



HTML5, CSS, and JavaScript: Why They Are Important to You and How They Fit Together

Gopinath Taget – Autodesk Inc.

SD6839 In this class we will talk about the client-side web technologies HTML5, CSS (Cascading Style Sheets), and the JavaScript API (application programming interface), and we will look at how they are relevant to Autodesk, Inc., technologies such as the Autodesk 3D viewer technology. You will discover how these technologies fit with each other, what they do well, and where they are best avoided. We will also examine examples that use these technologies together with Autodesk technologies, and we will discuss where you should start if you are new to these technologies. Finally, the class will talk about enabling frameworks built on top of the JavaScript API, such as jQuery, Angular, and ThreeJS, which make using these technologies more delightful and powerful.

Learning Objectives

At the end of this class, you will be able to:

- Discover where to start learning HTML5, CSS, and the Javascript API
- Discover the pros and cons of HTML5, CSS, and Javascript
- Discover how HTML5, CSS, and the JavaScript API can be used with each other
- Discover jQuery, Angular, and ThreeJS

About the Speaker

Gopinath Taget is a member of the Autodesk, Inc., Consulting Team. He has more than 16 years of experience developing and supporting AutoCAD software APIs, including ObjectARX technology, Microsoft .NET, Microsoft VBA, and AutoLISP programming language. He is also an expert on Revit APIs. Gopinath also has several years of experience in software development on other CAD platforms, including MicroStation software, SolidWorks software, and CATIA software, mainly using C++ and technologies such as Microsoft Foundation Class (MFC) and component object model (COM). Gopinath was also involved in the development of web-based applications for MapGuide software and AutoCAD Map 3D software. Gopinath has master's degrees in civil engineering and software systems.

gopinath.taget@gmail.com

Introduction – The Web browser as an Application Platform

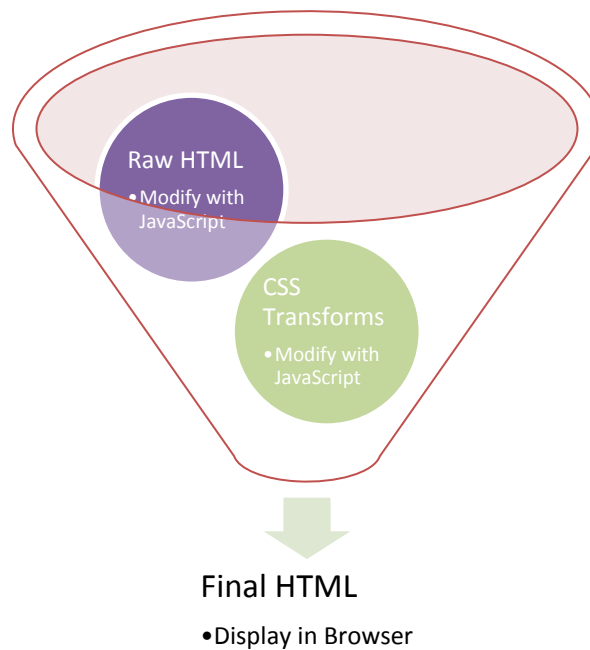
We are very used to thinking the web browser as just a portal to applications that run on the “internet”; more specifically on a web server somewhere on the internet. While this is not incorrect and was pretty accurate in the early 1990s at the beginning of the internet revolution, the web browser is not just a dumb viewer of a web server application any more. It is a lot more than that.

Over the last couple of decades the humble web browser has morphed into a rich application host platform with comprehensive APIs covering different platform services including UI, 2D/3D graphics, networking, Video/Audio and I/O. This has been helped in no less measure by aggressive competition between different browser vendors that brought on a browser war (a good war though) to add more features and fix bugs more quickly. Currently among the most popular browsers are [Chrome](#), [Firefox](#), [Internet Explorer](#) and [Safari](#). There are other browsers out there but those mentioned above are commonly available on multiple platforms.

Unfortunately this also brought on a certain level of fragmentation in the features and APIs supported by the browsers. This fragmentation is compounded by the proliferation of multiple hardware and Operating System configurations that the browser must run on including the hand held devices such as apple devices (iPhone, iPad), Android devices, net books, Unix desktops, Windows laptops and desktops and Mac laptops and desktops to name a few.

