Main features of HTML5

Lecturer – Mukazhanov N.K.
Department - CE&IS

Content:

- Reminders
- Philosophy
- Semantic markup
- Form elements
- Media

HTML: Reminder

Markup version

```
<!DOCTYPE html>
<html>
<head>
```

HTML header

```
  <title>School Announcement</title>
</head>
<body>
```

Heading

```
  <h1>JINR/CERN School 2014</h1>
```

link

```
  <p>JINR, <a href="http:\\www.cern.ch" >CERN</a> and MEPhI are
     organizing a school on grid and advanced information systems.</p>
```

Paragraph

```
  <p class="topics">
     The main topics of the school are:
```

List

```
     <ul>
       <li>NICA project</li>
       <li>Advanced Information Systems</li>
       <li>Introduction to the GRID technologies</li>
     </ul>
  </p>
</body>
</html>
```

CSS: Reminder

CSS = Cascading Style Sheets

```
<link rel="stylesheet" href="Style.css">
```

```
body { font-family: Arial, MS Sans Serif; background: url(gr1.jpg) repeat }
h1 { background: url(gr3.jpg); color: white; padding: 10px}
p { font-weight: bold; padding-left: 5px }
p.topics { color: #800517}
li { list-style-image: url(b.jpg); margin-top: 1em}
```

History of HTML Language

1991	Official birthday (20 elements)
1995	v.2.0
1996	CSS 1
1996	JavaScript
1997	3.2 and 4.0 (W3C Recommendation)
1999/2000	XHTML
2005	World is asynchronous (AJAX)
2009-…	5.0

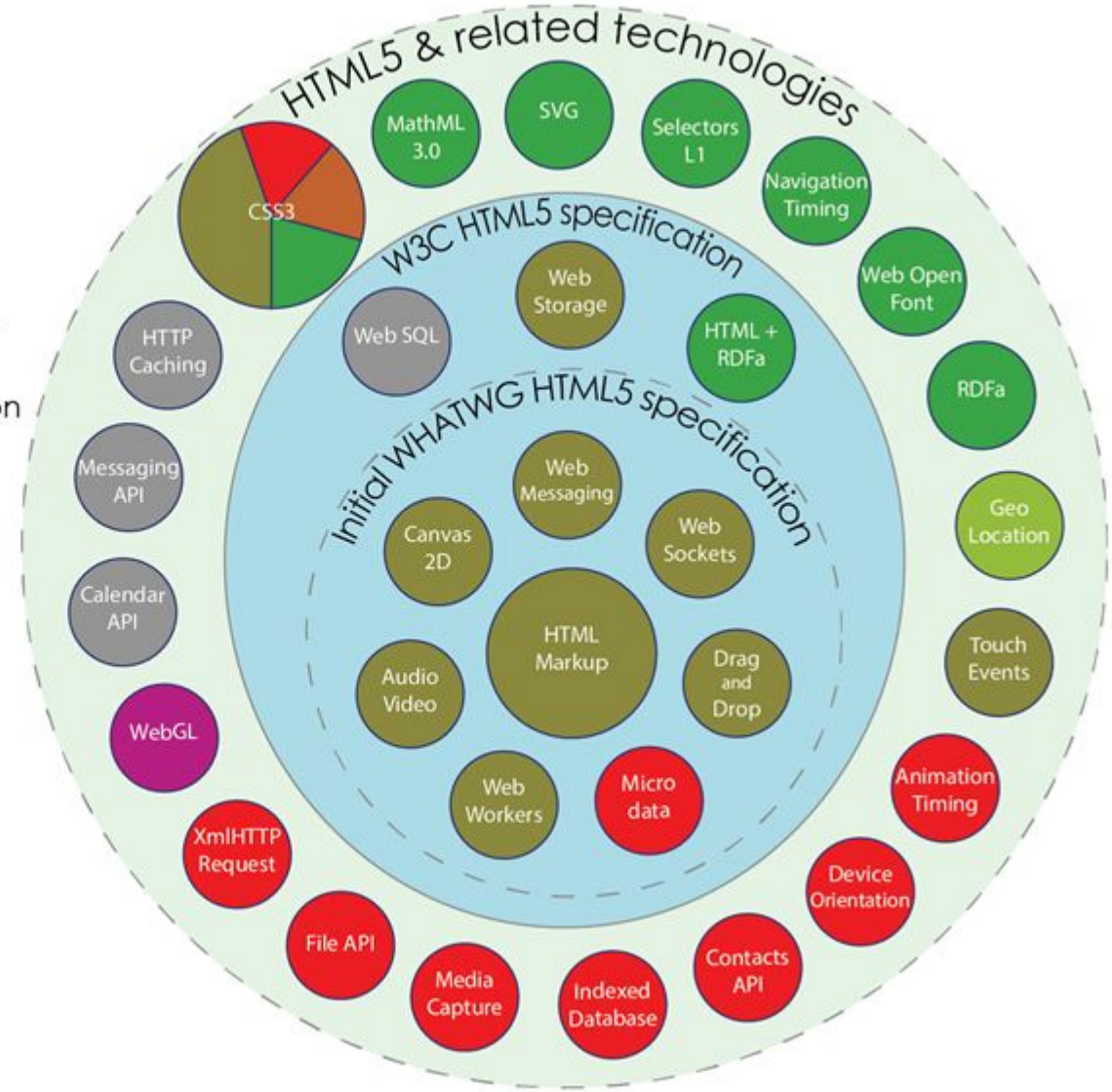
HTML5: Philosophy

- Reduce the need for external plug-ins
- Error handling
- Device independent
- Replace scripting with markup

