**Валидация ввода на стороне клиента**

Перед отправкой данных на сервер важно убедиться, что все поля формы заполнены в соответствующем формате. Такая проверка называется «Валидация на стороне клиента». С помощью такой проверки мы можем убедиться, что отправленные данные соответствуют некоторым, заложенным заранее в проект, требованиям.

Валидация на стороне клиента - это начальная проверка вводимых данных. Она является важным элементом правильного пользовательского опыта (UX, user experience). При обнаружении каких-либо ошибок в полях формы, пользователь может сразу их исправить. При этом не происходит отправки некорректных данных на сервер. Если бы данные ушли на сервер и после серверной валидации были бы им отклонены, то в этом случае мы заставляем пользователя ждать, а также нагружаем сервер лишними запросам. Также, если были допущены какие-либо ошибки в коде валидации на стороне сервера, то не валидные данные могут попасть в БД и разрушить ее структуру. Все это в конечном итоге может привести к неработоспособности проекта.

Проверка на стороне клиента не должна рассматриваться как исчерпывающая мера безопасности! Приложение всегда должно выполнять проверку безопасности любых данных, отправленных в форме, как на стороне сервера, так и на стороне клиента, поскольку проверку на стороне клиента достаточно легко можно обойти.

**Что такое Валидация форм?**

Ели зайти на любой сайт, где присутствует регистрационная форма или форма входа, то можно увидеть различные сообщения, которые появляются рядом с полями ввода при неправильном вводе данных. Эти сообщения представляют собой обратную связь для пользователя, чтобы он понимал, правильно ли были введены данные или, были допущены ошибки. Сообщения могут выглядеть следующим образом:

* «Поле является обязательным для заполнения» (это поле нельзя оставлять пустым).
* «Введите номер телефона в формате xxx-xxxx» (требуется определенный формат данных).
* «Введите действующий адрес электронной почты» (данные имеют неправильный формат).
* «Пароль должен содержать от 8 до 30 символов и содержать одну заглавную букву, один символ и цифру».

Это и называется валидацией формы. После ввода данных, браузер и/или веб-сервер проверяют, имеют ли данные правильный формат и находятся ли в пределах ограничений, установленных приложением. Проверка, которая выполняется в браузере, называется клиентской валидацией (или валидацией на стороне клиента), а проверка, выполняемая на сервере, называется серверной валидацией.

Причины, по которым необходимо валидировать ввода данных могу быть следующими:

* Мы хотим получать правильные данные в правильном формате. Приложение не будет работать должным образом, если данные наших пользователей хранятся в неверном формате или полностью отсутствуют;
* Мы хотим защитить данные наших пользователей. Необходимость вводить данные или пароли в требуемом формате упрощает защиту их учетных записей.
* Мы хотим защитить свое приложение. Существуют различные способы, с помощью которых хакеры могут завладеть незащищенными формами, чтобы нанести ущерб приложению. Валидация данных достаточно сильно снижает вероятность подобных ситуаций.

***ВАЖНО****: Никогда не доверяйте данным, передаваемым на ваш сервер от клиента. Даже если форма валидируется верно и препятствует вводу некорректных данных, хакеры все равно могут найти способы изменить запрос на сервер. Поэтому, очень важно после валидации на стороне клиента производить валидацию данных на стороне сервера.*

**Типы валидации на стороне клиента**

Существует два типа проверки на стороне клиента, с которыми, возможно, Вы будете когда-то работать:

• Валидация формы средствами HTML5 (встроенная валидация HTML5). Этот тип валидации данных обычно не требует большого количества JavaScript. Встроенная проверка формы имеет лучшую производительность, чем проверка на чистом JavaScript, но при этом не имеет необходимой гибкости.

• Проверка JavaScript пишется с использованием скриптового языка JavaScript. В этом случае валидация данных может быть очень гибко настроена. Но при этом необходимо писать код самостоятельно либо использовать различные библиотеки.

**Использование валидации HTML5**

Одна из наиболее важных особенностей валидации HTML5 - это возможность проверять бОльшую часть пользовательских данных без использования JavaScript. Она выполняется с помощью специальных атрибутов проверки в тегах формы. Используются следующие атрибуты:

• **required**: указывает на обязательность заполнения поля;

• **minlength и maxlength**: определяет минимальную и максимальную длину текстовых данных, введенных в поле;

• **min и max**: указывает минимальное и максимальное значения числовых типов ввода, введенных в поле;

• **type**: указывает, должны ли данные быть числом, адресом электронной почты или каким-либо другим предустановленным типом данных;

•**pattern**: в нем прописывается регулярное выражение, определяющее шаблон, которому должны соответствовать введенные данные.

