

arnaud.nauwynck@gmail.com

# Introduction to HTML / JavaScript / DOM

This document:

<http://arnaud-nauwynck.github.io/docs/Intro-Html-JS-DOM.pdf>

# Agenda

- Introduction to Client-Side Web Technologies
  - HTML = Elements / Custom Elements
  - CSS
  - JavaScript
- The DOM ... how JavaScript interact
- JavaScript framework(s)
  - AngularJS, binding principles

I will use Google before asking dumb questions. I will use Google before asking dumb questions.

**Google**  

All Images Videos News Books More ▾ Search tools

About 21,500,000,000 results (0.70 seconds)

**21 Billions...**

HyperText Markup Language (**HTML**) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS), and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.

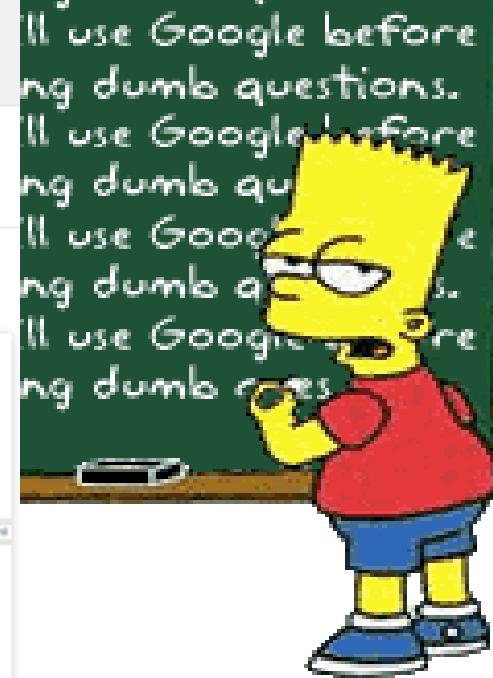
**HTML - Wikipedia**  
<https://en.wikipedia.org/wiki/HTML>

The HTML code above produces the following below

```
<!DOCTYPE html>
<html>
<!--
Created 14-10-2014
-->
<head>
<title>Sample</title>
</head>
<body>
<p>Sample text</p>
</body>
</html>
```

Sample text

[About this result](#) • [Feedback](#)



### People also ask

What is HTML language?

When HTML was invented?

What is the most recent version of HTML?

Rank#1 : w3schools (not Wikipedia?)

**HTML Tutorial - W3Schools**

[www.w3schools.com/html/](http://www.w3schools.com/html/) ▾

The **HTML** Certificate documents your knowledge of **HTML**. The CSS Certificate documents your



## HTML5 Tutorial

[HTML HOME](#)[HTML Introduction](#)[HTML Editors](#)[HTML Basic](#)[HTML Elements](#)[HTML Attributes](#)[HTML Headings](#)[HTML Paragraphs](#)[HTML Styles](#)[HTML Formatting](#)[HTML Quotations](#)[HTML Computercode](#)[HTML Comments](#)[HTML Colors](#)[HTML CSS](#)[HTML Links](#)[HTML Images](#)[HTML Tables](#)[HTML Lists](#)[HTML Blocks](#)[HTML Classes](#)[HTML Iframes](#)[HTML JavaScript](#)[HTML Head](#)[HTML Layout](#)[HTML Responsive](#)[HTML Entities](#)[HTML Symbols](#)[HTML Charset](#)

W3.CSS Templates



Templates

by

W3.CSS



COLOR PICKER



# HTML Introduction

[◀ Previous](#)[Next ▶](#)