HTML5 = HTML + CSS + JavaScript

HTML5

- W3C Recommendation
- Proposed Recommendation
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated



HTML5: New Features in a nutshell

- Semantic elements
- New form controls
- Local offline storage
- New JavaScript APIs
- Media: video and audio
- Canvas element for drawing
- User Interface improvements
- SVG and WebGL

New tags added in HTML5

LAYOUT	MEDIA	WEB APPS	RUBY	OTHER
<article>	Defines an article			
<aside>	Defines content aside from the page content			
<audio>	Defines sound content			
<canvas>	Defines graphics			
<command>	Defines a command button			
<datagrid>	References dynamic data in a tree or tabular data form			
<datalist>	Defines a dropdown list			
<details>	Defines details of an element			
<dialog>	Defines a conversation or people talking			
<embed>	Defines external interactive content or plugin			
<figcaption>	Defines the caption of a figure element			
<figure>	Defines a group of media content, and their caption			
<footer>	Defines a footer for a section or page			
<header>	Defines a header for a section or page			
<hgroup>	Defines information about a section in a document			
<keygen>	Defines a generated key in a form			
<mark>	Defines marked text			
<meter>	Defines measurement within a predefined range			
<nav>	Defines navigation links			
<output>	Defines some types of output			
<progress>	Defines progress of a task of any kind			
<rp>	Defines browser substitute for unsupported ruby elements			
<rt>	Defines explanation to ruby annotations			
<ruby>	Defines ruby annotations			
<section>	Defines a section			
<source>	Defines media resources			
<summary>	Defines the header of a "detail" element			
<time>	Defines a date/time			
<video>	Defines a video			
<wbr>	Defines a possible line-break			

http://www.testking.com/techking/infographics/ultimate-html5-cheatsheat/

HTML5: Simplification of code

Markup version

HTML4

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd>
```

The DOCTYPE declaration for HTML5 is very simple:

HTML5

```
<!DOCTYPE html>
```

Metadata

The character encoding (charset) declaration:

HTML4

```
<meta http-equiv="content-type" content="text/html;
charset=UTF-8" />
```

HTML5

```
<meta charset="utf-8"/>
```

Example

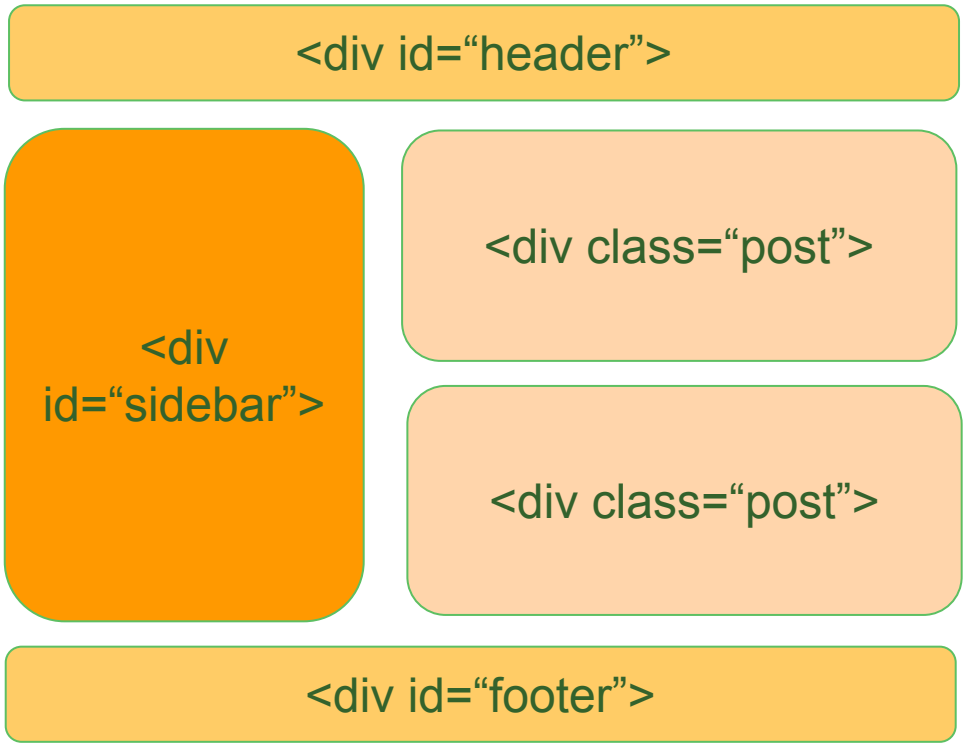
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Title of the document</title>
</head>

<body>
Content of the document.....
</body>

</html>
```

