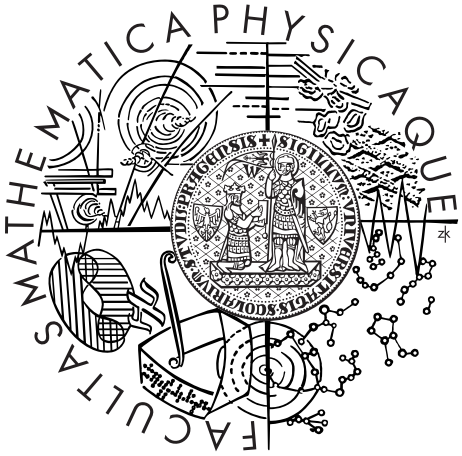
Charles University in Prague
Faculty of Mathematics and Physics

BACHELOR THESIS



Šimon Rozsíval

Vector Screencast

Department of Distributed and Dependable Systems

Supervisor of the bachelor thesis: Mgr. Martin Děcký
Study programme: Computer science
Specialization: Programming and software systems

Prague 2015

Dedication.

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... date ..... signature of the author

Název práce: Vektorový screencast

Autor: Šimon Rozsíval

Katedra: Katedra distribuovaných a spolehlivých systémů

Vedoucí bakalářské práce: Mgr. Martin Děcký

Abstrakt: Cílem bakalářské práce je vytvořit software pro záznam a přehrávání výukových videí pro potřeby Khanovy školy. Na rozdíl od běžných videí nejsou obrazová data uložena ve formě bitmap, ale jako vektory, což umožní snížit datovou náročnost a vykreslit obraz ostře při libovolně velkém rozlišení obrazovky uživatele. Přehrávač videa i nástroj pro nahrávání běží ve webovém prohlížeči. Součástí práce je také návrh a implementace vhodného formátu pro uchovávání obrazových a zvukových dat a implementace v softwarové architektuře klient/server.

Klíčová slova: screencast, vektory, video, on-line

Title: Vector Screencast

Author: Šimon Rozsíval

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Martin Děcký

Abstract: The goal of this bachelor thesis is to create a software for recording and playback of educational videos for Khanova škola (Czech clone of Khan Academy). Contrary to common videos the visual data is not stored as a sequence of bitmaps, but as vectors. This allows to reduce the data bandwidth and playback sharp images in any target resolution. The player and also the tool for recording the videos runs in a web browser. The thesis also focuses on designing and implementing a suitable file format for storing the visual and audio data and implementing the software according to the client/server paradigm.

Keywords: screencast, vector, video, on-line

Contents

Introduction

Each and every person on earth explores the world from the day of and continues to learn new things all his life. Education in the so-called developed world is essential for later employment.

@todo

Khan Academy

Khan Academy is an olnine tool providing free access to instructional videos and practice exercises covering various subjects including math, history, programming, economics, and more.

@todo more info

Vector Screencast project

A year ago two members of K̈hanova škola, the czech brach of the Khan Academy, with an idea of improving the current technical solution of recording and displayling videos on their website. Their idea was not to capture classical bitmap-based videos, but rather a vector-based animation that could be scaled to any display resolution or edited in the future to improve quality of their videos, as some of videos recorded only a few years ago don't look nice or are't even well readable on large displays. The other extreme are small displays of tablets and smartphones where the downscaled letters are too small to read and can't be zoomed well in most video players.

One of the other reasons for this type of solution was the possible decrease of the size of data transfered over the Internet, as most of the image does not change between each two frames and very often the image doesn't change at all.

Thesis structure

@todo

1. Distance education

Distance education is not only a phenomenon of the few last decades, but can be traced at least back to the 18th century, when Caleb Phillipps posted an advertisment called "Teacher of the New Method of Short Hand" in Boston Gazette, saying "Persons in the Country desirous to Learn this Art, may by having the several Lessons sent weekly to them, be as perfectly instructed as those that live in Boston." [1]

With the development of the Internet and its general accessibility in the developed world, providing distance education has become much easier and has spread widely. In some countries tuition rates are high and young people take loans. This topic is covered in a fitting way by John Oliver in his show [2]. The flexibility and low cost of distatnce education over the Internet gives people, who wouldn't be otherwise able to attend a traditional university, an opportunity to gain knowledge and train skills from their homes [3] spending much less money or even for free.

Students in the deveopling world are also taking the advantage of educational content available on the Internet. Several of the top U.S. universities, like Harvard, Stanford or MIT, put some of their materials on so called MOOCs (Massive Open Online Course) like Coursea, edX or Udacity. This content is then available to anyone with a computer and Internet connection and knowledge of the language. An great example is *Kepler* - non profit university project in Rwanda [5]. The goal of this project is to "provide an American-accredited degree, a world class education, and a clear path to good jobs for thousands of students for around $1,000 tuition per year." [6]

A concept of teaching often refered to as *Flipped Classroom* uses distance education. The idea is to let students study lecture materials at home at their own pace and then apply the things at school the next day by doing activities to illustrate the concepts. The teacher can then help them or explain details here in person. These materials are often in the form of videos either from an open source, like the Khan Academy, or created by the teacher himself.

1.1 Current systems

In the next few paragraphs I will try to pick some of current distance education services and tools available on the Internet. This list is not complete and is only meant to give the reader a notion of available technologies and their paradigms, on which the project of Vector Screencast is based.

1.1.1 Coursera

The mission of *Coursera* is to "provide universal access to the world's best education." [9] Anyone can, for free, go through materials published by universities and other organizations aimed at education.

Courses at Coursera consist mainly of video lectures commonly with a transcript and a presentation document attached to. These videos can be viewed directly in web browser on demand or downloaded to user's computer. After

studying the materials, students can submit assignments' solutions and take quizes and receive a Verified Certificate for the accomplishment of the course. These certificates are not free.

Most of the courses are in English and only a few of the courses are also translated into other languages. It is not possible for everyone to publish his materials through Coursera.

1.1.2 Youtube.com

Youtube.com [10] is not an educational service by design. Youtube allows people to create their own channels and upload their video content to share it with other Internet users for free.

Youtube was launched in 2005 and has became one of the most frequently visited websites on the Internet according to Alexa Internet [11]. Uploading video to Youtube is free, advertisment is displayed to the user while watching videos though.

The ease of making original-created videos available and the wide audience makes Youtube a perfect place for all individuals and organizations, who want to share their ideas or any video materials. Many educational channels can be found here, for example *Numberphile* [12], *Veritasium* [**?**], and *Khan Academy* [14].

Youtube videos can be viewed only online in a web browser or a specialized mobile app. There are only unofficial tools for downloading these videos.

The form and content of the video is practicaly unlimited, as long as does not violate the terms of the service. The maximum file size of a video is 128 GB in size and 11 hours in length. To upload larger or longer videos, user must split them into several parts. Video can have a text description which might contain the transcription of the video content and any subtitles can be attached to a video. Youtube also downscales videos to multiple resolutions so they can be viewed with a low speed Internet connection.