Если данные, введенные в поля формы, соответствуют всем правилам, указанным в атрибутах, форма считается валидной. В противном случае, форма будет не валидной и отправка данных на сервер не произойдет.

***Когда элемент является валидным, будет верно следующее:***

• Элемент соответствует псевдоклассу CSS :valid, который позволяет применять определенные стили CSS к валидным элементам;

• При отправке данных, браузер отправит форму на сервер при условии, что нет других каких-либо ограничений (например, дополнительные проверки с использованием JavaScript).

***Когда элемент невалиден, будет верно следующее:***

• Элемент соответствует псевдоклассу CSS :invalid, либо другим псевдоклассам пользовательского интерфейса (например,: out-of-range) в зависимости от ошибки;

• При попытке отправить данные, браузер заблокирует отправку (не будет отправлен запрос на сервер) и отобразит сообщение об ошибке.

**Примечание.** Существуют различные типы ошибок, которые не позволяют отправить форму на сервер, например:  *badInput, patternMismatch, rangeOverflow или rangeUnderflow, stepMismatch, tooLong или tooShort, typeMismatch, valueMissing или customError.*

**Пример валидации HTML5**

**Атрибут required**

Самый простой атрибут валидации - **required**. Чтобы сделать поле обязательным для ввода, необходимо просто добавить его к атрибутам поля. В случае использования этого атрибута, поле формы будет соответствовать псевдоклассу :required UI. Форма не будет отправляться на сервер, при этом появится сообщение сообщение об ошибке, если поле не содержит данных. Пока поле остается пустым, оно соответствует псевдоклассу :invalid UI.

<form>

        <label for="choose">What color do you like? (required)</label>

        <input id="choose" name="i\_like" required>

        <button>Submit</button>

      </form>

input:invalid {

    border: 2px dashed rgb(255, 0, 0);

  }

  input:invalid:required {

    background-image: linear-gradient(to right, pink, lightgreen);

  }

  input:valid {

    border: 2px solid black;

  }

Наличие **required** атрибута в любом элементе, поддерживающем этот атрибут, означает, что элемент соответствует псевдоклассу **:required**, независимо от того, имеет он значение или нет. Если в <input> не введено никакого значения, ввод данных будет соответствовать еще и псевдоклассу :invalid.

***Примечание.*** *Для удобства пользователя, необходимо указывать, какие поля являются обязательными. Это не только повышает лояльность пользователя к Вашему ресурсу, но и рекомендуется правилами доступности WCAG (Правила* *обеспечении доступности веб-контента).*

**Атрибуты Pattern,** [**minlength**](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/minlength)**и [maxlength](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/maxlength),** [**min**](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/min)**и [max](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/max), регулярные выражения,**

**Pattern**

Это еще один полезный атрибут, значение которого проверяет вводимые данные с использованием регулярных выражений. Регулярное выражение (regex) - это шаблон, который можно использовать для сопоставления комбинаций символов в текстовых строках. Они отлично подходят для проверки форм, а также широко используются при написании JavaScript кода.

Примеры регулярных выражений для проверки различных шаблонов входных данных можно легко найти в сети. Например, здесь: <https://habr.com/ru/post/349860/>.

<https://htmlacademy.ru/blog/js/regexp-howto>

**Ограничение длины вводимых данных**

Можно ограничить длину вводимых данных в текстовых полях, таких как <input> или <textarea>, используя атрибуты **minlength** и **maxlengt**h. Поле остается валидным, если оно содержит значение, и это значение имеет длину больше, чем minlength, или меньше, чем maxlength.

**Ограничение максимального и минимального значения**

Для числовых полей (т.е. <input type = "number">) атрибуты min и max могут использоваться для определения диапазона допустимых значений. Если поле содержит значение вне этого диапазона, оно будет не валидным.

