**Какое основное назначение технологии Windows Forms, ее особенности, преимущества и недостатки?**

Windows Forms интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью фреймворка

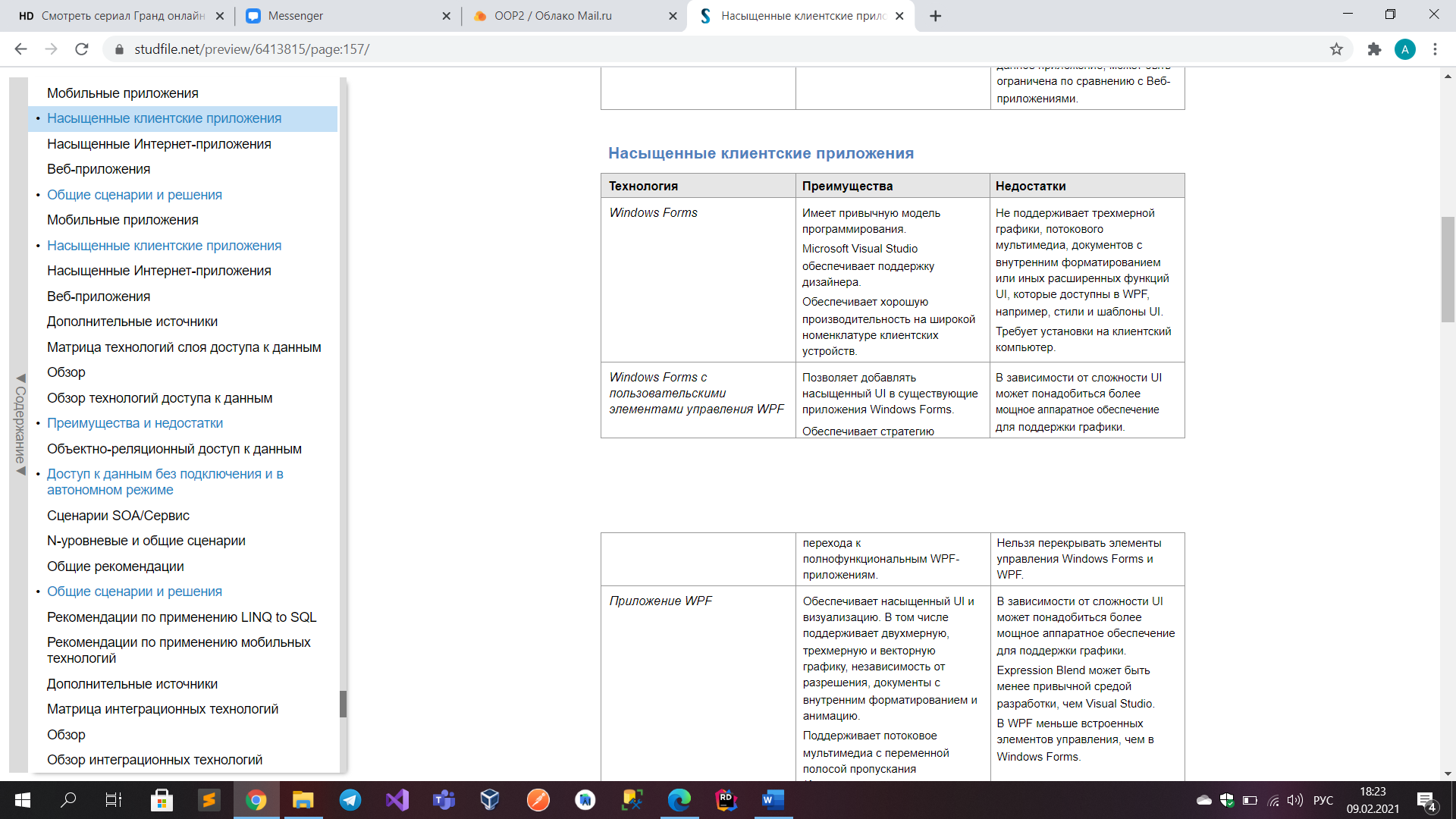
Форма — это объект, который задается свойствами, определяющими их внешний вид, методами, определяющими их поведение, и событиями, определяющими их взаимодействие с пользователем. System.Windows.Forms.Form

Особенности:

Единообразная обработка всех форм приложения. Чтобы единообразно обработать все формы приложения, нужно подписаться на глобальные события.

