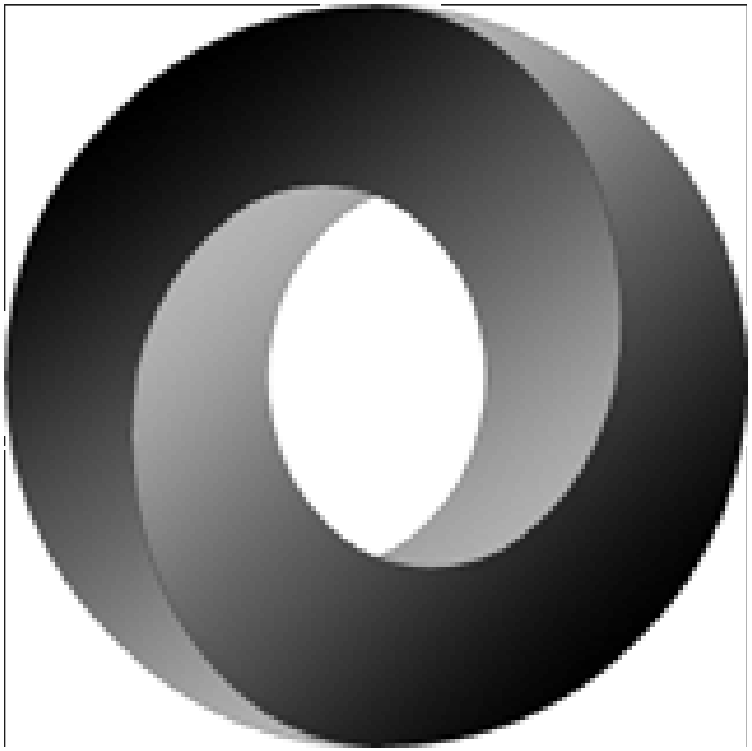
JAVASCRIPT IN BROWSERS



These slides have been prepared by Cyril Concolato.

JAVASCRIPT VS. ECMASCRIPT

- What is ECMAScript?
  - Programming/Scripting Language
  - Interpreted code (not compiled into machine code), Portable code
  - Standard syntax
  - Invented by Brendan Eich at Netscape (and Microsoft JScript)
- Versions
  - JavaScript 1.5-2.0
  - ECMA-262 3rd, (4th), 5th, 6th (2015), 7th edition (draft)
- In the Web Browser: JavaScript
  - Executed by the JavaScript engine of the browser according to a model
  - Used with specific interfaces (DOM, ...)
- More: Tutorial Videos by Douglas Crockford

JAVASCRIPT BASICS

- How to declare/assign a variable?
- How to define a function?
- How to call a function?
- Arrays
- Strings
- Objects
- Properties

BROWSERS AND JAVASCRIPT

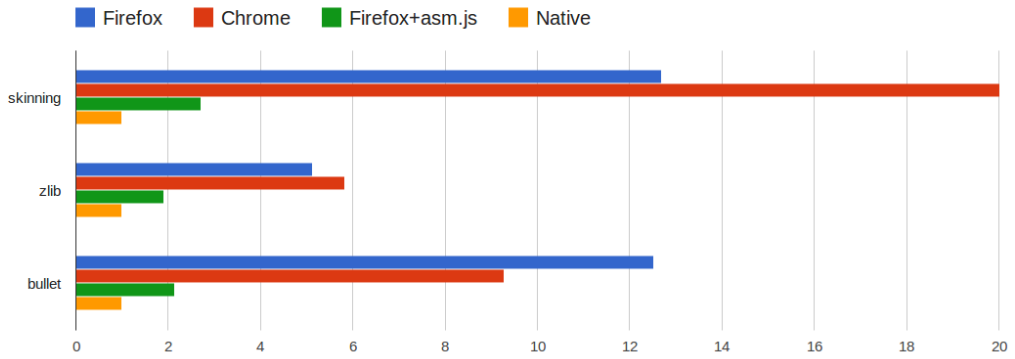
- The JavaScript Engine is a core component of browsers
  - Used for:
    - Interactivity, animations, media manipulations (Canvas, audio API, ...)
  - Potential problems
    - Security
    - Performance

JAVASCRIPT ENGINES RACE

- IE9 Chakra, Opera Carakan, Safari Nitro, Firefox JägerMonkey, Chrome V8
- Benchmarks:
  - SunSpider
  - V8 Benchmark
  - Octane
  - ...
- Optimizations: JIT, dead-branch ...

