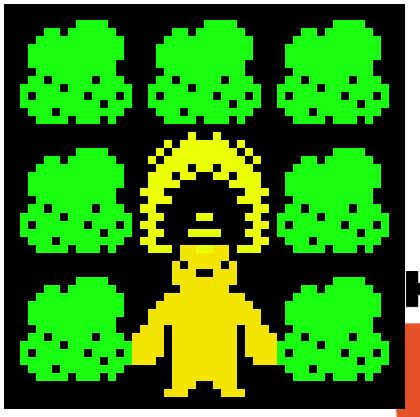
HTML5 Workshop

Mike Kasprzak
Sykhronics Entertainment
www.sykhronics.com



Agenda

- Crash course HTML5
- Hands on, try what we're talking about
  - “Show, don't tell”
- … once we get past the basics
- Need to start basic, because everything HTML builds off the basics.

What is HTML5?

- **WARNING:** HTML5 is a lot of things!
- HyperText Markup Language, version 5
- HTML4 came out ~15 years ago (1997)
- HTML5 isn't even done yet!?
- Final Spec expected in 2014

What is HTML5?

- HTML5 is an umbrella term for many standards
  - **HTML** – “Fancy” Documents. Think MS Word.
  - **CSS** – Styling or “dressing up” Documents.
  - **JavaScript** – Dynamic Documents, Interactivity.
  - And a loooong list of Extensions. 15 years worth!
    - We'll cover a few as we go along.

What is HTML5?

- HTML is about creating Documents.
- Documents are static... usually.
- HTML4 was good at making documents. Just think about the things you've done with the Internet since Y2k. That was HTML4.
- 5 years ago, “Apps” got popular.
- HTML has been evolving since HTML4.
- HTML likes to be trendy. It does apps too!
- HTML5 makes “Documents” in to “Applications”

What is HTML?

- HyperText Markup Language
  - HyperText – Text on a computer (whoa)
  - Markup Language – a way to annotate docs
    - E.g. Highlighting or Circling text
    - Crossing out mistakes
    - a Markup Language is rules like these

What is HTML?

- HTML is Fancy Documents
- HTML Documents are viewed with Web Browsers
- HTML does everything MS Word does
  - Bold, Italics, Fonts, Text Size
  - Alignment
  - Bullet Point lists (like these)

What is HTML?

- How is HTML Fancy?
- Hyperlinks!
  - Let you Navigate a document by clicking.
  - Click a chapter in a TOC, goes there instantly.
    - .. not that impressive by today's standards
- Hyperlinks let you open **OTHER** documents.
- That is the Internet.
- That's all it took. Links to other documents. World changed!

Creating HTML5 Documents

- Text Files
- **.html** or **.htm** file extensions (e.g. **index.html**)
- Can use any text editor to make them
  - Notepad, even “vi”
- Artist and Designer friendly tools are available too. Adobe Dreamweaver.
- Many tools can export, even MS Word.
- We'll use a Text Editor w/ Syntax Highlighting.
  - Don't have one? Try NotePad++, SublimeEdit.

Viewing HTML5 Documents

- Use a Web Browser!
  - Firefox
  - Chrome
  - Internet Explorer (10 only)
- Take this moment to upgrade your browser!
  - Firefox and Chrome, go to the About box
  - IE you'll have to download and reboot :(
- Everyone go install **Chrome**!