Реализация интерфейсов независимыми и зависимыми формами и отношения между ними. Диаграмма классов, представленная ниже, показывает, как относятся между собой независимые и зависимые формы списков, универсальные, а также реализуемые на основе базовых конкретные формы списков (могут реализовываться как вручную, так и посредством генераторов кода для Flexberry):

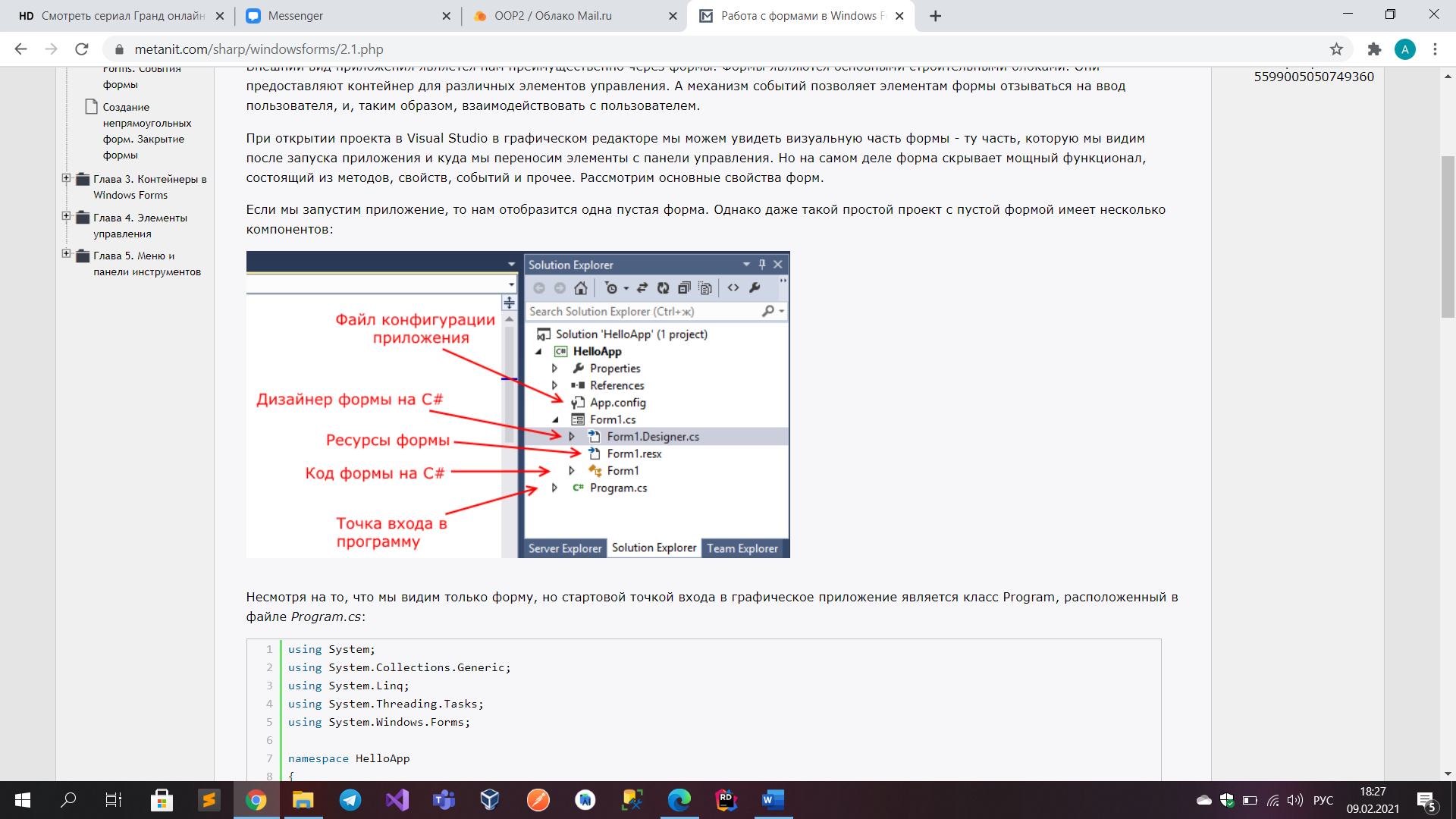
Положение формы на экране. В некоторых случаях восстоновление положения формы приводило к тому, что форма не была видна на экране. Такая ситуация, например, возникает при смене разрешения монитора на более низкое: при новом разрешении форма могла оказаться за пределами видимой области.