JAVASCRIPT COMPILATION

- Google Web Toolkit (GWT): from Java to Javascript
- Emscripten: from C/C++ to Javascript
- Asm.js: restricted JS with improved execution speed



Run time normalized to Native (clang -O2), lower values are better

SCRIPT PROCESSING IN HTML

WEB APPLICATIONS=

- HTML +
  - Document structure
  - Textual content and media resources (images, ...)
- CSS +
  - Presentation information
- JavaScript (=ECMAScript + Web APIs)
  - Browser-interpreted code to provide the intelligence, behavior of the application

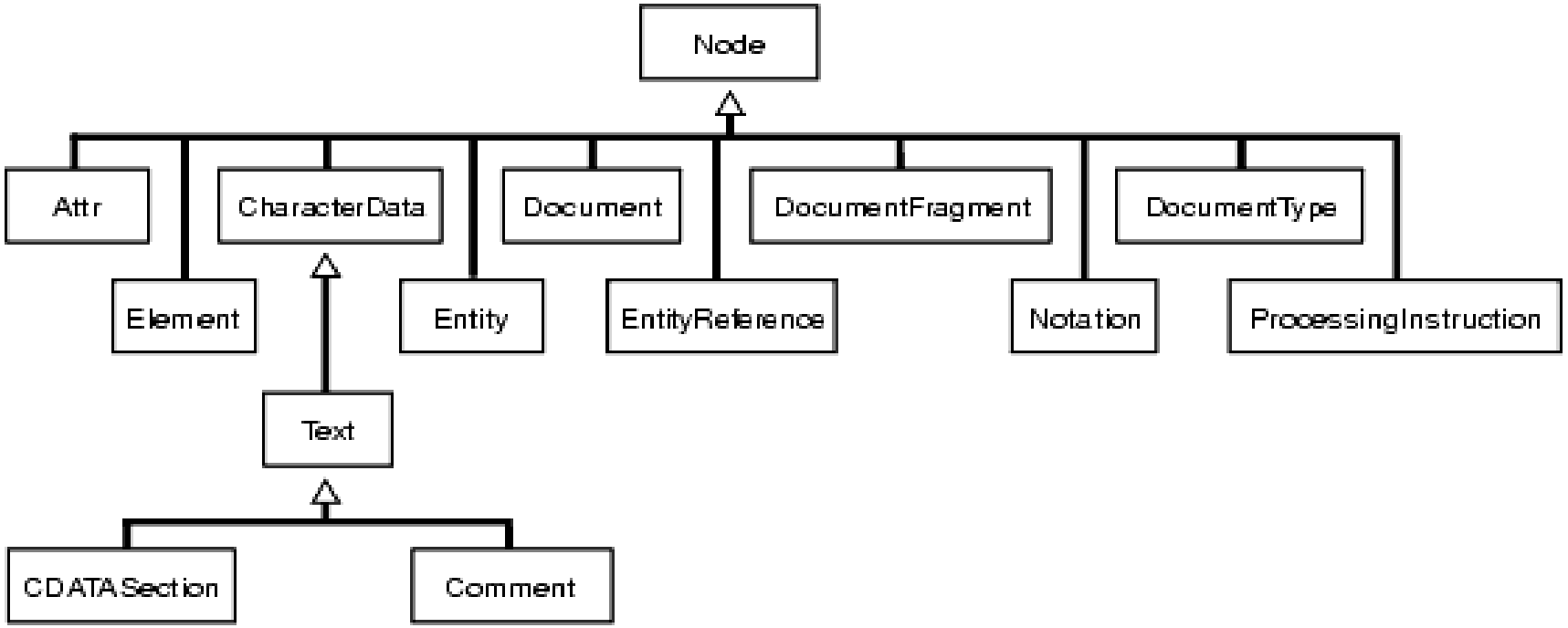
“THERE IS A WEB API FOR EVERYTHING”

- Basic APIs
  - Document Object Model (DOM): Core, Events, Window, ...
- Specific APIs
  - Communication APIs
    - XHR, Push, WebSockets, ...
  - Drawing APIs
    - Canvas, WebGL, ...
  - Storage APIs
    - Files, Cookies, Database, ...
  - Multimedia APIs
    - Audio, video, streaming, ...
  - Device APIs
    - Battery, AdressBook, WebCam ...
  - System APIs

DOCUMENT OBJECT MODEL (DOM) INTERFACES

- Interfaces to the document tree
  - For access and modifications of content, structure, and style of documents
- Specifications
  - Level 1 (one single specification)
  - Level 2 (6 specs): Core, Style, (Views), ...
  - Level 3 (3 specs): Core, ...
  - Level 4