**Пример клиентской валидации HTML5**

HTML

<form>

        <p>

          <fieldset>

            <legend>Do you have a driver's license?<abbr title="This field is mandatory" aria-label="required">\*</abbr></legend>

            <!-- While only one radio button in a same-named group can be selected at a time,

                 and therefore only one radio button in a same-named group having the "required"

                 attribute suffices in making a selection a requirement -->

            <input type="radio" required name="driver" id="r1" value="yes"><label for="r1">Yes</label>

            <input type="radio" required name="driver" id="r2" value="no"><label for="r2">No</label>

          </fieldset>

        </p>

        <p>

          <label for="n1">How old are you?</label>

          <!-- The pattern attribute can act as a fallback for browsers which

               don't implement the number input type but support the pattern attribute.

               Please note that browsers that support the pattern attribute will make it

               fail silently when used with a number field.

               Its usage here acts only as a fallback -->

          <input type="number" min="12" max="120" step="1" id="n1" name="age"

                 pattern="\d+">

        </p>

        <p>

          <label for="t1">What's your favorite fruit?<abbr title="This field is mandatory" aria-label="required">\*</abbr></label>

          <input type="text" id="t1" name="fruit" list="l1" required

                 pattern="[Bb]anana|[Cc]herry|[Aa]pple|[Ss]trawberry|[Ll]emon|[Oo]range">

          <datalist id="l1">

            <option>Banana</option>

            <option>Cherry</option>

            <option>Apple</option>

            <option>Strawberry</option>

            <option>Lemon</option>

            <option>Orange</option>

          </datalist>

        </p>

        <p>

          <label for="t2">What's your e-mail address?</label>

          <input type="email" id="t2" name="email">

        </p>

        <p>

          <label for="t3">Leave a short message</label>

          <textarea id="t3" name="msg" maxlength="140" rows="5"></textarea>

        </p>

        <p>

          <button>Submit</button>

        </p>

      </form>

CSS

form {

    font: 1em sans-serif;

    max-width: 320px;

  }

  p > label {

    display: block;

  }

  input[type="text"],

  input[type="email"],

  input[type="number"],

  textarea,

  fieldset {

    width : 100%;

    border: 1px solid #333;

    box-sizing: border-box;

  }

  input:invalid {

    box-shadow: 0 0 5px 1px red;

  }

  input:focus:invalid {

    box-shadow: none;

  }

**Проверка форм с использованием JavaScript**

Если есть необходимость контролировать ввод и отправку данных самостоятельно, контролировать стили сообщений об ошибках, а также использовать валидацию в старых браузерах, которые не поддерживают валидацию HTML5, в этом случае необходимо писать javaScript.

[**The Constraint Validation API**](https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation#the_constraint_validation_api) **(Программный интерфейс проверки ограничений)**

Большинство браузеров поддерживают [The Constraint Validation API](https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation#the_constraint_validation_api), который состоит из набора методов и свойств, доступных в следующих интерфейсах DOM элементов форм:

* HTMLButtonElement (представляет элемент <button>);
* HTMLFieldSetElement (представляет элемент <fieldset>);
* HTMLInputElement (представляет элемент <input>);
* HTMLOutputElement (представляет элемент <output>);
* HTMLSelectElement (представляет элемент <select>);
* HTMLTextAreaElement (представляет элемент <textarea>).

[The Constraint Validation API](https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation#the_constraint_validation_api) делает следующие свойства доступными для указанных выше элементов:

* validationMessage: возвращает сообщение, описывающее ограничения проверки.
* validity: возвращает объект ValidityState, который содержит несколько свойств, описывающих состояние элемента:
* patternMismatch: возвращает true, если значение не соответствует указанному шаблону, и false, если оно совпадает.
* tooLong: возвращает true, если значение больше максимальной длины, указанной атрибутом maxlength, или false, если оно меньше или равно максимуму.
* tooShort: возвращает true, если значение меньше минимальной длины, указанной атрибутом minlength, или false, если оно больше или равно минимуму.
* rangeOverflow: возвращает истину, если значение больше максимального, указанного в атрибуте max, или ложь, если оно меньше или равно максимуму.
* rangeUnderflow: возвращает true, если значение меньше минимума, указанного атрибутом min, или false, если оно больше или равно минимуму.
* typeMismatch: возвращает true, если значение не соответствует требуемому синтаксису (если тип - адрес электронной почты или URL), или false, если синтаксис правильный.
* valid: возвращает true, если элемент соответствует всем ограничениям валидации и, следовательно, считается валидным, или false, если не выполняется какое-либо ограничение.
* valueMissing: возвращает true, если элемент имеет обязательный атрибут, но не имеет значения, или false в противном случае.
* willValidate: возвращает true, если элемент будет проверен при отправке формы; в противном случае - ложь.