**2. Зачем используется класс Form? Назовите основные методы, свойства и события данного класса.**

Представляет окно или диалоговое окно, составляющее пользовательский интерфейс приложения.

**3. Поясните структуру проекта и назначение всех файлов?**



**4. Зачем нужен атрибут STAThreadAttribute?**

STA-single-threaded apartment. Это означает, что все потоки в этой программе выполняются в рамках одного процесса, а управление программой осуществляется одним главным потоком.

**5. Как в вашем проекте используются события и делегаты?**

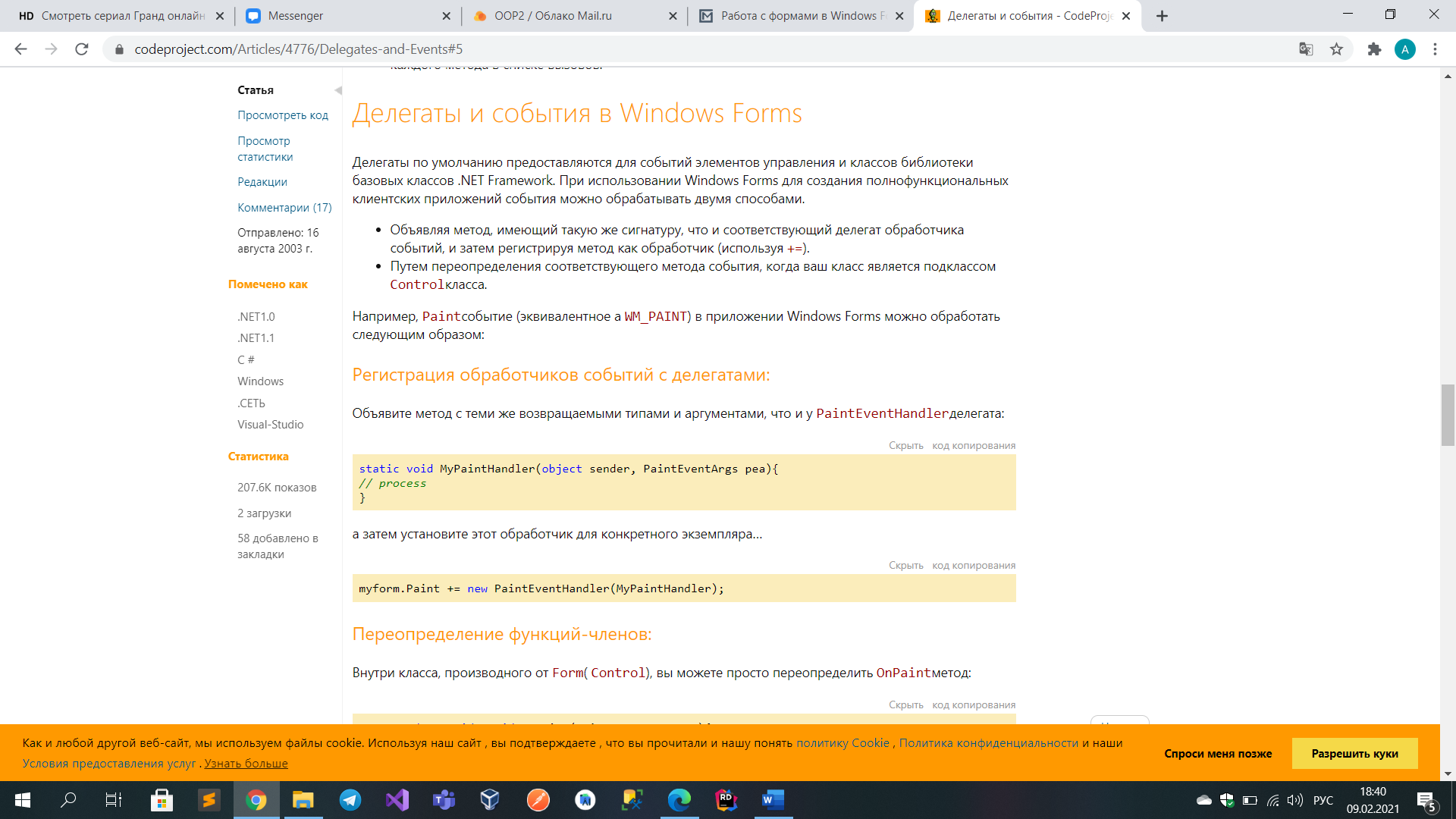
Для взаимодействия с пользователем в Windows Forms используется механизм событий. События в Windows Forms представляют стандартные события на C#, только применяемые к визуальным компонентам и подчиняются тем же правилам, что события в C#. Но создание обработчиков событий в Windows Forms все же имеет некоторые особенности.