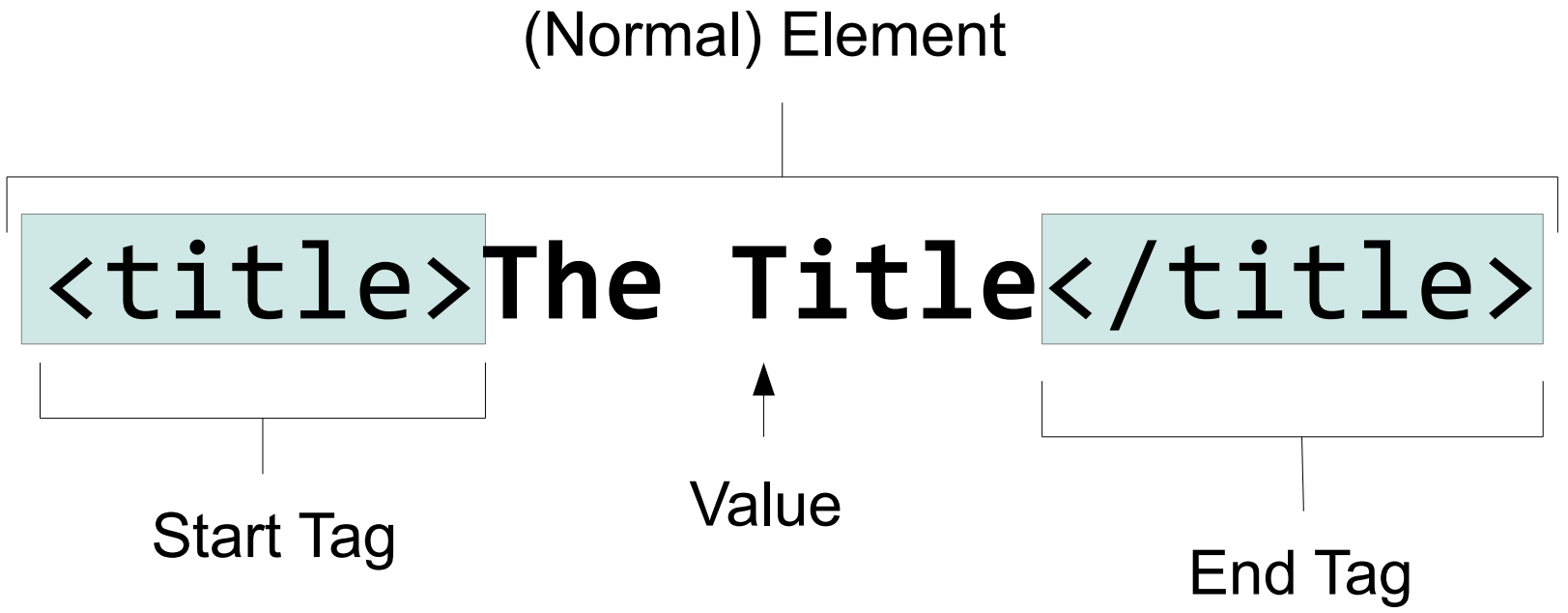
Debugging HTML5 Documents

- Chrome and IE 10 have debuggers built in. **Chrome** has *the best* debugger.
- Firefox, go download Firebug
  - Add-ons, Get Add-ons, then search for Firebug
  - It looks like a bug, how clever!
- Press **F12** in all 3 to open the debugger
- FF and IE, press **CTRL+F5** to cache refresh
- Chrome, right click on the reload button
  - May need to enable the debugger first

Simple HTML5 Document

```
<!DOCTYPE html>
<html>
 <head>
   <title>The Title</title>
 </head>
 <body>
   <p>Hello World!</p>
 </body>
</html>
```

HTML Elements



Simple HTML5 Document

```
<!DOCTYPE html>
<html>
 <head>
   <title>The Title</title>
 </head>
 <body>
   <p>Hello World!</p>
 </body>
</html>
```

Core Document HTML Tags

- **<html> </html>** is the HTML Document. An HTML document is contained within.
- **<body> </body>** is the visible content.
  - Visible content is what you see in the Browser Window (i.e. the box with the scroll bar)
- **<head> </head>** is the invisible content.
  - Content you don't want to see in the Browser Window goes here.

Spacing and Indentation

- Spacing and Indentation is for readability.
- Not like python.
- **WEIRD:** No matter how many spaces, tabs, or newlines you add, it displays as **only 1 space**.
- Yes you can get around this, but (SPOILER) you should use **CSS** instead.

Simple HTML5 Document

```
<!DOCTYPE html>
<html>
 <head>
   <title>The Title</title>
 </head>
 <body>
   <p>Hello World!</p>
 </body>
</html>
```

Body Tags

- ** ** is Paragraph.
- If you want more paragraphs, you make a 2nd paragraph, or 3rd.