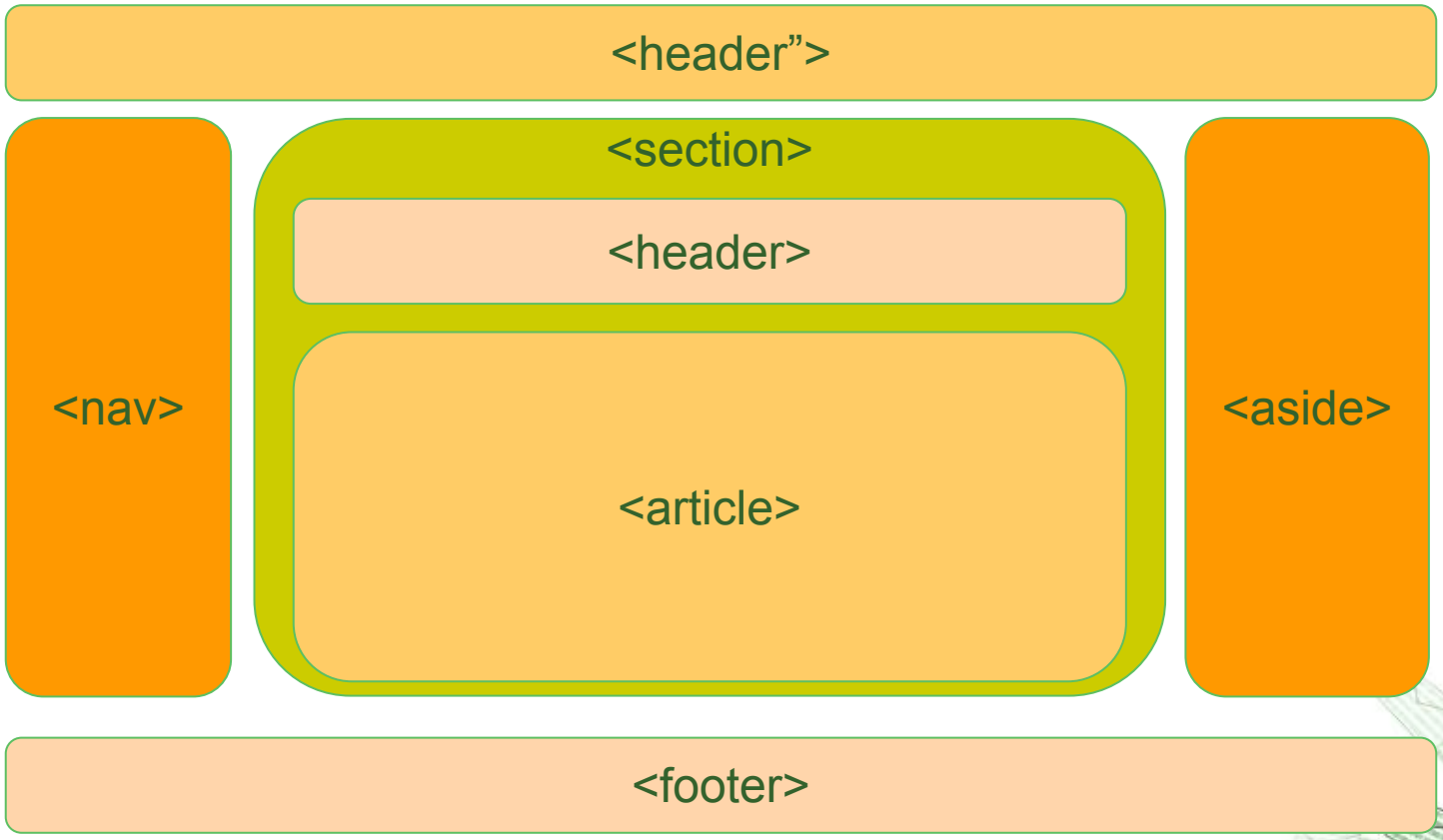
HTML: Structure

- No semantics in layout



HTML5: Semantic markup

Semantic elements = elements with a meaning

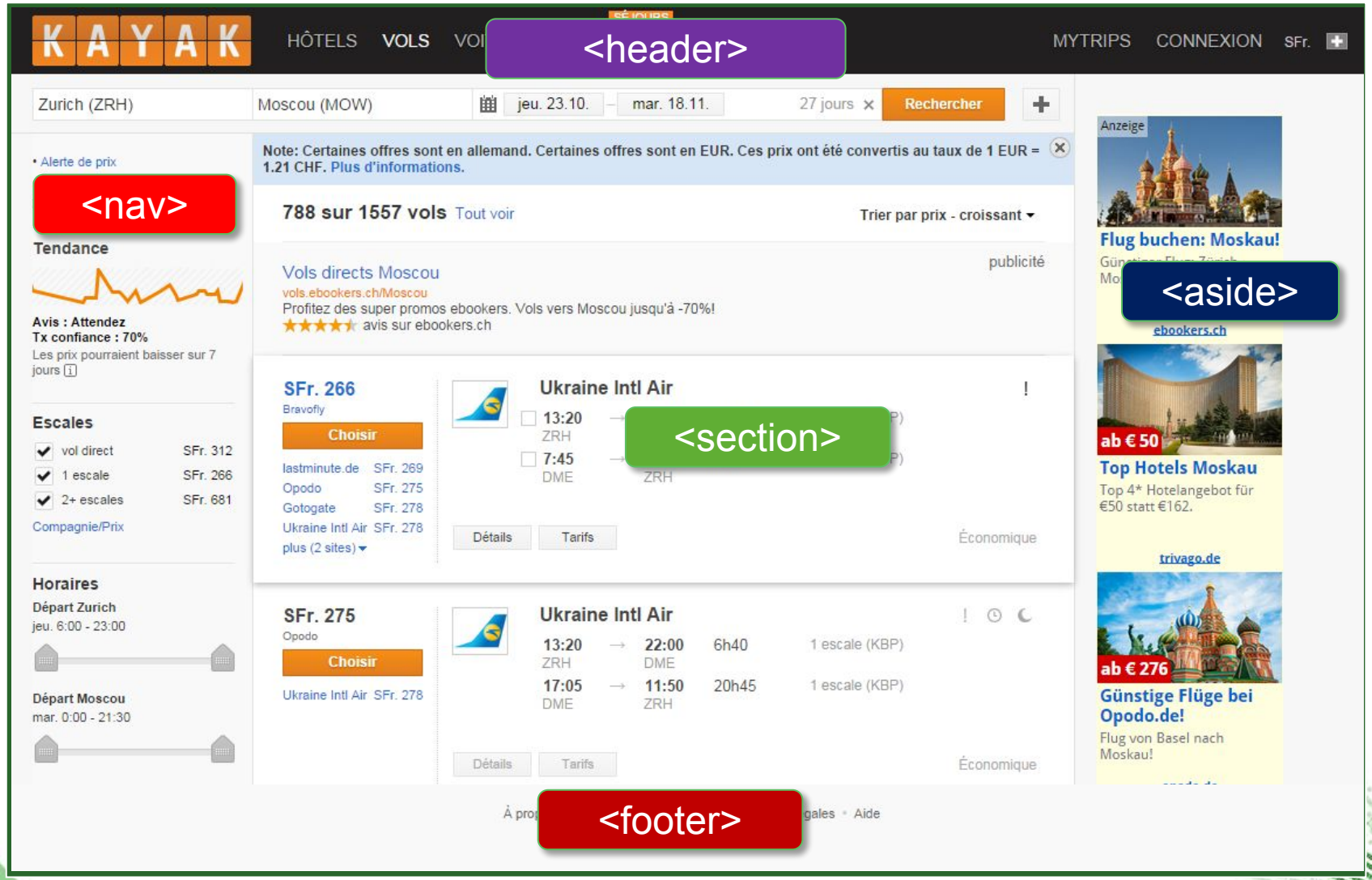


HTML Layout Elements

new semantic elements :

- <header> - Defines a header for a document or a section
- <nav> - Defines a container for navigation links
- <section> - Defines a section in a document
- <article> - Defines an independent self-contained article
- <aside> - Defines content aside from the content (like a sidebar)
- <footer> - Defines a footer for a document or a section
- <details> - Defines additional details
- <summary> - Defines a heading for the <details> element

HTML5: Semantic elements example



HTML Layout Techniques

There are five different ways to create multicolumn layouts. Each way has its advantages and disadvantages:

- HTML tables (not recommended)
- CSS float property
- CSS flexbox
- CSS framework
- CSS grid

Layout examples:

CSS Float Layout

It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work. Disadvantages: Floating elements are tied to the document flow, which may harm the flexibility.

The float Property

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

- left - The element floats to the left of its container
- right - The element floats to the right of its container
- none - The element does not float (will be displayed just where it occurs in the text). This is default
- inherit - The element inherits the float value of its parent

CSS Float Layout

The clear Property

The clear property specifies what elements can float beside the cleared element and on which side.

The clear property can have one of the following values:

- none - Allows floating elements on both sides. This is default
- left - No floating elements allowed on the left side
- right- No floating elements allowed on the right side
- both - No floating elements allowed on either the left or the right side
- inherit - The element inherits the clear value of its parent

CSS Flexbox Layout

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

Flex container:

- flex-direction: row, column, column-reverse, row-reverse
