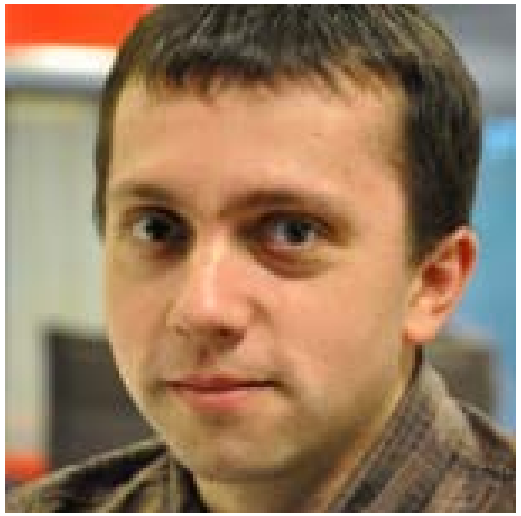
Using Video and Adding Graphical Elements with a Canvas



Gill Cleeren

@gillcleeren

Outline

Hear and see with the new audio and video elements

Drawing with JavaScript on a Canvas

Target of This Module

We'll have added a video to our coffee detail page

We can draw complex items using a Canvas

Hear and see

with the new audio and video elements

Audio and Video in HTML5

- Playing media content without plugins
- JavaScript API is available
- Possible to synchronize with other content (subtitles)

The Audio and Video Elements

Common Audio and Video Properties

- src attribute: points to the actual media content

```
<video src="coffee-introduction.mp4"></video>
<audio src="intro.mp3"></audio>
```

- autoplay
  - Automatic playback is disabled by default
  - Can be controlled using autoplay attribute

```
<video src="coffee-introduction.mp4" autoplay="autoplay"></video>
<audio src="intro.mp3" autoplay></audio>
```

- loop
  - Playing the media once is the default
  - loop property can be used to control this behavior

```
<video src="coffee-introduction.mp4"></video>
<audio src="intro.mp3"></audio>
```

Common Audio and Video Properties

- preload
  - Defines the buffering type to be used
  - Can be overruled by the browser though
  - Possible values
    - none
    - metadata
    - auto

```
<video src="coffee-introduction.mp4" preload="auto"></video>
<audio src="intro.mp3" preload="none"></audio>
```

- controls
  - Used to control display of playback controls

```
<video src="coffee-introduction.mp4" controls></video>
<audio src="intro.mp3" controls></audio>
```

Common Audio and Video Properties

- muted
  - Controls if media is muted or not

```
<video src="coffee-introduction.mp4" muted></video>
<audio src="intro.mp3" muted ></audio>
```

Video Specific Properties

- width and height
  - Defaults to video width and height
  - Without video, 300x150 is used for layout
  - Keeps aspect ratio by default if one of two is defined

```
<video src="coffee-introduction.mp4" width="175px"></video>
```

  - Possible to overrule the behaviour by setting both with and height
    - Even using a different aspect ratio is possible
- poster
  - Points to an image to display before playback of video

```
<video src="coffee-introduction.mp4" poster="/images/joeslogo.png" controls> </video>
```

Fallback Content

- If the browser doesn't understand audio or video, it can display the default content: fallback

```
<video src="coffee-introduction.mp4"
  poster="/images/joeslogo.png" controls>
  Sadly your browser doesn't support video…
</video>
```

Defining More than One Source

- Different browsers support different media types
  - Defining just one source can cause problems!
- Instead of using the src attribute, you should define some options using source elements

```
<video width="320" height="240" poster="/images/joeslogo.png" controls>
     <source src=" coffee-introduction.mp4">
     <source src=" coffee-introduction.ogg">
     Sadly your browser doesn't support video…
</video>
```

  - type attribute can be used to further define the type of the content

```
<source src=" coffee-introduction.mp4" type="video/mp4">
<source src=" coffee-introduction.ogg" type="video/ogg">
```

Supported formats

Video	MP4/H.264 WebM Ogg Theora
Audio	MP3 AAC Ogg Vorbis

Audio and Video API

- Controlling playback is done using a JavaScript API
- play/pause/paused property

```
function playPause() {
    if (myVideo.paused)
        myVideo.play();
    else
        myVideo.pause();
}
```

- autoplay: can also be set from JavaScript

```
function setAutoplay() {
    myVideo.autoplay = "true";
}
```

Audio and Video API

- play and pause events
  - Fired when the media playback starts or is being paused

```
video.onpause = function(e){
  hidepauseButton();
  showPlayButton ();
};

video.onplay = function(e) {
  hidePlayButton();
  showPauseButton ();
};
```