  ** **Paragraph 1** **Paragraph 2** **

  ** **Paragraph 3** **
- You don't actually have to use Paragraphs, but it is a good way to structure a document.

Body Tags

- Many more tags.
- `<p>Hey, <strong>this text is bold</strong></p>`
- `<p>This text is <em>italics</em></p>`
- `<p>This <u>is underlined</u></p>`
- `<p>This has <del>a line through it</del></p>`
- `This <strong><em><del>is wow</del></em></strong>`
- `<h1>Try me instead of Paragraph</h1>`

Simple HTML5 Document

```
<!DOCTYPE html>
<html>
 <head>
   <title>The Title</title>
 </head>
 <body>
   <p>Hello World!</p>
 </body>
</html>
```

Head Tags

- Invisible Content
- **<title>**The Title**</title>** is the only exception. The title shows up in the Title Bar or Browser Tab of your Web Browser.
- Other Invisible Content goes here.
  - CSS Style Sheets
  - JavaScript Code
  - Metadata
- We'll come back to these

Simple HTML5 Document

```
<!DOCTYPE html>
<html>
 <head>
   <title>The Title</title>
 </head>
 <body>
   <p>Hello World!</p>
 </body>
</html>
```

Void Elements

```
<!DOCTYPE html>
```

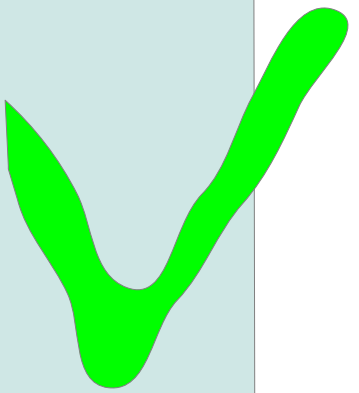
- Special Element
- Has no End Tag, and thus no “Value”
- HTML has a list of them, and unfortunately, you just have to know if they're Void or not.