This fragmentation has kept organizations like [W3C](#), a web standards organization, pretty busy in trying to define and maintain a platform agnostic set of standards for web technologies that works well with all these platforms. Considering the level of fragmentation, they have done an admirable job in keeping the standards consistent.

The most important technologies for web browsers are [HTML](#), [CSS](#) (Cascading Style Sheets) and [JavaScript](#). HTML together with CSS defines the language that the browsers understand. You need to use this to tell the browsers what to do (mostly it's about what to show in the browser and how to show it). W3C manages the standards for both the technologies. JavaScript is a programming language that allows you to define and manipulate HTML and CSS dynamically, typically, in response to events. There is a lot more to HTML, CSS and JavaScript but this is basically how they work with each other.



You will find a lot of sites on the internet that teach these web technologies but the single most useful site for tutorials and reference documentation for these technologies is [w3schools](https://www.w3schools.com).

HTML and CSS have been evolving over the years mainly in response to the increasing need for more feature rich and responsive websites. The latest versions of HTML and CSS are denoted as HTML5 and CSS3 respectively. Unless otherwise mentioned, the rest of this handout refers to these versions when talking about HTML and CSS.

Hyper Text Markup Language

HTML is the language of the browser. It is a clear text, human readable language. The browser knows what to display by reading the “instructions” in the HTML text. The instructions are specified as “tags” (similar to XML tags) nested in a hierarchy. Here is a very simple example:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

In the case above, `html`, `body`, `h1` and `p` are tags that are interpreted by browsers and displayed. For a comprehensive list of HTML tags and what they mean, please refer to [this web page](#).


In general the HTML tags can be [categorized as follows](#) based on their function in a web page:

- Basic Tags
- Formatting
- Forms and Input
- Frames
- Images
- Audio / Video
- Links
- Lists
- Tables
- Styles and Semantics
- Meta Info
- Programming

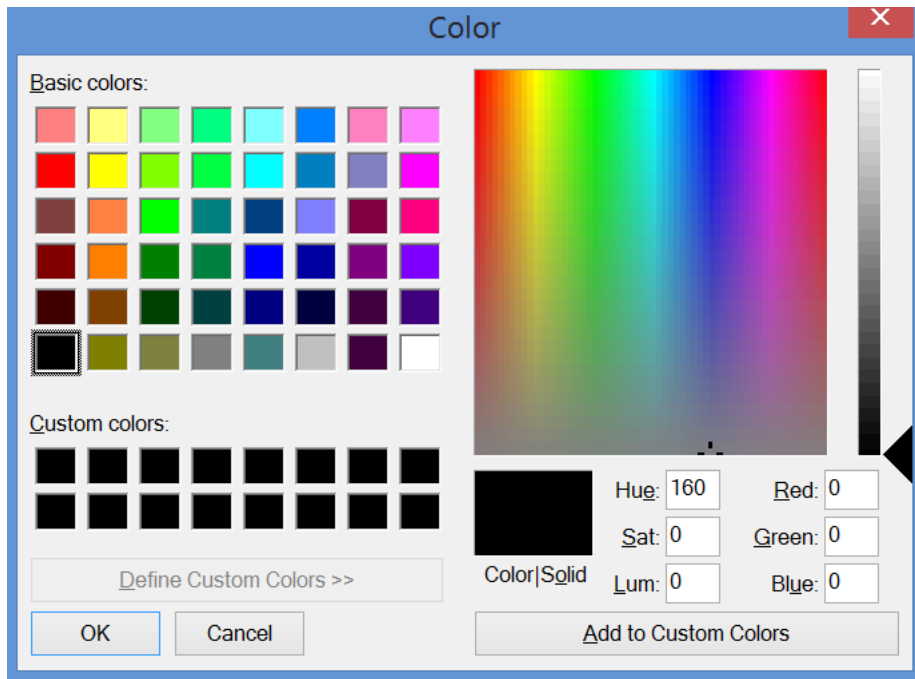
One point to note is that while most HTML tags would be supported by all browsers, the way they are displayed by browsers is not standardized. The browsers could (and often do) differ in the way the tags are interpreted and displayed. In fact, some types of tags may not be supported at all by some browsers. One common approach to verify if a browser supports a specific tag is to use the [caniuse](#) website. For instance, an `input` tag of type “color” would be defined like this:

```
<input type="color" name="favcolor">
```

It would be interpreted and displayed like this in Chrome:

Select your favorite color: 

And displayed the color like this, on click:



In IE, it would not be interpreted as color at all and would just be shown as a text box:

Select your favorite color:

This is the fallback behavior for all input types that the browser does not understand.