Audio and Video API

- playbackRate gives control over playback speed

```
function setPlaySpeed()
{
  myVideo.playbackRate=0.7;
}
```

- currentTime: used to find current location in the media
  - Can also be used for seeking a specific time

```
function setCurrentTime()
{
   myVideo.currentTime=5;
}
```

Audio and Video API

- duration: returns length of the media

```
function getVidDuration()
{
  alert(myVideo.duration);
}
```

- progress event: fires on progress, couple of times per second
  - Used in combination with currentTime and duration

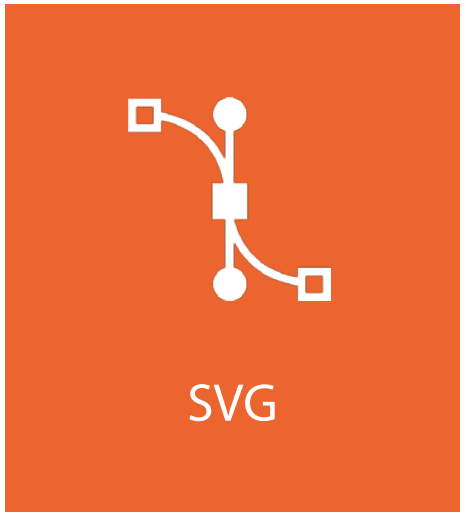
```
function checkProgress(){
  myVideo=document.getElementById("myVideo");
  myVideo.onprogress=alert("Downloading your video");
};
```

Demo: Adding a Video

Drawing with JavaScript on a Canvas

Rich Graphics in the Browser

- Previous versions of HTML didn't support rich graphics natively
  - Had to be done using a plugin like Flash or Silverlight
    - May not be supported in all browsers and OS's
  - Was added to the page using an object tag



Drawing pane

Pixel-based

Lookless

Commonly rendered by the GPU

Getting Started with the Canvas

- Canvas is placed on the page, defaults to 300x150
  - Setting size is preferable

```
<canvas id="mainCanvas" width="200" height="100">
```

- From there on, it's JavaScript!

Steps to Add a Canvas to Your Page

- Step 1: retrieve the canvas using getElementById

```
var canvas = document.getElementById("mainCanvas");
```

- Step 2: retrieve the context using the getContext("2d")
  - Context is used as middle man to draw on the canvas

```
var ctx = canvas.getContext("2d");
```

- Step 3: Optionally, specify fallback content

```
<canvas id="mainCanvas" width="200" height="100" >
  Canvas not supported, sorry!
</canvas>
```

Drawing on the Canvas

- The canvas uses the top left as 0,0
- Simple drawing methods
  - fillRect
  - strokeRect
  - clearRect

```
<body>
<canvas id="mainCanvas" width="300" height="200"
  style="border:1px solid #c3c3c3;">
  Canvas not supported, sorry!
</canvas>
<script>
  var c = document.getElementById(" mainCanvas ");
  var ctx = c.getContext("2d");
  ctx.fillStyle = "#FFFF00";
  ctx.fillRect(0,0,100,80);
</script>
</body>
```

Resetting the Canvas

```
var canvas= document.getElementById("mainCanvas");
canvas.width = canvas.width;
```

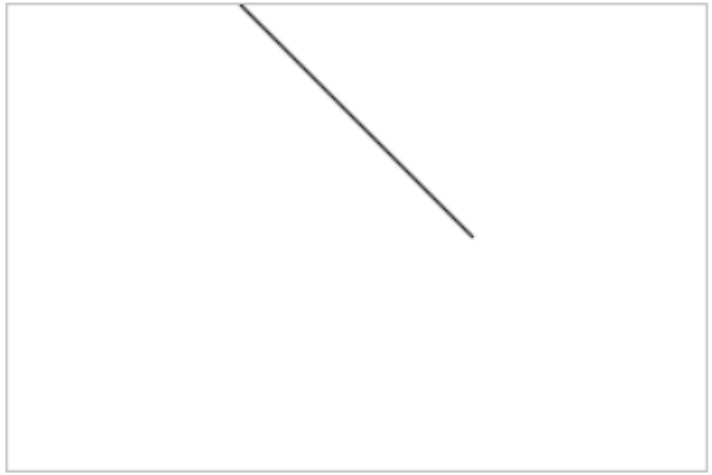
Drawing on the Canvas

- Complex shapes are drawn using a path
  - Contains one or more subpaths
    - Each subpath is a series of points to form a line
  - It's possible to add lines or figures to a path
    - Also possible to specify how the drawing of subsequent lines should happen

```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(100,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
```