- flex-wrap: wrap, nowrap, wrap-reverse
- justify-content: center, flex-start, flex-end …
- align-items: center, flex-start, flex-end …
- align-content: space-between, space-around …

CSS Flexbox Layout

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

Flex Items

- order
- flex-grow
- flex-shrink
- flex-basis
- flex
- align-self

HTML5: Form elements

<input type="text">

<input type="email">

<input type="URL">

<input type="date">

type={time, month, week}

<input type="number">

<input type="range">

<input type="color">

Examples:

- <form>-element

<form>

.

form elements

.

</form>

Examples:

Type	Description
<input type="text">	Defines a one-line text input field
<input type="radio">	Defines a radio button (for selecting one of many choices)
<input type="submit">	Defines a submit button (for submitting the form)

<input type="text"> defines a one-line input field for **text input**:

Example

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

Example of action attribute:

```
<form action="page.php">
  First name:<br>

  <input type="text" name="firstname" value="Mickey"><br>
  Last name:<br>

<input type="text" name="lastname" value="Mouse"><br><br>

<input type="submit" value="Submit">

</form>
```

Grouping Form Data with <fieldset>

The <fieldset> element is used to group related data in a form.

The <legend> element defines a caption for the <fieldset> element.

```
<form action="/action_page.php">
  <fieldset>
   <legend>Personal information:</legend>
   First name:<br>
     <input type="text" name="firstname" value="Mickey"><br>
      Last name:<br>
     <input type="text" name="lastname" value="Mouse"><br><br>
      <input type="submit" value="Submit">
 </fieldset>
</form>
```

List of form elemets

- <input>
- <select> -> <select name="cars">
  <option value="v1">first</option>
  <option value="v2"> second</option>
  </select>
- Multiple Selections: -> <select name="cars" size="4" **multiple**>
- <textarea> -> <textarea name="message" rows="10" cols="30"> AAA </textarea>
- <button>

HTML5 form elemets

- <datalist> Element

The <datalist> element specifies a list of pre-defined options for an <input> element.