Constraint Validation API также предоставляет следующие методы для указанных выше элементов и элемента form:

* checkValidity (): возвращает true, если значение элемента валидно; в противном случае - false. Если элемент невалиден, этот метод запускает событие невалидности для элемента;
* reportValidity (): сообщает о невалидных полях. Используется в сочетании с preventDefault () в обработчике событий onSubmit;
* setCustomValidity (message): добавляет к элементу настраиваемое сообщение об ошибке.

**Реализация настраиваемого сообщения об ошибке**

Каждый раз, когда пользователь пытается отправить невалидную форму, браузер отображает сообщение об ошибке. Способ отображения этого сообщения зависит от браузера.

У подобных автоматических сообщений есть два недостатка:

* Не существует стандартного способа изменить их внешний вид с помощью CSS;
* Они зависят от языка, на котором работает браузер. Т.е., страница может быть на одном языке, но сообщение об ошибке отображается на другом языке.

**Пример использования** Constraint Validation API

HTML

<form>

        <label for="mail">Please, write your e-mail address:</label>

        <input type="email" id="mail" name="mail">

        <button>Submit</button>

      </form>

JS

const email = document.getElementById("mail");

email.addEventListener("input", function (event) {

  if (email.validity.typeMismatch) {

    email.setCustomValidity("I am expecting an e-mail address!");

  } else {

    email.setCustomValidity("");

  }

});

**Пример использования валидации средствами javaScript и Constraint Validation API без использования встроенной валидации браузера.**

HTML

 <form novalidate>

        <p>

          <label for="mail">

            <span>Please enter an email address:</span>

            <input type="email" id="mail" name="mail" required minlength="8">

            <span class="error" aria-live="polite"></span>

          </label>

        </p>

        <button>Submit</button>

      </form>

CSS

body {

    font: 1em sans-serif;

    width: 200px;

    padding: 0;

    margin : 0 auto;

  }

  p \* {

    display: block;

  }

  input[type=email]{

    -webkit-appearance: none;

    appearance: none;

    width: 100%;

    border: 1px solid #333;

    margin: 0;

    font-family: inherit;

    font-size: 90%;

    box-sizing: border-box;

  }

  /\* This is our style for the invalid fields \*/

  input:invalid{

    border-color: #900;

    background-color: #FDD;

  }

  input:focus:invalid {

    outline: none;

  }

  /\* This is the style of our error messages \*/

  .error {

    width  : 100%;

    padding: 0;

    font-size: 80%;

    color: white;

    background-color: #900;

    border-radius: 0 0 5px 5px;

    box-sizing: border-box;

  }

  .error.active {

    padding: 0.3em;

  }

JS

// There are many ways to pick a DOM node; here we get the form itself and the email

// input box, as well as the span element into which we will place the error message.

const form  = document.getElementsByTagName('form')[0];

const email = document.getElementById('mail');

const emailError = document.querySelector('#mail + span.error');

email.addEventListener('input', function (event) {

  // Each time the user types something, we check if the

  // form fields are valid.

  if (email.validity.valid) {

    // In case there is an error message visible, if the field

    // is valid, we remove the error message.

    emailError.textContent = ''; // Reset the content of the message

    emailError.className = 'error'; // Reset the visual state of the message

  } else {

    // If there is still an error, show the correct error

    showError();

  }

});

form.addEventListener('submit', function (event) {

  // if the email field is valid, we let the form submit

  if(!email.validity.valid) {

    // If it isn't, we display an appropriate error message

    showError();

    // Then we prevent the form from being sent by canceling the event

    event.preventDefault();

  }

});

function showError() {

  if(email.validity.valueMissing) {

    // If the field is empty,

    // display the following error message.

    emailError.textContent = 'You need to enter an e-mail address.';

  } else if(email.validity.typeMismatch) {

    // If the field doesn't contain an email address,

    // display the following error message.

    emailError.textContent = 'Entered value needs to be an e-mail address.';

  } else if(email.validity.tooShort) {

    // If the data is too short,

    // display the following error message.

    emailError.textContent = `Email should be at least ${ email.minLength } characters; you entered ${ email.value.length }.`;

  }

  // Set the styling appropriately

  emailError.className = 'error active';

}

**Валидация клиентского ввода без использования, встроенного API**

В некоторых случаях, например, при работе с устаревшим браузером или пользовательскими элементами управления, вы не сможете или не захотите использовать API проверки ограничений. В этом случае, Вы можете использовать JavaScript, но его придется писать вручную ☺.