1.1.3 Moodle

Moodle [7] is an open-source project used by millions of users [8] providing a robust tool for creating custom learning materials and providing them to students. The source code of Moodle can be downloaded and deployed it any server. Teachers then publish their materials and resources for students or for assigning homework.

1.1.4 Educreations and ShowMe

Educreations is a service for creating and sharing educational videos similar to Khan Academy videos. The idea is that teachers create a their own videos by drawing onto a digital whiteboard and then share these videos with their classes.

Educreations is designed for an iPad device, but can be used also as a web application from any web browser, that supports Adobe Flash Player. Web application supports playing and recording and video player can be easily embeded into any website. Content of the video is scaled appropriately to the output screen and is drawn smoothly.

Figure 1.1: Khan Academy lesson

There is a free variant, which is limited. For more storage capacity and advanced tools, users must pay monthly fees. The software is proprietary and is is the file format of the video.

ShowMe [1] is a very similar service. An iPad app is available for recording and viewing videos. Recorded videos can then be played in a web browser, but not recorded. Pricing is very similar to Educreations.

Both ShowMe and Educreations of these services target mainly on tablets. The advantage of tablets is their touchscreen, which can be used for drawing in a natural way with fingers or a special stylus.

1.1.5 Khan Academy

A similar service to Coursera is *Khan Academy*. Khan Academy originated in 2003 when it's founder, Salman Khan, began tutoring his cousin over an instant messenger via drawing pictures with a computer mouse. Salman then started to record these videos and put them on his Youtube channel, so someone could watch them later. This channel became the basis of Khan Academy.

Khan Academy became known and has grown a lot, but the style of Khan Academy videos remained the same. A person draws lines and diagrams using a bitmap editor on his computer and talks about the subject aloud while recording his computer screen and recording his voice using a microphone. This style is sometimes called the "Khan-style-video" (KSV) [2]. These videos are then uploaded to Youtube and embedded in the Khan Academy website [15].

Apart form the video lectures, the website also contains exercises and quizes to encourage students in learning. The pace of the lesson depends on the student. He can pause the videos or watch them multiple times before continuing with the lesson 1.1.

Most of the videos are recorded in English, but many of the videos are translated into other languages - by replacing the audio track with a different one or with subtitles. One of the projects working on the localization of Khan Academy videos is a czech branch called Khanova Škola[16].

[1] http://www.showme.com/

[2] https://www.youtube.com/watch?v=Ohu-5sVux28

2. The Vector Video project

In the spring of 2014 the people behind "Khanova Škola" – the Czech branch of the Khan Academy – were looking for a person to develop an experimental video technology based on vector graphics. Their idea was to record raw data of user's input as he creates a Khan-style-video and later, when another user watches the video, draw the video scaled to match user's device's resolution and thus achieving maximum quality of the output. The result of this will be that the video will never become obsolete because it's quality isn not keeping up with the times.

Thanks to the sparse nature of vectors, in comparison to bitmaps, the data consumption might also be reduced or at least similar. This animation would also be linked to an audio track of authors voice commentary forming a complex KSV video suitable for educational purposes.

2.1 Video recording tool requirements

User, who wants to create a new video, should enter a website in his web browser and without installing any additional software start recording a video using his mouse, touchscreen or digital stylus and a microphone. This video should then be uploaded to the server.

If the user uses a pressure-sensitive stylus, then the applied pressure during drawing should be recorded too. The more pressure the user applies, the thicker the line will be and vice versa. Using this specific hardware might require additional drivers or specific software installed.

Different brush sizes, brush colors and background colors are available for the user. User should be able to erase certain parts of the canvas or the whole canvas at once. User should be able to pause and continue recording at any time.

Lines are immediately rendered on the screen as the user draws them, so he sees exactly the same output as the viewer will when he is playing the video later. All the raw data collected from the user should be stored, including the data that have no effect for video playback, like recording cursor movement while recording is paused. This will allow the user to simulate the process of video recording in the future and use it for further post-processing in any distant future. Recording of this redundant information should be optional.

2.2 Video player requirements

Any user should be able to play vector video on-line in all major modern web browsers without any special software or plugin installed, including mobile browsers.

Video should be scaled appropriately to the size of the player and device's screen resolution. The lines should have the same shape as the author has intended.

User can pause and continue with the playback of the video at any time. User can skip to any point of the video either forward or backward.

If the author of the video had recorded his voice, it must be played along the video. Audio must be synchronized with the video whenever user plays or pauses the video or when he skips to a different point on the time line.

User interface should be intuitive and easy to use either on a desktop computer using mouse and keyboard, but also with touchscreens on mobile devices.

2.3 Goal of this thesis

The result of this thesis should be an open-source library suitable for extending any web application with the abilities of recording and playing Khan Academy style videos in modern web browsers. An appropriate vector-based format should be chosen or defined to store video data.

The library should be easily adjustable and configurable for different purposes. User interface should be fully translatable to any language.

3. Analysis

3.1 Technical requirements

The overall project consists of two separate tools — the recorder and the player. Each of these tools behaves differently and will be used by different people. While the video player will be used by general audience, the recorder will be used by a much narrower group of content creators.

The recording tool will capture the movement of a virtual chalk and lines drawn on a virtual blackboard, as well as voice of the author using a microphone. The recorder data will be then sent to the server, where it should be stored. The video player will receive previousely recorded data and display the movement of the chalk and lines created by the author while playing the voice commentary.

For the purposes of recording, the tool must be able to capture the input from a microphone, track mouse movement and left mouse button state, collect information from Wacom graphics tablet devices and draw lines on screen. What the creator sees should be the same as what the end user will watch later.

3.2 Available techonologies

Web is a huge and fast growing environment. In only a few years, it has become a universal place for presenting and exchanging information. This put the web in the focus of many software companies and organizations and as a result, many different technologies for developing *Rich Internet Applications* (RIA) have been created. Some of them have already faded into obscurity, other are just emerging. One of the main limitations in the selection of the right technology for developing web application is their compatibility with different operating systems and web browsers.

3.2.1 Java applets

Java applets are used for creating interactive applications withing web browser [17]. Java applets meet all the specified technical requirements of both video player and recording tool.

Java applets are written in any language, that can be compiled into bytecode, this bytecode is then downloaded to the web browser and then run using Java Virtual Machine (JVM). This means that to be able to run a Java applet, user needs to have JVM installed on the device and an installed and allowed Java plugin in user's browser. There is no support for mobile operating systems such as iOS and Android [18] and Google Chrome has also disabled Java applets by default in most recent versions for security reasons.

3.2.2 Adobe Flash

Adobe Flash is a multimedia and software platform used for creating vector graphics, animations and games. Flash has all required features: vector graphics ma-

nipulation, working with XML, mouse input capturing, microphone input, and audio streaming [19].