Users will see a drop-down list of the pre-defined options as they input data.

The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.

- ```
  <form action="/action_page.php">
    <input list="browsers">
    <datalist id="browsers">
      <option value="Internet Explorer">
      <option value="Firefox">
      <option value="Chrome">
      <option value="Opera">
      <option value="Safari">
    </datalist>
  </form>
  ```


  ```

HTML5 <output> Element

- The <output> element represents the result of a calculation (like one performed by a script).

```
<form action="/action_page.php“
oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0
  <input type="range"  id="a" name="a" value="50">
  100 +
  <input type="number" id="b" name="b" value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
  <input type="submit">
</form>
```

HTML5 form elements

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

Input types

- Input Type Text
- Input Type Password
- Input Type Submit
- Input Type Reset
- Input Type Radio
- Input Type Checkbox
- Input Type Button

HTML5 Input Types

- color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Input Type Color

- The <input type="color"> is used for input fields that should contain a color.
- Depending on browser support, a color picker can show up in the input field.

- <form>
  Select your favorite color:
  <input type="color" name="favcolor“ value =“#c00000”>
</form>

Input Type Date

- The <input type="date"> is used for input fields that should contain a date.
- Depending on browser support, a date picker can show up in the input field

```
<form> Date:
    <input type="date" name="bday“ value =“inputdate”>
</form>
```

Date type

- You can also use the **min** and **max** attributes to add restrictions to dates:

```
<form>
Enter a date before 1980-01-01:
 <input type="date" name=“maxday" max="1979-12-31"><br>
 Enter a date after 2000-01-01:
 <input type="date" name=“minday" min="2000-01-02"><br>
</form>
```

-

Input Type Datetime-local

- The <input type="datetime-local"> specifies a date and time input field, with no time zone.
- Depending on browser support, a date picker can show up in the input field.

- <form>
  Local date (date and time):
  <input type="datetime-local" name="bdaytime">
  </form>

Input Type Email

- The <input type="email"> is used for input fields that should contain an e-mail address.
- Depending on browser support, the e-mail address can be automatically validated when submitted.

```
<form action="/action_page.php">
E-mail:
<input type="email" name="email" value="enter email">
<input type="submit">
</form>
```

Input Restrictions

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

Input Restrictions

- The following example displays a numeric input field, where you can enter a value from 0 to 100, in steps of 10. The default value is 30:

```
<form action="">
    Quantity:
    <input type="number" name="quantity"
        min="0" max="100" step="10" value="30">
    <input type="submit">
</form>
```

Input Type TIME

- The <input type="time"> allows the user to select a time (no time zone).

```
<form action="">
    Select a time:
    <input type="time" name="usr_time">
    <input type="submit">
</form>
```

Input Type Url

- The <input type="url"> is used for input fields that should contain a URL address.
-

```
<form action="">
    Add your homepage:
     <input type="url" name="homepage">
     <input type="submit">
</form>
```

Input Type Tel

- The <input type="tel"> is used for input fields that should contain a telephone number.

```
<form action="">
    Add your homepage:
     <input type="url" name="homepage">
     <input type="submit">
</form>
```

HTML5 Input Attributes

HTML5 added the following attributes for <input>:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max

The following attributes for <form>:

- autocomplete
- novalidate

- required
- Step

Examples - https://www.w3schools.com/html/html_form_attributes.asp

The autocomplete Attribute

- The autocomplete attribute specifies whether a form or input field should have autocomplete on or off.
- When autocomplete is on, the browser automatically completes the input values based on values that the user has entered before.
- It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.
- The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

```
<form action="" autocomplete="on">
    First name:<input type="text" name="fname"><br>
    Last name: <input type="text" name="lname"><br>
    E-mail: <input type="email" name="email" autocomplete="off"><br>
    <input type="submit">
</form>
```

Other Attributes

- The **novalidate** attribute is a <form> attribute. When present, novalidate specifies that the form data should not be validated when submitted.
- The **autofocus** attribute specifies that the input field should automatically get focus when the page lo
- The **formaction** attribute specifies the URL of a file that will process the input control when the form is submitted.

The formaction attribute overrides the action attribute of the <form> element.

The formaction attribute is used with type="submit" and type="image"

The pattern Attribute

- The pattern attribute specifies a regular expression that the <input> element's value is checked against.
- The pattern attribute works with the following input types: text, search, url, tel, email, and password.

```
<form action="">
Country code: <input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Three letter country code">
<input type="submit">
</form>
```

HTML5: Media

Multimedia on the web is sound, music, videos, movies, and animations.

Multimedia Formats

- Multimedia elements (like audio or video) are stored in media files.
- The most common way to discover the type of a file, is to look at the file extension.
- Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Common Video Formats

There are many video formats out there.
The MP4, WebM, and Ogg formats are supported by HTML.
The MP4 format is recommended by YouTube.

Codecs Challenge

- MPEG-4/H.264: Commonly used video compression format (not royalty-free)
- OGG/Theora: Royalty-free codec not encumbered by any known patents
- WebM: Multimedia format designed to provide a royalty-free, high-quality open video compression format for use with HTML5 video.

Video codecs support in different browsers

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Chrome for Android
MPEG-4/H.264	11	32	37	7.1	24	8	7	4.4	37
Ogg/Theora	11	32	37	7.1	24	8	7	4.4	37
WebM	11	32	37	7.1	24	8	7	4.4	37

http://caniuse.com

Works with an installed WebM codec

No single combination of codecs works in all HTML5 browsers and this is not likely to change in the near future. To make your video watchable across all of these devices and platforms, you're going to need to encode your video more than once.

HTML Video

The HTML <video> element is used to show a video on a web page.

