CS411 Short Notes
Chapter No 24 Short Notes

1). Commands are a more robust and loosely coupled version of:
- Events

2). WPF defines a number of
- Built-in Commands

3). Any object can work as a command by:
- Implementing icommand

4). icommand includes:
- Execute
- CanExecute
- CanExecuteChanged

5). For cut, copy and paste we can define and implement three classes mplementing
- icommand

6).Pre-defined Built-in Commands:
- Application Commands
- Component Commands
- Media Commands
- Navigation commands
- Editing Commands

7). All instances of RoutedUiCommand implement icommand and support bubbling

8). Custom Commands don't get the treatment of CommandConverter.

9). You can add KeyBinding and MouseBinding yourself.

10). Example of KeyBinding:

- This.InputBinding.Add(New KeyBinding(ApplicationCommands.Help, new KesyGeasture(Key.F2)));
- <Windows.InputBindings> <KeyBinding Command = “Help” Key = “F2”> <KeyBinding Command = “NotaCommand” Key = “F1”> </Windows.InputBindings>
- This.InputBinding.Ass(New KeyBinding(ApplicationCommands.NotaCommand, new KeyGesture(Key.F1)));

11) TextBox is an example of Control with builtin command binding that responds to ctrl-z etc.

12) WPF Input Events make possible rich interactive content.

13) Application Commands:

Close	Save
Copy	SaveAll
New	Stop
Open	Undo
Cut	Delete
Print	Find
Properties	Replace
Save	SaveAs
Stop	Undo

14) Component Commands:

MoveDown	ScrollPageRight
MoveLeft	ScrollPageUp
MoveRight	SelectToHome
SelectToEnd	MoveUp
ScrollByLine	SelectToPageDown
ScrollPageDown	SelectToPageUp
ScrollPageLeft	

15) Media Commands:

ChannelDown	ChannelUp
DecreaseVolume	FastForward
IncreaseVolume	MuteVolume
NextTrack	Pause
Play	PreviousTrack
Record	Rewind

Select	Stop

16) Navigation Commands:

BrowseBack	BrowseForward
BrowseHome	BrowseStop
Favorites	FirstPage
GoToPage	LastPage
NextPage	PreviousPage
Refresh	Search
Zoom	

17) Editing Commands:

AlignCenter	AlignJustify
AlignLeft	AlignRight
CorrectSpellingError	DecreaseFontSize
DecreaseIndentation	EnterLineBreak
EnterParagraphBreak	IgnoreSpellingError
IncreaseFontSize	IncreaseIndentation
MoveDownByLine	MoveDownByPage
MoveDownByParagraph	MoveLeftByCharacter
MoveLeftByWord	MoveRightByCharacter
MoveRightByWord	

Chapter NO.25 Short Notes

1).WPF Window is a:
- win32 window

2). Any Number of child windows can be made by
- instantiating a window derived class and calling show

3). Child windows gets closed or minimized:
- on closing or minimizing of parent window

4). Window Events include:
- Activated
- Deactivated

5).Using Window.Show() a window appears and then disappears

6). StartupUri in xaml is
<Application x:Class = "PhotoGallery.App"
xmlns = http://schemas.microsoft.com.winfx/2006/xaml/presentation
xmlns: x = http://schemas.microsoft.com/winfx/2006/xaml
StartupUri = "MainWindow.xaml"/>

7). main() is created dynamically

8). System.Environment.GetCommandLineArgs is used to get command line arguments in WPF

9). Application Level Events are:
- Startup
- Exit
- Activated
- Deactivated
- SessionEnding

10). Window.show() is used to instantiate (Read-only) window.

11). Properties Dictionary is a simple Dictionary used to share things among different windows.

12). Application.Current() is used to take current window from any app.

13). Dispatcher.Run() is used to make multiple UI threads.

14). Single Instance App:

```
Bool mutexIsNew;
using (System.Threading.Mutex m = New System.Threading.Mutex(True, uniqueName, out
    mutexIsNew))
    {
        if(mutexIsNew)
        // This is the first instance. Run the application
        else
        // There is already an instance running
}
```

15). Dialogs are just child windows

16).There are two types of dialogs:

- Modal
- Modeless
-

17). Modeless Dialog are more like normal Windows.

18). Modal Dialog are such windows that are necessary to terminate explicitly in order to interact with main window.

19). In modeless Window we can work in parallel with main window just like using Tools window in Photoshop.

20).Common Dialogs are actually provided by win32 and are part of Modal dialog.
21). Custom Dialogs are user-defined.

22). ShowDialog() returns us Null-able Boolean with options True/False/Null.

23). Apllication Deploy include:

- ClickOnce

- Windows Installer

24). EULA stand for End User License.

25). Benefits of Windows Installer:

- showing custom setup UI like EULA
- Control over where files are installed
- Arbitrary code at install time
- Register COM components and File Association
- Install for All users
- Installation From CD

26). Benefits of ClickOnce:

- Built-in support for automatic updates and roll back to previous version
- A web-like go-away experience
- Start-menu or control panel program list
- All files in isolated area
- Clean Un-installation

Chapter NO.26 Short Notes

1) Wizard is a Dialog Box with multiple User-Interfaces.

2). Difference Between Page and Window:
Page in itself cannot Display. Window is an area containing page.

3). Child windows gets closed or minimized:
- on closing or minimizing of parent window

4). Navigation Based apps are:
- Windows Explorer
- Media Player
- Photo Gallery

5). Host Windows are:
- Navigation Window
- Frame

6). Navigation Containers Provide:
- Navigating
- History Journal
- Navigation Related events

7). Difference between Navigation Window and Frame:
Navigation window gives you a navigation Bar by default while Frame does not. Navigation Window is more like a top-level window wherease Frame more like an HTML frame or iFrame.

8). Page does everything that windows does except OnClosed and OnClosing.

9). Navigation can also happen between HTML files.

10). A page can interact with its Navigation Container by using NavigationService Class.