To view Flash animations or to execute Flash applications, Adobe Flash Player is needed. Adobe Flash Player is available and being developed for all major operating systems, although that is not true for mobile platforms. There was never any support for Apple iOS [20] and in 2012 deveolpment of Flash for Android was discontinued [**?**]. Using Adobe Flash would mean to exclude most users of tablets and smartphones [23], which is a large disadvantage of this technology.

3.2.3 Microsoft Silverlight

Microsoft Silverlight [24] a development tool for creating web applications. It is based on the .NET Framework and it is similar to Java applets and Adobe Flash. It was Microsoft's attempt to compete Adobe Flash, but wasn't has not gained much popularity. Development of Silverlight was also discontinued by Microsoft in 2012 and it will not be supported in the Microsoft's new web browser Microsoft Edge[25]. These facts make Silverlight unsuitable for this project.

3.2.4 HTML5

HTML5 is the fifth revision of *HyperText Markup Language* (HTML) standard by the *World Wide Web Consortium* (W3C). It has been given the Recommendation status in the end of 2014 and all crutial aspects of both tools can be implemented using the proposed standard. Tracking mouse was long supported even in HTML and ECMAScript/JavaScript[1] through DOM Events[29].

Vector graphics are supported through the Scalable Vector Graphics (SVG) format [27] and it can be manipulated through Document Object Model (DOM) API [28], as well as any other XML content. MediaStream API [] enables access to audio input from user's microphone. The `<audio>` tag can be used to stream audio files and play them in the web browser.

While HTML5 is a new standard, many of the most important features have already been implemented in some web browsers, like Google Chrome and Mozilla Firefox. More specifically all the features needed to implement video player are supported in the latest versions of all major web browsers. The only catch might be a disagreement on supported audio file formats among the developers of web browsers. This can be overcome by converting the audio into the most widely used formats and providing them to the browser, which will then choose the one it supports.

The features needed by the recording tool, like the MediaStream API, are more specific and the number of browsers supporting these features is smaller, but this won't be an issue for content creators, who can easily install a supported web browser on all desktop platforms and even some mobile ones. Also a fast development in this area is expected and these features will be most likely implemented in all major browsers soon.

[1] ECMAScript is familiarly known as JavaScript after it's most common implementation[26]

Scalable Vector Graphics (SVG) and Synchronized Multimedia Integration Language (SMIL)

SVG SVG is an Extensible Markup Language (XML)[2] based file format designed for describing two-dimensional vector images[27]. It is an open format developed and maintained by the W3C SVG Working Group []. Current W3C Recommendation is SVG 1.1 (Second Edition).

SMIL *SMIL* is an Estabelished standard for are a way of define animation of SVG images using a declarative approach. It is an extension of SVG specification that lets the user to define key-frame animations of SVG elements attributes. SMIL is not supported by all major browsers[3]. SMIL's future is unsure at the moment, as Blink rendering engine development team has announced to deprecate the technology in Chromium and Google Chrome in favor of CSS animations and the *Web Animation API* [] [4].

Web Animation API

A very promissing technology is the *Web Animation API* [5], which will provide a very effective way of declaring and controling animations in a web page through a JavScript API. This technology is currently (3rd July 2015) in the state of Editor's Draft. This means that it is provided for discussion only and may change at any moment [] [6].

Unfortunatelly, at the time of writing this thesis, support for this technology is not supported by several major browsers[7][8] and is only partially supported in the remaining few [9][10].

3.2.5 Conclusion

The bottleneck of most of the technologies is their support in mobile devices. These devices don't allow some of the above mentioned technologies to run inside them. As the popularity and market share of mobile devices grows, supporting them is a high priority. HTML5 APIs features needed to create the video player are implemented major web browsers, including the versions of browsers for mobile devices. Audio recording is not supported by all major browsers at the moment, but this will change with the release of Microsoft Edge []11. After considering these facts, HTML5 was selected to implementat eigher the recording tool and the video player.

[2] http://www.w3.org/XML/
[3] https://status.modern.ie/svgsmilanimation
[4] https://www.chromestatus.com/features/5371475380928512
[5] http://www.w3.org/TR/web-animations/
[6] http://w3c.github.io/web-animations/
[7] https://status.modern.ie/webanimationsjavascriptapi
[8]
[9] http://caniuse.com/#feat=web-animation
[10] https://birtles.github.io/areweanimatedyet/
[11] http://blogs.windows.com/msedgedev/2015/05/13/announcing-media-capture-functionality-in-microsoft-edge/

3.2.6 Possible issues and known limitations

As mentioned earlier, not all browsers implement all features described in the HTML5 specification in their lastest versions. But the trend is to implement as many technologies as possible to bring better browsing experience to users to gain bigger market share, so this situation will get better soon.

Another fact, that might cause trouble, is that many users still use an old version of their browser. This is mainly the case of Microsoft Internet Explorer (MSIE), which does not have automatic update mechanism. Fortunatelly, the number of MSIE users is dropping and Microsoft Edge, the ancestor of MSIE, will have automatic updates built in (@todo: some relevant source).

So far, there hasn't been consensus on formats and codecs support in HTML5 *WebAudio API*. The most supported format across web browsers is MP3. This format isn't supported by some bowsers on some platforms due to licence reasons. An up-to-date table with media formats' support can be found at *Mozzila Developer Network* (MDN) website [] [12]. This inconvenience can be easily solved by providing several `<source>` elements to the `<audio>` element with different formats. Browser will den select the format it supports and it will download and play this audio track. The flaw is the need of having the data stored on the server in several formats and the video takes more disk capacity.

3.3 Drawing algorighm

One of the key aspects of success of this library is the way it draws the lines. The lines must be nice, smooth and feel natural, as if sombody drew physicaly draw them on a blackboard with a chalk, and not disturbing or distracting. This library cannot improve the contents of the author's video, but the way it is displayed to the user can have possitive impact on viewer's impression of the video. It is also required that stylus pressure has direct proportion to the width of the drawn line.

After a research, the algorithm called "Dynadraw" by Paul Haeberli [**?**] from 1989, seems appropriate for this project. It is used in other open-source projects like the *Inkscape*[13] for it's calligraphic tool[14].

This algorighm models the tip of the brush as a physical object and simulates it's movement. User applies force on the brush by moving the mouse or any other pointing device. The original algorighm is designed for calligraphic strokes and uses mouse speed to calculate dynamic line widths. This technique makes lines drawn by a mouse or other pressure-insensitive device more interesting and can be combined with pressure level of pressure-sensitive devices.

For details of the implementation see section 5.3.1 on page 20.