## What is HTML?

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

## A Simple HTML Document

### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
```

# HyperText Markup Language

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)



**WIKIPEDIA**  
The Free Encyclopedia

[Article](#) [Talk](#)

[Read](#) [View source](#) [View history](#)

Search Wikipedia



## HTML

From Wikipedia, the free encyclopedia

(Redirected from [Html](#))

*".htm" and ".html" redirect here. For other uses, see [HTM \(disambiguation\)](#).*

*For the use of HTML on Wikipedia, see [Help:HTML in wikitext](#).*

**HyperText Markup Language (HTML)** is the standard [markup language](#) for creating [web pages](#) and [web applications](#). With [Cascading Style Sheets](#) (CSS), and [JavaScript](#), it forms a triad of cornerstone technologies for the [World Wide Web](#).<sup>[1]</sup> [Web browsers](#) receive HTML documents from a [webserver](#) or from local storage and render them into multimedia web pages. HTML describes the structure of a web page [semantically](#) and originally included cues for the appearance of the document.

[HTML elements](#) are the building blocks of HTML pages. With HTML constructs, [images](#) and other objects, such as [interactive forms](#) may be embedded into the rendered page. It provides a means to create [structured documents](#) by denoting structural [semantics](#) for text such as headings, paragraphs, lists, [links](#), quotes and other items. HTML elements are delineated by [tags](#), written using [angle brackets](#). Tags such as `<img />` and `<input />` introduce content into the page directly. Others such as `<p>...</p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a [scripting language](#) such as [JavaScript](#) which affect the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The [World Wide Web Consortium](#) (W3C), maintainer of both the HTML and the CSS standards, has encouraged the use of CSS over

**HTML**  
(HyperText Markup Language)

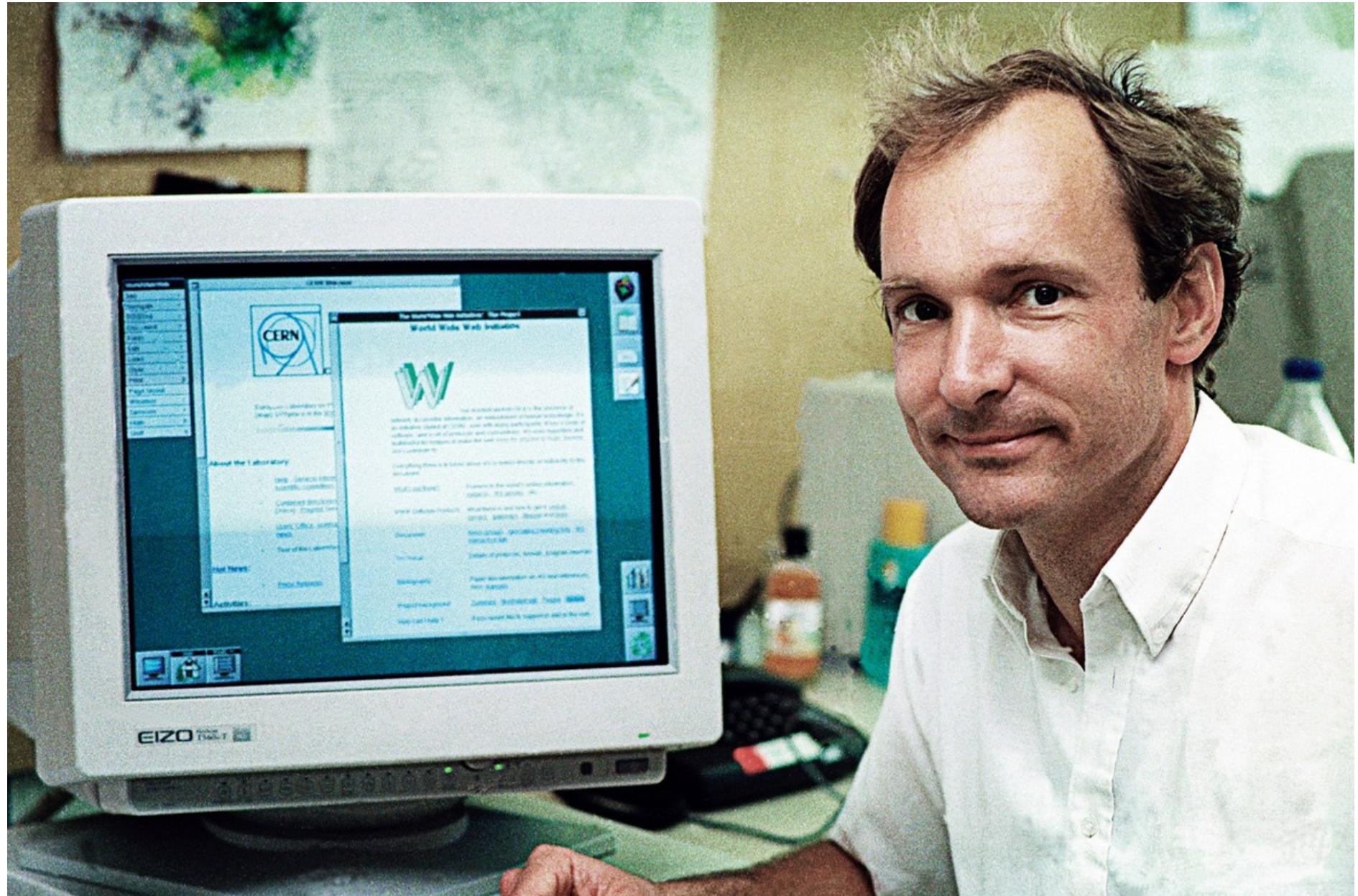
```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
<head>
<title>sample</title>
</head>
<body>
<p>Voluptatem accusantium  
totam rem aperiam.</p>
</body>
</html>
```

HTML

Filename extension	.html .htm text/html
Internet media type	
Type code	TEXT
Developed by	W3C & WHATWG
Initial release	1993; 23 years ago
Latest release	5.0 / 5.1 (working)

# 1989 : HTTP + HTML

## Tim Berners-Lee





HyperText Markup Language (HTML)  
is the standard markup language  
for creating web pages and web applications.

### Example

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

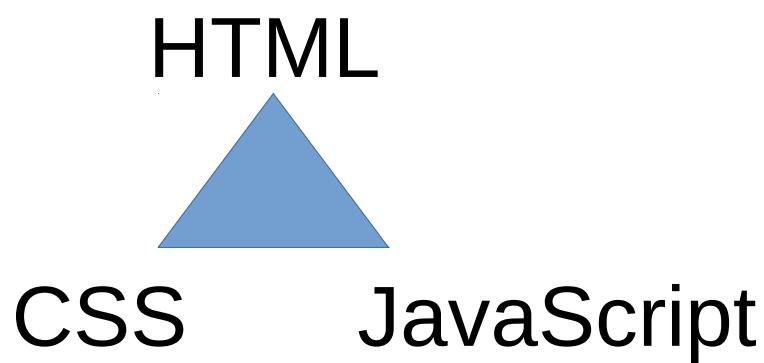


**WIKIPEDIA**  
The Free Encyclopedia

With Cascading Style Sheets (**CSS**), and **JavaScript**,  
it forms a triad of cornerstone technologies  
for the World Wide Web

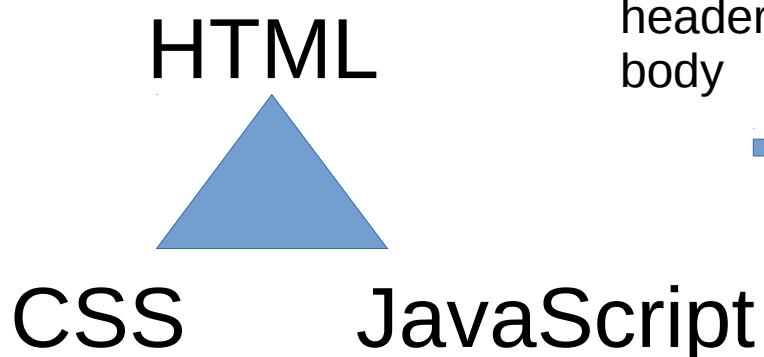
**www**

**W3C Technologies =**



# Client(Html+CSS+JS) – Protocol(Http) - Server(\*)

Client-Side

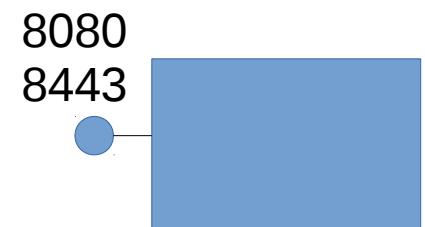


Server-Side  
expose IP<TCP<HTTP port

HTTP Request (GET,POST,..) URL  
headers  
body

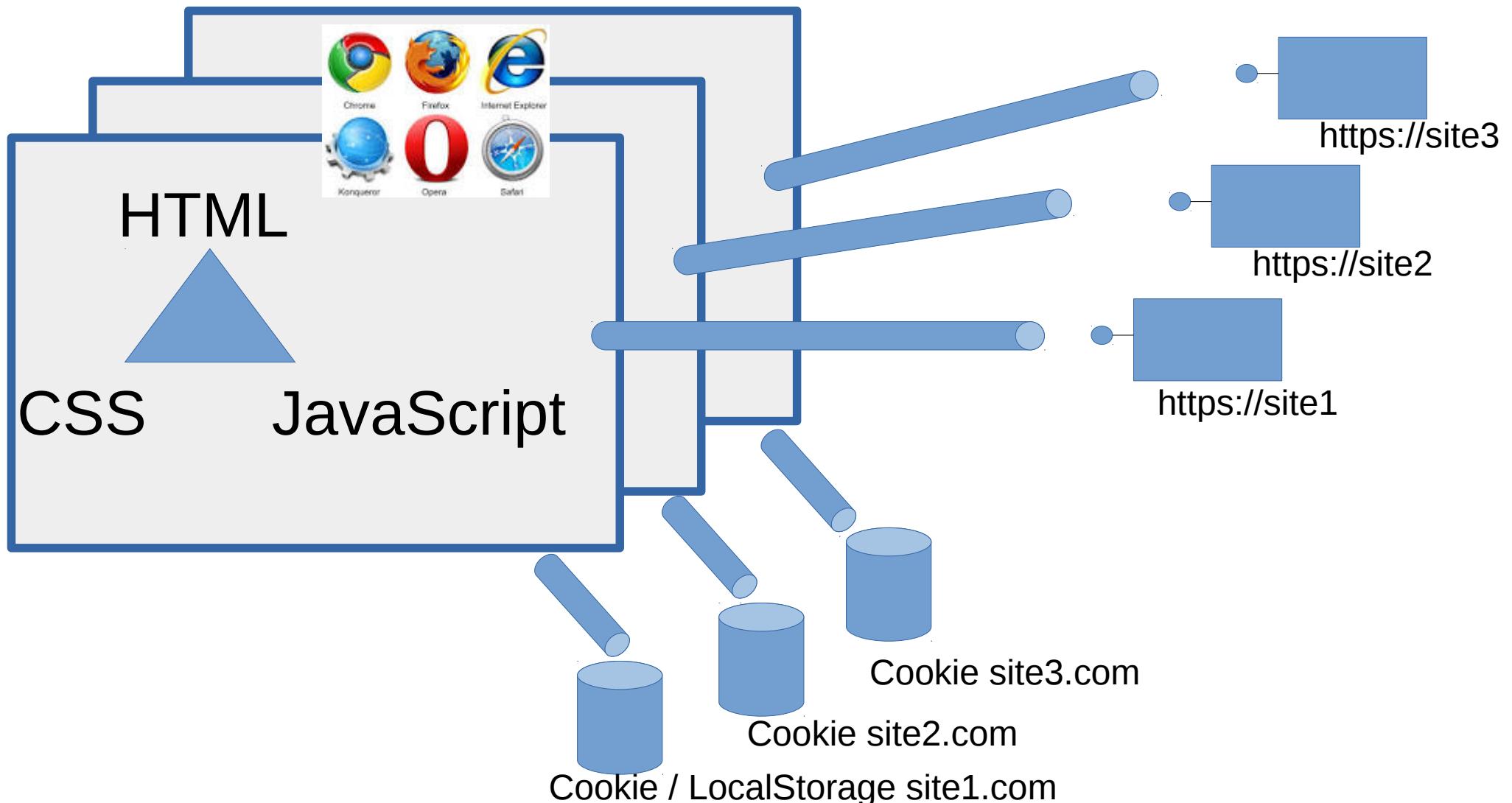


HTTP Response (20x,30x,40x,50x)  
headers  
body



WebSocket events

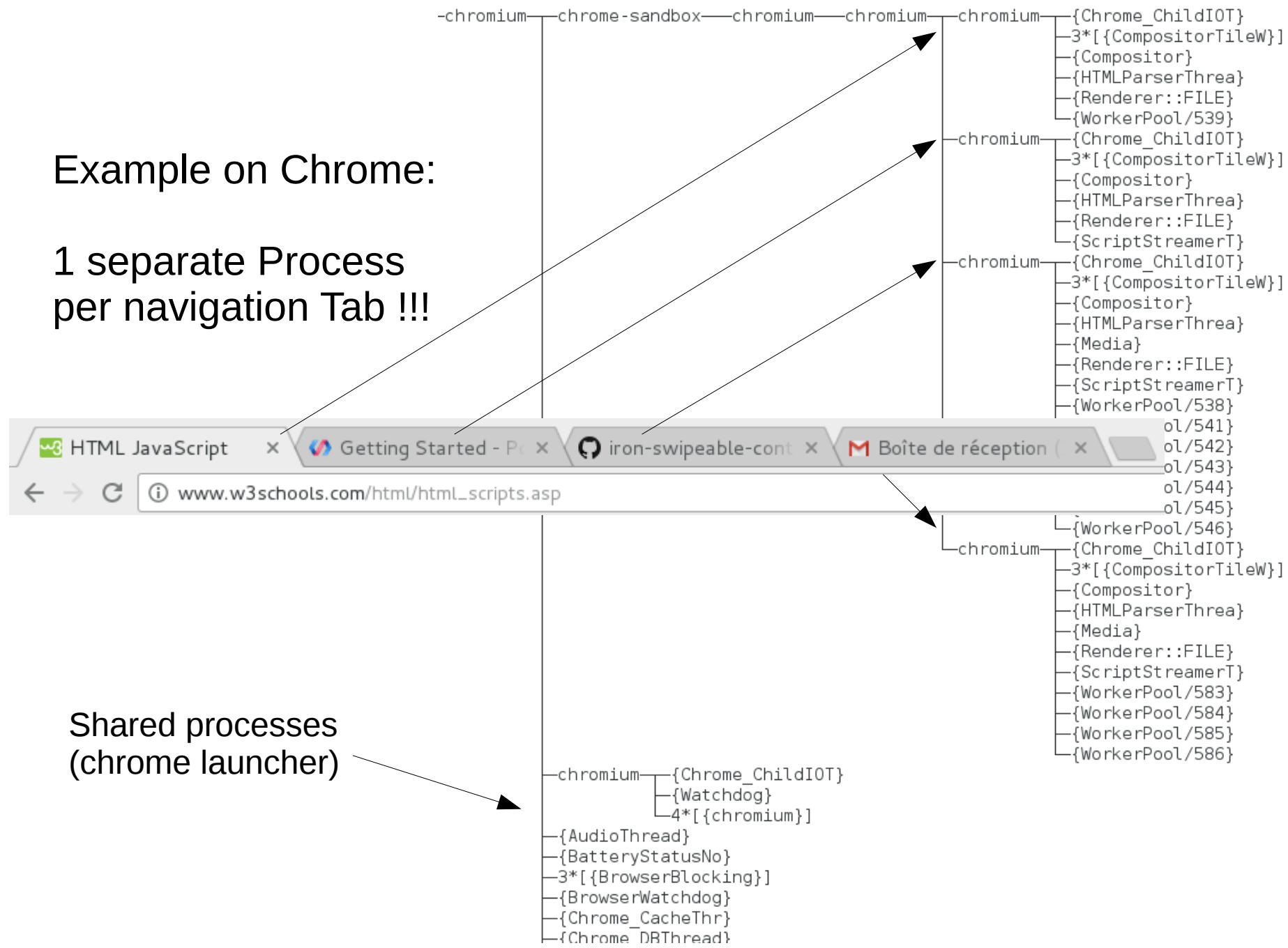
# Client-Side = Browser = Security Sandbox



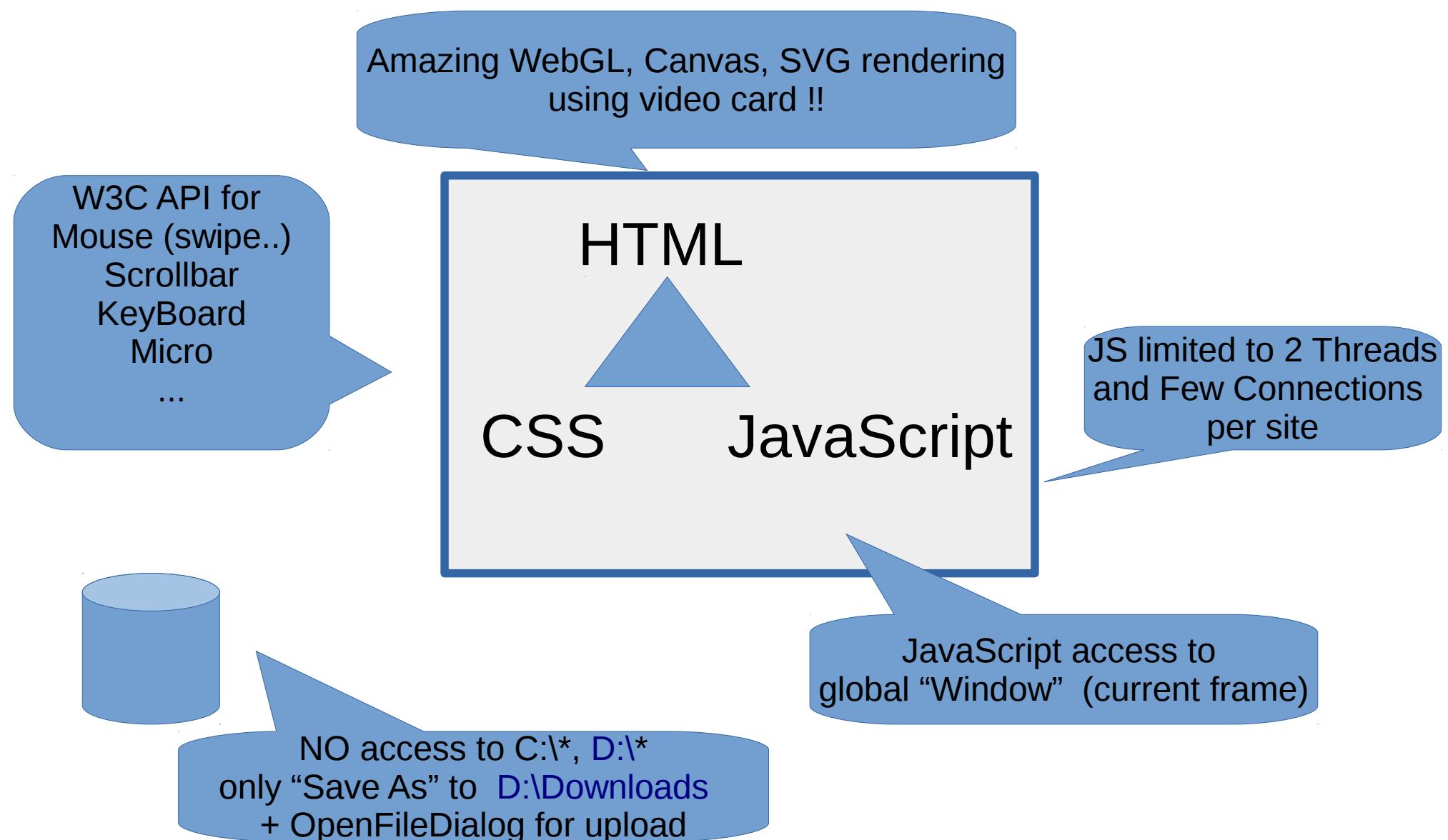
# Site Navigation Isolation ...

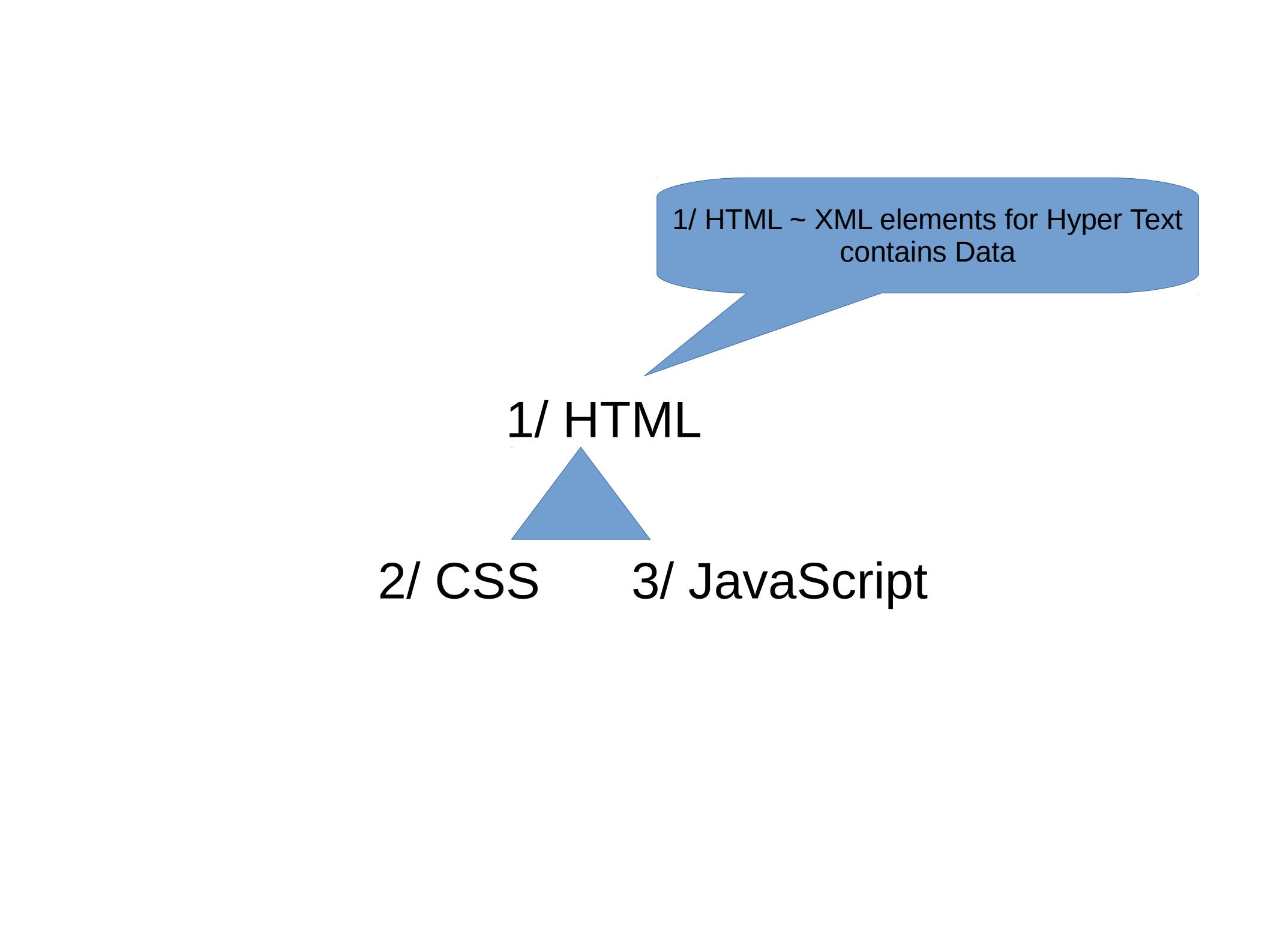
# Example on Chrome:

# 1 separate Process per navigation Tab !!!



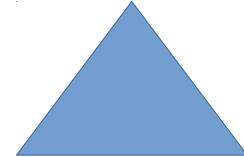
# On Client-Side = ONLY Html / CSS / (Limited) JavaScript





1/ HTML ~ XML elements for Hyper Text  
contains Data

1/ HTML



2/ CSS

3/ JavaScript

# Element = <tagName> ... </tagName>

The screenshot shows a navigation bar at the top with links for Home, HTML (which is highlighted in green), CSS, JavaScript, SQL, PHP, References, and Tutorials. On the left, there's a sidebar with links for HTML Basic, HTML Elements (also highlighted in green), HTML Attributes, HTML Headings, HTML Paragraphs, HTML Styles, and HTML Formatting. The main content area has a title 'HTML Elements' and a paragraph explaining that an HTML element consists of a start tag and end tag with content in between, followed by a code example.

HTML Basic

HTML Elements

HTML Attributes

HTML Headings

HTML Paragraphs

HTML Styles

HTML Formatting

## HTML Elements

An HTML element usually consists of a **start tag** and **end tag**, with the content inserted in between:

<tagname>Content goes here...</tagname>

# <elt attribute1="value1" attribute2 ..>

[HTML](#)[CSS](#)[JAVASCRIPT](#)[SQL](#)[PHP](#)[REFERENCES ▾](#)[HTML Elements](#)[HTML Attributes](#)[HTML Headings](#)[HTML Paragraphs](#)[HTML Styles](#)[HTML Formatting](#)[HTML Quotations](#)

## HTML Attributes

- All HTML elements can have **attributes**
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

### Example