11). You can get an instance of NavigationService by calling the static NavigationService.GetNavigationService method and passing an instance of the page.

12). this.NavigationService.Refresh() results in reload of current page.

13). Performing navigation in 3 main ways:

- calling navigation method
- Using Hyperlinks
- Using the Journal

14). Properties of page:

- WindowHeight
- WindowWidth
- WindowTitle

15). Example Of Navigate Method:

```
PhotoPage nextPage = new PhotoPage();
this.NavigationService.Navigate(nextPage);
```

16). Navigate to a page via Uri

```
this.NavigateService.Navigate(new Uri("photoPage.xaml", uriKind.Relative));
```

17). Example of navigation using Hyperlink.

```
<textblock>
Click
<Hyperlink NavigateUri = "photoPage.xaml">
Here
</Hyperlink>
to view the photo
</textblock>
```

18). Journal provides logic behind back and frwd.

19). Two internal stacks are:

- Undo
- Redo

20).Navigation Window always has a journal but frame may depending on its JournalOwnership = OwnsJournal, UseParenJournal, Automatic

21). When frame has journal it has back/frwd buttons but can set NavigationUiVisibility = Hidden

22). Navigation Events that raise when loading first page:
See in handouts

23). Navigation Events that raise when navigating between pages.
See handouts

23). NavigationIsstopped is an event called instead of LoadCompleted if an error occurs or navigation is cancelled.

24). Html to html event firing is not allowed.

25). PageFunction has special mechanism for returning data.

Chapter NO.27 Short Notes

1). Browser based Applications are partial trust applications.

2). Partial Trust means with some restrictions.

3). Basic benefit of Silverlight is that it is cross-platform. But WPF browser based applications just run on windows.

4). In Web Based applications the journal of Application and Web Browser are integrated.

5). GAC is a Global Assembly Cache

6). ClickOnce Applications are chached.

7). How to open file from local system:

```
String filecontents = null;
OpenFileDialog ofd = new openFileDialog();
if(ofd.ShowDialog()==true)
{
using(stream s = ofd.OpenFile())
using(StreamReader sr = new StreamReader())
{
filecontent = sr.ReadToEnd();
}
}
```

8). Data can be given to any web site using two method:

- URL Parameter
- Cookies

9). BrowserInteropHelper.Source to retrieve the complete URL.

10). Browse Cookies retrieved by Application.GetCookie method.

11). Two steps of making full-trusted Web Browser App.

- Change:

<targetzone>Internet</targetzone>

to

<targetzone>Custom</targetzone>

- Add:

<permissionset class = "System.Security.Permissionset" version = "1" ID = "Custom" samesite = "site" Unrestricted = "true"/>

12). To publish a web based application use VS publishing wizard or mage tool in SDK and copy files to webserver which must be configured to serve it.

13). Users can install and run a web based application just by navigating to a URL.

14). Resources can be:

- Binary
- Logical

15). .Net has binary resources in WPF.

16). Binary resources can be:

- Embedded(part of exe)
- Stored at Known place

17). Localization:

Localization means that your application can target more than one target languages or target Grammers.

18). In Order to localize resources its necessary to make them embedded resources.

Chapter NO.28 Short Notes

1) How to specify Default Culture of Application:

```
<project>
<propertygroup>
<uiculture>en-US</uiculture>
[assembly: NautralResourceLanguage("en-US",
    UltimateResourceFallBackLocation.Satellite]
```

2). Logical Resources are arbitrary .net objects stored and named in an element Resource Properties.

3). By using logical resources we can save change in one place and have different effects. Like you can change brushes in one place and have different effects .

4). Logical Resources can further be categorized in "Static Resources" and "Dynamic Resources".

5). In Dynamic Resources we actually subscribe to the updates of the resource.

6). In static resource we just want its value once.

7). Static Resource markup extension walks the logical tree or even application level or system Resources.

8). Example of Static Resource:

```
<windows xmlns:http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns: x = http://schemas.microsoft.com/winfx/2006/xaml>
<window.Resources>
<Image x: key = "Zoom" Height = "21" Source = "Zoom.gif"/>
</window.Resources>
<StackPanel>
<StaticResource ResourceKey = "zoom"/>
</StackPanel>
</window>
```

9). Static Resources do not support forward references.

10). Static Resources do not support Sharing.

11). Procedural Code/Static Resource:

```
window.Resources.add("BackgroundBrush", new SolidColorBrush(Colors.Yellow));
Window.Resources.Add("BorderBrush", new  SolidColorBrush(Colors.Red));
Button button = new Button();
StackPanel.Children.Add(button);
Button.Background = (Brush)button.FindResource("BackgroundBrush");
Button.BorderBrush = (Brush)button.FindResource("BorderBrush);
```

12). Procedural Code/Dynamic Resource

```
Button button = new Button();
Button.SetResourceReference(Button.BackgroundProperty, "BackgroundBrush");
Button.SetResourceReference(Button.BorderBrushProperty, "BorderBrush");
Button button = new Button();
Button.Background = (Brush)window.Resources["BackgroundBrush"];
Button.BorderBrush = (Brush)window.Resources["BorderBrush "];
```

13). The key benefit of logical resources is that Use and definition becomes seprate.

14). System Resources are such resources which are shared by al application For Example: System Color, System font, System Parameters.

Chapter NO.29 Short Notes

1). Data Binding allows to declaratively bind two different properties in xaml.

2). Binding Object has one source and target Object.

3). Binding binds two propertiews together and keeps a communication channel open.

4). Example code of binding:

```
public mainWindow()
{
InitializeComponent();
Binding binding = new Binding();
Binding.Source = TreeView;
Binding.Path = new PropertyPath("SelectedItem.Header");
CurrentFolder.SetBinding(TextBlock.TextProperty, binding);
}
```

5). Property.Pah can also use

- BindingOperation.SetBinding
- BindingOperation.ClearBinding
- BindingOperation.ClearAllBindings

6). The DisplayMemberPath property can be used to improve display:

```
<listbox x:Name= "pictureix" DiaplayMemberPath = "Name"
ItemSource = "{Binding source ={StaticSource photos}}">
</listbox>
```

7). Synchronization means binding multiple targets with single source.