```
<video width=“850" height="600" controls>
  <source src="lecture2.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

HTML Audio

The HTML <audio> element is used to play an audio file on a web page.

```
<audio controls>
  <source src="konil_tolkyny.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML YouTube Videos

To play your video on a web page, do the following:

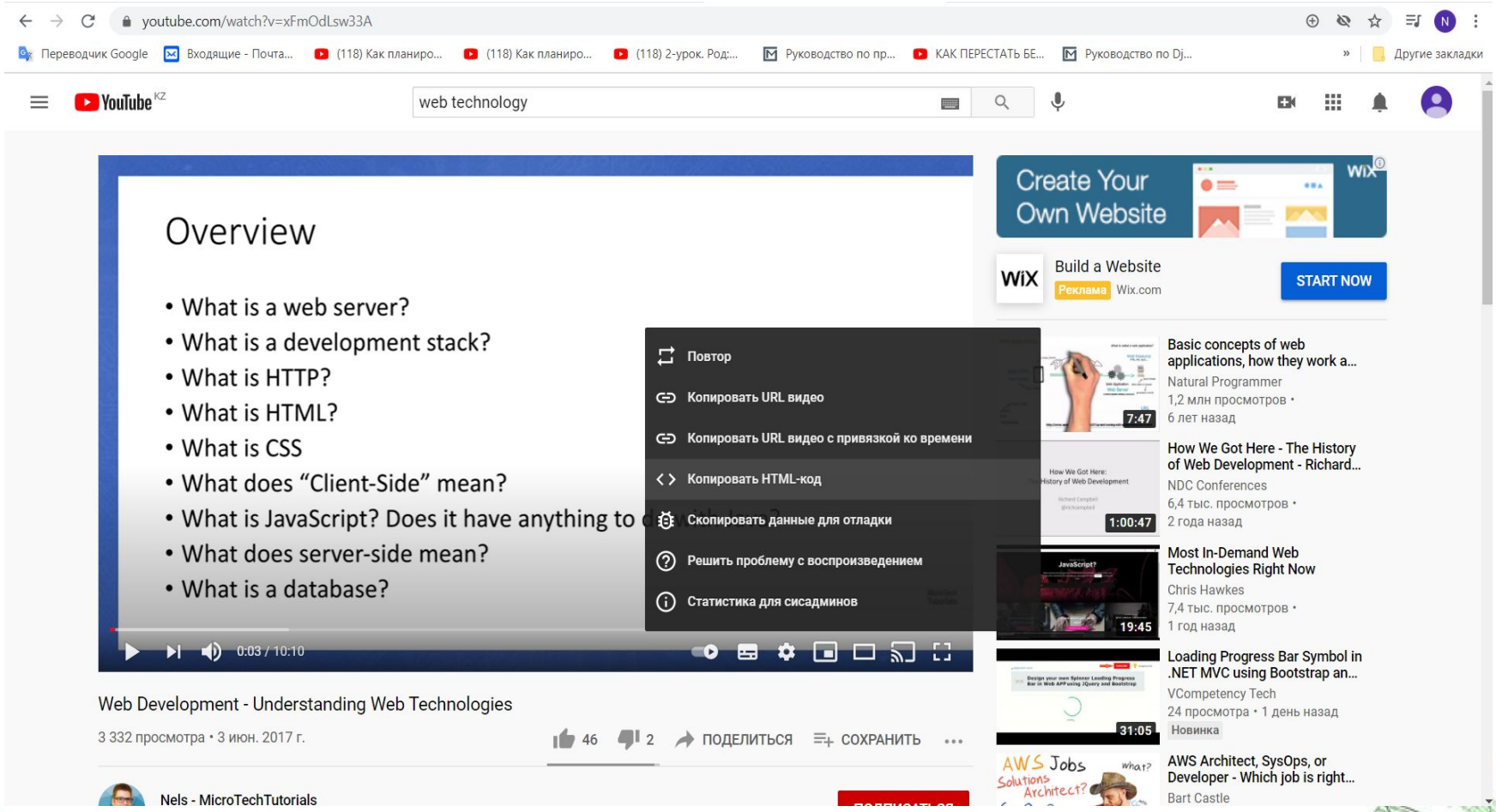
- Upload the video to YouTube
- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player
- Add any other parameters to the URL

```
<iframe width="853" height="480"

src="https://www.youtube.com/embed/gG-geeJxGgM"

frameborder="0" allow="accelerometer; clipboard-write;
encrypted-media; gyroscope; picture-in-picture"
allowfullscreen>
```

HTML YouTube Videos



http://www.youtube.com

HTML Graphics

HTML Canvas

- The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

- <canvas id="myCanvas" width="200" height="100"></canvas>

HTML5: Canvas

Canvas is an API for 2D drawing

<canvas/>

```
<body>
    <canvas></canvas>
</body>
```

Context selector

```
var context = document.querySelector('canvas').getContext('2d');
```

Lines, shapes, path, …