Прежде всего в WinForms есть некоторый стандартный набор событий, который по большей части имеется у всех визуальных компонентов. Отдельные элементы добавляют свои события, но принципы работы с ними будут похожие.

Чтобы добавить обработчик, можно просто два раза нажать по пустому полю рядом с названием события, и после этого Visual Studio автоматически сгенерирует обработчик события. Например, нажмем для создания обработчика для события Load:

Для добавления обработчика используется стандартный синтаксис C#: this.Load += new System.EventHandler(this.Form1\_Load)

Поэтому если мы захотим удалить созданный подобным образом обработчик, то нам надо не только удалить метод из кода формы в Form1.cs, но и удалить добавление обработчика в этом файле.



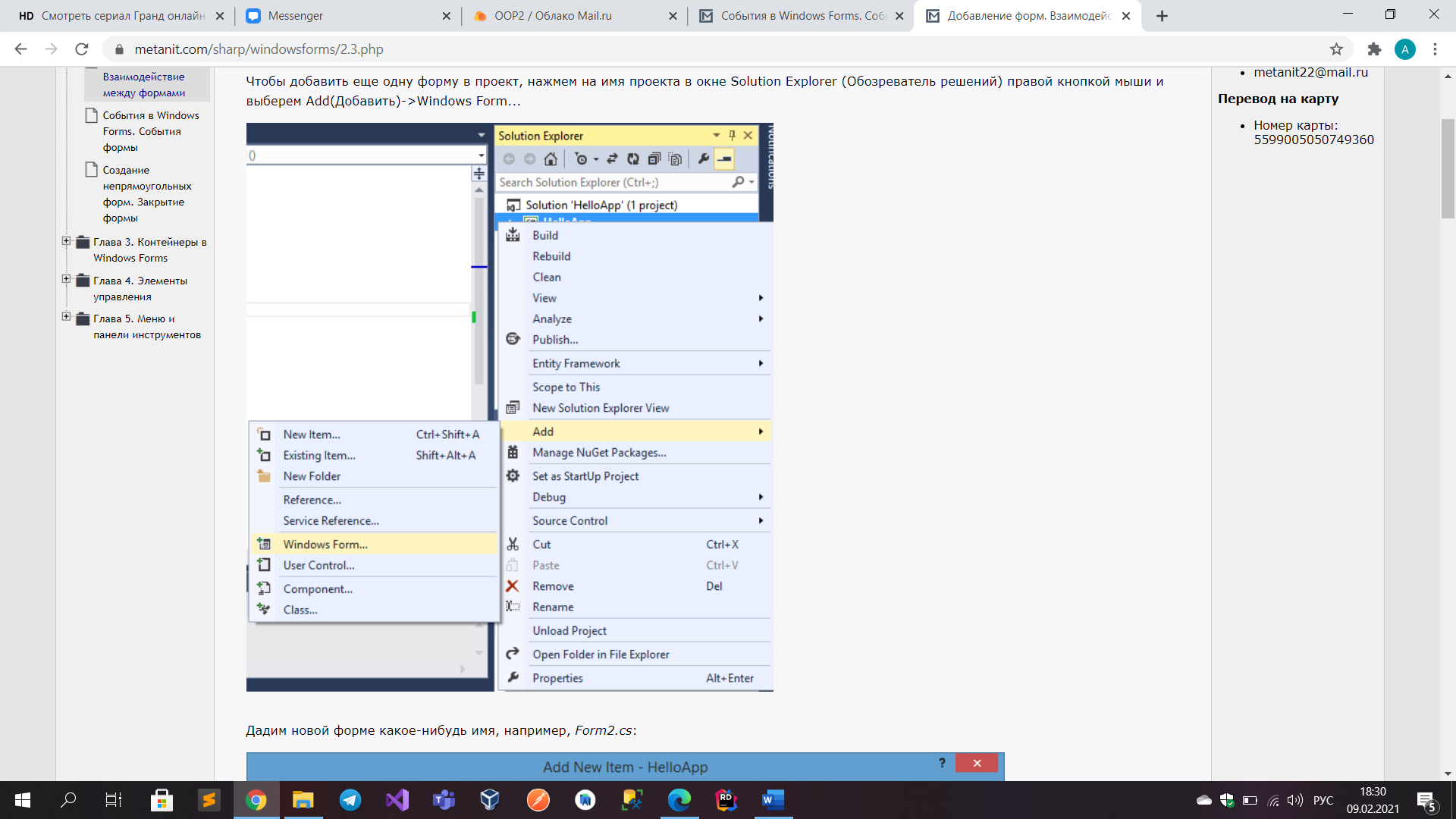
https://www.codeproject.com/Articles/4776/Delegates-and-Events#5