```
<a href="http://www.w3schools.com">This is a link</a>
```

# Standard W3c Tags

## Basic Structure

```
<h1>Header 1</h1>
<h2>..</h2>
<h3>..<h3>
<h4>..</h4>
<p>paragraph</p>
<br/>
```

## Text Formatting

```
<pre> - pre-formatted
<b> - Bold text
<strong> - Important text
<i> - Italic text
<em> - Emphasized text
<mark> - Marked text
<small> - Small text
<del> - Deleted text
<ins> - Inserted text
<sub> - Subscript text
<sup> - Superscript text
```

## Hyper Link..

```
<A href="..">

<video >
<script >
<iframe>
```

## Other..

```
<table> <tr><td>
<ul> <li>
    <>
    <>...
    ...
    ...
```

Html 1.0 ... a simple MarkupLangage for Text

=> many specific tags for text formatting

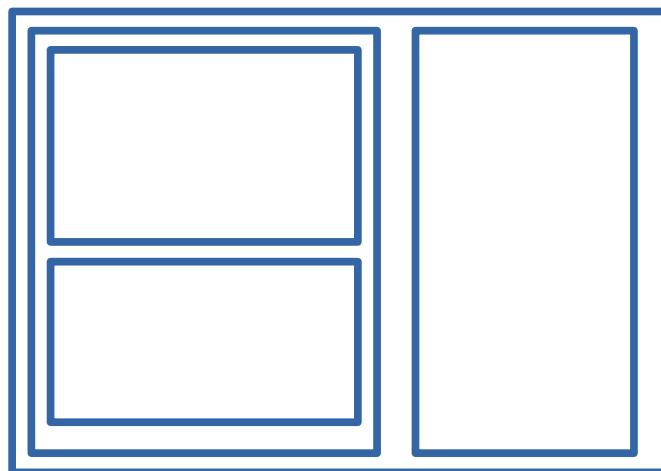
Only 2 families of tag layout: **Blocks & Inline**

=> technically no “needs” to add more  
than <div>&<span>

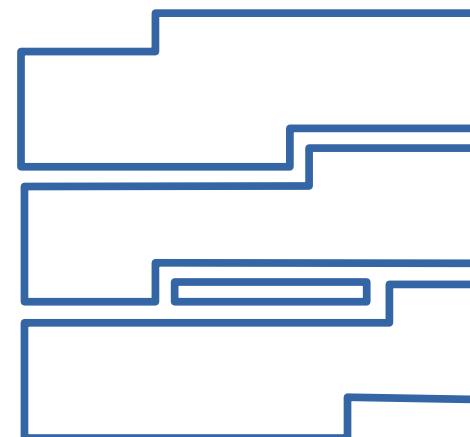
with CSS & JS everywhere

# <div> and <span> ..

<div> = blocks



<span> = inline



# Html = soup of <div><div><div> (example: gmail)

The screenshot shows the browser's developer tools with the 'Inspector' tab selected. The DOM tree is displayed, illustrating the complex nested structure of div elements that make up a Gmail interface. The tree starts with the root <html> tag and branches down through various classes like 'aAX', 'fr', 'body', 'nH', 'no', 'nn', 'ar4', 'z', 'A0', and 'aeF'. Specific elements like iframes for scripts and styles, and input fields, are also visible.