DOM INTERFACES HIERARCHY



DOM INTERFACES: METHODS AND PROPERTIES

- The Node interface

```
nodeType
parentNode
firstChild
nextChild
hasChildNodes()
hasAttributes()
appendChild()
removeChild()
```

- The Document interface

```
documentElement
getElementById()
getElementsByTagName()
querySelector()
createElement()
createTextNode()
```

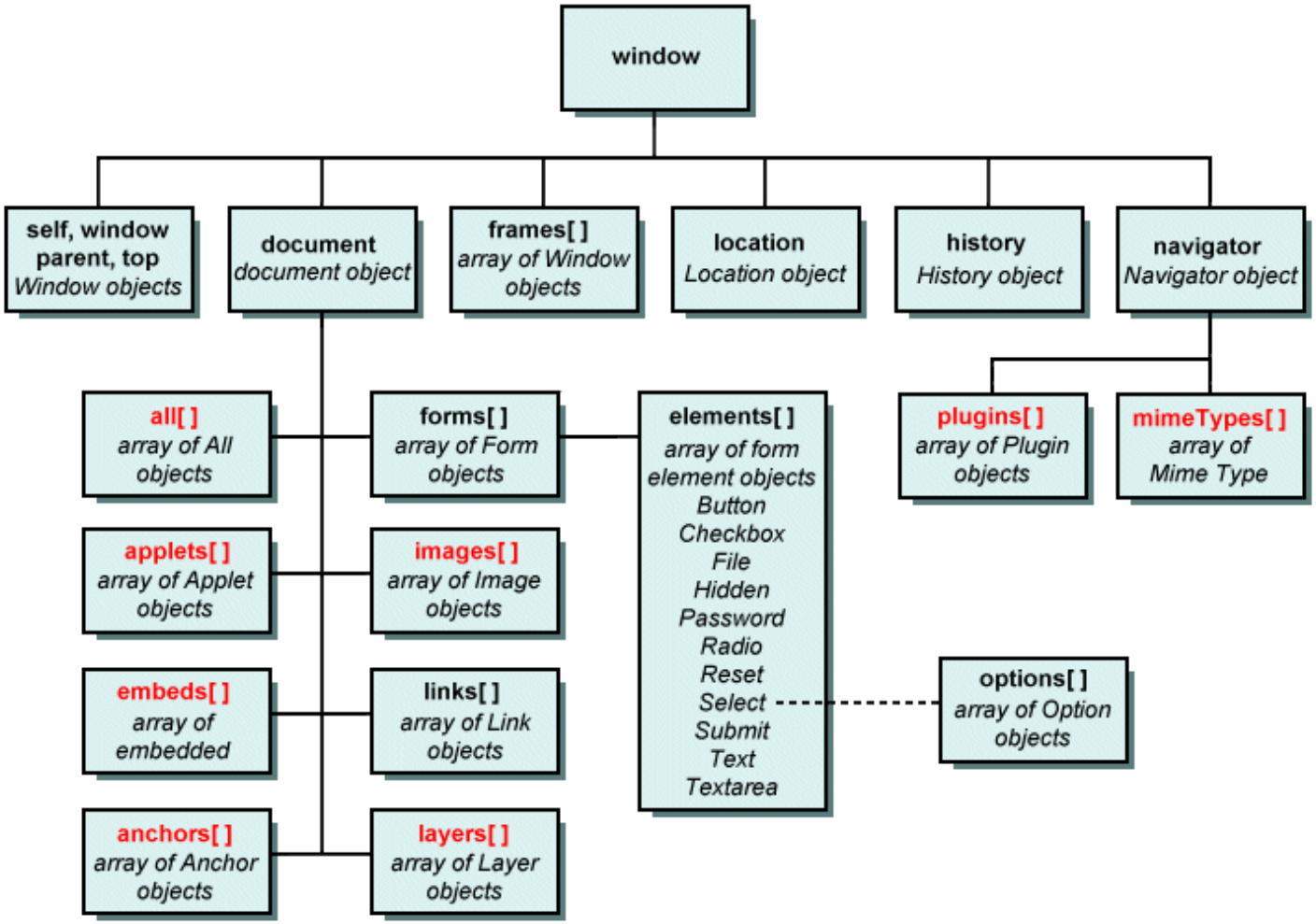
- The Element interface

```
innerHTML
```

THE WINDOW OBJECT

- API corresponding to the browser window or tab
- Convenient API for various usages
  - Timing (animations)
  - General events (load, ...)
  - Navigation (history)
  - Embedding (openURL)
- JavaScript `global` object in browser

THE WINDOW OBJECT



CSS AND JAVASCRIPT

- The JavaScript style property
  - Used to set a new style on an element
  - Used to query the style on this element

```
var e = document.getElementById("SomeElementId");
e.style.top = 10px;
```