For elements that are new to HTML5, please refer to [this webpage](#). Among these, you might find a couple of tags particularly interesting:

Tag	Description
<canvas>	Defines graphic drawing using JavaScript
<svg>	Defines graphic drawing using SVG

The [canvas tag](#) is particularly useful for all kinds of graphics applications including 2D and 3D graphics, animations and games. You will find a number of examples of the usage of the canvas tag with the JavaScript API on the web.

SVG stands for Scalable Vector Graphics and is the tag that will allow you to draw vector graphics as opposed to raster graphics.

One point to note though is that even though Canvas and SVG essentially support only 2D graphics natively, we can use [math techniques](#) to simulate 3D on them.

Cascading Style Sheets

Cascading Style Sheets are the makeup kit for HTML. The same piece of web page can be made to look completely different using different CSS styling instructions.

A simple example of what CSS instructions look like:

```
body {  
    background-color: #d0e4fe;  
}  
  
h1 {  
    color: orange;  
    text-align: center;  
}  
  
p {  
    font-family: "Times New Roman";  
    font-size: 20px;  
}
```

In the example above, we set the different visual properties of the body, h1 (header 1) and p (paragraph) tag elements of a HTML page. Here is an example of the application of the CSS on HTML:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <style>  
      body {  
        color: red;  
      }  
  
      h1 {  
        color: #00ff00;  
      }  
  
      p.ex {  
        color: rgb(0,0,255);  
      }  
    </style>  
  </head>  
  <body>  
  
    <h1>This is heading 1</h1>
```

```
<p>This is an ordinary paragraph. Notice that this text is red. The default text-  
color for a page is defined in the body selector.</p>  
<p class="ex">This is a paragraph with class="ex". This text is blue.</p>  
</body>  
</html>
```

And here is what it looks like:

This is heading 1

This is an ordinary paragraph. Notice that this text is red. The default text-color for a page is defined in the body selector.

This is a paragraph with class="ex". This text is blue.

You will find comprehensive information on the different kinds of CSS instructions and more [here](#).

One of the common uses of CSS is to make a web page work well both on desktop as well as mobile platforms. [This website](#) discusses some of those techniques. Some other the advantages of using CSS:

- Consistency: You can have a single CSS file applying consistent style across multiple web pages
- Small size: Using CSS reduces the size of your web site because you do not have to repeat styles for each web page.
- Browser compatibility: The variety of browsers available makes creation of browser neutral web sites hard. CSS provides browser instructions that make it easy to maintain consistency across browsers.

The CSS instructions are categorized into the following property groups:

CSS Property Groups

- | | | |
|--|--|--|
| • Color | • Table | • Paged Media |
| • Background and Borders | • Lists and Counters | • Generated Content |
| • Basic Box | • Animation | • Filter Effects |
| • Flexible Box | • Transform | • Image/Replaced Content |
| • Text | • Transition | • Masking |
| • Text Decoration | • Basic User Interface | • Speech |
| • Fonts | • Multi-column | • Marquee |
| • Writing Modes | | |

JavaScript

JavaScript is a first class programming language with deep programming semantics and advanced features. It is designed from ground up to be used in a browser. It has syntax similar to C++ and Java which gives a head start for many programmers learning it.

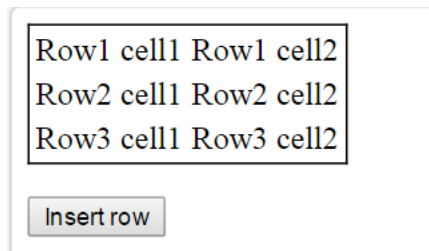
The most common usage of JavaScript is to manipulate the HTML tag elements and CSS property elements. In JavaScript parlance, this is termed as DOM (Document Object Model) manipulation. DOM manipulation typically happens on an event like clicking a button or clicking an image. Here is an example of manipulating a table in HTML on button click:

```
<!DOCTYPE html>
<html>
<head>
<script>
function changeContent(id, row, cell, content) {
    var x = document.getElementById(id).rows[row].cells;
    x[cell].innerHTML = content;
}
</script>
</head>

