Hi! My name is Nika Neider and today I will tell you about Css & Shaders.

Slide 1

First I speak you about standard filters. Then I will go to the Shaders.

Filter is a CSS property that **allows you to apply** graphic effects to elements on your web page.

Slide 2

In Twenty oh eight, Robert O'Callaghan from Mozilla first described the idea of using SVG filters **through the application of** CSS to DOM's elements in his blog. 4 years later, on 25 of October Twenty oh twelv, the W3C released the first version of the CSS filter specification.

Slide 3

Since then, there have been several specifications updates.

On 17 of May Twenty oh eighteen was the last time the specification was updated.

On 9 of November Twenty oh seventeen was created a filter module number 2 that refers to the backdrop filter.

Slide 4

Let's see how filters work. When browser loads the page, **it performes the following** steps:

- Creates layout of page; - Applies Styles; - And render the page;

Filters **are applied** after all these steps. But before the rendered page gets the screen. They **process** the page **pixel-by-pixel applying the specified effects**. These effects are **drawn over** the original page. **Therefore**, you can use multiple filters. The more filters, the more time **is required for** the browser to render the page.

Slide 5

The filter **can be applied** to a picture / block / element. You can also apply multiple filters **at once**.

The syntax of the filters is simple - we add the filter property to css with the following values.

Slide 6

filter: url(); The url() helps you **to create your own** filters using the SVG filter element and **refer to them from** CSS. Each filter has its own id.

Slide 6

filter: blur(*px*, *em*, *rem*); Applies a blur effect to the image.

A larger value will create more blur. The edges **are blurred like** a border-shadow.

Slide 7

filter: brightness(%); Adjusts the brightness of the image. Default value is 100%

0 will make the image completely black.

Slide 8

Value over 100% will provide brighter results.

Slide 9

filter: contrast(%); Adjusts the contrast of the image. Default value is 100%

0 will make the image completely black.

Slide 10

Value over 100% will give more contrast of the image.

Slide 11

filter: drop-shadow; Creates a shadow **similar to** the box-shadow property, but only the filter **supports hardware acceleration**.

Adds a shadow **along the outline of the shape.**

X, y - Specifies a pixel value for the horizontal shadow, vertical shadow.

blur - The blur distance, default = 0. c - color of the shadow.

spread - Optional. The size of shadow. Some browsers, do not support spread;

Slide 12

filter: grayscale(%); Converts the image to grayscale. Default value is 0%

100% will make the image completely gray.

Slide 13

filter: hue-rotate(deg); Applies a hue rotation on the image. The value **defines** the number **of degrees around** the color circle **the image samples will be adjusted**. 0deg is default, and represents the original image.

Slide 14

filter: invert(%); Inverts the samples in the image. Default value is 100%

100% will make the image completely inverted.

Slide 15

filter: opacity(%); Sets the opacity level for the image. Default value is 100%

0% or 0 is completely transparent.

Slide 16

filter: saturate(%); Saturates the image. Default value is 100%

0% (0) will make the image completely un-saturated.

Slide 17

Values over 100% provides super-saturated results.

Slide 18

filter: sepia(%); Converts the image to sepia. Default value is 0%

100% will make the image completely sepia.

Slide 19

To use multiple filters, **separate each filter with a space**.

Slide 20

The backdrop-filter CSS property lets you apply graphical effects such as blurring or color shifting to the area behind an element.

The property of the backdrop-filter is the same as for the filter property.

However, not all browsers support this property.

Slide 21

An example of blur (10%). As you can see the background behind the block is blurred.

Slide 22

From this slide you can see that the history of shaders was short.

They were a part of the first Filter specification.

26 November 2013 custom filters were removed. Chrome has stopped support them in 2014.

However, the Shaders are a part of the story and we will study it.

Slide 21

Shaders were the small programs **that process the vertices** of 3D geometry and the color of pixels.

There was the 2 types of shaders which was **available** in custom filters.

Vertex shaders — were about geometry and deforming and displacing objects in space.

Fragment shaders — were about painting the surfaces of objects and modifying how pixels look.

Slide 22

1. An html or svg element is placed on a mesh. By default the mesh will have only two triangles.
2. The mesh **is then processed with** a vertex shader, which **distorts and deforms** the element in 3D space.
3. Then called a fragment shader, which processes the pixels inside the mesh.
4. The **rasterized output is displayed** on the screen **or go to** the next filter **primitive**.

Slide 23

The mesh is based on **triangulated geometry**. The default vertex mesh is a rectangle with two triangles. It can be divided into more triangles to create a more flexible grid.

By moving the vertices around, the triangles **inside are deformed or displaced** in different ways. There are 2 types of **grid geometry** that can be created through css **using the geometry descriptor**.

1. **Detached** — Mesh has been divided into individual components.
2. **Attached** — If one triangle is deformed, then entire mesh is also deformed.

Slide 24

The syntax is the same as in filter base. We write custom atribute. Because that the shaders are also called custom filters. It contains 4 parameters. V_Shader and F_Shader **separated by a space and** may contain url () or mix ().

Slide 25

Mix() can take input parameters of 3 general kinds.

- uniform parameters — have a single value for all vertices and pixels of the mesh
- attribute parameters — have their own values for every vertex of the mesh
- varying parameters — are passed from vertex shader to fragment shader.