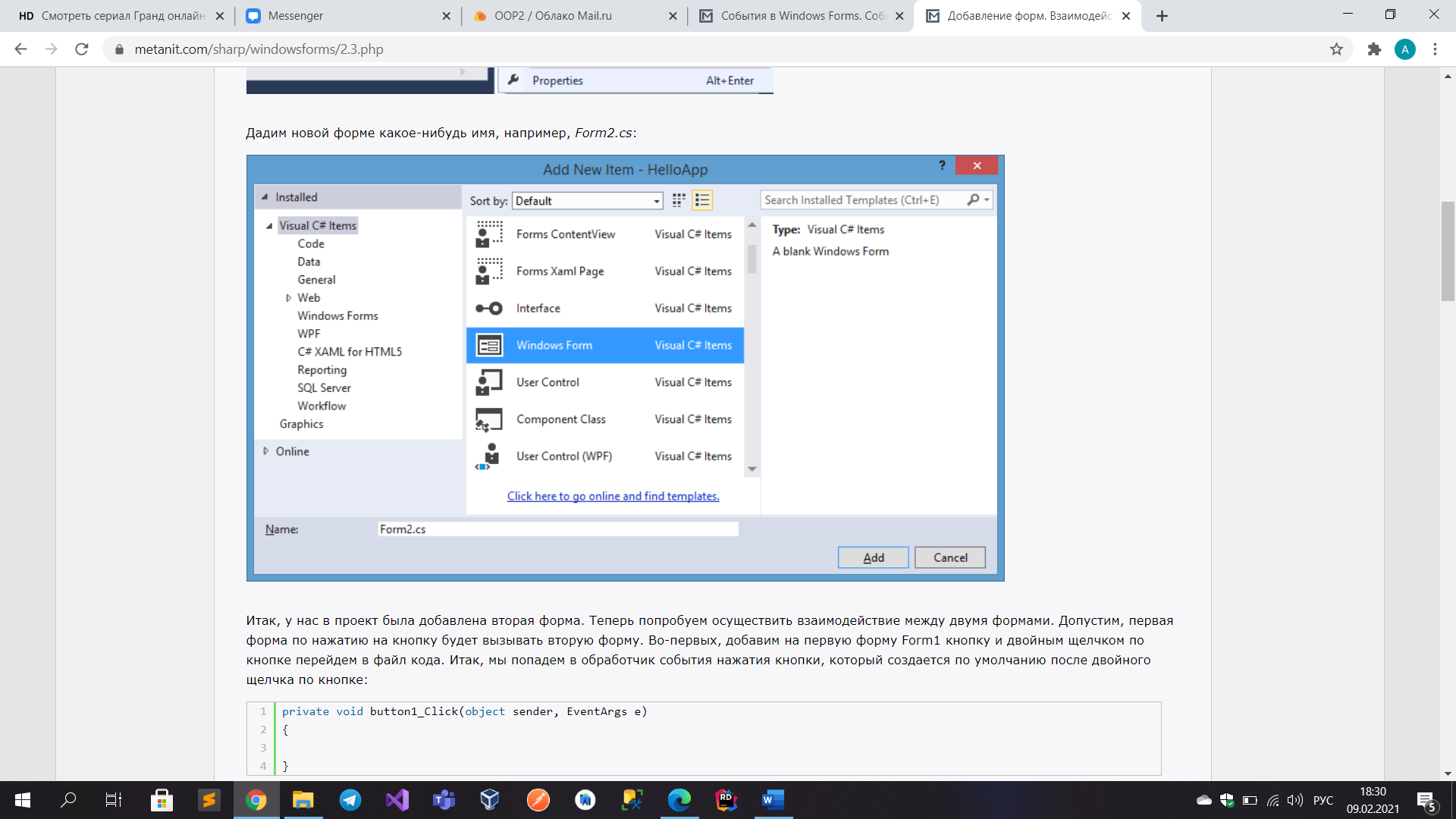
**6. Объясните схему работы цепочек делегатов.**

**7. Объясните механизм подписки и отмены подписки на события.**

**8. Как создать вторую форму и передать туда данные? Есть ли другие способы?**

Добавление форм. Взаимодействие между формами 1) Создать форму 2) код





9. Как во время выполнения приложения добавить/удалить элемент управления?

### Распространенные задачи при разработке приложений — это добавление элементов управления и удаление элементов управления из любого контейнера в формах (например, [Panel](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.panel) или [GroupBox](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.groupbox) , или даже самой формы). Во время разработки элементы управления можно перетаскивать непосредственно на панель или в группу. Во время выполнения эти элементы управления поддерживают коллекцию Controls, которая отслеживает размещенные в них элементы управления.

### Программное добавление элемента управления в коллекцию

1. Создайте экземпляр элемента управления, подлежащий добавлению.
2. Задайте свойства нового элемента управления.
3. Добавьте этот элемент управления в коллекцию Controls родительского элемента управления.

### Программное удаление элементов управления из коллекции

1. Удалите обработчик событий из события. В Visual Basic используйте ключевое слово [оператора RemoveHandler](https://docs.microsoft.com/ru-ru/dotnet/visual-basic/language-reference/statements/removehandler-statement) . в C# используйте [оператор-=](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/subtraction-operator).
2. Используйте метод Remove для удаления требуемого элемента управления из коллекции Controls панели.
3. Вызовите [Dispose](https://docs.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.dispose) метод, чтобы освободить все ресурсы, используемые элементом управления.