```
<!DOCTYPE html>
<html class="aAX" lang="fr">
  <head></head>
  <body class="aAU" tabindex="-1">
    <iframe aria-hidden="true" tabindex="-1" style="position: absolute; width: 9em; height: 9em; top: -99em;"></iframe>
    <div style="font-size:0; color:white; z-index:-9; position:absolute; height:0; width:0; overflow:hidden; left:30%; top:30%;"></div>
    <noscript><style> #loading {display:none} </style><form action="#"></noscript>
    <div id="loading" style="display: none;"></div>
    <div id="roster_comm_link" style="display:none"></div>
    <input id="hist_state" name="hist_state" style="display:none" type="text"></input>
    <script type="text/javascript"> //<![CDATA[ document.getElementById("reloadurl")...&lt;/script&gt;
    &lt;iframe id="js_frame" class="invfr" name="j1ffbeopa7vygl" src="/_scs/mail-static/_/js/k=gmail.main.fr.uh_As34DTMg.0/m=m_i,...-z_wf4N_a:&gt;&lt;/iframe&gt;
    &lt;iframe id="sound_frame" class="invfr" name="slffbeopa7vygl" src="?ui=2&amp;view=bsp&amp;ver=ohhl4rw8mbn4" tabindex="-1" title="empty" aria-hidden="true" style="width: 2144px; height: 917px; position: absolute; top: -1000px; height: 1px; overflow: hidden;"&gt;&lt;/iframe&gt;
    &lt;div class="ata-asE" style="display: none;"&gt;&lt;/div&gt;
    &lt;div aria-atomic="true" aria-live="assertive" style="position: absolute; top: -1000px; height: 1px; overflow: hidden;"&gt;&lt;/div&gt;
    &lt;div aria-atomic="true" aria-live="polite" style="position: absolute; top: -1000px; height: 1px; overflow: hidden;"&gt;Forums&lt;/div&gt;
    &lt;div style="position: relative; min-height: 100%;"&gt;
      &lt;div class="vI8oZc cS"&gt;&lt;/div&gt;
      &lt;div tabindex="0"&gt;&lt;/div&gt;
      &lt;div class="nH" style="width: 2144px; position: relative; height: 917px; margin-top: -917px; border: 1px solid black; background-color: white; z-index: 1; border-radius: 10px; padding: 10px; font-family: sans-serif; font-size: 14px; color: black; transition: all 0.3s ease; "&gt;
        &lt;div class="nH" style="width: 1933px; height: 1933px; margin: auto; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
          &lt;div class="nH" style="width: 100%; height: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
            &lt;div class="nH ar4 z" style="width: 100%; height: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
              &lt;div class="mq nH oy8Mbf" style="display: none; width: 100%; height: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
                &lt;div&gt;
                  &lt;div id=":a" class="aeH"&gt;&lt;/div&gt;
                  &lt;div class="A0"&gt;
                    &lt;div id=":9" class="Tm aeJ" style="height: 899px; width: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
                      &lt;div id=":7" class="aeF" style="min-height: 472.283px; width: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
                        &lt;div class="nH" style="width: 100%; height: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
                          &lt;div class="BltHke nH oy8Mbf aE3" role="main" style="width: 100%; height: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
                            &lt;div style="width: 100%; height: 100%; border: 1px solid black; border-radius: 10px; padding: 10px; background-color: white; z-index: 1; position: relative; "&gt;
</pre>
```

Code quality with Only technical <div>&<span> ?

Need more semantic tags for web applications

# CustomElement

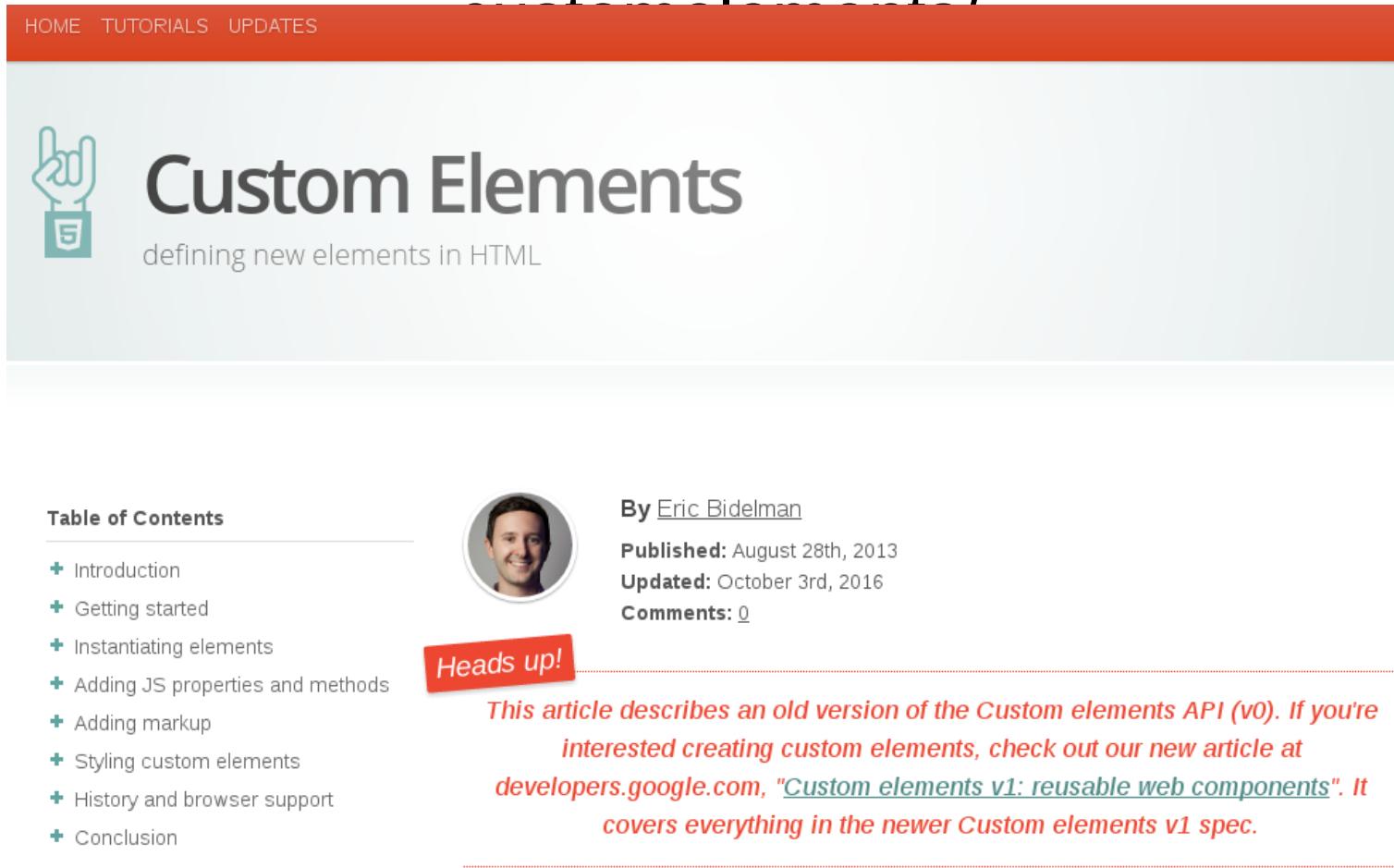


Define your own application specific tags:  
`<your-custom-tag>`

`<your-menu> <your-view> <your-object>`

# Eric Bidelman's article

<https://www.html5rocks.com/en/tutorials/webcomponents/custom-elements/>



The screenshot shows a web browser displaying an article titled "Custom Elements" by Eric Bidelman. The page has a red header bar with links for "HOME", "TUTORIALS", and "UPDATES". Below the header, there's a teal icon of a hand making a rock-on sign next to a small "5" symbol. The main title "Custom Elements" is in large, bold, dark gray letters. Below it, the subtitle "defining new elements in HTML" is in smaller gray text. On the left, a "Table of Contents" sidebar lists various sections with plus signs. In the center, there's a circular profile picture of Eric Bidelman, followed by his name and publication details: "Published: August 28th, 2013" and "Updated: October 3rd, 2016". A red callout box labeled "Heads up!" contains a note about an old version of the API. The main content area starts with an "Introduction" section.

HOME TUTORIALS UPDATES

# Custom Elements

defining new elements in HTML

**Table of Contents**

- + Introduction
- + Getting started
- + Instantiating elements
- + Adding JS properties and methods
- + Adding markup
- + Styling custom elements
- + History and browser support
- + Conclusion

**Localizations**

- + 简体中文
- + 中文 (简体)
- + 日本語
- + Contribute another

**By** Eric Bidelman

**Published:** August 28th, 2013

**Updated:** October 3rd, 2016

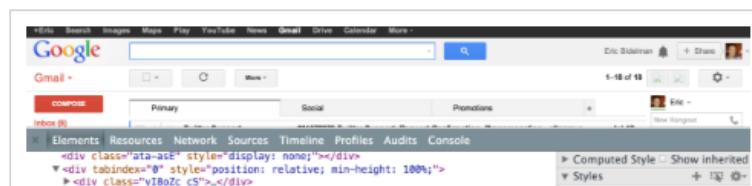
**Comments:** 0

**Heads up!**

This article describes an old version of the Custom elements API (v0). If you're interested creating custom elements, check out our new article at [developers.google.com](#), "[Custom elements v1: reusable web components](#)". It covers everything in the newer Custom elements v1 spec.

## Introduction

The web severely lacks expression. To see what I mean, take a peek at a "modern" web app like GMail:





*Modern web apps: built with <div> soup.*

There's nothing modern about <div> soup. And yet,...this is how we build web apps. It's sad. Shouldn't we demand more from our platform?

## Sexy markup. Let's make it a thing.

HTML gives us an excellent tool for structuring a document but its vocabulary is limited to elements the [HTML standard](#) defines.

What if the markup for GMail wasn't atrocious? What if it was beautiful:

```
<hangout-module>
  <hangout-chat from="Paul, Addy">
    <hangout-discussion>
      <hangout-message from="Paul" profile="profile.png"
                      profile="11807591949626375791" datetime="2013-07-17T12:02">
        <p>Feelin' this Web Components thing.</p>
        <p>Heard of it?</p>
      </hangout-message>
    </hangout-discussion>
  </hangout-chat>
  <hangout-chat>...</hangout-chat>
</hangout-module>
```

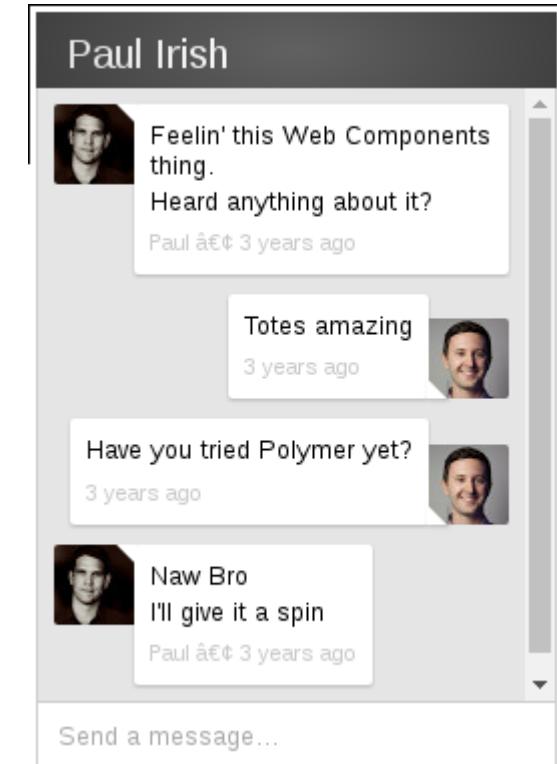
Try the demo!

How refreshing! This app totally makes sense too. It's **meaningful, easy to**

# Example Demo :

## http://html5-demos.appspot.com/hangouts

```
<!DOCTYPE html>
<html>
  ><head>...</head>
  ><body>
    ><div class="newchat">...</div>
    ><div id="hangouts">
      <!-- <hangout-module from="Paul Irish"
      profile="118075919496626375791"></hangout-module> -->
      ><hangout-module>
        >#shadow-root (open)
          ><style>...</style>
          ><hangout-header>...</hangout-header>
          ><hangout-discussion id="discussion">
            >#shadow-root (open)
              ><template repeat="{{message in messages}}">...</template>
              ><hangout-message isother="{{message.isother}}" from="{{from}}"
              profile="{{message.profile}}" datetime="{{message.datetime}}"
              now="{{now}}" class="isother">
                >#shadow-root (open)
                  ><template repeat="{{msg in message.msg}}">...</template> == $0
                    <p>Feelin' this Web Components thing.</p>
                    <p>Heard anything about it?</p>
                  </hangout-message>
                  ><hangout-message isother="{{message.isother}}" from="{{from}}"
                  profile="{{message.profile}}" datetime="{{message.datetime}}"
                  now="{{now}}" class="self">...</hangout-message>
                  ><hangout-message isother="{{message.isother}}" from="{{from}}"
                  profile="{{message.profile}}" datetime="{{message.datetime}}"
                  now="{{now}}" class="self">...</hangout-message>
                  ><hangout-message isother="{{message.isother}}" from="{{from}}"
                  profile="{{message.profile}}" datetime="{{message.datetime}}"
                  now="{{now}}" class="isother">...</hangout-message>
```



## TABLE OF CONTENTS

1.	<b>Conformance</b>
2.	<b>HTML: Custom elements</b>
2.1	Introduction
2.1.1	Creating an autonomous custom element
2.1.2	Creating a customized built-in element
2.1.3	Drawbacks of autonomous custom elements
2.1.4	Upgrading elements after their creation
2.2	Requirements for custom element constructors
2.3	Core concepts
2.4	The CustomElementRegistry interface
2.5	Upgrades
2.6	Custom element reactions
3.	<b>HTML: HTML element constructors</b>
4.	<b>Miscellaneous patches</b>
4.1	HTML: The <code>window</code> object
4.2	HTML: Pseudo-classes
4.3	HTML: Creating and inserting nodes
4.4	HTML: Parsing XHTML documents
4.5	HTML: Content model
4.6	HTML: Wide-ranging patches
4.7	DOM: Elements
4.8	DOM: Cloning
4.9	DOM: Mutation algorithms
4.10	DOM: Wide-ranging patches
A.	<b>Acknowledgments</b>
B.	<b>References</b>
D.1	Normative references

# Custom Elements

W3C Working Draft 13 October 2016

**This version:**

<https://www.w3.org/TR/2016/WD-custom-elements-20161013/>

**Latest published version:**

<https://www.w3.org/TR/custom-elements/>

**Latest editor's draft:**

<https://w3c.github.io/webcomponents/spec/custom/>

**Previous version:**

<https://www.w3.org/TR/2016/WD-custom-elements-20161002/>

**Editor:**

[Domenic Denicola](#), Google, Inc.

**Repository:**

[We are on Github.](#)

[File a bug.](#)

[Commit history.](#)

**Mailing list:**

[public-webapps@w3.org](mailto:public-webapps@w3.org)

**Implementation:**

[Can I use Custom Elements?](#)

[Test Suite](#)

[Test Suite repository](#)

[Copyright](#) © 2016 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C [liability](#), [trademark](#) and [permissive document license](#) rules apply.

## Abstract

This specification describes the method for enabling the author to define and use new types of DOM elements in a document. It is a copy of the relevant parts of [\[HTML\]](#) up through commit

# Custom Elt