- The `getComputedStyle()` method
  - To ask for all styles (inherited, computed, ...) of an element

```
var e = document.getElementById("SomeElementId");
var style = window.getComputedStyle(e);
var height = style.getPropertyValue("height");
```

JSON – JAVASCRIPT OBJECT NOTATION

- Format for exchanging data
  - Text based
  - Structured
  - Easy seraliazing/parsing
- Based on JavaScript
  - Literal notations
    - Object {}
    - Array []
    - String ""

JSON - EXAMPLE

```
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    { "type": "home", "number": "212 555-1234" },
    { "type": "fax", "number": "646 555-4567" }
  ]
}
```

JSON VS. XML

```xml
<person>
 <age>12</age>
 <name>Danielle</name>
</person>
```

```json
{
  "age" : 12,
  "name" : "Danielle"
}
```

JAVASCRIPT LIBRAIRIES

- Principles
  - Simplify the JS code written by Web Developers
  - Provide a unique interface for all browsers (bugs)
- Many librairies
  - JQuery,
  - Angular,
  - Bootstrap ...
- JavaScript “beautifier”/“minifier”

SCRIPTED ANIMATIONS

- Use of timers and callback functions
  - Ex: using the `window` object
  - Ex: using an `SVGTimer` object
  - Ex: using `requestAnimationFrame`
- Management of the synchronization by the script

ANIMATIONS WITH JS

```
<rect id='R' width="120" height="50" fill="blue">
<script>
function doAnimation(){
  var rect=document.getElementById('R');
  x=x+xincr;
  rect.setAttribute('x', x);
  window.setTimeout("doAnimation()", 10);
}
</script>
```

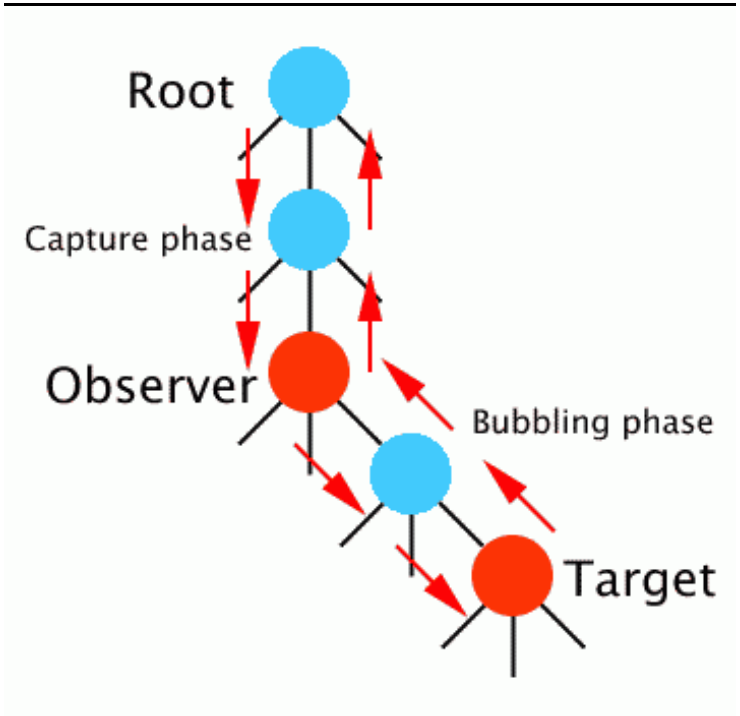
```
function animloop() {
  requestAnimFrame(animloop);
  render();
}
```

INTERACTIVITY & SCRIPTING

- Simple interactivity does not require scripting
  - Forms filing and submitting
  - Navigation
  - Triggering animations or transitions
  - ...
- More complex interactions require Javascript with
  - DOM events
  - AJAX Pattern

DOM EVENTS

- API to indicate to the browser how to process events in JavaScript
- Based on a specific Event Propagation model
  - Capture phase, target phase, bubbling phase
  - Cancellation of events,
  - Default action



EXAMPLES OF DOM EVENTS CODE

```
<script type="application/ecmascript" >
  function doSomething(evt) { … }
</script>
<text onclick=“doSomething(evt)” >Hello World!</text>
```