Пример простейшей валидации на javaScript Вы можете найти примере, который загружен в MS Teams.

HTML

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

    <title></title>

    <link href="StyleSheet.css" rel="stylesheet" />

    <script src="JavaScript.js"></script>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css" integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">

</head>

<body>

    <section class="wrapper">

        <!--Header with Logo strats here-->

        <div class="container-fluid">

            <div class="row">

                <div class="col-md-12 logo"><img src="https://thumbs.dreamstime.com/z/d-image-yellow-taxi-41596598.jpg" alt="logo" width="150" /></div>

            </div>

            <div class="row">

                <div class="col-md-12 text">To make an order, please enter information in the form below...</div>

            </div>

        </div>

        <!--End Header-->

        <!--Layout for the form starts here-->

        <form id="form" action="#">

            <div class="container">

                <div class="row">

                    <!--Begining of the Main Form-->

                    <div class="col-md-8 orderInfo">

                            <h5>Order information</h5>

                            <div class="row">

                                <div class="col-md-4">First Name:</div>

                                <div class="col-md-4"><input class="form-control" id="first" type="text" onblur="checkFirst();" required /></div>

                                <div class="col-md-4"><span id="first\_Check"></span></div>

                            </div>

                            <div class="row">

                                <div class="col-md-4">Last Name:</div>

                                <div class="col-md-4"><input class="form-control" id="last" type="text" onblur="checkLast();" required /></div>

                                <div class="col-md-4"><span id="last\_Check"></span></div>

                            </div>

                            <div class="row">

                                <div class="col-md-4">Phone number:</div>

                                <div class="col-md-4"><input class="form-control" id="phone" type="text" onblur="checkPhone();" placeholder="xxx-xxx-xxx" required /></div>

                                <div class="col-md-4"><span id="phone\_Check"></span></div>

                            </div>

                            <div class="row">

                                <div class="col-md-4">Destination:</div>

                                <div class="col-md-4"><input class="form-control" id="dest" type="text" onblur="checkDest();" required /></div>

                                <div class="col-md-4"><span id="dest\_Check"></span></div>

                            </div>

                            <div class="row">

                                <div class="col-md-4">Date and Time:</div>

                                <div class="col-md-4">

                                    <input class="form-control" id="date" type="date" required />

                                    <input class="form-control" id="time" type="time" required />

                                </div>

                                <div class="col-md-4"></div>

                            </div>

                            <div class="row">

                                <div class="col-md-4">Quantity:</div>

                                <div class="col-md-4"><input class="form-control" id="quantity" type="number" min="1" max="5" required /></div>

                                <div class="col-md-4"></div>

                            </div>

                            <div class="row">

                                <div class="col-md-4">Payment method:</div>

                                <div class="col-md-4">

                                    <select class="form-control" name="paymethod" id="paymethod" required>

                                        <option>Cash</option>

                                        <option>Card</option>

                                        <option>PayPal</option>

                                    </select>

                                </div>

                                <div class="col-md-4"></div>

                            </div>

                            <div class="row">

                                <div class="col-md-12 order"><span id="sendOrder" onclick="outputData();">Send Order</span></div>

                            </div>

                    </div>

                    <!--End of the form-->

                    <!--Order summary-->

                    <div class="col-md-4 orderSummary">

                        <div class="col-md-12">

                            <h5>Order Summary</h5>

                            <span id="summary"></span>

                        </div>

                    </div>

                    <!--End of Order summary-->

                </div>

            </div>

        </form>

        <!--End of Layout for the form-->

    </section>

</body>

</html>

CSS

body {

    padding: 0;

    margin: 0;

    font-family: Arial;

    font-size: 1.0em;

    color: black;

}

.wrapper {

    margin: auto;

    width: 100%;

    min-height: 100vh;

    padding-top: 50px;

    /\*background-color: rgb(128, 128, 128);\*/

    padding: 10px;

}

#form {

    width: 100%;

    margin: auto;

}

.row {

    margin: 10px 10px;

}

.logo {

    width: 100%;

    text-align: center;

    background-color: white;

}

.text {

    width: 70%;

    height: 40px;

    text-align: center;

    font-size: 1.1em;

}

.order {

    padding-top: 20px;