8). Example Code Of Synchronization:

```
<listbox IsSynchronizedWithCurrentItem = "True" DisplayMemberPath = "Name"
ItemSource = "{Binding source={StaticSource  photos}}"
</listbox>
<listbox IsSynchronizedWithCurrentItem = "True" DisplayMemberPath = "DateTime"
ItemSource = "{Binding source={StaticSource  photos}}"
```

```
</listbox>
<listbox IsSynchronizedWithCurrentItem = “True” DisplayMemberPath = “Size”
ItemSource = “{Binding source={StaticSource  photos}}”
</listbox>
```

9). Data Context is a method of defining Default Data Source.

10). example Code of Data Context:

```
<stackpanel DataContext = “{StaticSource   photos}”>
<label x:Name =”numitemslabel”
content = “{Binding  Path=Count}”/>
<listbox s:Name = “picturebox” DisplayMemberPath = “Name”
Itemsource = “{Binding}”>
</listbox>
</stackpanel>
```

11). Three Ways of Control Rendering are:

- String formatting
- Data Template
- Value Converter

12). String Formatting can be used even without Data Binding

13). With Data Template User Interface are auto-applied to arbitrary .net object when it is rendered.

14). Example Code of Data Template:

```
<listbox x:Name = “picturebox”
ItemSource = “{Binding  Source={StaticResource  photos}}”>
<listbox.ItemTemplate>
<DataTemplate>
<Image Source = “placeholder.jpg” Height = “35”/>
</DataTemplate>
</listbix.ItemTemplate>
</listbox>
```

15). With Data Template there is an implicit DataContext.

16). Value Converter is a custom Logic that morphs Source value into Target Type.

Chapter NO.30 Short Notes

1). View Supports:

- Sorting
- Grouping
- Filtering
- Navigation

2). Example of sorting on one field:

```
SortDescription sort = new SortDescription("Name", ListSortDirection.Ascending);
```

3). Example of sorting on two Different Fields:

```
View.SortDescription.Add(new SortDescription("DateTime",
    ListSortDirection.Descending);
View.SortDescription.Add(new SortDescription("Name", ListSortDescription.Ascending);
```

4). Method of Getting Default View:

```
IcollectionView view = CollectionViewSource.getDefaultSource(this.FindSource("photos"));
```

5). Example of Grouping:

```
IcollectionView view = CollectionViewSource.getDeafultView(this.FindSource("photos"));
View.GroupDescriptions.Add(new PropertyGroupDescription("DateTime"));
```

6). Example Of Filtering:

```
IcollectionView view =
    CollectionViewSource.getDefaultView(this.FindResource("photos"));
View.Filter = delegate(object o){
Return((o As photo).DateTime-DateTime.Now).Days<=7;
};
```

7). Example of Navigation in View:

```
void previous_Click(Object sender, Routedevent e)
{
IcollectionView view =
    CollectionViewSource.getDefaultView(this.FindResource("photos"));
view.MoveCurrentToPrevious();
```

```
if(view.IsCurrentBeforeFirst)
view.MoveCurrentToLast();
}
void next_Click(Object sender, Routedevent e)
{
IcollectionView view =
    CollectionViewSource.getDefaultView(this.FindResource("photos"));
view.MoveCurrentToNext();
if(view.IsCurrentAfterLast)
view.MoveCurrentToFirst();
}
```

8). Binding to the whole object:
"{Binding Path = /}"

9). Binding to the DateTime Property
"{Binding path=/DateTime}"

10). Binding on a current item of a different Data Source:
"{Binding Path=Photo/}"

11). Binding to the DateTime Property of different data source:
"{Binding Path=Photos/DateTime}"

12). Sorting is always applied before Grouping

13). Navigation means managing the current item in view.

14). If you have multiple target elements connected to the same Custom View then their IsSynchronizedWithCurrentItem = "True" by default and it is "false" in Default View.

15). Data Providers are kind of classes which made it simple for us to access certain kind of data.

16). There are two types of Data Providers:

- xaml Data Providers
- Object Data Providers

Chapter NO.31 Short Notes

1). Object Data Provider provides us some facility which are useful for binding to objects which are not designed for binding.

2). Asynchronous Data Binding means that the property that you want to access should be done in background.

3). xmlDataProvider and objectDataProvider have an IsAsynchronous property. Which is false by default on ObjectDataProvider and true by default for xmlData Provider.

4). Example of Object Data Provider:

```
<window.Resources>
<local:Photos x:key = “photos”/>
<ObjectDataProvider x:key = “dataprovider”
ObjectInstance = “{StaticSource   photos}”/>
</window.Resources>
```

5). Binding to a method is useful for classes that are not designed for Data Binding.

6). Convert Method is used when bringing data from Source to Target

7). Convert Back is used when bringing data from target to source.

8). Binding Modes:

- One-Way
- Two-Way
- OneWayToSource
- OneTime

9). Example of Method Data Binding:

```
<ObjectDataProvider.ConstructorParameter>
<Sys:Int32>23</sys:Int32>
</ObjectDataProvider.ConstructorParameters>
</ObjectDataProvider>
```

<ObjectDataProvider x:key=”DataProvider”
ObjectType=”{x:Type local:photos}”
MethodName = “getFolderName”/>

10). One-Way Binding means the target is updated whenever the source changes.

11). Two-Way Binding means change to either the target or source updates the other

12). OneWayToSource is the opposite of One-Way binding. the source is changed whenever the target changes.

13). One-Time binding is like one-way except changes to the source are not reflected at the target. The target retains a snapshot of the source at the time the Binding is initiated.

14). when Do you want the two way source to be updated:

- PropertChanged
- LostFocus
- Explicit

15). Validation Rules are just like simple classes that are used to ensure the proper working of the application.