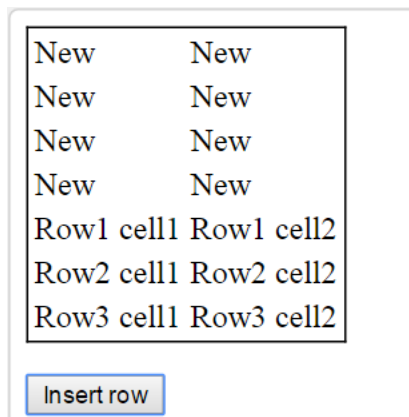
<body>
<table id="myTable" border="1">
<tr>
<td>Row1 cell1</td>
<td>Row1 cell2</td>
</tr>
<tr>
<td>Row2 cell1</td>
<td>Row2 cell2</td>
</tr>
<tr>
<td>Row3 cell1</td>
<td>Row3 cell2</td>
</tr>
</table>
<p>
<input type="button" onclick="changeContent('myTable', 0, 0, 'Hello')"
value="Change content">
</p>

</body>
</html>
```


Here is the page before the button click:



And here it is after a few clicks:



Another example of manipulating an image on click:

```
<!DOCTYPE html>
<html>
<body>

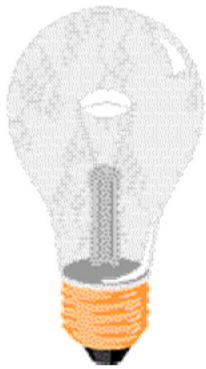


<p>Click the light bulb to turn on/off the light.</p>

<script>
function changeImage() {
    var image = document.getElementById('myImage');
    if (image.src.match("bulbon")) {
        image.src = "pic_bulboff.gif";
    } else {
```

```
        image.src = "pic_bulbon.gif";  
    }  
}  
</script>  
  
</body>  
</html>
```

Here is the image before the click:



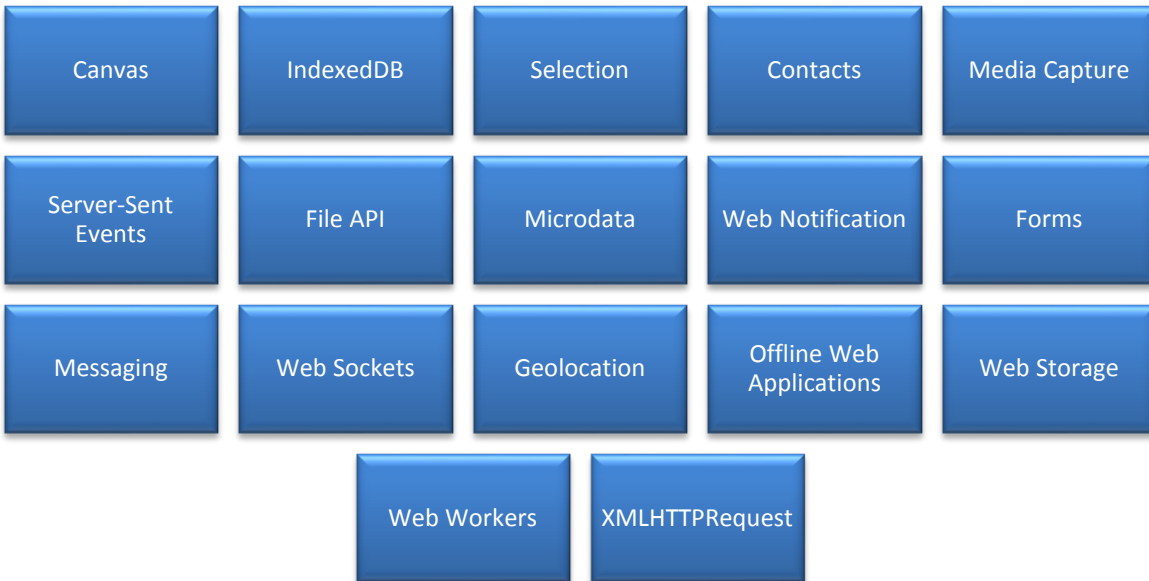
Click the light bulb to turn on/off the light.

And the image after:



Click the light bulb to turn on/off the light.

Javascript however is not just for DOM manipulation. The browser provides APIs for several other functional areas:



I will discuss many of these APIs and more in my AU class. See you there!