```
context.fillRect(10, 10, 100, 100); //x,y,w,h

context.beginPath();
context.arc(10, 10, 30, 0, Math.PI*2, true); // x,y,r,s,e,d
context.closePath();
```

Pixels

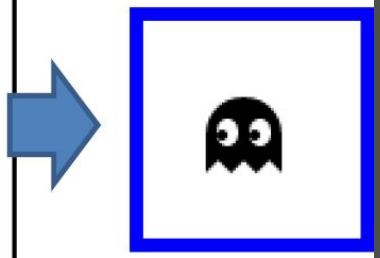
```
var pixels = context.getImageData(0, 0, w,h);
pixels.data[i+0] = ... //R
pixels.data[i+1] = ... //B
...
context.putImageData(pixels, 0, 0);
```

Save image (Data URL)

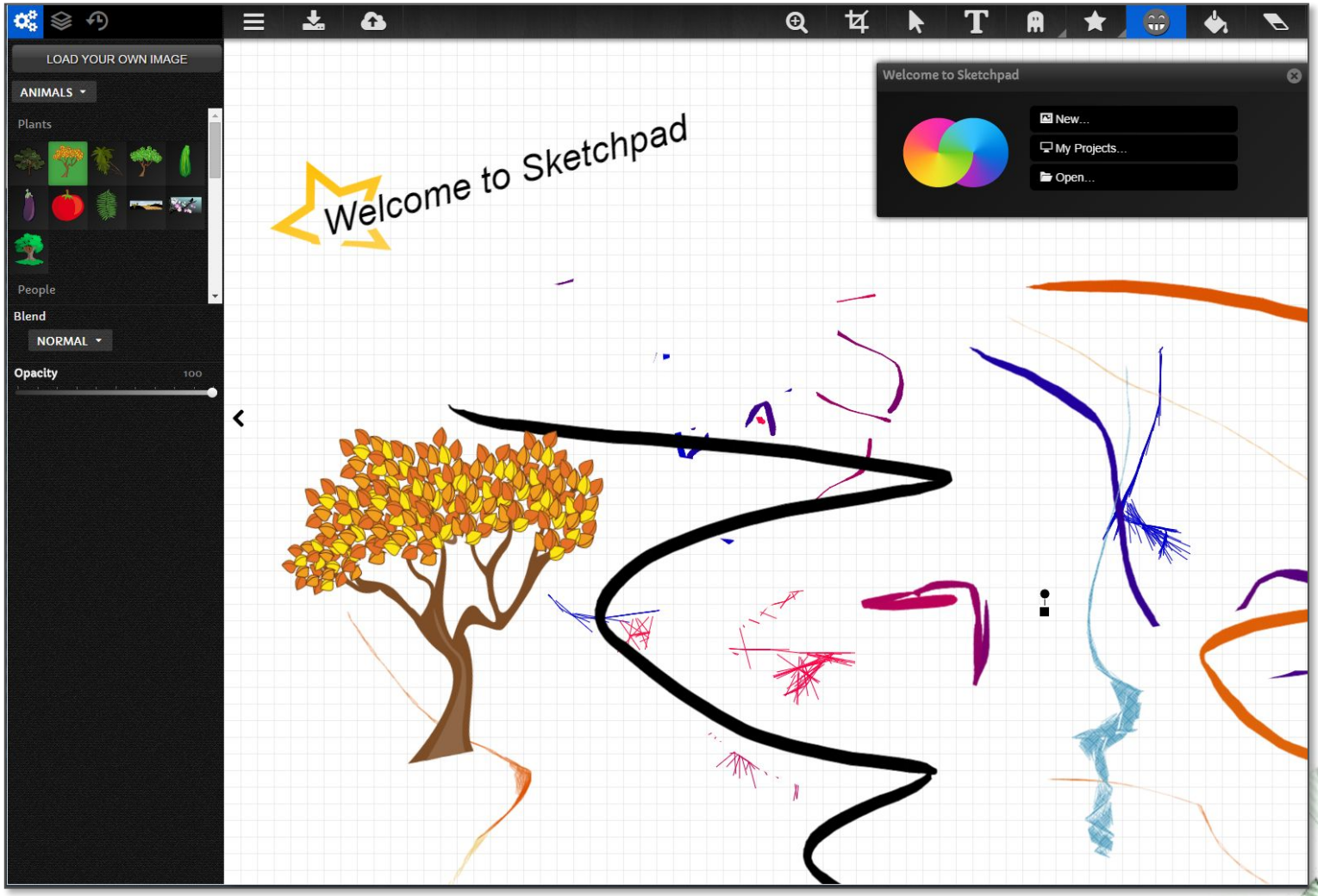
```
window.location = context.canvas.toDataURL('image/png');
// "..."
```

Canvas example

