2. Capturing a Face image

The following procedures are recommendations for acquiring the highest quality image from the end-user as the recognition process results are highly dependent on this input. The intended user behavior and an actual process for capturing the user's facial image consists of several key actions:

1. Requesting from the User to place his/her face in front of the camera.
2. Guiding the User in real-time, so the placement of the face can be in a most ideal position.
3. Capturing the best possible image (manually or automatically).

Since we need the front aka selfie camera for taking this image, both mobile and desktop have the same device configuration:

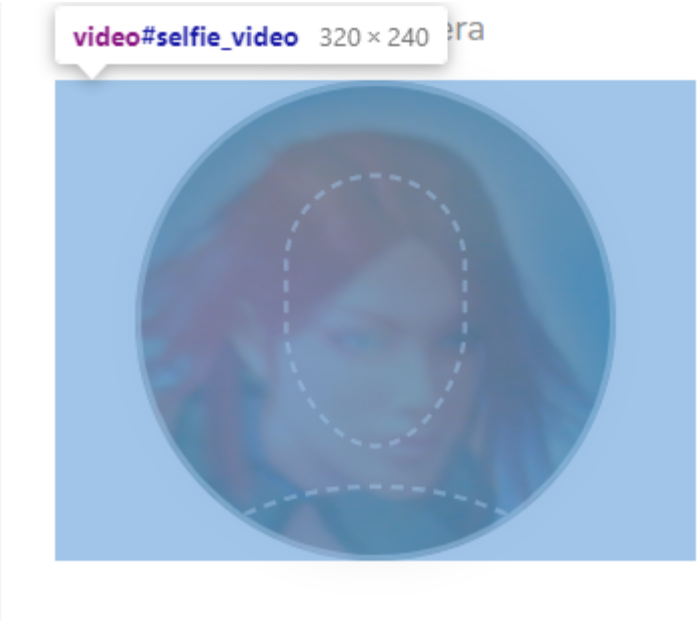
```
let constraints

constraints = {
  audio: false,
  video: {
    facingMode: 'user',
    width: { min: 640, ideal: 640 },
    height: { min: 480, ideal: 480 }
  }
};
```

As we can see, the **facingMode** is set to 'user' and we don't need a big and variable resolution to acquire the user's picture, so 640x480 will suit our needs.

Now we should prepare the video object to adapt to the size and layout UI of the app.

The following example implies that you have previously configured and activated the video stream from the user's device.
On how to set the user's camera with video, refer to **Working with Camera Input** document.

Capturing the image from the stream depends on the position of the visual frame which serves to a User as a guideline to position their face within this frame's boundaries:



In the example above we can see the relation between the video element and the frame which will be the only part of the video which we will extract in order to acquire the card image.

Let's prepare the video to fit our layout. In our example, the frame size is 240x240px and the video is in portrait mode. We need to scale the video to fit the frame's height as well as to center it in relation to the frame.

```
let video = document.getElementById('selfie')
let w = video.videoWidth
let h = video.videoHeight
let ratio = 0

// if portrait video ( desktop )
if( w >= h )
{
  ratio = w / h
  video.style.width = ( 240 * ratio ) +"px"
  video.style.height = 240+"px"
  video.style.marginLeft =  ( 240 * ratio - 240 ) / -2 +"px"
}
```

```
else // if landscape video ( mobile's front camera )
{
  ratio = h / w
  video.style.height = ( 240 * ratio ) +"px"
  video.style.width = 240+"px"
  video.style.marginTop =  ( 240 * ratio - 240 ) / -2 +"px"
}

video.style.width = video.width +"px"
video.style.height = video.height +"px"

// show video only after its been resized to avoid visual artefacts
video.style.visibility = "visible"
```

Once the user has placed the face within the frame we're ready to capture the base64 image, either manually via the button or when auto-detected using Detecting Face Size, Position & Light methods.

In the routine bellow, we can see how we require **canvas** element as a placeholder for our image extraction, on which we will draw the part of video that is within the **frame** guideline boundaries:

```
var canvas = document.createElement("canvas")
var w = video.videoWidth
var h = video.videoHeight

if( w > h )
{
  canvas.width = h
  canvas.height = h
  canvas.getContext('2d').drawImage(video, -( w - h )/2, 0, w, h )
}
else
{
  canvas.width = w
  canvas.height = w
  canvas.getContext('2d').drawImage(video, 0, -( h - w )/2, w, h )
}
```

Now that we have the canvas with a video snapshot on it, we can easily extract the base64 image:

```
var base64 = canvas.toDataURL('image/jpeg');
```

```
// image data needs additional manipulation before being sent to the service
base64 = base64.substr( 23, base64.length );
```