решения задачи о прогулке кота.

Какая температура за окном?

15

☐ Есть ли дождь?

изображение погоды

КОТ

“  
Идем гулять XX минут!

Изначально на экране есть только вопрос с полем ввода числа и чекбокс наличия дождя. При изменении данных в них - справа появляется изображение, соответствующее погоде. И ниже появляется изображение кота с текстом времени прогулки.

Напомню задачу: Если идёт дождь, гулять я не хожу. В этом случае длительность прогулки равняется 0.

А вот если дождя нет, всё зависит от температуры на улице:

Во-первых, если температура от 10°C (включительно) до 15°C (не включая это значение), я гуляю 30 минут.

Во-вторых, если температура от 15°C (включительно) до 25°C (не включая значение), я гуляю 40 минут — погода идеальна.

В-третьих, при температуре от 25°C (включительно) до 35°C (включительно), я гуляю 20 минут.

В остальных случаях я никуда не выхожу: либо очень холодно, либо очень жарко.

Изображения:

дождь:

<https://www.flaticon.com/svg/static/icons/svg/3026/3026385.svg>

<https://www.flaticon.com/svg/static/icons/svg/763/763771.svg>

никуда не выхожу:

<https://www.flaticon.com/svg/static/icons/svg/2965/2965893.svg>

<https://www.flaticon.com/svg/static/icons/svg/763/763758.svg>

40 минут:

<https://www.flaticon.com/svg/static/icons/svg/869/869767.svg>

<https://www.flaticon.com/svg/static/icons/svg/763/763765.svg>

30 минут:

<https://www.flaticon.com/svg/static/icons/svg/578/578116.svg>

<https://www.flaticon.com/svg/static/icons/svg/763/763747.svg>

20 минут:

<https://www.flaticon.com/svg/static/icons/svg/993/993377.svg>

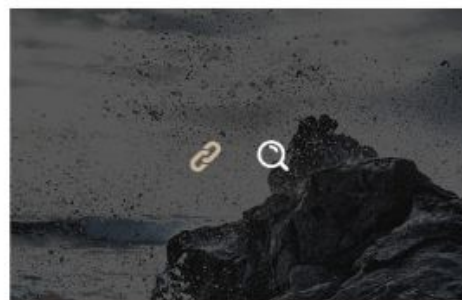
<https://www.flaticon.com/svg/static/icons/svg/763/763785.svg>

## Еще сортировка

Сделай сортировку фотогалереи размером 3\*4 по 5ти параметрам: тематика 1, тематика 2, тематика 3, тематика 4 и все фото



Claritas Etiam Processus  
Photography, Nature



Quam Nutamus Farum  
Graphic Design, Mock-Up



Usus Legentis Videntur  
Photography, Holiday



Claritas Etiam Processus  
Photography, Nature



Quam Nutamus Farum  
Graphic Design, Mock-Up



Usus Legentis Videntur  
Photography, Holiday

---

## Новый список дел \*\*

Создай улучшенный список дел, в котором:

- задача считается выполненной и исчезает, если юзер кликнул по чекбоксу;
- если все задачи выполнены, появляется сообщение, что больше задач нет;
- если в пустой список добавляется новая задача, сообщение исчезает;
- чтобы добавить новую задачу, надо ввести описание в поле ввода и нажать «Добавить задачу», задача появится в конце списка.

☐ Покормить кота

☐ Уехать из страны

☐ Вдариться в бега

Добавить задачу

Верстальщик уже всё подготовил: <https://codepen.io/lipa88/pen/Bazpwdm>

---

## А теперь с перетаскиванием\*\*\*

Ознакомься со статьей [Тutorial. Список задач с drag & drop](#) и добавь в задачу выше возможность сортировать пункты списка дел с помощью перетаскивания.