```
<input id="todoItemText" type="text"/>
<button type="button"
  onclick="onClickAddTodoItem()"
>Add</button>
<ul id="todoItemList">
  <todo-item
    todo="first todo(attr??)" />
</ul>
```

```
"use strict";
class TodoItem extends HTMLElement {
  constructor(self) {
    super(self);
    console.info("ctor");
  }
  createdCallback() {
    console.info("created");
    this._li = document.createElement("li");
    this._textNode = document.createTextNode("");
    this._li.appendChild(this._textNode);
    this.appendChild(this._li);
  }
  attachedCallback() { console.info("attached"); }
  detachedCallback() {
    console.info("detached");
  }
  set todo(text) {
    console.info("set attribute");
    this._textNode.data = text;
  }
  get todo() {
    return this._textNode.data;
  }
  attributeChangedCallback(name, oldValue, newValue) {
    if (name === 'todo') {
      console.info("attributeChange");
      this._textNode.data = newValue;
    }
  }
}
document.registerElement("todo-item", TodoItem);

function findById(id) { return document.getElementById(id); }
function onClickAddTodoItem() {
  let text = findById('todoItemText').value;
  let newTodoItemElt = document.createElement("todo-item");
  newTodoItemElt.todo = text;
  findById('todoItemList').appendChild(newTodoItemElt);
}
```

# Custom Elements ...

attribute not working in this sample?

- 
- test
- attribute not working in this sample??

The screenshot shows the Chrome DevTools interface with the 'Elements' tab selected. On the left, the DOM tree displays a simple HTML structure with a script element and a list of todo items. The 'todo-item' element is expanded to show its internal structure, including a list item ('li') and some text nodes. On the right, the 'Properties' panel is open for the first 'todo-item' node. It lists various properties such as '\_li', '\_textNode', 'accessKey', 'assignedSlot', 'attributes', 'baseURI', 'childElementCount', 'childNodes', 'children', 'classList', 'className', 'clientHeight', 'clientLeft', 'clientTop', and 'clientWidth'. The 'Attributes' section shows a 'data-todo' attribute with the value 'first todo(attr??)'.

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <script>...</script>
    <input id="todoItemText" type="text">
    <button type="button" onclick="onClickAddTodoItem()">Add</button>
    <ul id="todoItemList">
      <todo-item data-todo="first todo(attr??)">...
        <li>test</li>
      </todo-item>
      <todo-item> == $0
        <li>test</li>
      </todo-item>
    </ul>
  </body>
</html>
```

Properties
_li: li
_textNode: text
accessKey: ""
assignedSlot: null
attributes: NamedNodeMap
baseURI: "file:///mnt/a_1terate2/homeData/arnaud/personal/development/CustomElements/elements.html"
childElementCount: 1
childNodes: NodeList[1]
children: HTMLCollection[1]
classList: DOMTokenList[0]
className: ""
clientHeight: 0
clientLeft: 0
clientTop: 0
clientWidth: 0

# WebComponent ... Simplify CustomElement

Goals :

- 1) simplify Javascript ...replace by Html  
`<dom-module><template>`
- 2) supports old browser via PolyFill
- 3) use Template + ShadowDom



Please help us make Web Components & PWA's better by completing this survey



# WebComponents.org

a place to discuss and evolve web component best-practices

## POLYFILLS

The webcomponent.js polyfills enable Web Components in (evergreen) browsers that lack native support.

Install with [Bower](#)

```
bower install webcomponentsjs
```

Install with [npm](#)

```
npm install webcomponents.js
```

[Download  
webcomponents.js](#)

0.7.12 (117KB minified, 34KB gzipped)

[learn more about the polyfills](#)

## BROWSER SUPPORT

CHROME   OPERA   FIREFOX   SAFARI   IE/EDGE

## DISCOVER



### CUSTOMELEMENTS.IO

A gallery to display Web Components created by the community.



### POLYMER CATALOG

The official product line catalog for Polymer Web Components.



### BUILT WITH POLYMER

A curated collection of web apps and websites using Polymer.

## SPECS



### CUSTOM ELEMENTS

This specification describes the method for enabling the author to define reusable UI components.

## PRESENTATIONS



Accessibility  
is My Favorite Part  
of the Platform

### ACCESSIBILITY IS MY FAVORITE PART OF THE PLATFORM

My favorite part of my job is when I get to work on accessibility. I see this as removing uncertainty and anxiety from people's day, and helping them access the information they need. But working with accessibility can be frustrating, especially when you feel like you're fighting the platform instead of working with it.

[Read More >](#)

[see all presentations](#)

# WebComponents = 4 w3c specs

= CustomElement + Import + Template + ShadowDom

### BROWSER SUPPORT

	CHROME	OPERA	FIREFOX	SAFARI	IE/EDGE
Custom Element	Green	Green	Green	Green	Green
Imports	Green	Green	Yellow	Red	Red
Template	Green	Green	Yellow	Yellow	Red
Shadow DOM	Green	Green	Yellow	Green	Red

### LIBRARIES

**POLYMER**  
Polymer is a new type of library for the web, built on top of Web Components, and designed to leverage the evolving web platform on modern browsers.

**X-TAG**  
X-Tag is a small JavaScript library, initially created by Mozilla & now supported by Microsoft, that brings Web Components Custom Element capabilities to all modern browsers.

**BOSONIC**

### SPECS

**CUSTOM ELEMENTS**  
This specification describes the method for enabling the author to define and use new types of DOM elements in a document.

**HTML IMPORTS**  
HTML Imports are a way to include and reuse HTML documents in other HTML documents.

**TEMPLATES**  
This specification describes a method for declaring inert DOM subtrees in HTML and manipulating them to instantiate document fragments with identical contents.

**SHADOW DOM**  
This specification describes a method of establishing and maintaining functional boundaries between DOM trees and how these trees interact with each other within a document, thus enabling better functional encapsulation within the DOM.

# [www.polymer-project.org](http://www.polymer-project.org)

## = Google Library of WebComponents

 Polymer Project

Start Polymer 2.0 Preview App Toolbox News Element Catalog

---

**#UseThePlatform to build a better web**

---



The Polymer Project helps you deliver amazing user experiences by unlocking the full potential of the web platform.

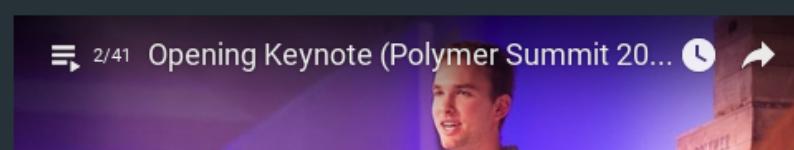
Use the [Polymer library](#) to make the most of Web Components, a powerful new platform feature for extending HTML and componentizing your apps.

Use the [Polymer App Toolbox](#) to build cutting-edge Progressive Web Apps that load quickly, respond instantly and work from anywhere.

[GET STARTED](#) [TRY THE SHOP DEMO](#) [ABOUT THE PROJECT](#)

### Polymer Summit 2016

Polymer Summit 2016 is done and dusted! Missed it, or want to replay your favorite talks? You can find all the videos here!

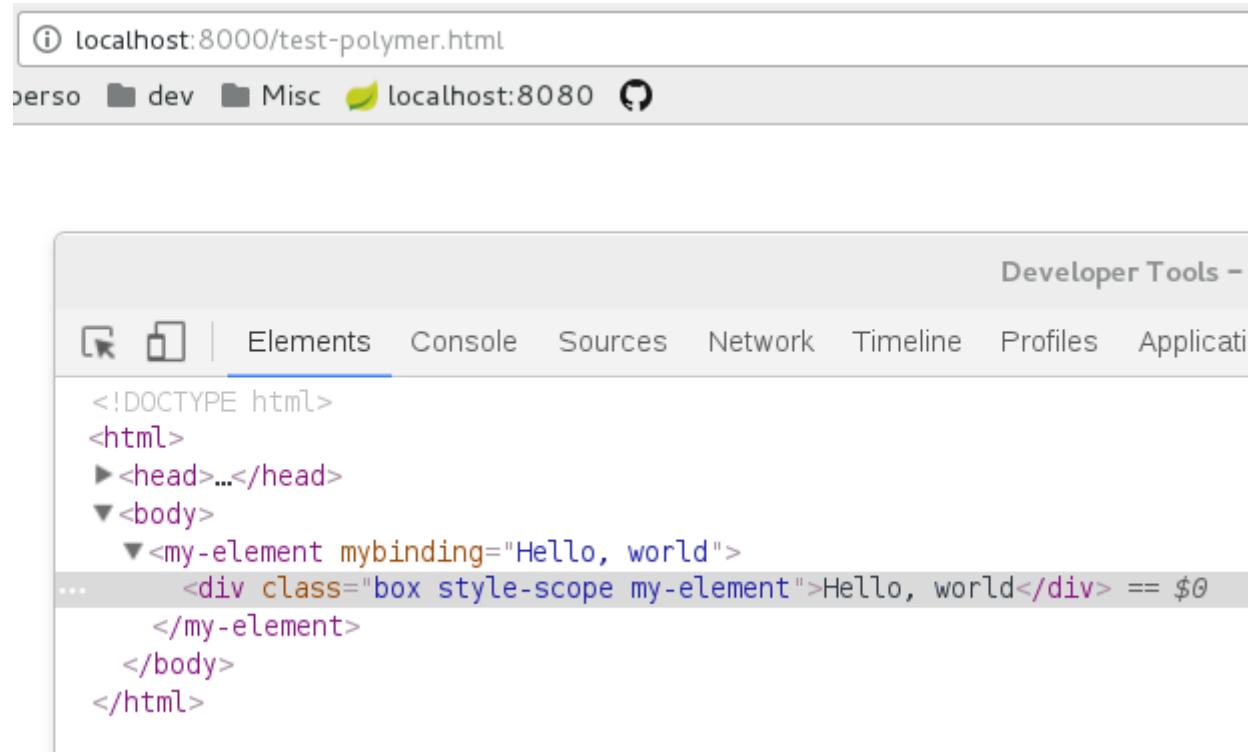


# Polymer example

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <script src="bower_components/webcomponentsjs/webcomponents-lite.js"></script>
  <link rel="import" href="bower_components/polymer/polymer.html">
  <link rel="import" href="my-element.html" />
  <!-- notice... does not work on file:// ... need use server, example: "python -m SimpleHTTPServer
</head>
<body>
  <my-element mybinding="Hello, world"></my-element>
</body>
</html>
```

File “my-element.html”