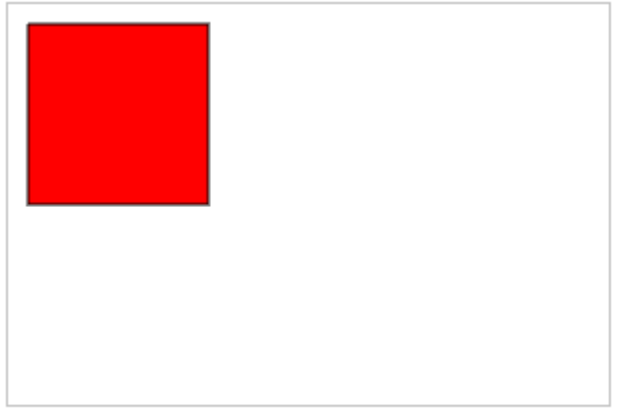
Drawing on the Canvas

```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(100,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>
```



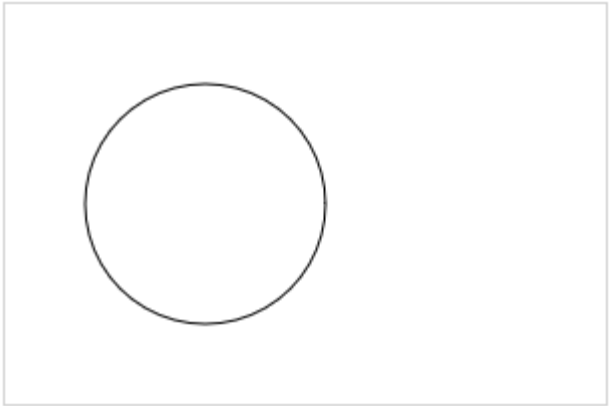
Drawing on the Canvas

```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.moveTo(10, 10);
ctx.lineTo(100, 10);
ctx.lineTo(100, 100);
ctx.lineTo(10, 100);
ctx.closePath();
ctx.fillStyle = "red";
ctx.fill();
ctx.stroke();
</script>
```



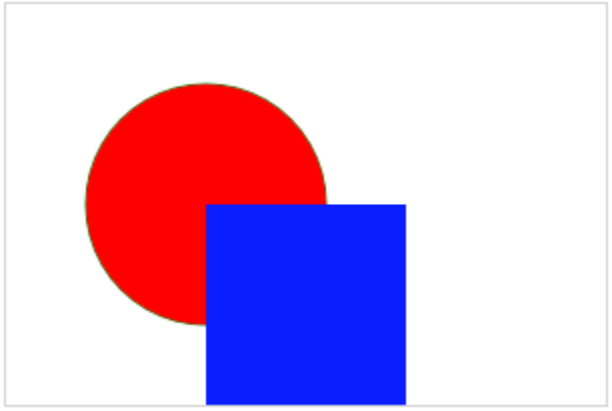
Drawing on the Canvas

```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(100,100,60,0,2*Math.PI);
ctx.fillStyle = "red";
ctx.fill();
ctx.stroke();
</script>
```



Styling the Canvas Content

- Canvas supports solid colors and gradients for fill and stroke
  - Can be set using fillStyle and strokeStyle
  - Accepts named color, hex, RGB and aRGB



```
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(100,100,60,0,2*Math.PI);
ctx.fillStyle = "red";
ctx.strokeStyle = "green";
ctx.stroke();
ctx.fill();
ctx.fillStyle = "rgb(10, 30, 255)";
ctx.fillRect(100, 100, 100, 100);
</script>
```

Adding Gradient Fills

- Adding gradients requires 3 steps
  - Creating the gradient
  - Adding color stops
  - Assigning the gradient to the style
- Can be linear or radial

```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
var gradient = ctx.createLinearGradient(0,0,200,0);
gradient.addColorStop(0,"red");
gradient.addColorStop(0.5, "blue");
gradient.addColorStop(1,"green");
ctx.fillStyle = gradient;
ctx.fillRect(10,10,150,80);
</script>
```

Adding Images to the Canvas

- Canvas can be combined with images for more advanced graphics
  - drawImage method is used for this

```
context.drawImage(img, x, y);
```

  - Image can be retrieved from DOM using getElementById

```
<script>
var canvas = document.getElementById("mainCanvas");
var ctx = canvas.getContext("2d");
var img = document.getElementById("im");
ctx.drawImage(img,10,10);
</script>
```