}

#sendOrder {

    width: 150px;

    height: 40px;

    background-color: rgb(248, 146, 38);

    text-align: center;

    padding-top: 5px;

    border: 1px solid gray;

    border-radius: 5px;

    display: block;

    margin: auto;

}

    #sendOrder:hover {

        cursor: pointer;

    }

.orderInfo {

    border-bottom: 1px solid gray;

    border-top: 1px solid gray;

    padding: 10px;

}

.orderSummary {

    border-bottom: 1px solid gray;

    border-top: 1px solid gray;

    padding: 10px;

}

JS

//Checking of First name field

function checkFirst() {

    let first = document.getElementById("first").value;

    let regex = /^[a-zA-Z\s]{2,15}$/;

    if (regex.test(first)) { // if testing of first is true

        document.getElementById("first\_Check").style.color = "green";

        document.getElementById("first\_Check").innerHTML = "Input correct :)";

        return true;

    }

    else {// if it's not

        document.getElementById("first\_Check").style.color = "red";

        document.getElementById("first\_Check").innerHTML = "Please, enter 2-15 characters";

        return false;

    }

}

//Checking of Last name field

function checkLast() {

    let last = document.getElementById("last").value;

    let regex = /^[a-zA-Z\s]{2,20}$/;

    if (regex.test(last)) { // if testing of first is true

        document.getElementById("last\_Check").style.color = "green";

        document.getElementById("last\_Check").innerHTML = "Input correct :)";

        return true;

    }

    else {// if it's not

        document.getElementById("last\_Check").style.color = "red";

        document.getElementById("last\_Check").innerHTML = "Please, enter 2-20 characters";

        return false;

    }

}

//Checking of Destination field

function checkDest() {

    let dest = document.getElementById("dest").value;

    let regex = /^[a-zA-Z\s\.\,\d\ ]{5,50}$/;

    if (regex.test(dest)) { // if testing of dest is true

        document.getElementById("dest\_Check").style.color = "green";

        document.getElementById("dest\_Check").innerHTML = "Input correct :)";

        return true;

    }

    else {// if it's not

        document.getElementById("dest\_Check").style.color = "red";

        document.getElementById("dest\_Check").innerHTML = "Please, enter 2-50 characters";

        return false;

    }

}

//Checking of Phone number field

function checkPhone() {

    let phone = document.getElementById("phone").value;

    let regex = /^\d{3}-(\d{3})-\d{3}$/;

    if (regex.test(phone)) { // if testing of first is true

        document.getElementById("phone\_Check").style.color = "green";

        document.getElementById("phone\_Check").innerHTML = "Input correct :)";

        return true;

    }

    else {// if it's not

        document.getElementById("phone\_Check").style.color = "red";

        document.getElementById("phone\_Check").innerHTML = "Input is incorrect!";

        return false;

    }

}

//Collect and output all the data

function outputData() {

    let first = document.getElementById("first").value;

    let last = document.getElementById("last").value;

    let dest = document.getElementById("dest").value;

    let phone = document.getElementById("phone").value;

    let date = document.getElementById("date").value;

    let time = document.getElementById("time").value;

    let dateAndTime = date + " at " + time;

    let quantity = document.getElementById("quantity").value;

    let paymethod = document.getElementById("paymethod").value;

    if (first != "" && last != "" && dest != "" && phone != "" && date != "" && time != "" && quantity != "") {

        document.getElementById("summary").style.color = "black";

        document.getElementById("summary").innerHTML = "Hello, " + first + " " + last + "!";

        document.getElementById("summary").innerHTML += "<p>Your order was successfully placed. We will contact you as soon as possible. You can see the details of your order below: </p>";

        document.getElementById("summary").innerHTML += "<p>Phone number: " + phone + ".</br>";

        document.getElementById("summary").innerHTML += "Destination: " + dest + ".</br>";

        document.getElementById("summary").innerHTML += "Date and Time: " + dateAndTime + ".</br>";

        document.getElementById("summary").innerHTML += "Number of people: " + quantity + ".</br>";

        document.getElementById("summary").innerHTML += "Payment Method: " + paymethod + ".</p>";

        document.getElementById("summary").innerHTML += "<p style='font-weight: bold;'>We are glad that You use our company's services :)</p>";

    }

    else {

        document.getElementById("summary").style.color="red";

        document.getElementById("summary").innerHTML = "Please, fill in all the fields...";

    }

}