```
<dom-module id="my-element">
  <template>
    <style>
      .box{
        color: grey;
      }
    </style>
    <div class="box">{{mybinding}}</div>
  </template>
  <script>
    Polymer({
      is: "my-element",
      properties: {
        mybinding: {
          type: String,
          value: "My text here"
        }
      });
    </script>
</dom-module>
```





# Polymer custom elements catalog

0.10.1

## App

App Elements

---

App elements

1.0.10

## Fe

Iron Elements

---

Polymer core elements

1.0.7

## Md

Paper Elements

---

Material design elements

1.1.1

## Go

Google Web Components

---

Components for Google's APIs and services

1.0.1

## Au

Gold Elements

---

Ecommerce Elements

1.0.0

## Ne

Neon Elements

---

Animation and Special Effects

2.0.0

## Pt

Platinum Elements

---

Offline, push, and more

1.0.0

## Mo

Molecules

---

Wrappers for third-party libraries

## Element Guides

[Flexbox layout with iron-flex-layout](#)  
Simple flexbox layout

[Responsive Material Design layouts](#)  
How to create responsive Material Design layouts with Paper and Iron elements.

[Using Elements](#)  
Learn how to install and use Polymer Elements in your own projects.

[neon-animation](#)  
A short guide to neon-animation and neon-animated-pages

# Polymer Core Elements : Fe (Iron)

The screenshot shows the Polymer Elements Catalog interface. On the left is a sidebar with colored squares and labels: Catalog (red), Search Elements (blue), All Elements (grey), App Elements (purple), Iron Elements (green, selected), Paper Elements (grey), Google Web Components (blue), Gold Elements (orange), Neon Elements (teal), Platinum Elements (grey), and Molecules (red). The main content area has a header "Iron Elements" and a description: "A set of visual and non-visual utility elements. Includes elements for working with layout, user input, selection, and scaffolding apps". Below is a table with columns "Name" and "Description".

Name	Description
iron-a11y-announcer	A singleton element that simplifies announcing text to screen readers.
iron-a11y-keys	A basic element implementation of iron-a11y-keys-behavior, matching the legacy core-a11y-keys.
iron-a11y-keys-behavior	A behavior that enables keybindings for greater a11y.
iron-ajax	Makes it easy to make ajax calls and parse the response
iron-autogrow-textarea	A textarea element that automatically grows with input
iron-behaviors	Provides a set of behaviors for the iron elements
iron-checked-element-behavior	Implements an element that has a checked attribute and can be added to a form
iron-collapse	Provides a collapsable container
iron-component-page	Turns a raw element definition into beautiful documentation
iron-demo-helpers	Utility classes to make building demo pages easier

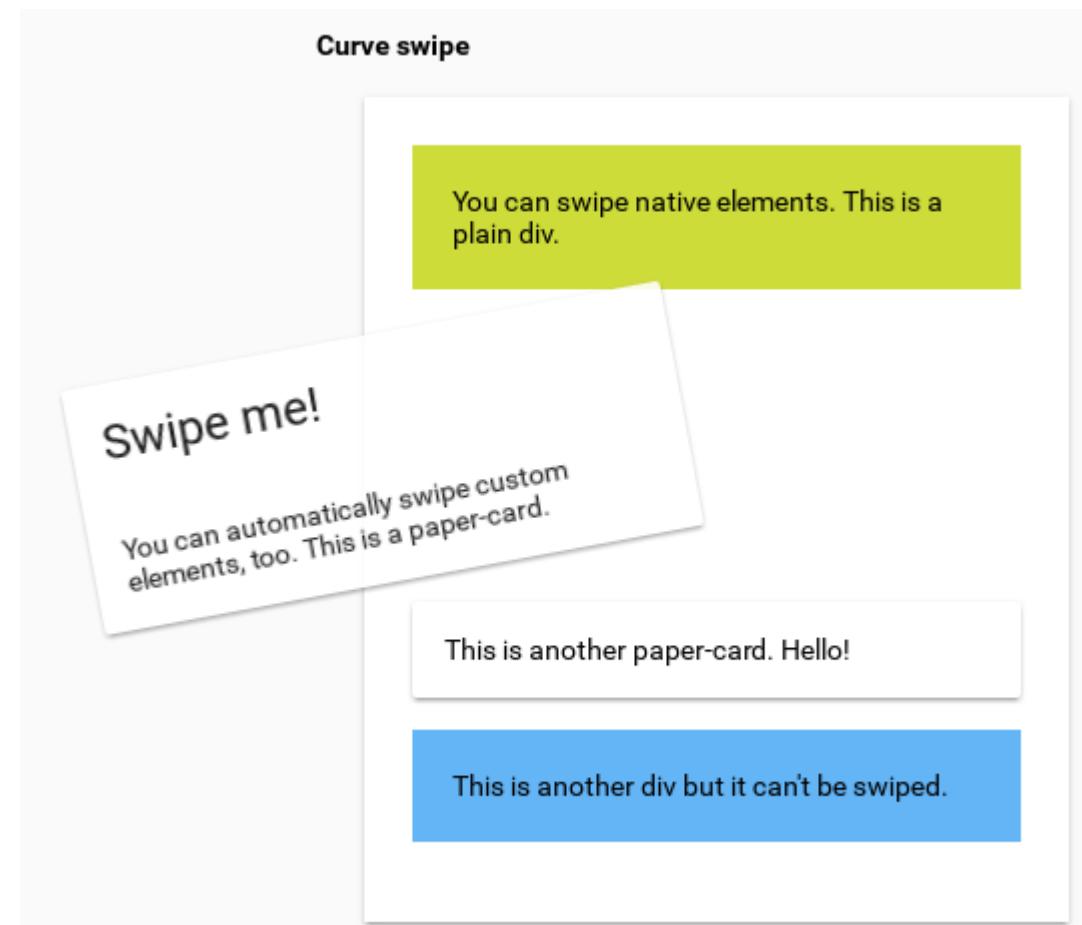
At the bottom right of the table, there are icons for star, document, link, and refresh.

# Iron example

## <iron-swipeable-container>

<https://github.com/PolymerElements/Iron-swipeable-container/blob/master/demo/index.html>

```
<h4>Curve swipe</h4>
<div class="vertical-section">
  <iron-swipeable-container swipe-style="curve">
    <div class="item swipe">You can swipe native
      <paper-card class="swipe" heading="Swipe me!">
        <div class="card-content">
          You can automatically swipe custom elements, too. This is a paper-card.
        </div>
      </paper-card>
    </div>
  </iron-swipeable-container>
</div>
```



# Material Design : Md (Paper)

The screenshot shows the Polymer Elements Catalog interface. On the left is a sidebar with colored tabs for different element categories: Catalog (blue), Search Elements (grey), All Elements (grey), App Elements (purple), Iron Elements (green), Paper Elements (selected, grey), Google Web Components (blue), Gold Elements (orange), Neon Elements (teal), Platinum Elements (grey), and Molecules (red). The main content area has a header "Paper Elements" and a description: "Paper elements are a set of visual elements that implement Google's Material Design." Below this is a table listing ten paper elements, each with a name and a brief description. The table includes columns for Name and Description. The last two columns of the table are icons for star, copy, link, and eye.

Name	Description	Star	Copy	Link	Eye
paper-badge	Material design status message for elements				
paper-behaviors	Common behaviors across the paper elements				
paper-button	Material design button				
paper-card	Material design piece of paper with unique related data				
paper-checkbox	A material design checkbox				
paper-dialog	A Material Design dialog				
paper-dialog-behavior	Implements a behavior used for material design dialogs				
paper-dialog-scrollable	A scrollable area used inside the material design dialog				
paper-drawer-panel	A responsive drawer panel				
paper-dropdown-menu	An element that works similarly to a native browser select				
paper-fab	A material design floating action button				
paper-header-panel	A header and content wrapper for layout with headers				

# Md Paper Example

Catalog

Md Paper Elements

paper-checkbox 1.4.1

A material design checkbox

Docs

Demo

Source

Add to Collection

Bower Command

```
bower install --save PolymerEle
```

### Checkboxes can be checked or unchecked, or disabled entirely

Checkbox    Checkbox    Disabled

```
<paper-checkbox>Checkbox</paper-checkbox>
<paper-checkbox checked>Checkbox</paper-checkbox>
<paper-checkbox disabled>Disabled</paper-checkbox>
```

### Checkboxes can hide the ripple effect using the noink attribute

Checkbox

```
<paper-checkbox noink>Checkbox</paper-checkbox>
```

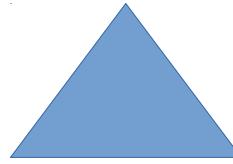
### Checkboxes can be styled using custom properties

Checkbox  
With a subtitle

Checkbox  
With a longer subtitle that wraps to another line

Checkbox

**1/ HTML**



**2/ CSS**

**3/ JavaScript**

**2/ CSS ~ size,border,colors,fonts  
contains Style**

# `style=" ...inlined CSS property:value;"`

[HTML](#)[CSS](#)[JAVASCRIPT](#)[SQL](#)[PHP](#)[BOOTSTRAP](#)[JQUERY](#)[EXAMPLES ▾](#)[REFERENCES ▾](#)[HTML Paragraphs](#)[HTML Styles](#)[HTML Formatting](#)[HTML Quotations](#)[HTML Computercode](#)[HTML Comments](#)[HTML Colors](#)[HTML CSS](#)[HTML Links](#)[HTML Images](#)[HTML Tables](#)[HTML Lists](#)[HTML Blocks](#)[HTML Classes](#)

## The HTML Style Attribute

Setting the style of an HTML element, can be done with the **style attribute**.

The HTML style attribute has the following **syntax**:

```
<tagname style="property:value;">
```

The **property** is a CSS property. The **value** is a CSS value.

You will learn more about CSS later in this tutorial.

# Styles ... using CSS files

[HTML](#)[CSS](#)[JAVASCRIPT](#)[SQL](#)[PHP](#)[BOOTSTRAP](#)[JQUERY](#)[EXAMPLES ▾](#)[REFERENCES](#)

HTML Colors

[HTML CSS](#)[HTML Links](#)[HTML Images](#)[HTML Tables](#)[HTML Lists](#)[HTML Blocks](#)[HTML Classes](#)[HTML Iframes](#)[HTML JavaScript](#)[HTML Head](#)[HTML Layout](#)[HTML Responsive](#)[HTML Entities](#)[HTML Symbols](#)[HTML Charset](#)[HTML URL Encode](#)[HTML XHTML](#)

## HTML Forms

[HTML Forms](#)[HTML Form Elements](#)[HTML Input Types](#)[HTML Input Attributes](#)

## HTML Styles - CSS

[◀ Previous](#)

CSS = Styles and  
Colors

M a n i p u l a t e   T e x t

C o l o r s ,   B o x e s

## Styling HTML with CSS

**CSS** stands for **Cascading Style Sheets**.

CSS describes **how HTML elements are to be displayed on screen, paper, or in other media**.

# Internal CSS Internal / External CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element:

## Example

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1 {color: blue;}
p {color: red;}
</style>
```

## External CSS

An external style sheet is used to define the style for many HTML pages.