16). Binding has ValidationRules Property that can be set to one or more validation rules

17). ExceptionValidationRule says that you can update a source if updating a source does not cast an exception.

Chapter NO.32 Short Notes

1). Concurrency means doing two things at the same time.

2). Purposes of Concurrency are:

- Responsive User Interface
- Simultaneous Requests
- Parallel Programming

3). Time slicing means giving time to a thread then we slice and run another thread then come back and so on.

4). Creating a Simple thread in C#:

```
Class ThreadTest
{
Static void main()
{
Thread t = new Thread(WriteY);
t.Start();
for(int i = 0; i<1000; i++) Console.Write("x");
}
Static void WriteY()
{
for(int I = 0; i<1000; i++) Console.Write("y");
}
}
```

5). A Thread isAlive = true once starts and until end.

6). Thread.CurrentThread is currently executing thread.

7). When you call t.join() it means you want to wait for t to finish.

8). Sleep() means a thread wants to give away the CPU.

9). Local Variables are kept on stack.

10). Locks are a way to ensure that if you are in area of a code then no other thread will be allowed to enter that area of thread. And are used when threads are sharing shared data.

11). Example of Lock:

```
Class ThreadSafe
{
Static Bool _done;
Static ReadOnly Object _locker = new object();
Static void main()
{
new Thread(Go).Start();
Go();
}
Static void Go()
{
Lock(_locker)
{
if(!_done){Console.WriteLine("Done"); _done = true;}
}
}
}
```

12). Example of passing Data to Threads:

```
 Static void main()
{
Thread t = new Thread(()=> Print("Hello From t!"));
t.Start();
}
Static void Print(String message)
{Console.WriteLine(message);}
```

13). Lambda ()=> is used to pass data to threads.

14). for(int i = 0; i<10; i++)
new Thread(()=>Console.Write(i)).Start(); will result in some typical output like 0223557799 just because of sharing a common variable and thread concurrency

15). above code can be rectified as follows:

```
for(int i = 0; i<10;  i++)
{
int temp = i;
new Thread(()=> Console.Write(temp)).Start();
}
```

Chapter NO.33 Short Notes

1). BeginInvoke and Invoke are such methods that can be called from any other thread.

2). BeginInvoke is a low-level primitive

3). SynchronizationContext is a higher level primitive.

4). Example of BeginInvokde:

```
Partial Class mywindow: Window
{
public mywindow()
{
InitializeComponent();
new Thread(work).start();
}
void work()
{
Thread.Sleep(5000);
UpdateMessage("The Answer");
}
void UpdateMessage(String message)
{
Action action=()=>TextMessage.Text = message;
Dispatcher.BeginInvoke(action);
}
}
```

5). SynchronizationContext can be used for generalized Thread Marshaling.

6). Thread.CurrentThread is currently executing thread.

7). Thread.CurrentThread.IsThreadPoolThread is used to check whether the current thread is running in ThreadPool or independently.

8). There are two ways of calling thread from ThreadPool:

- ThreadPoool.QueueUserWorkItem(notUsed=> Console.WriteLine("Hello"));
- Task.Run(()=>Console.WriteLine("Hello From Task"));

9). ThreadPool saves the overhead of thread creation, management and termination.

10). ThreadPool creates or reduces real threads using hillclimbing algo to maximize CPU usage and reduce time slicing.

11). Threads of ThreadPool are always Background threads which means they are not going to be deleted they are just waiting for work items.

12). Three benefits of Tasks are:
- Easy return value from threads
- Exception Handling
- Continuation

13). Starting a task is like creating a thread except started right away(hot) means that there is no need to explicitly start a task.

14). Example of Task.Wait()

```
Task task = Task.Run(()=>
{
Thread.Sleep(2000);
Console.WriteLine("Foo");
});
Console.WriteLine(task.IsCompleted);
task.Wait();
```

15). Example of exception handling with tasks:

```
Task task = Task.Run(()=>{throw null;});
Try
{
Task.Wait();
}
catch(AggregateException aex)
{
if(aex.InnerException is NullReferenceException)
Console.WriteLine("Null");
    else
```

```
throw;
}
```

16). Create a task which you can wait and attach continuations. Controlled by the following operations:

```
 public class TaskCompletionSource<TResult>
{
public void SetResult(TResult result);
public void SetException(Exception exception);
public void SetCanceled();
public bool TrysetResult(TResult result);
public bool TrySetException(Exception exception);
public bool TrySetCanceled();
}
```

17). What would be the result of following piece of code:

```
Var tcs = new TaskCompletionSource<int>();
new Thread(()=>{Thread.Sleep(5000);
tcs.SetResult(42); }).Start();
Task<int> task = tcs.Task;
Console.WriteLine(task.Result);
```

This code will print 42 after 5s.

18). How to use Delay method:

- Task.Delay(5000).GetAwaiter().OnCompleted(()=>Console.WriteLine(42));
- Task.Delay(5000).ContinueWith(ant=>Console.WriteLine(42));

Chapter NO.34 Short Notes

1). Asynchronous takes return quickly.

2). Asynchronous are non-blocking.

3). For I/O bound we can usually work without thread.

4). For CPU bound we can start and return task and are asynchronous tasks.