3.4 Audio recording

Recorded audio will have the form of raw uncompressed *Pulse-code modulation* (PCM) data[46]. The sample rate is chosen by the web browser and cannot

[12] https://developer.mozilla.org/en-US/docs/Web/HTML/Supported_media_formats

[13] Inkscape: http://inkscape.org

[14] http://bazaar.launchpad.net/ inkscape.dev/inkscape/trunk/view/head:/src/ui/tools/calligraphic-tool.h

be changed by the programmer. The sample rate must be between 22.05 to 96 kHz. If we choose the bit depth of 16 bits per sample, than we can calculate the minimum data load of uncompressed video recording. Therefore we receive at least 44.1 kB of data per second, 2.646 MB per minute of recording. From experience, desktop browsers target 44.1 kHz sampling rate, which even doubles the data load. This data must be upload to the server so users can watch the screencasts with voice commentary.

Buffering recorded data The first possible approach is to buffer all the recorded data, create an uncompressed audio file structure and upload this file to the server at the end of the recording. Creating a *Waveform Audio File Format* (WAVE or WAV) file is relatively simple because of it's simple header and the data is a stream of LPCM data.

Compression of data in browser The uncompressed audio is relatively large and the upload will take much more time, than uploading a compressed file would take. Audio compression is a complex process. There are several JavaScript libraries, that implement audio compression inside a web browser[15][16], but they have shown to be large, slow and unstable.

Streaming data in background A different method is possible with the use of the *WebSocket protocol*. The WebSocket protocol enables two-way communication between a client and a server [48]. It is built on top of TCP sockets, which means it provides ordered and reliable transmission of messages. Either textual or binary messages can be sent over a WebSocket. If the data is transfered over the WebSocket at the time of recording in background, then a considerable part of the data might be already uploaded to the server as soon as the user stops recording, depending on his Internet connection upstream speed.

3.4.1 Conclusion

Streaming the recorded audio data in background over a WebSocket seems to be convenient for the user. It will fasten the uploading process at the end of video recording considerably, as the size of uncompressed audio data is relatively big.

Server must implement WebSocket connections and serve clients. Server might also use a native program, like the open-source FFmpeg[17] to compress the audio and covert to different file formats.

3.5 File format

Recorded data must be stored in a data storage, so it can be played later. Vector video file must contain all the information about cursor movement and precise data of the lines, including their variable width, and the times at which every single segment of a line is drawn. This data is then used to reconstruct the precise

[15] http://bgrins.github.io/videoconverter.js/
[16] https://github.com/akrennmair/libmp3lame-js
[17] https://www.ffmpeg.org

movement of the author's cursor and draw the very sqme lines at the very same pace.

EVA There are several existing vector based formats available for animations. One of them is *EVA* by SHARP[38] developed in 1996. EVA is a binary format popular in Japan. Unfortunatelly the documentation and resourcecs are available only in Japanese[39].

SWF Another well-known format is Adobe's *SWF*. It is a binary format with a wide rage of usage. It can contain either bitmap-based video, vector primitives, and ActionScript[40]. It is possible to manipulate contained vector primitives and morph their properties to create a key-frame animation[22]. Documentation of the format is open since 2008.

SVG *Scalable Vector Graphics* was already mentioned earlier. This format can contain SMIL animation definitions, which makes it also a vector animation format. SVG is based on XML and it is therefore a human-readable textual format. As discussed earlier SMIL animations are not considered a good option for this project.

Custom extension of SVG JavaScript is not very good at manipulating binary files. Until the specification of typed arrays[18], manipulating binary files in JavaScript was very ineffective and unintuitive. Even with these new technologies, working with XML in JavaScript is more straighforward. The fact, that SVG is based on XML means, that it can be extended with custom namespace and define custom animation, that will be better for this project.

3.5.1 Conclusion

After considering available options, creating a custom format based on SVG seems like the best option for it's human-readability, simplicity, versatility, ease of manipulation using JavaScript, and the possibility of previewing static scenes of the animation in existing software. This format will serve well as a proof-of-concept and can be replaced with a more sophisticated format in the future and the format must not be therefore hard-coded into the library.

For the specification of this new format, see chapter 4 on page 14.

[18] https://www.khronos.org/registry/typedarray/specs/latest/

4. Vector Screencast file format

SVG format is common and is implemented in web browsers and in many programs. These programs can open an SVG image and draw it's contents. The idea is to create an SVG file, which contains all the information needed using custom namespace attributes and elements and shows the state of the blackboard at the end of recording.

A valid SVG document must have an *svg* root element with specific namespace attributes and specified *width* and *height* attributes. For the purpose of extending the document, a new namespace *http://www.rozsival.com/2015/vector-screencast* is added with the prefix *a*. An example of an empty SVG document with this namespace looks as follows:

```
1 <?xml version="1.0"?>
2 <svg version="1.1"
3         width="470"
4         height="100"
5         xmlns="http://www.w3.org/2000/svg"
6         xmlns:a="http://www.rozsival.com/2015/
            vector-screencast">
7
8 </svg>
```

SVG specification allows inclusion of elements and attributes from foreign namespaces anywhere with the SVG context. Attributes from foreign namespaces might be attached to any element. Both elements and attributes will be included in the DOM by the SVG user agent, but will be ignored otherwise [**?**].

Vector Screencast root element must have two child elements: `<metadata></metadata>` and `<g a:type="chunks"></g>`

SVG `<g>` element[1] is intended for grouping related graphics elements. These groups might be also nested.

4.1 Video Metadata

The first child element of the `<svg>` element must be a `<metadata>` element. This element must contain these child elements:

`<a:width>` The content is the original width of the blackboard. This number will be used to correct coordinates when playing the video with different video resolution. The `width` attribute of the `<svg>` element might contain different value.

`<a:height>` The content is the original height of the blackboard. This number will be used to correct coordinates when playing the video with different video resolution. The `height` attribute of the `<svg>` element might contain different value.

[1] http://www.w3.org/TR/SVG/struct.html#GElement

<a:length> The content is the duration of the video in milliseconds.

<a:audio> Contains the list of audio sources as child elements.

<a:source > Defines one audio source. It has two attributes: a:src – containing the URL of the audio source, and a:type – containing the MIME type of the audio source.

4.2 Video Chunks

Video is divided into smaller consequent parts, called *chunks*. Chunks have varaible time duration based on user's behaviour. They are a logical unit of the video, each of them can be rendered at once as one primitive. This structuring of the whole video into smaller chunks helps optimalizing skipping parts of the video. Some parts can be skipped entirely, as the author cleared the canvas at some point, others are rendered at once without evaluating any animation commands.

There are three types of chunks: *path*, *erase*, and *void*. Each chunk is stored as one SVG group elment with a specific a:type attribute (*path*, *erase*, *void*) and an a:t attribute, which is the time, when this chunk starts to be processed, in milliseconds. Chunks contain the prerendered primitive and a list of animation commands.

"Path" This type of chunk represents one line the user draws. The first child element is an SVG <path> element. The fill attribute should correspond to the real color of the path, but is not neccesary for later use. The data attribute d includes serialized information about the segments of the path in the form of a valid SVG path instructions[2]. For more details about the serialization and deserialization of this data, see section @todo on page @todo. One or more child elements might follow after the <path>, all of which must be animation command elements.