**With an external style sheet, you can change the look of an entire web site, by changing one file!**

To use an external style sheet, add a link to it in the <head> section of the HTML page:

## Example

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="styles.css">
```

# Style with < class="..">

HOME HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP JQUERY EXAMPLES ▾ REF

HTML Classes

HTML Iframes  
HTML JavaScript  
HTML Head  
HTML Layout  
HTML Responsive  
HTML Entities  
HTML Symbols  
HTML Charset  
HTML URL Encode  
HTML XHTML

HTML Forms

HTML Forms  
HTML Form Elements  
HTML Input Types  
HTML Input Attributes

HTML5

HTML5 Intro  
HTML5 Support  
HTML5 Elements  
HTML5 Semantics  
HTML5 Migration  
HTML5 Style Guide

HTML Graphics

HTML Canvas

## HTML The class Attribute

◀ Previous

### Using The class Attribute

The HTML class attribute makes it possible to define equal styles for elements with the same class name.

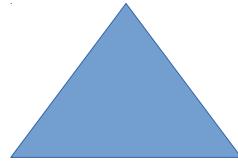
Here we have three <div> elements that points to the same class name:

#### Example

```
<!DOCTYPE html>
<html>
<head>
<style>
div.cities {
    background-color: black;
    color: white;
    margin: 20px 0 20px 0;
    padding: 20px;
}
</style>
</head>
<body>

<div class="cities">
<h2>London</h2>
```

1/ HTML



2/ CSS

3/ JavaScript

3/ Script  
contains actions,events,behavior

A blue rounded rectangular box with a white speech bubble pointing towards it from the top left. The text "3/ Script" is at the top, followed by "contains actions,events,behavior" below it.

## HTML JavaScript

HTML Head

HTML Layout

HTML Responsive

HTML Entities

HTML Symbols

HTML Charset

HTML URL Encode

HTML XHTML

## HTML Forms

HTML Forms

HTML Form Elements

HTML Input Types

HTML Input Attributes

## HTML5

HTML5 Intro

HTML5 Support

HTML5 Elements

HTML5 Semantics

HTML5 Migration

HTML5 Style Guide

## HTML Graphics

HTML Canvas

HTML SVG

HTML Google Maps

# HTML JavaScript

[◀ Previous](#)

JavaScript makes HTML pages more dynamic and interactive.

## Example

### My First JavaScript

Click me to display Date and Time

Sat Nov 26 2016 21:58:34 GMT+0100 (CET)

[Try it Yourself »](#)

```
<div class="w3-padding w3-white notranslate">
  <h2>My First JavaScript</h2>
  <button type="button" onclick=
    "document.getElementById('demo').innerHTML=Date()">
    Click me to display Date and Time</button> == $0
  <p id="demo"></p>
</div>
```

## The HTML <script> Tag

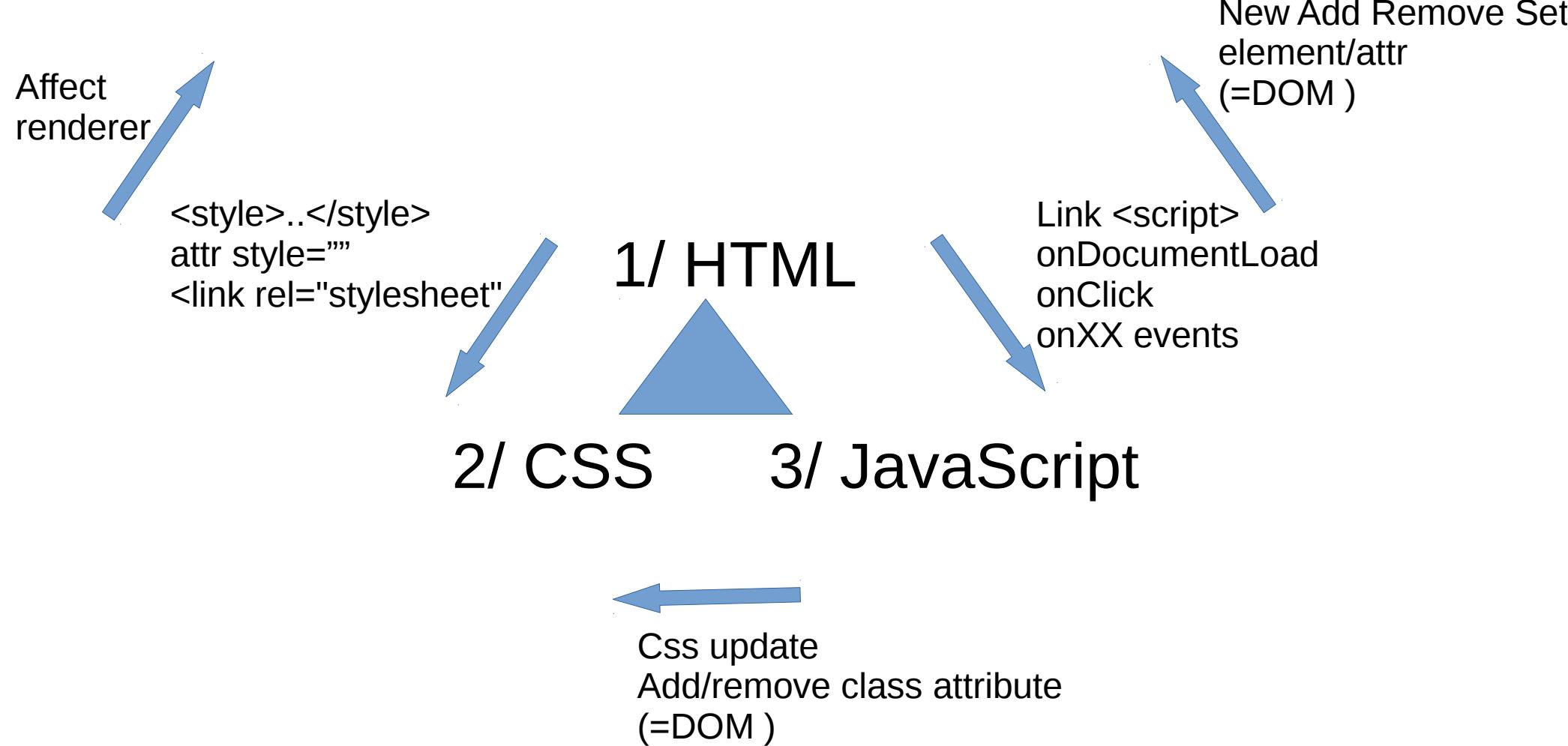
The **<script>** tag is used to define a client-side script (JavaScript).

The **<script>** element either contains scripting statements, or it points to an external script file through the **src** attribute.

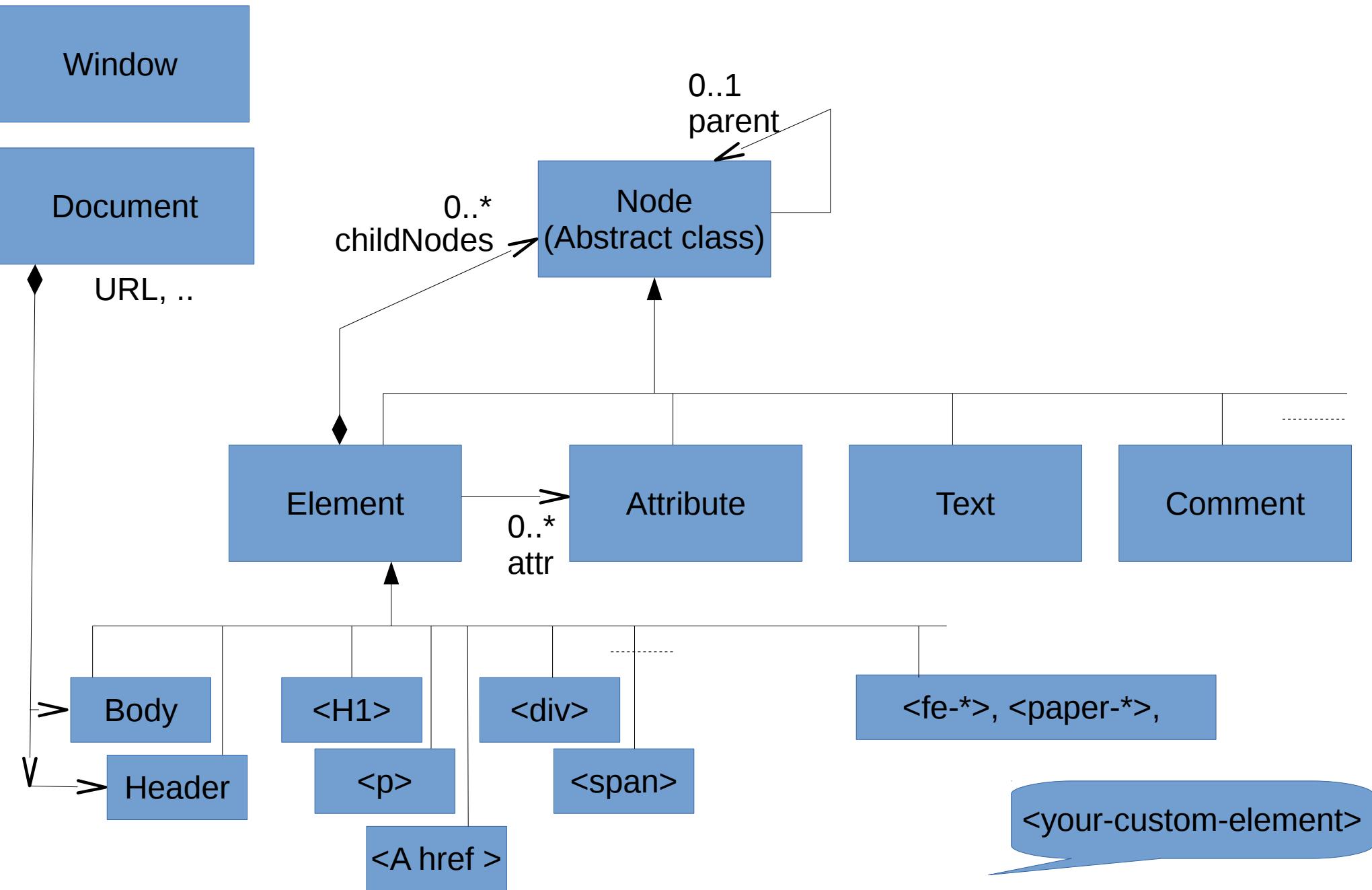
Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.

To select an HTML element, JavaScript very often use the `document.getElementById(id)` method.

# Html / CSS / JavaScript Interactions



# DOM = Document Object Model



# DOM = Class Hierarchy

```
public abstract class Node {  
    private Node parent;  
  
    ..  
    public Node getParent() { .. }  
}
```

```
public abstract class Element extends Node  
{  
    private List<Node> childList;  
    private List<Attribute> attributes;  
  