5). Example of CPU bound asyn Task:

```
Int GetPrimesCount(int start, int count)
{
ParallelEnumerable.Range(start, count).count(n=>Enumerable.Range(2,(int)Math.Sqrt(n)-
    1).All(i=>n%i>0));
}
void DisplayPrimeCounts()
{
for(int i = 0; i<10; i++)
Console.WriteLine(GetPrimesCount(i*1000000+2, 1000000)+" Primes between
    "+(i*100000)+" and "+((i+1)*100000-1));
Console.WriteLine("Done");
}
```

6). Example of Corse Grained:

```
Task.Run(()=> DisplayPrimeCounts());
Task<int> GetPrimesCountAsyn(int start, int count)
{
Return Task.Run(()=> ParallelEnumerable.Range(start,
    count).Count(n=>Enumerable.Range(2, (int)Math.Sqrt(n)-1).All(i=>n%i>0)));
}
for(int i = 0; i<10; i++)
{
Var awaiter = GetPrimesCountAsync(i*1000000+2, 1000000).GetAwaiter();
Awaiter.OnCompleted(()=> Console.WriteLine(awaiter.GetResult()+" primes
    between…."));
```

```
}
Console.WriteLine("Done");
```

7). The above code is simple in C#5.0:

```
Asyn Task DisplayPrimeCounts()
{
for(int i = 0; i<10; i++)
Console.WriteLine(await GetPrimesCountAsyn(i*1000000+2, 1000000)+ "primes between
    "+(i*1000000)+ " and "+((i+1)*1000000-1));
Console.WriteLine("Done");
}
```

8). Syntax of await:

```
var result = await expression;
statement(s);
```

9). Asynchronous tasks means doing some work then returning some result then completing the rest.

10). What is the message loop

```
while(!ThisApplication.Ended)
{
wait for something to appear in message queue
Got Something: What kind of message is it?
keyboard/mouse message->fire an event handler
user BeginInvoke/Invoke message-> execute delegate
}
```

Chapter NO.35 Short Notes

1). Write Downloading Code:

```
asyn void Go()

_button.IsEnabled = false;
String[] urls = www.albahri.com www.orielly.com www.linqpad.net.Split();
Int totallength = 0;
try{
foreach(String url in urls)
{
var uri = new Uri("http://"+url);
Byte[] data = await new WebClient().DownloadDataTaskAsync(url);
_results.Text+ "Length of "+url+ " is "+data.Length+Environmentr.newLine;
TotalLength+=data.Length;
}
_result.Text+="Total length : "+TotalLength;
}
catch(WebException ex)
{
_result.Text+="error: "+ex.Message;
}
finally{_button.IsEnabled = true;}
}
```

2). We can return a Task from void function without explicitly returning it.

3). If you have to write asynchronous function then follow three steps:

- Write its totally synchronous version
- Then use await and async
- Then use returnand return Task in place of void

4). Example of async chaining:

```
async Task Go()
{
var task = PrintAnswerToLife();
```

```
await task;
Console.WriteLine("Done");
}
async Task<int> GetAnswerToLife()
{
var task = Task.Delay(5000);
await task;
int answer = 21*2;
return answer;
}
```

5). Example of Parallelism:

```
var task1 = PrintAnswerToLife();
var task2 = PrintAnswerToLife();
await task1;
await task2;
async Task<int> GetAnswerToLife()
{
_x++;
await Task.Delay(5000);
return 21*2;
}
```

6). Tasks provide us the facility of handling cancellation and progress.

7).Example of Cancellation:

```
class CancellationToken
{
public bool IsCancellationRequested{get; private set;}
public void Cancel(){IsCancellationRequested = true;}
public void ThrowIfCancellationRequested();
{
If(IsCancellationRequired)
Throw new OperationCancelledException();
}
}
```

8). Example of cancellation using Task:

```
var CancelResource = new CancellationTokenSource();
```

Task.Delay(5000).ContinueWith(ant=>CancellationSource.Cancel());

9). Cancellation can even be used Task.Wait i.e. Synchronous methods but have to call cancel from another task. Cancellation is useful for timeouts.

10). In Task-Based Async Pattern (TAP) we create a "Hot" task From Task or TaskResult.

11). Task Combinator:

Task<int> WinningTask – await Task.WhenAny
Console.WriteLine("Done");
Console.WriteLine(WinningTask.Result);
(Delay1(), Delay2(), Delay3());
int answer = await await Task.WhenAny(Delay1(), Delay2(), Delay3());

12). Async call Graph execution:

- brief sync execution on thread calling Go
- Every await and return
- when delay fires a thread
- remaining statements run
- eventually Go's task is marked as completed.

13). Pseudo Concurrency means a program is running and awaiting for something then resuming and then awaiting.

14). Task Combinator uses Task.WhenAny and Task.WhenAll.

Chapter NO.36 Short Notes

1). Example of WhenAll in Task Combinators:

```
await Task.WhenAll(Delay1(), Delay2(), Delay3());
Task task1 = Delay1(), task2 = Delay2(), task3 = Delay3();
await task1(); await task2; await task3;
Task task1 = Task.Run(()=>{throw null;});
Task task2 = Task.Run(()=>{throw null;});
Task all = Task.WhenAll(task1, task2);
try{await all;}
catch
{
Console.WriteLine(all.Exception.InnerExceptions.Count);
}
```

2). there are two types of parallelism:

- Data Parallelism
- Task Parallelism

3). Three steps of Parallelism:

- Partition Problem
- Process
- Combine the results

4). Data Parallelism is easier to perform and scales well, it is also structured.

5). Concurrent collections are useful when you want a thread-safe collection.

6). Tasks provide us the facility of handling cancellation and progress.

7). Three static method of the parallel class are:

- Parallel.Invoke
- Parallel.For
- Parallel.ForEach

8). Parallel.Invoke:
Executes an array of delegates in parallel.

9). Example of Parallel.Invoke:
public static void Invoke(params Action[] actions);
Parallel.Invoke(
()=>new WebClient().DownloadFile("http://www.linqpad.net", "lp.html"),
()=>new WebClient().DownloadFile(http://www.jaon.dk, "jaon.html"));

10).Example of Parallel.For:
for(int i=0; i<100; i++)
foo(i);
Parallel.For(0, 100, i=>foo(i));
Parallel.For(0, 100, foo);

11). Example of Parallel.ForEach():
Foreach(char c in "Hello World)
foo(c);
Parallel.Foreach("Hello, World", foo);

12). ParallelLoopState is a thing which allows you to Break out of loops.