```
var ctx = document.getElementById('canvas').getContext('2d');
ctx.fillStyle = "blue";
ctx.fillRect(55, 55, 90, 90);
ctx.clearRect(60, 60, 80, 80);
ctx.fillStyle = "black";
ctx.beginPath();
ctx.moveTo(83, 116);
ctx.lineTo(83, 102);
ctx.bezierCurveTo(83, 94, 89, 88, 97, 88);
ctx.bezierCurveTo(105, 88, 111, 94, 111, 102);
ctx.lineTo(111, 116);ctx.lineTo(106.333, 111.333);
ctx.lineTo(101.666, 116);ctx.lineTo(97, 111.333);
ctx.lineTo(92.333, 116);ctx.lineTo(87.666, 111.333);
ctx.lineTo(83, 116);
ctx.fill();ctx.fillStyle = "white";
ctx.beginPath();
ctx.moveTo(91, 96);
ctx.bezierCurveTo(88, 96, 87, 99, 87, 101);ctx.bezierCurveTo(87, 103, 88, 106, 91, 106);
ctx.bezierCurveTo(94, 106, 95, 103, 95, 101);ctx.bezierCurveTo(95, 99, 94, 96, 91, 96);
ctx.moveTo(103, 96);
ctx.bezierCurveTo(100, 96, 99, 99, 99, 101);ctx.bezierCurveTo(99, 103, 100, 106, 103, 106);
ctx.bezierCurveTo(106, 106, 107, 103, 107, 101);ctx.bezierCurveTo(107, 99, 106, 96, 103, 96);
ctx.fill();ctx.fillStyle = "black";
ctx.beginPath();
ctx.arc(101, 102, 2, 0, Math.PI * 2, true);
ctx.fill();
ctx.beginPath();
ctx.arc(89, 102, 2, 0, Math.PI * 2, true);
ctx.fill();
```



… a more advanced example



https://sketch.io/sketchpad/

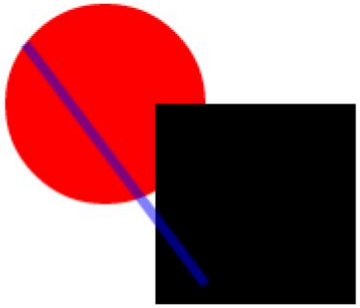
HTML5: Scalable Vector Graphics (SVG)

SVG is an XML-based format for describing 2D vector graphics

SVG in HTML5:

```
<svg xmlns="http://www.w3.org/2000/svg">
...
</svg>
```

```
<svg id="svgelem" height="200" xmlns="http://www.w3.org/2000/svg">
    <circle id="redcircle" cx="50" cy="50" r="50" fill="red" />
    <rect x="75" y="50" id="bluerect" width="100" height="100" fill="black" />
    <line x1="10" y1="20" x2="100" y2="140"
        style="stroke:blue;stroke-width:5" opacity="0.5" />
</svg>
```



HTML5 SVG

What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

The HTML <svg> Element

- The HTML <svg> element is a container for SVG graphics.
- SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

```
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
Sorry, your browser does not support inline SVG.
</svg>
```

Differences Between SVG and Canvas

- SVG is a language for describing 2D graphics in XML.
- Canvas draws 2D graphics, on the fly (with a JavaScript).
- SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
- In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

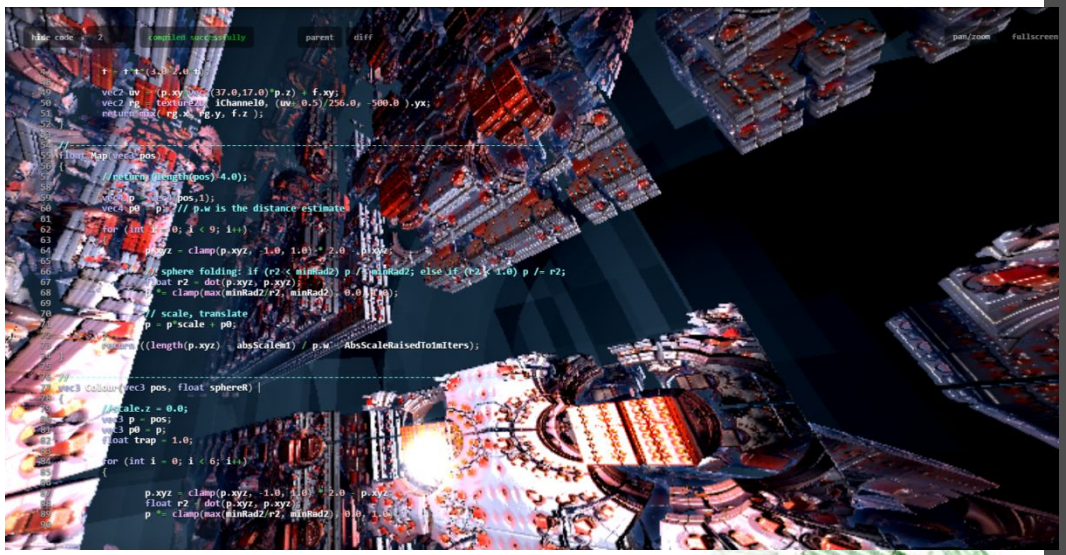
Canvas or SVG?

Canvas	SVG
Pixel-based	Object model-based (XML)
Single HTML container	Multiple elements (part of the DOM)
Created and modified via API	Created with markup and modified by API or CSS
Interaction manually programmed from mouse coordinates	Interaction is object based on the tree of elements (DOM)
Zooming	Scaling

Canvas + WebGL

- WebGL is a JavaScript API for interactive 2D/3D graphics
- Based on the OpenGL ES standard
- Supported by most modern browsers without plug-ins

Compatibility



http://glslsandbox.com/

HTML5 or Flash?

HTML5	Flash
Not (yet) fully supported by all browsers	Support for wide variety of browsers (via plugin)
HTML, JavaScript	ActionScript
Semantic, Indexed	No semantic, not indexed
Access to source code	Limited access to source code
Supported on Apple mobile devices	Not supported on Apple mobile devices
Basic interactivity	Rich interactivity

https://en.wikipedia.org/wiki/Comparison_of_HTML5_and_Flash

Thank you for your attention!