    ..  
}
```

# DOM Reference

w3schools.com

THE WORLD'S LARGEST WEB DEVELOPER SITE



HTML

- HTML Tag Reference
- HTML Event Reference
- HTML Color Reference
- HTML Attribute Reference
- HTML Canvas Reference
- HTML SVG Reference
- Google Maps Reference

CSS

- CSS Reference
- CSS Selector Reference
- W3.CSS Reference
- Bootstrap Reference
- Icon Reference

JavaScript

- JavaScript Reference
- HTML DOM Reference
- jQuery Reference
- jQuery Mobile Reference
- AngularJS Reference

XML

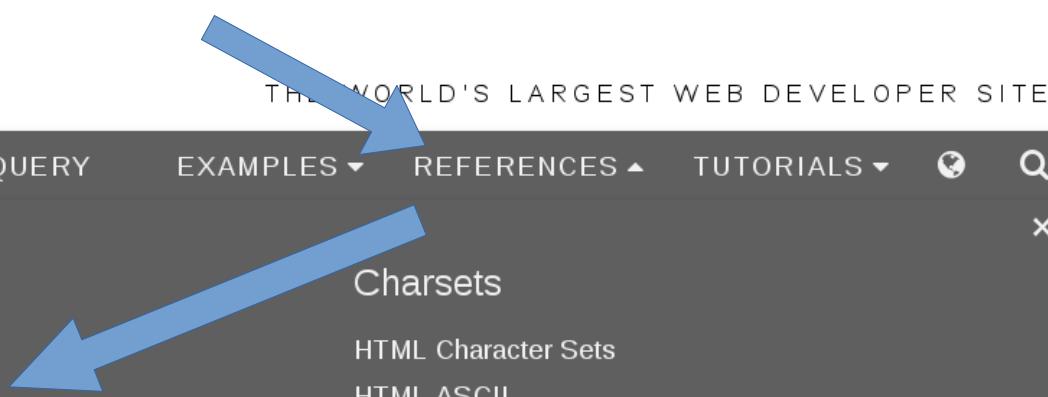
- XML Reference
- XML Http Reference
- XSLT Reference
- XML Schema Reference

Charsets

- HTML Character Sets
- HTML ASCII
- HTML ANSI
- HTML Windows-1252
- HTML ISO-8859-1
- HTML Symbols
- HTML UTF-8

Server Side

- PHP Reference
- SQL Reference
- ASP Reference



# DOM Elements

## Properties, Method, Events

### HTML DOM

DOM Document

DOM Elements

DOM Attributes

DOM Events

DOM Style

### HTML Objects

<a>

<abbr>

<address>

<area>

<article>

<aside>

<audio>

<b>

<base>

<bdo>

<blockquote>

<body>

<br>

<button>

<canvas>

<caption>

## Properties and Methods

The following properties and methods can be used on all HTML elements:

Property / Method	Description
<code>element.accessKey</code>	Sets or returns the accesskey attribute of an element
<code>element.addEventListener()</code>	Attaches an event handler to the specified element
<code>element.appendChild()</code>	Adds a new child node, to an element, as the last child node
<code>element.attributes</code>	Returns a NamedNodeMap of an element's attributes
<code>element.blur()</code>	Removes focus from an element
<code>element.childElementCount</code>	Returns the number of child elements an element has
<code>element.childNodes</code>	Returns a collection of an element's child nodes (including text and comment nodes)
<code>element.children</code>	Returns a collection of an element's child element (excluding text and comment nodes)
<code>element.classList</code>	Returns the class name(s) of an element
<code>element.className</code>	Sets or returns the value of the class attribute of an element
<code>element.click()</code>	Simulates a mouse-click on an element
<code>element.clientHeight</code>	Returns the height of an element, including padding

# DOM Elements Reference

HOME HTML CSS JAVASCRIPT SQL PHP BOOTSTRAP JQUERY EXAMPLES ▾ REFERENCES ▾ TUTORIALS

## HTML DOM

DOM Document  
DOM Elements  
DOM Attributes  
DOM Events  
DOM Style

## HTML Objects

<a>  
<abbr>  
<address>  
<area>  
<article>  
<aside>  
<audio>  
<b>  
<base>  
<bdo>  
<blockquote>  
<body>  
<br>  
<button>  
<canvas>  
<caption>  
<cite>  
<code>

## HTML DOM Reference

The references describe the properties and methods of the HTML DOM, along with examples.

DOM Document

DOM Elements

DOM Attributes

DOM Events

DOM Style

## HTML Element Objects Reference

The references describe the properties and methods of each HTML object, along with examples.



# Typical JS Code to Find & Edit Elts

test

Add

- test
- test
- test
- test
- test

```
<script>
  function onClickAddTodoItem() {
    let todoItemTextElt = document.getElementById('todoItemText');
    let text = todoItemTextElt.value;
    let newTodoItemElt = document.createElement("li");
    let newTodoItemTextElt = document.createTextNode(text);
    newTodoItemElt.appendChild(newTodoItemTextElt);
    let todoItemListElt = document.getElementById('todoItemList');
    todoItemListElt.appendChild(newTodoItemElt);
  }
</script>
<input id="todoItemText" type="text"></input>
<button type="button" onclick="onClickAddTodoItem()">Add</button>
<ul id="todoItemList"></ul>
```

Find:

Document  
.getElementById()

Create:

Document  
.createElement("li")  
.createTextNode()  
...

Update:

element.appendChild()  
value = element.getAttribute()  
element.setAttribute = value  
Element.innerHTML  
...

# jQuery for Portable/Concise DOM Manipulation?

Plugins Contribute Events Support jQuery Foundation

Your donations help fund the continued development and growth of jQuery.

SUPPORT THE PROJECT

Download API Documentation Blog Plugins Browser Support Search

 write less, do more.

 Lightweight Footprint  
Only 32kB minified and gzipped. Can also be included as an AMD module

 CSS3 Compliant  
Supports CSS3 selectors to find elements as well as in style property manipulation

 Cross-Browser  
[Chrome](#), [Edge](#), [Firefox](#), [IE](#), [Safari](#), [Android](#), [iOS](#), and more

 Download jQuery v3.1.1

[View Source on GitHub →](#)  
[How jQuery Works →](#)

## What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

## Corporate Members

Make DOM manipulation, event, ... “simpler” ...

## Resources

- [jQuery Core API Documentation](#)
- [jQuery Learning Center](#)
- [jQuery Blog](#)
- [Contribute to jQuery](#)
- [About the jQuery Foundation](#)
- [Browse or Submit jQuery Bugs](#)

# Same TodoItem example using JQuery

test  Add

- test
- test
- test
- test
- test

```
<head>
    <script src="jquery.min.js"></script>
</head>
<body>
<script>
$(document).ready(function(){
    $("#todoItemAddButton").click(function(){
        let text = $("#todoItemText").val();
        $("#todoItemList").append("<li>" + text + "</li>");
    });
});
</script>
<input id="todoItemText" type="text"></input>
<button id="todoItemAddButton" type="button">Add</button>
<ul id="todoItemList"></ul>
</body>
```

# Jquery \$() : wrapper for Document.getElementById(..)

Code is smaller ??  
by including “<script src=”jquery.min.js”></script>

Well ... If you use only 1 method of Jquery...

You could have use “jqueryLight”

or even

```
function my$(id) {  
    return Document.getElementById(id);  
}
```

# Jquery Risk... Script Code Injection!

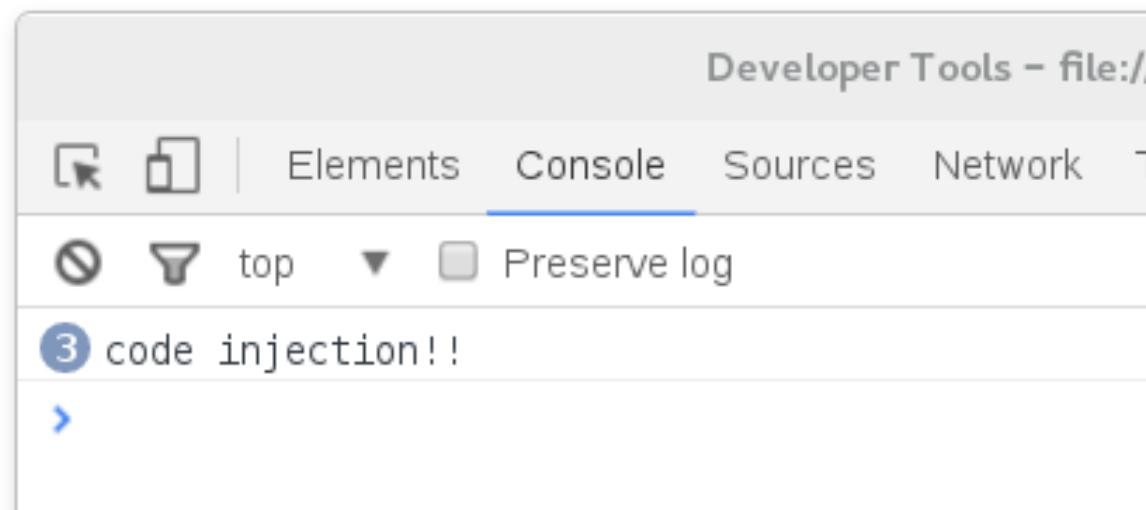
Example in previous example :

append("<li>" + text + "</li>")

... text can contain html script!!

```
AA <script>console.log("code injection!!")</script> Ho!! 
```

- AA Ho!!
- AA Ho!!
- AA Ho!!



# Jquery vs DOM

Naive retrial after google it  
... same pb !!

```
$( '<li />', {html: text} ).appendTo( '#todoItemList' );
```

This one works ...

```
$("#todoItemList").append($('<li />').append(document.createTextNode(text)));
```

```
AA <script>console.log("code injection!!")</script> Ho!! 
```

- test
- test
- AA <script>console.log("code injection!!")</script> Ho!!
- AA <script>console.log("code injection!!")</script> Ho!!

Compare with Explicit DOM  
+ findById

```
function findById(id) { return document.getElementById(id); }
function onClickAddTodoItem2() {
    let text = findById('todoItemText').value;
    let newTodoItemElt = document.createElement("li");
    newTodoItemElt.appendChild(document.createTextNode(text));
    findById('todoItemList').appendChild(newTodoItemElt);
}
```

# Jquery Bashing in 2016 ...

`$()` code not simple ! ... not “typed”

JQuery is a very low-level framework  
... for writing mostly imperative code  
Or applying plugins

No easy bi-direction Binding  
Better use Richer Framework, like AngularJS

Browser Portability no more an issue in 2016?

[Home](#)[Learn](#)[Develop](#)[Discuss](#)

## HTML enhanced for web apps!

 [Download AngularJS 1](#)

(1.6.0-rc.2 / 1.5.9 / 1.2.32)

[Try the new Angular 2](#) [View on GitHub](#) [Design Docs & Notes](#)[Follow +AngularJS on G+](#)[Follow @angularjs](#)

187K followers

[Tweet](#)

Learn Angular in your browser for free!

# Same TodoItem example using AngularJS-1.x

```
<!DOCTYPE html>
<html>
<head>
    <script src="angular.min.js"></script>
<script>
angular.module("myApp", []).controller("MyController", function() {
    var self = this;
    self.todoText = '';
    self.todoItems = [];
    self.onClickAddTodoItem = function() {
        self.todoItems.push(self.todoText);
    };
});
</script>
</head>
<body ng-app="myApp">
<div ng-controller="MyController as ctrl">
    <input ng-model="ctrl.todoText" type="text" size="40"></input>
    <button ng-click="ctrl.onClickAddTodoItem()" type="button">Add</button>
    <ul>
        <li ng-repeat="item in ctrl.todoItems track by $index">{{item}}</li>
    </ul>
</div>
</body>
</html>
```

# DOM vs Jquery vs AngularJS

- AngularJS code ~ 25 lines / jquery ~ 21 lines
- AngularJS code much more readable



Imperative code

Direct DOM manipulation

NO Binding, Events

events history required



Declarative code

MVC : Model-View-Controller

Bi-Directionnal Binding

(no events required, only Model)

# Much more to say on Angular MVC Binding...

- very handy writing quick web app
- not very efficient having 1000 objects in scope
- ...digest() loop is horrible
- Angular 2.x is complete rewrite of 1.x
- ...still not very clean code (digest zone?)  
Model should be controlled.. and fire chg Event
- ReactJs, Aurelia, ...  
... JS Word = “1 fwk a day, the doctor away”?

# Conclusion

- Only a “very short” introduction to Html-Js-Dom
- I Hope you feel
  - “I Did not know all this before”
  - “I am happy I have learned something”
  - “I want to learn more”

This document:

<http://arnaud-nauwynck.github.io/docs/Intro-Html-JS-DOM.pdf>