13). The ParallelLoopState class and its methids are:
public class ParallelLoopState
{
public void Break();
public void stop();
public bool IsExceptional{get;}
public bool IsStopped{get;}
public long LowestBreakIteration{get;}
public bool ShouldExitCurrentIteration{get;}
}

14). Concurrent Collections are three times slower than normal collections in writing but not the case with reading.
15). Concurrent Collections include :

- Concurrent Stack
- Concurrent Queue
- Concurrent Bag
- Concurrent Dictionary

16). A Producer Consumer Queue:

```
public class PCQueue : IDisposable
{
  BlockingCollection<Action> _taskQ = new BlockingCollection<Action>();

  public PCQueue (int workerCount)
  {
    // Create and start a separate Task for each consumer:
    for (int i = 0; i < workerCount; i++)
      Task.Factory.StartNew (Consume);
  }

  public void Enqueue (Action action) { _taskQ.Add (action); }

  void Consume()
  {
    // This sequence that we're enumerating will block when no elements
    // are available and will end when CompleteAdding is called.

    foreach (Action action in _taskQ.GetConsumingEnumerable())
      action();     // Perform task.
  }

  public void Dispose() { _taskQ.CompleteAdding(); }
}
```

Chapter NO.37 Short Notes

1). HTML is a Hyper Text Markup Language

2). Java Script is built-in all platforms and is supported by all web-browsers.

3). Web pages have three kinds of layers:

- HTML: Gives content/structure
- CSS: Presentation Layer
- Java script: Behavioral Layer

- **4). JS was introduced in 1995, originally named with LiveScript but got populated with most hot name "Java", MS introduced jScript... their version of JS for IE,**

5). JQuery:

- JS is a prog language... can be hard for some... also browser incompatibilities make testing difficult
- jQuery is a JS library intended to make JS programming easier
- jQuery soles JS complexity and browser incompatibilities
- can do things in single LOC that would take hundreds
- many advanced features come as jQuery plugins
- used on millions of websites

6). In HTML:

- at least three tags ... html root tag, head tag containing title etc, and body tag containing all parts to be rendered in browser window
- is paragraph is emphasis is a hyperlink... XML attribute and value
- validating html means checking if all tags appropriately closed etc.

7). HTML Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title>Hey, I am the title of this web page.</title>
</head>
<body>
Hey, I am some body text on this web page.
</body>
</html>
```

8). CSS stands for Cascading Style Sheets, it is a formatting language

9). CSS says that take "this" and make "this" look like "that"

10).JQuery says that take "this" and act like "that"

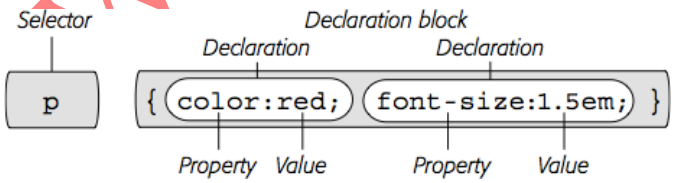
11). CSS can do more powerful stuff, like add border, change margins, and even control exact placement on web page

12). JS can add/remove/change CSS properties based on input or mouse clicks... even animate from one property to another... like yellow to red... or animate across screen by changing position

13). Example of CSS Style:

```
p {
    color: red;
    font-size: 1.5em;
}
```

Explanation:



14). Concurrent Collections are three times slower than normal collections in writing but not the case with reading.

15). JS is an interpreted language means each line of code is compiled at run time.

16). Tell Web Browser about JS by using <script> </script> tag

17). Using Script in HTML $.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
<html>
<head>
<title>My Web Page</title>
<script type="text/javascript">

</script>
</head>
```

18). Using Script in HTML 5:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>My Web Page</title>
<script>

</script>
</head>
```

19). Writing Hello World in HTML:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<head>
<title>My Web Page</title>
<script>
alert('hello world!');
</script>
</head>
```

20). Basics of JS:

- lot of basic syntax like C++, C#
- create a variable using “var x”, names begin with letter, $, or _
- var days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun'];
- alert(days[0]); // Accessing element of array
- var playList = []; // Declaring empty array
- var prefs = [1, 223, 'www.oreilly.com', false]; //Array having different data types//
- prefs.push('test'); prefs.push('test','test2'); ... arrays grow
- prefs.unshift('test'); prefs.unshift('test','test2'); ... insert at start
- shift() gets/removes the first element... queue using push/shift
- pop() removes last... stack

21). Declaring Function in JS:

```
function functionName(parameter1, parameter2, parameter3) {
  // the JavaScript you want to run
}
```

- There is no specified return type
- Function Declaration… no type

22). JQuery:

- many JS programs select elements, add new content, hide and show content, modify tag’s attributes, determine value of form fields, and react to user actions
- the details complicated specially with browser interoperability
- libraries offer a set of functions to make these tasks easy

23). Benefits of JQuery:

- only 30k compressed library size
- easy to learn
- used on millions of sites
- free
- dev community
- plugins... !!!

Chapter NO.38 Short Notes

1). DOM stands for Document Object Model

2). Selection By element ID:
document.GetElementById(‘banner’);

3). Selection By element name:
document.GetElementByName(‘a’);

4). Basic Selectors are:
- By Class Name
- By ID Name
- By Tag Name

5). Example of Select by ID:
$(#banner) is the tag with id banner

6). Advanced Selectors are as follows:

7). Descendent Selectors:
$(‘#navbar a’): select all the elements whose id is navbar then descendent a.

8). Child Selector:
$(‘body>p’): Select p tags which are within body tag

9). Adjacent Sibling:
$(‘h2+div’): Select tag “div” that comes immediately after “h2” tag.