"Erase" This type of chunk represents clearing the whole canvas with one color. The first child element is an SVG <rect> element. The fill attribute should correspond to the real color of the new background. One or more child elements might follow after the <rect>, all of which must be animation command elements.

"Void" This type of chunk does not render anything on the canvas. It might contain one or more child elements, all of which must be animation command elements.

4.3 Animation Commands

Animation commands are necessary for correct actions timing during the video. Chunks contain the information of how the resulting primitive looks like, com-

[2]http://www.w3.org/TR/SVG/paths.html#PathData

mands user's gradual forming of these primitives.

Animation command must be a child element of a chunk. Every chunk must have a `a:t` attribute with the precise time of the action in milliseconds. The value of the time attribute of an element must be greater or equal to the values of the preceding sibling action elements.

<m> *Move cursor* command tells the player to move cursor to a specific position defined by attributes `a:x` and `a:y`.

<c> *Change color* command tells the player to switch current brush color according to the value of `a:c`. Value of the attribute must be valid CSS color value[3].

<s> *Change brush size* command tells the player to change current brush size to the value of `a:w` attribute. The units of this value are pixels and the size is relative to the width and height stated in the *metadata*.

<d> *Draw next segment* segment of the line will be rendered with current brush color and size. This command doesn't carry any aditional information, all the information about the segment is inside the *path* chunk. *Draw next segment* commands can be present only in an *path* chunk. There must not be any *change color* or *change brush size* command between the first *draw next segment* command and the last one inside one *path* chunk – color or size can't be changed in the middle of the path.

[3]http://www.w3.org/TR/CSS2/syndata.html#value-def-color

5. Implementation

5.1 ECMAScript and JavaScript

ECMAScript [1] is a standardized scripting language widely used in website development. Latest approved edition of ECMAScript is ECMAScript 5.1, which is implemented in most major web browsers, and it's implementations in web browsers are commonly called JavaScript [2][3].

JavaScript is a dynamic programming language [4], which combines multiple aspects of imperative, functional, and object-oriented programming.

Functions are so called first-class citizens. This means they can be stored in variables, passed as function parameters, returned as results of functions, and included in data structures.

Object oriented programming (OOP) [5] in JavaScript differs from class-oriented OOP in the way inheritance is implemented. While in class-oriented languages, for example in C++ or C#, inheritance is achieved by declaring classes of objects. In JavaScript, objective oriented programming is implemented through object prototypes. Prototype is just a link to another object, which has another prototype. This way a prototype chain is created. The last object in this chain has a `null` prototype. When trying to access a property of an object, it is searched in it's own properties. If it is not found, then it is searched in it's prototype's properties and so on until the end of the prototype chain is reached [31].

JavaScript doesn't provide any mean of static type checking. All types are created during runtime and they are also checked only during runtime. This lack of static checking during compilation might lead to rarely occurring errors and it is important to take this in mind while writing JavaScript code. A good practice is to write documentation comments[6], where the types are stated.

JavaScript is an interpreted language, some implementations also use a Just-in-time compilation (JIT) [7] for better performance. Another very important aspect of ECMAScript which affects it's performace is the absence of direct control over memory usage – memory is released when it is not needed any more. This mechanism is called *Garbate collection*. As of 2012, all modern browsers use mark-and-sweep garbage-collector [**?**].

5.1.1 TypeScript

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript[33]. It is an open-source project developed by Microsoft. It is, as it's name suggests, is a strongly typed programming language compatible with JavaScript.

[1] http://www.ecmascript.org/
[2] https://developer.mozilla.org/en/docs/Web/JavaScript
[3] https://msdn.microsoft.com/cs-cz/library/d1et7k7c(v=vs.94).aspx
[4] http://en.wikipedia.org/wiki/Dynamic_programming_language
[5] http://en.wikipedia.org/wiki/Object-oriented_programming
[6] Commonly used syntax in JavaScript projects is JSDoc (http://usejsdoc.org/)
[7] For example Google V8 engine used in Google Chrome. See https://code.google.com/p/v8/

TypeScript extends capabilities of JavaScript by static type checking in the time of compilation, which helps finding errors in source code, and speeds up the process of coding. TypeScript also introduces class-based object oriented programming to JavaScript. It includes concepts of interfaces and polymorphism, which makes programs written in TypeScript more understandable to programmers familiar with other object-oriented programming languages like Java, C# or C++.

TypeScript uses several features of ECMAScript 6, but is transcompiled into ECMAScript 5, and therefore doesn't bring any new functionality. The reason for choosing TypeScript is the clarity of code and more convenient development process for the programmer.

5.2 HTML5

HTML5 is used to refer to modern web technologies. The core is the *HyperText Markup Language*, designed for semantic description of documents [34]. HTML5 extends this semantics with new HTML tags such as `<header>` or `<footer>`, but also defines a huge set of new APIs, that allow web developers to create much richer and universal websites and web applications. Some of these new technologies are needed by the Vector Screencasts project.

Working with XML data AIn JavaScript, working with XML data is very similar to working with regular website DOM[8]. `Document`[9] interface is used for traversing the XML tree, modifying it or for creating a new tree. In HTML5, XML files can be opened using a HTTP GET request using `XMLHttpRequest`[10] object, which returns `Document` instance in it's `responseXML` property, if the document meets specified criteria [36].

Web sockets Web sockets allow applications to open bidirectional communication channels with server-side processes [][11]. Web sockets are built on top of *TCP* (Transmission Control Protocol) connection. This means that the delivery of data is reliable and ordered [][12].

Web Workers JavaScript code of a website normaly runs in a single thread. It is common to run asynchronous functions as callbacks and event handlers, but these are still executed in a one thread. Web workers are a means of running heavy tasks without affecting the user interface in background. A new web worker is creating as an instance of `Worker` object with an URI of the script with it's source. A real OS-level thread is spawn for each worker.

WebWorkers have several limitations. They can't change the DOM and have direct links to objects in the main thread. WebWorkers receive messages from the JavaScript source that created them through `onmessage` event and can send

[8]Document Object Model, http://www.w3.org/DOM/
[9]https://dom.spec.whatwg.org/#interface-document
[10]https://xhr.spec.whatwg.org/
[11]https://html.spec.whatwg.org/multipage/comms.html#network
[12]https://tools.ietf.org/html/rfc6455

a response via the `postMessage` function. These messages contain serialized JavaScript objects, which cannot contain any references. As a result, concurrency problems are not typically a problem. WebWorkers can perform a HTTP requests using the `XMLHttpRequest` object, but the content is not parsed, if it is an XML file. For more details see the specification [] [13]

Rendering graphics using HTML5

Displaying text and static visual content is the main purpose of HTML and Cascade Style Sheets (CSS) and it is widely used this way across the web. Creating a complex polygon or curve would be very hard and would involve various tricks [14] or would be even impossible.