  - Size can be specified

```
ctx.drawImage(img,10,10, 100, 75);
```

Adding Images to the Canvas

- Zooming is also possible

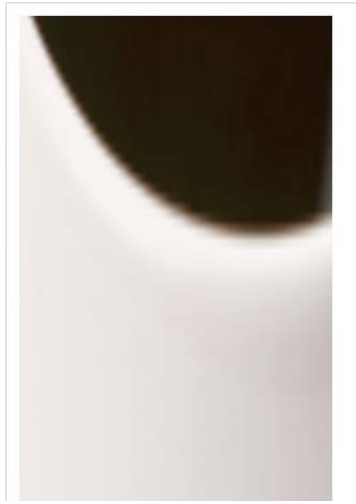
```
ctx.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);
```

```
<script>
var canvas = document.getElementById("mainCanvas");
var ctx = canvas.getContext("2d");
var img = document.getElementById("im");

ctx.drawImage(img,100,100,120,75,10,10,250,600);
</script>
```

A cup of coffee:

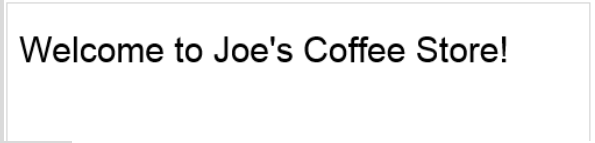
The cup on the canvas:



Adding Text to the Canvas

- Adding text to the canvas is done using the fillText or strokeText methods

```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.font = "22px Arial";
ctx.fillText("Welcome to Joe's Coffee Store!",10,50);
</script>
```




```
<script>
var c = document.getElementById("mainCanvas");
var ctx = c.getContext("2d");
ctx.font = "22px Arial";
ctx.strokeText("Welcome to Joe's Coffee Store!",10,50);
</script>
```

Transformations in the Canvas

- A transformation changes the grid
  - Results in impressive options for the canvas

Transformations in the Canvas

- Available transformations
  - Rotate: allows to rotate the coordinate system of a number of radians

```
<script>
  var canvas = document.getElementById('mainCanvas');
  var ctx = canvas.getContext('2d');
  var rectWidth = 150;
  var rectHeight = 150;
  ctx.translate(canvas.width / 2, canvas.height / 2);
  ctx.rotate(Math.PI / 4);
  ctx.fillStyle = 'red';
  ctx.fillRect(0, 0, rectWidth, rectHeight);
</script>
```

Transformations in the Canvas

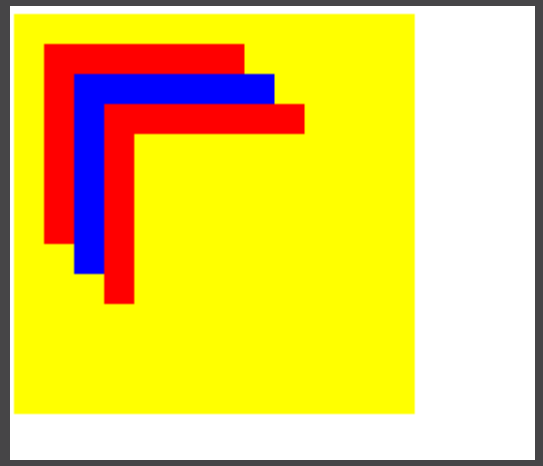
- Scaling the grid is also possible

```
<script>
  var canvas = document.getElementById('mainCanvas');
  var ctx = canvas.getContext('2d');
  var rectWidth = 150;
  var rectHeight = 150;
  ctx.scale(1, 0.5);
  ctx.fillStyle = 'red';
  ctx.fillRect(0, 0, rectWidth, rectHeight);
</script>
```

Canvas State

- The canvas has a system to "remember" the applied settings on a stack
  - save: adds the settings to the stack
  - restore: gets the last settings from the stack, replacing current settings

```
<script>
    var canvas = document.getElementById("mainCanvas");
    var ctx = canvas.getContext('2d');
    ctx.fillStyle = 'yellow';
    ctx.fillRect(0,0,200,200);
    ctx.save();
    ctx.fillStyle = 'red';
    ctx.fillRect( 15,15,100,100);
    ctx.save();
    ctx.fillStyle = 'blue';
    ctx.fillRect(30,30,100,100);
    ctx.restore();
    ctx.fillRect(45,45,100,100);
    ctx.restore();
    ctx.fillRect(60,60,100,100);
</script>
```



Demo: Drawing with the Canvas

Summary