10). Attribute Selectors :
- $(‘img[alt]’): we need those images in which attribute of “alt” exists
- $(‘input[type = “text”]’): we need all input fields of type text
- $(‘a[href^=”mailto:”]’): we need tags whose link “href” starts with “mailto”
- $(‘a[href $= “.pdf”]’): we need all that tags whose links are of “.pdf” type

- $(‘a[href *= “missingmanual.com”]’): we need tags with “href” having string “missingmanual.com”

11). JQuery Filters:

- :even :odd
- $('.striped tr:even'): we need those rows on which class.stripped is applied
- :first :last :not
- $('a:not(.navButton)'): We need all the links that do not have .navButton class on them
- :has
- $('li:has(a)') --- diff from descendent : we need all these list items in which tag(a) exists
- :contains
- $('a:contains(Click Me!)') : we need that list in which the text “Click Me!” exists.
- :hidden :visible
- $('div:hidden').show(): find all div tags that are hidden and call .show() and they will show

12). JQuery Provides automatic looping

13). Automatic Looping:

- $('#slideshow img').hide(): Slideshow id containing all img tag choose them and hide them

14). Chaining Functions:

- $('#popUp').width(300).height(300): pop ID tag selected set width 300 and height 300
- $('#popUp').width(300).height(300).text('Hi!').fadeIn(1000);

14). Concurrent Collections are three times slower than normal collections in writing but not the case with reading.

15). JS is an interpreted language means each line of code is compiled at run time.

16). Tell Web Browser about JS by using <script> </script> tag

17). Using Script in HTML $.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">
<html>
<head>
<title>My Web Page</title>
<script type="text/javascript">

</script>
</head>
```

18). Using Script in HTML 5:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<title>My Web Page</title>
<script>

</script>
</head>
```

19). Writing Hello World in HTML:

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<head>
<title>My Web Page</title>
<script>
alert('hello world!');
</script>
</head>
```

20). Basics of JS:

- lot of basic syntax like C++, C#
- create a variable using “var x”, names begin with letter, $, or _
- var days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun'];
- alert(days[0]); // Accessing element of array
- var playList = []; // Declaring empty array
- var prefs = [1, 223, 'www.oreilly.com', false]; //Array having different data types//
- prefs.push('test'); prefs.push('test','test2'); ... arrays grow

- prefs.unshift('test'); prefs.unshift('test',’test2’); ... insert at start
- shift() gets/removes the first element... queue using push/shift
- pop() removes last... stack

21). Declaring Function in JS:

```
function functionName(parameter1, parameter2, parameter3) {
 // the JavaScript you want to run
}
```

- There is no specified return type
- Function Declaration… no type

22). JQuery:

- many JS programs select elements, add new content, hide and show content, modify tag’s attributes, determine value of form fields, and react to user actions
- the details complicated specially with browser interoperability
- libraries offer a set of functions to make these tasks easy

23). Benefits of JQuery:

- only 30k compressed library size
- easy to learn
- used on millions of sites
- free
- dev community
- plugins... !!!

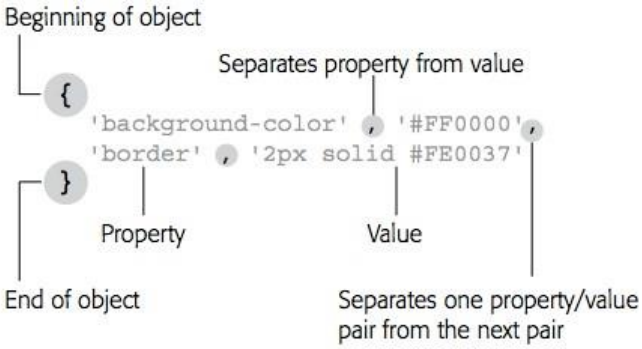
24). Attributes can be manipulated with

- addclass
- removeclass
- toggleclass

25). Example of multiple css Properties

```
$('#highlightedDiv').css({
  'background-color' : '#FF0000',
  'border' : '2px solid #FE0037'
});
```

26). Explanation of above code:



27). When acting on each element in selection key word Each is used like this:

```
$('selector').each(function() {
 // code goes in here
});
```

```
$('a[href^=http://]').each(function() {
  var extLink = $(this).attr('href');
 $('#bibList').append('<li>' + extLink + '</li>');
});
```

28). Web mouse Events are :
click, dblclick, mousedown,

29). Doc/Window Events:
load, resize, scroll, unload

30). Form Events:
submit, reset, change, focus, blur

31). Keyboard Events:
- keypress (over n over), keydown, keyup

32). Steps in setting Events:
- step 1: select elements
- step 2: assign an event
- step 3: pass function to event
- $('#menu').mouseover(function() {
  $('#submenu').show();
  }); // end mouseover

33). Examples of Events:

```
<script src="../_js/jquery-1.6.3.min.js"></script>
<script>
$(document).ready(function() {
  $('html').dblclick(function() {
    alert('ouch');
  }); // end double click
  $('a').mouseover(function() {
    var message = "<p>You moused over a link</p>";
    $('.main').append(message);
  }); // end mouseover
  $('#button').click(function() {
    $(this).val("Stop that!");
  }); // end click
}); // end ready
</script>
```

34).JQuery Events:

- hover
- toggle is like hover except worked on and off by clicks
- event object is passed to all functions handling events

```
$('#menu').hover(function() {
   $('#submenu').show();
}, function() {
   $('#submenu').hide();
}); // end hover
```

Chapter NO.39 Short Notes

1). Web events also have an event object just like desktop event object.

2). ent.PreventDeafult() or return false to stop the default behavior.

3). For removing enet us: $('.tabbutton').unbind('click')

4). For stopping default event bubbling: evt.StopPropagation()

5). Generic way to bind events: $('#selector').bind('click', mydata, functionname)

6). Event Properties:

Event property	Description
pageX	The distance (in pixels) of the mouse pointer from the left edge of the browser window.
pageY	The distance (in pixels) of the mouse pointer from the top edge of the browser window.
screenX	The distance (in pixels) of the mouse pointer from the left edge of the monitor.
screenY	The distance (in pixels) of the mouse pointer from the top edge of the monitor.
shiftKey	Is *true* if the shift key is down when the event occurs.
which	Use with the keypress event to determine the numeric code for the key that was pressed (see tip, next).
target	The object that was the "target" of the event—for example, for a *click()* event, the element that was clicked.
data	A jQuery object used with the *bind()* function to pass data to an event handling function (see page 177).

7). Examples:

```
$(document).bind('click keypress', function() {
  $('#lightbox').hide( );
}); // end bind
```

Binding multiple events:

```
$('#theElement').bind('click', function() {
  // do something interesting
}); // end bind
$('#theElement').bind('mouseover', function() {
  // do something else interesting
}); // end bind
```

8). FAQ Example:

```
<script src="../_js/jquery-1.6.3.min.js"></script>
<script>
$(document).ready(function() {
  $('.answer').hide();
  $('#main h2').toggle(
    function() {
      $(this).next('.answer').fadeIn();
      $(this).addClass('close');
    },
    function() {
      $(this).next('.answer').fadeOut();
      $(this).removeClass('close');
    }
  ); //end toggle
});
</script>
```

9). JQuery Animations:

- $('element').fadeOut('slow');
- fast normal slow or number of ms
- default 200 400 600
- fadeIn, fadeOut, fadeToggle
- slideDown, slideUp, slideToggle

10). Example of Login Slider:

```
$(document).ready(function() {
    $('#open').toggle(
        function() {
            $('#login form').slideDown(300);
            $(this).addClass('close');
      },
      function() {
         $('#login form').fadeOut(600);
          $(this).removeClass('close');
      }
  ); // end toggle
}); // end ready
```

11). Forms:

<input name=”quantity” type=”text” id=”quantity”>

<input name=”total” type=”text” id=”total”>
var unitcost = 9.95;
var amount =\$(‘\#quantity’).val();
var total = amount*unitcost;
total = total.ToFixed(2);
\$(‘\#total’).val(total);

12). JQuery Form selectors:

Selector	Example	What it does
:input	$(':input')	Selects all input, textarea, select, and button elements. In other words, it selects all form elements.
:text	$(':text')	Selects all text fields.
:password	$(':password')	Selects all password fields.
:radio	$(':radio')	Selects all radio buttons.
:checkbox	$(':checkbox')	Selects all checkboxes.
:submit	$(':submit')	Selects all submit buttons.
:image	$(':image')	Selects all image buttons.
:reset	$(':reset')	Selects all reset buttons.
:button	$(':button')	Selects all fields with type *button*.
:file	$(':file')	Selects all file fields (used for uploading a file).
:hidden	$(':hidden')	Selects all hidden fields.

Chapter NO.40 Short Notes

1). Ajax is a term which allows us to talk to server remaining within JS.

2). What Ajax can do:

- Display new HTML content without reloading the page
- Submit form and instantly display the result
- Login without leaving the page
- Ajax make pages feel more responsive and Desktop like

3). JS, server-side programming, and web browser, all work together

4). Role of Java Script in Ajax:

- Send Request
- Wait for response
- Process Response
- Update Web Page

5). Role of Web Server:

Receive Request
Respond as HTML, plain text, JASON

6).XMLHttpRequest is also known as XHR in short

7). Steps (Talking to Web Server)

- create XMLHttpRequest (also called XHR in short)

var newXHR = new XMLHttpRequest();

- call open to specify what kind of data and where it will go

can GET or POST

newXHR.open('GET', 'shop.php?productID=34');

- write a callback function

it will remove, add, change elements

- send data

newXHR.send(null); GET

newXHR.send('q=javascript'); POST

- receive response

callback invoked and XHR receives status, text response, and possibly an XML response

8). The basic concept of POST is that if you want to post a lot of data then instead of sending it via URL you can insert this into the request body.

9). The Web Server Request can respond in the following states:

- status = 200/304 all ok, 404 file not found, 500 internal server error, 403 access forbidden

8). The JQuery simplifies all steps except that of changing the webpage

9). Example of loading news in a div from web-server using jQuery:

- $('#headlines').load('todays_news.html');

10). Get() and Post() in jQuery:

- need server side to do anything else
- server may not return html e.g. database records as xml or json
- jQuery handles differences of GET and POST
- $.get(url, data, callback);
- $.post(url, data, callback);

11). Examples of using Get() and Post():

- GET has limit... often thousands of chars
- $.get('rateMovie.php','rating=5');
- $.post('rateMovie.php','rating=5');
- $.post('rateMovie.php','rating=5&user=Bob');
- 'favFood=Mac & Cheese' // incorrect
- 'favFood=Mac%20%26%20Cheese' // properly escaped
- var queryString = 'favFood=' + encodeURIComponent('Mac & Cheese');
- $.post('foodChoice.php', queryString);

12). The better way to use Get() and Post() is object literal i.e.

```
var data = $.post('rankMovie.php',
    {
rating: 5,
user: 'Bob' }
); // end post
```

13). Serialize is used for form submission in JS by so that the page may not reload.

14). Example of using serialize method:

```
var formData = $('#login').serialize();
    $.get('login.php',formData,loginResults);
```

15). third parameter in get() and post() is call back. We write a function that will process data returned by server.

16). Example of Processing Returned Data:

17). php is a server side programming language.

18). Error Handlers:

```
$.get(url, data, successFunction).error(errorFunction);
 $.get('rate.php', querystring, processResponse).error(errorResponse);
function errorResponse() {
        var errorMsg = "Your vote could not be processed right now.";
        errorMsg += "Please try again later.";
        $('#message').html(errorMsg);
}
```

19). JASON:

- JS format... method for exchanging data
- JSON is JS so its quick n easy for JS
- no XML like parsing
- JSON is a JS obj literal

```
{
    firstName: 'Frank',
    lastName: 'Smith',
    phone: '503-555-1212'
}
```

- OR

- {
- 'firstName': 'Frank',
- 'lastName': 'Smith',
- 'phone': '503-555-1212'
- } (MUST use if names have spaces etc.)
- server returns a string formatted like a JSON obj literal
- jQuery getJSON method
- like get but data passed to callback will be a JSON object
- e.g. var bday = {
- person: 'Raoul',
- date: '10/27/1980'
- };
- bday.person // 'Raoul'
- bday.date // '10/27/1980'
-