The new HTML specification takes this in mind and brings ways of creating more rich and dynamic content within a web page. There are two technologies that should be taken into account - Canvas 2D Context and SVG.

Canvas 2D Context The Canvas element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, or other visual images on the fly [35].

Using canvas seems appropriate for this project. Canvas could be created with respect to user's resolution and web browser window size. All elements can be scaled to fit this viewport. This will make them look sharp and there won't be any artifacts, noise and blur caused by interpolation which would be caused by scaling normal bitmap video.

The problem with Canvas might occur when user resizes his window or enters full-screen after the canvas is initialized. Canvas contains a bitmap image consisting of graphical primitives drawn onto it. All content must be redrawn so it remains sharp.

5.3 Drawing lines

Khan Academy videos are known for it's consistent and simple style. A person draws on a virtual canvas (evoking a school blackboard) with a brush (or possibly a chalk) of a round shape.

Tutor has a pointing device, typically a computer mouse or digital pen, and it's position on the canvas is marked with a moving cursor. When the tutor clicks, a dot is marked on canvas in the current position of the cursor. Tutor can produce a line (typically a curve) following this cursor when he presses a mouse button or increases digital pen pressure and while moving the cursor. The curve ends when he releases the button or pen pressure. The color and size of the dot or line corresponds to the current settings.

At the time of recording, mouse coordinates relative to the drawing board are captured along with current pressure of the digital pointing device (mouse, graphical tablet pen). This data is then used to draw a curve with variable thickness at the moment of recording as visual feedback for the person recording and the same process is done every time the video is replayed.

[13] https://html.spec.whatwg.org/multipage/workers.html#worker
[14] http://nicolasgallagher.com/pure-css-gui-icons/

The outcome of the rendering phase should be the same every time so the intention of the creator is preserved. On the other hand, the rendering algorithm might be improved (i.e. by making the lines smoother) in the future and the video could be rendered using to this algorithm without any editing, while the information in the video will remain untouched.

Rendering at the time of playback gives us the opportunity to adjust the outcome to the environment of the end user. This means that the result can be sharp on every display resolution without the need of having many versions of the same video for each resolution.

5.3.1 DynaDraw algorithm

Paul Haeberli has created a simple algorithm called *"DynaDraw"* in 1989, which is suitable for calligraphy. Brush is modeled as a physical object with it's mass, velocity and friction coefficient [15] []. Mouse movement is interpreted as a way of exerting force on the brush – the faster you move the brush, the greater the force applied on the brush is. Acceleration is then calculated according to Newton's second law of motion considering brush's mass and velocity of the brush is derived with respect to the amount of brush's friction. This velocity is then applied and brush is moved. The trace brush should leave behind is then drawn onto the blackboard.

The advantage of this algorithm is it's simplicity and the possibilities of configuring the brush with different values of mass and friction (the author of the algorithm refers to this constant also as a drag, which better fits the purpose of slowing down the brush).

Heavier brushes move slowly, but the path they leave behind is much smoother, as the hand shaking is eliminated by composition of forces in different directions.

Light brushes move faster and are often very close to the cursor during the movement. When the cursor stops abruptly, light brushes with little friction tend to keep moving past the cursor and wrap around it. This produces little curls at the end of lines.

This approximation of cursor movement with appropriate constants selected improves quality of user's input and leads to nice results event when the user is using an ordinary mouse instead of a digital drawing pen.

Brush movement simulation

One step of simulation applies force on the brush according to current mouse position and thus moves it in the direction of the pointer. This process of applying force must be done periodically, at the frequency of 60 Hz in ideal case[16]. Implementation of this simulation is not identical to the original *DynaDraw* algorithm, but all of it's key aspects are preserved. Pseudocode 1 describes the algorithm used in this theses.

The main difference between the original implementation and the one used in Vector Video is brush width calculation. The original algorithm calculates the

[15] DynaDraw: http://www.graficaobscura.com/dyna/index.html

[16] HTML5 provides requestAnimationFrame function, which is intended for animations and which targets 60 frames per second

width of the line in a specific point by measuring it's velocity – the faster the brush is moving, the thinner the drawn line is. This width dynamics gave the algorithm it's name.

In our implementation, this effect is implemented, but is much more subtle. Brush dynamic in this implementation relies mainly on the pressure of a digital pen on a graphics tablet. Since the exact value of pressure in the point of current brush's location is not always known, the value is linearly interpolated between the value of pressure in the previous position of the brush and the value of pressure in the current mouse position according to distance from each of these points.

As this simulation isn't deterministic, this process cannot be reconstructed afterwards and when the video is being played, only the already computed values of points along the path must be used. The precise value of pressure is therefore redundant for later playback of the video and

Pseudocode 1 One step of brush movement simulation

```
function OneStep(M, Δt)                    ▷ M - mouse position, Δt - elapsed time
    if M ≠ ∅ then
        brushMoved ← apply(M⃗, Δt)
        if brushMoved = true then
            DrawSegment
        end if
    end if
end function
function apply(M, Δt)
    F⃗ ← M − P                              ▷ P - current position of the brush
    a⃗ = F⃗/m                               ▷ m - mass of the brush
    if ‖a⃗‖ ≤ C_a then                      ▷ C_a - minimum acceleration constant
        return false;
    end if
    v⃗ ← v⃗ + a⃗
    if ‖v⃗‖ > C_v then                      ▷ C_v - minimum velocity constant
        P ← P + μΔtv⃗                       ▷ μ - coefficient of friction
        return true
    end if
    return false
end function
```

Rendering of one line segment

Line consists of many segments that are drawn after every simulation step, which causes brush to move. There are several ways to draw the segment, the most straightforward is to draw a simple quadrilateral between two points as shown in Figure 5.1 and Pseudocode 2.

The curves drawn with quadrilaterals are not very smooth. For smoother curves, straight lines must be replaced with interpolation splines. Both SVG and Canvas 2D Context implement cubic Bézier cures. Cubic Bézier curves are defined by four control points, $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$. Path starts in $\mathbf{P}_0$ and end in $\mathbf{P}_3$, it