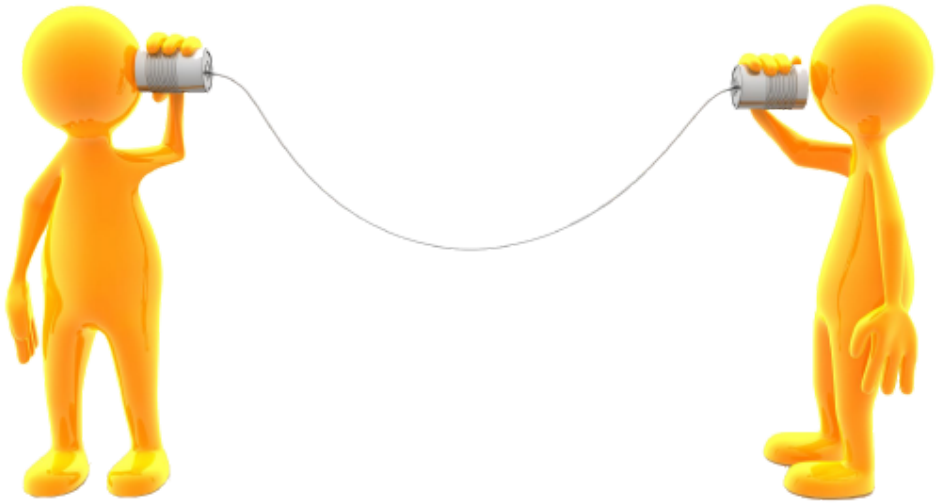
```
<script type="application/ecmascript" >
 function doSomething(evt) { … }
 e=document.getElementById(‘T’);
 e.addEventListener(‘click’, doSomething, false);
</script>
<text id=“T” >Hello World!</text>
```

```
<script type="application/ecmascript" >
 function doSomething(evt) { … }
 e=document.getElementById(‘T’);
 e.onclick=doSomething;
</script>
<text id=“T” >Hello World!</text>
```

DOM EVENT TYPES

- Mouse Events
  - click, mousedown, mouseup, mouseover, mousemove, mouseout
- Key Events
  - keypress, keyrelease
- Touch events
  - touchstart, touchend, touchleave, touchmove, ...
- Drag events
  - dragstart, dragend, ...
- Network events
  - load, error, abort, progress
- Form events
  - submit, focus ...
- Media events
  - play, pause ...

WEB PAGES CAN COMMUNICATE



WEB API FOR COMMUNICATIONS

- The `XMLHttpRequest` object
  - HTTP communication with a server
  - Core of the AJAX programming model
- The `WebSocket` Interface
  - Lightweight communication with servers
  - Companion with IETF WebSocket protocol
  - Upgrade from HTTP but different from HTTP
- Server-Sent Events
  - Used to deliver messages in push mode (notifications)
- Web Messaging
  - Messaging between Javascript contexts in the same browser (pages, workers)
- WebRTC
  - Peer-to-peer communication

AJAX "ASYNCHRONOUS JAVASCRIPT AND XML"

- Used to make asynchronous HTTP requests and retrieve data (e.g. text, XML, binary ...)
- Combined usage of different technologies
  - HTML (or SVG, ...)
  - ECMAScript
  - XML (or JSON, ...)
  - HTTP Download
- Exemples
  - HTML/SVG + XML + DOM + XMLHttpRequest
  - Flash + ActionScript + LoadVars + XML
- Benefits
  - Requests are asynchronous to the rendering
    - Avoids waiting for the response to further interact
  - Enables client-side heavy interactivity
    - Data base requests and response handling

AJAX EXAMPLE

```
var xhr = new XMLHttpRequest();
xhr.open("GET", "test.txt");
xhr.onload = function() {
  alert(this.responseText);
}
xhr.send();
```

WEBSOCKET EXAMPLE

```
var socket=new WebSocket('ws://example.com:12010/');
socket.onopen=function () {
 setInterval(function() { if (socket.bufferedAmount==0) socket.send(getUpdateData()
};
socket.onmessage=function (evt) {
 var received_msg=evt.data;
 alert("Message is received...");
};
socket.onclose=function() {
 // websocket is closed.
 alert("Connection is closed...");
};
```

WEB WORKERS

- Equivalent to threads in JavaScript (without shared memory)
- Used for long-running scripts, in background, in parallel on multi-core CPU
- Thread and Messaging across scripts

```
var worker=new Worker('worker.js');
worker.onmessage=function (event) {
document.getElementById('result').textContent=event.data;
};
var otherWorker=/* findotherworker */;
otherWorker.postMessage("A message");
```

FILE & WEB STORAGE

- Web Storage
  - Part of HTML5 Scope
  - Similar to HTTP Cookies mechanism with extensions
    - Persistent Storage of structured data at the client side
    - Cross window storage for the same site
    - Larger storage capacity than cookies

- File API
  - Handling files, directories, file systems in browsers taking security issues into account
  - API: FileReader, FileWriter