Pseudocode 2 Draw one segment of a line

function DRAWSEGMENT
- $\vec{n} \leftarrow \frac{(-\vec{v}_y, \vec{v}_x)}{\|\vec{v}\|}$
- $w \leftarrow$ CURRENTBRUSHPRESSURE $\cdot\, b$ ▷ b - brush size
- $L \leftarrow P - w\vec{n}$
- $R \leftarrow P + w\vec{n}$
- BEGINPATH
- MOVETO(L') ▷ L' and R' - previousely drawn point
- LINETO(R')
- LINETO(R)
- LINETO(L)
- CLOSEPATH
- FILL(c) ▷ c - current brush color
- $L' \leftarrow L$
- $R' \leftarrow R$

end function

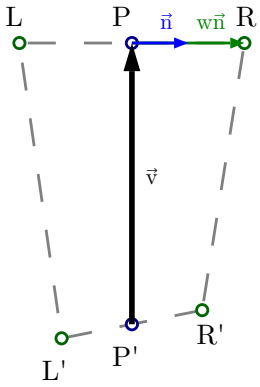


Figure 5.1: Drawn quadrilateral segment of a line

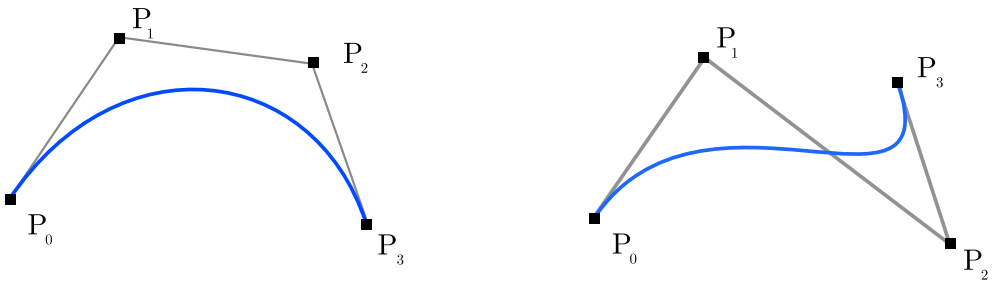


Figure 5.2: Bézier curve interpolation examples

doesn't usually go through points $\mathbf{P}_1, \mathbf{P}_2$. The interpolation forumla of the curve is

$$\mathbf{B}(t) = (1-1)^3\mathbf{P}_0 + 3(1-t)^2t\mathbf{P}_1 + 3(1-t)^2t^2\mathbf{P}_2 + t^3\mathbf{P}_3, 0 \leq t \leq 1$$

[][17]

To calculate the control points of a cubic Bézier curve for segment between points $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$, calculated by the DynaDraw algorithm, we can also take points $\mathbf{X}_{i-1}$ and $\mathbf{X}_{i+2}$ and look at these four points as Catmull-Rom spline control points. Catmull-Rom is a special type of Cardinal spline, with the tension parameter $\tau = 0$[][18]. This approach gives us a nice smooth curve calculated just from consequent points and also guarantee of C^1 continuity (@todo: citation) of the whole path.

A special conversion matrix between Catmull-Rom spline and Bézier curve is defined [][19] and so the conversion is very straighforward. The formula for calculating cubic Bézier control points $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, from the points $\mathbf{X}_{i-1}, \mathbf{X}_i, \mathbf{X}_{i+1}, \mathbf{X}_{i+2}$ is

$$\begin{pmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \mathbf{P}_3 \end{pmatrix} = \frac{1}{6} \begin{pmatrix} 0 & 6 & 0 & 0 \\ -1 & 6 & 1 & 0 \\ 0 & 1 & 6 & -1 \\ 0 & 0 & 6 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{X}_{i-1} \\ \mathbf{X}_i \\ \mathbf{X}_{i+1} \\ \mathbf{X}_{i+2} \end{pmatrix}$$

The first and the last segments must be processed in a special way, as there is no preciding point, or following respectively.

For an example of cubic Bézier curve, see figure 5.2.

[17] An introduction to splines for use in computer graphics and geometric modeling, Bartels, Richard H, p. 160, https://cs.uwaterloo.ca/research/tr/1983/CS-83-09.pdf

[18] http://people.cs.clemson.edu/ dhouse/courses/405/notes/splines.pdf

[19] http://therndguy.com/papers/curves.pdf

6. Integration of the library in a HTML web page

Using Vector Screencast library is intended to be as simple as possible. All the user needs to do is include a JavaScript file of the library and a CSS file of a theme into his HTML code and configure the player in a few lines of code.

6.1 Obtaining the Vector Screencast library

Prepared library files can be obtained either from the attached files or from the GIT repository https://github.com/simonrozsival/vectorvideo in the */release/VectorScreencast* and */release/themes/* folders. You only need to copy files *vector-screencast.min.js* and *theme-default.min.css* into your project.

The library files must be linked to your document and you should make sure that your website will be displayed properly on mobile devices by specifying the *viewport* meta tag in the *head* section of your document[1]. Also create an empty element with a specific *id* attribute – this will be the container, into which either the screencast player or the recorder will be placed.

An example of a HTML5 template with correct setup can look similarly (irrelevant parts of the document were let omitted and replaced by suspension points):

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      ...
5      <meta name="viewport" content="width=device-width,
           initial-scale=1">
6      <link rel="stylesheet" type="text/css" href="/path/
           to/theme-default.min.css" media="screen,
           projection">
7      ...
8    </head>
9    <body>
10     ...
11     <div id="some-specific-id"></div>
12     ...
13     <script src="/path/to/vector-screencast.min.js">
           </script>
14   </body>
15 </html>
```

The initialisation scripts must be executed afte all web page resources are downloaded and the DOM is ready – this can be achieved by putting this code inside a handler of the *window.onload* event.

[1] https://developers.google.com/speed/docs/insights/ConfigureViewport

6.2 Vector Screencast Player

Inside your scripts, create a new instance of *VectorScreencast.Player*. The constructor takes two arguments, first of them is the *id* attribute of a container element and the second is a configuration object. The only obligatory property of the configuration object is the *Source* property – the URL of the source SVG file.

There are several other interesting optional settings, that will help you customize the screencast player. One of them is the *Localization* property, which takes an object implementing the *VectorScreencast.Localization.PlayerLocalization* interface. To see the complete list of all configuration options and further details, please refer to the *VectorScreencast.Settings.PlayerSettings* interface in the API reference of the project in the */docs/* folder of the attached files. You can see an advanced example of *VectorScreencast.Player* usage */demo/public/play.html* in the attached files.

6.3 Vector Screencast Recorder

Inside your scripts, create a new instance of *VectorScreencast.Player*. The constructor takes two arguments, first of them is the *id* attribute of a container element and the second is a configuration object. The only obligatory property of the configuration object is the *Source* property – the URL of the source SVG file.

There are several other interesting optional settings, that will help you customize the screencast player. One of them is the *Localization* property, which takes an object implementing the *VectorScreencast.Localization.PlayerLocalization* interface. To see the complete list of all configuration options and further details, please refer to the *VectorScreencast.Settings.PlayerSettings* interface in the API reference of the project in the */docs/* folder of the attached files. You can see an advanced example of *VectorScreencast.Player* usage */demo/public/play.html* in the attached files.

6.4 Embedding Vector Screencast Player

@todo

6.5 Custom theme

@todo

7. Users' documentation

Users' documentation with screenshot similar to the one I have already done.

Conclusion

Bibliography

[1] Bower, Beverly L. Hardy, Kimberly P. *From correspondence to cyberspace: Changes and challenges in distance education* [online]. New Directions for Community Colleges, Volume 2004, Issue 128, pages 5–12.

[2] John Oliver: Student Debt, HBO [online] 9/7/2015, available at https://www.youtube.com/watch?v=P8pjd1QEA0c

[3] Bill Gates: Online, All Students Sit in the Front Row [online], posted on November 18, 2014, seen on 3/12/2015, available at http://www.gatesnotes.com/Education/Colleges-Without-Walls-Arizona

[4] Jessica Leber: In the Developing World, MOOCs Start to Get Real [online], MIT Technology Review, published on March 15, 2013, available at http://www.technologyreview.com/news/512256/in-the-developing-world-moocs-start-to-get-real/

[5] Kepler [online], http://kepler.org/

[6] Generation Rwanda [online], http://www.generationrwanda.org/

[7] Moodle, https://moodle.org

[8] Moodle statistics, https://moodle.net/stats/

[9] Coursera [online], 3/13/2015, available at https://www.coursera.org/about/

[10] Youtube.com, https://www.youtube.com

[11] Alexa Internet, http://www.alexa.com/topsites

[12] Numberphile Youtube channel, https://www.youtube.com/user/numberphile

[13] Veritasium Youtube channel, https://www.youtube.com/user/veritasium

[14] Khan Academy Youtube channel, https://www.youtube.com/user/khanacademy

[15] Khan Academy, viewed July 8, 2015 [online] https://www.youtube.com/user/khanacademy

[16] Khanova škola, https://khanovaskola.cz

[17] Java applet, Wikipedia, the free encyclopedia, viewed July 8, 2015 [online] https://en.wikipedia.org/wiki/Java_applet

[18] Oracle, Java.com, How do I get Java for Mobile device? [online] viewed April 16, 2015 http://www.java.com/en/download/faq/java_mobile.xml

[19] Features, Adobe Flash Player, Adobe.com, viewed July 8, 2015 [online] http://www.adobe.com/cz/products/flashplayer/features.html

[20] Steve Jobs, Apple.com, Thoughts on Flash, published April 2010, viewed April 16, 2015 [online] http://www.apple.com/hotnews/thoughts-on-flash

[21] Adobe Blog, An Update on Flash Player and Android, published June 28, 2012, viewed April 16, 2015 [online] http://blogs.adobe.com/flashplayer/2012/06/flash-player-and-android-update.html

[22] SWF and AMF Technology Center, Adobe.com, SWF and AMF Technology Center, viewed July 8, 2015 [online] http://www.adobe.com/devnet/swf.html

[23] Bosomworth Danyl, SmartInsights.com, Mobile Marketing Statistics 2015, published January 15, 2015, viewed April 16, 2015 [online] http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/

[24] Microsoft Silverlight, Get Microsoft Silverlight, viewed April 16, 2015 [online] http://www.microsoft.com/silverlight/

[25] Moving to HTML5 Premium Media, Microsoft Edge Dev Blog, viewed July 8, 2015 [online] http://blogs.windows.com/msedgedev/2015/07/02/moving-to-html5-premium-media/

[26] A brief history of ECMAScript versions (including Harmony/ES.next), Axel Rauschmayer, dzone.com, viewed July 8, 2015 [online] https://dzone.com/articles/brief-history-ecmascript

[27] Scalable Vector Graphics (SVG) 1.1 (Second Edition), The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/SVG/

[28] Document Object Model, Wikipedia, the free encyclopedia, viewed Junde 8, 2015 [online] https://en.wikipedia.org/wiki/Document_Object_Model

[29] Document Object Model (DOM) Level 2 Events Specification, The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/DOM-Level-2-Events/

[30] SVG Extensibility, The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/SVG/extend.html

[31] Inheritance and the prototype chain, Mozilla Developer Network (MDN), viewed July 8, 2015 [online] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Inheritance_and_the_prototype_chain

[32] Memory Management, Mozilla Developer Network (MDN), viewed July 8, 2015 [online] https://developer.mozilla.org/cs/docs/Web/JavaScript/Memory_Management

[33] TypeScript, viewed July 8, 2015 [online] http://www.typescriptlang.org/

[34] HTML Living Standard, The Web Hypertext Application Technology Working Group (WHATWG), viewed July 8, 2015 [online] https://html.spec.whatwg.org/#is-this-html5

[35] The canvas element, The World Wide Web Consortium (W3C), viewed July 8, 2015 [online] http://www.w3.org/TR/2010/WD-html5-20100624/the-canvas-element.html

[36] XMLHttpRequest, HTML Living Standard, The Web Hypertext Application Technology Working Group (WHATWG), viewed July 8, 2015 [online] https://xhr.spec.whatwg.org/#document-response

[37] Dynadraw algorithm, Paul Haeberli, viewed July 8, 2015 [online] http://www.graficaobscura.com/dyna/

[38] EVA animater plaza, SHARP, viewed July 8, 2015 [online] http://www.sharp.co.jp/sc/excite/evademo/evahome.htm

[39] Extended Vector Animation, Wikipedia, the free encyclopedia, viewed July 8, 2015 [online] https://en.wikipedia.org/wiki/Extended_Vector_Animation

[40] SWF, Wikipedia, the free encyclopedia, viewed July 8, 2015 [online] https://en.wikipedia.org/wiki/SWF

[41] WAVE Specifications, Prof. Peter Kabal, viewed July 9, 2015 [online] http://soundfile.sapp.org/doc/WaveFormat/

[42] Media Capture and Streams, W3C Last Call Working Draft 14 April 2015, viewed July 9, 2015 [online] http://www.w3.org/TR/mediacapture-streams/#dom-mediadevices-getusermedia

[43] The ScriptProcessorNode Interface, Web Audio API, W3C Working Draft 10 October 2013, viewed July 9, 2015 [online] http://www.w3.org/TR/webaudio/#ScriptProcessorNode-section

[44] The ScriptProcessorNode Interface - DEPRECATED, Web Audio API, W3C Editor's Draft 21 June 2015, viewed July 9, 2015 [online] https://webaudio.github.io/web-audio-api/#the-scriptprocessornode-interface—deprecated

[45] AudioWorkerNodeProcessor Interface, Web Audio API, W3C Editor's Draft 21 June 2015, viewed July 9, 2015 [online] http://webaudio.github.io/web-audio-api/#the-audioworkernodeprocessor-interface

[46] The AudioBuffer Interface, Web Audio API, W3C Working Draft 10 October 2013, viewed July 9, 2015 [online] http://www.w3.org/TR/webaudio/#AudioBuffer

[47] Pulse-code modulation, Wikipedia, the free encyclopedia, viewed July 9, 2015 [online] https://en.wikipedia.org/wiki/Pulse-code_modulation

[48] The WebSocket Protocol, Internet Engineering Task Force (IETF) RFC 6455, viewed July 9, 2015 [online] https://tools.ietf.org/html/rfc6455

[49] The WebSocket API, W3C Candidate Recommendation 20 September 2012, viewed July 9, 2015 [online] http://www-mmsp.ece.mcgill.ca/documents/AudioFormats/WAVE/WAVE.html

List of Tables

List of Abbreviations

Attachments