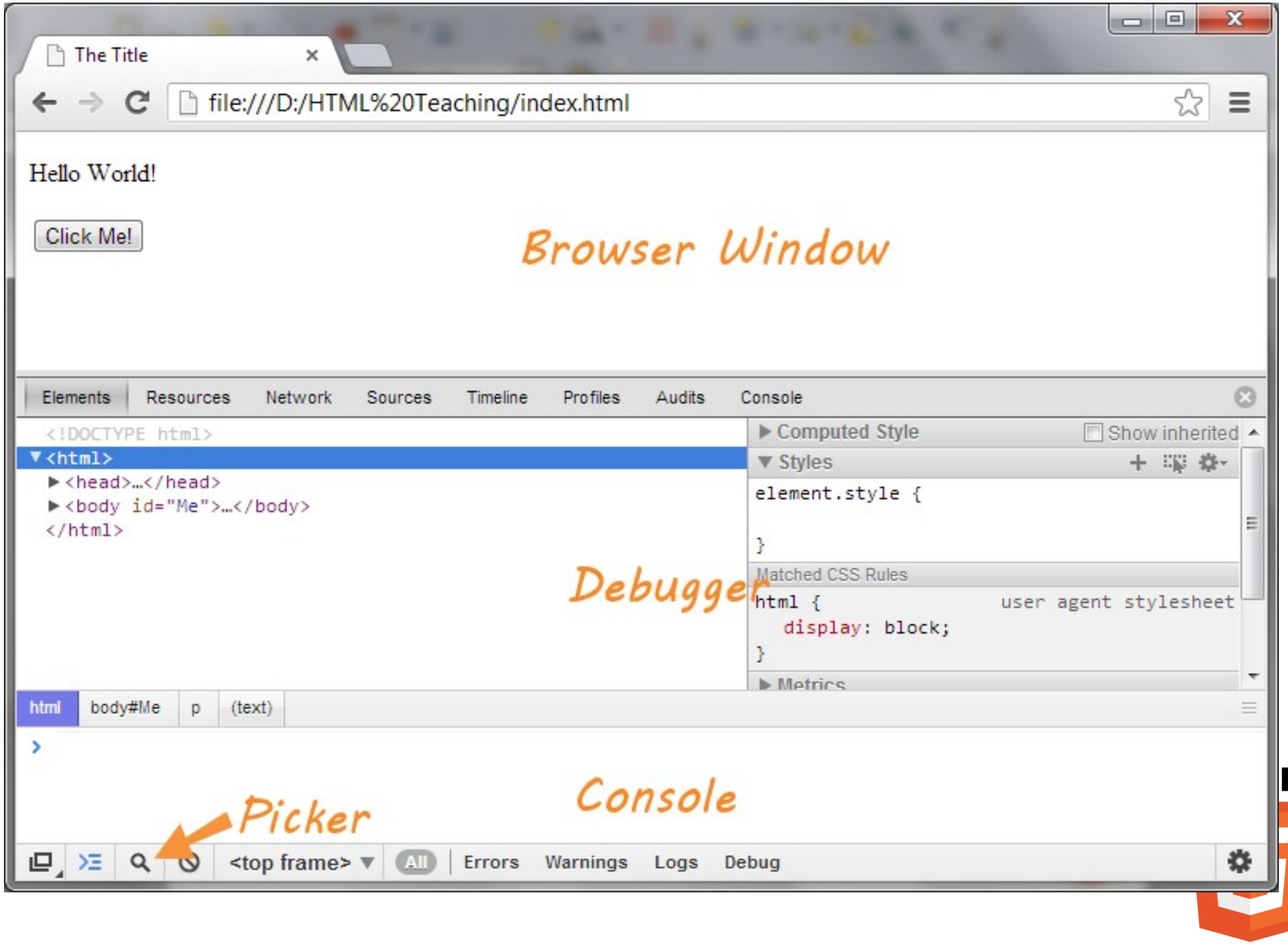
<!DOCTYPE html>

- What is <!DOCTYPE html>
- Actually a special Void Element (even specialer than normal Void Elements)
- This says a document is an HTML5 document.
- Most tags are lower case, but this is upper.
- A document beginning with <!DOCTYPE html> is saying “I am a standards compliant HTML5 document”.

Simple HTML5 Document

```
<!DOCTYPE html>
<html>
 <head>
   <title>The Title</title>
 </head>
 <body>
   <p>Hello World!</p>
 </body>
</html>
```




The Title
file:///D:/HTML%20Teaching/index.html
Hello World!
Click Me!
Browser Window
Elements
Resources
Network
Sources
Timeline
Profiles
Audits
Console
<!DOCTYPE html>
<html>
<head>…</head>
<body id="Me">…</body>
</html>
Computed Style
Show inherited
Styles
element.style {
}
Matched CSS Rules
html {
user agent stylesheet
display: block;
}
Metrics
Debugger
html
body#Me
p
(text)
Console
Picker
<top frame>
All
Errors
Warnings
Logs
Debug

Body Tags (repeat)

- Many more tags.
- `<p>Hey, <strong>`**this text is bold**`</strong></p>`
- `<p>This text is <em>`*italics*`</em></p>`
- `<p>This <u>`<u>is underlined</u>`</u></p>`
- `<p>This has <del>`~~a line through it~~`</del></p>`
- `This <strong><em><del>`~~***is wow</del>***~~`</em></strong>`
- `<h1>`**Try me instead of Paragraph**`</h1>`

Objects

- Non-text things you can insert in a document.

  <img src=”MyImage.jpeg”>

- Image is a Void Element

HTML Tag Attributes

- Parameters for Tags. It's how you specialize.
- Start Tag only (and Void tags).
- <tag attribute=value another=value>
- Surrounding values with quotes “” is optional, but recommended for strings and text.
- <!DOCTYPE html>, the **html** is an attribute.
- Some attributes can be set without a value (not many though).

Image Objects

- <img **src**=”MyImage.jpeg” **width**=500>
- **width** and **height** (in pixels)
- **src** is what file.
- If you specify just the width (or height), it'll scale the image equally about the other axis.

Other Objects

- There are more object types
- <**video**> and <**audio**> can be used to insert video and audio files. New as of HTML5.
- <**object**> can be used to insert other data.
  - Data from Plugins: Flash, Java, ... (was <embed>)
  - HTML Documents (also via <iframe>)
  - <**object** data=”myswf.swf” width=200 height=200> </**object**>
- **<canvas>** is what game developers use.
  - A surface we can render to (using JavaScript).

One More Thing

- Hyperlinks!
- Google
- "a" creates hyperlinks.
- **href** is the URL to open.
- To make an image clickable as a link, wrap it in an <a> tag.

  <img src="any.gif">

End of Easy stuff

XML

- eXtensible Markup Language

```
<?xml version="1.0" encoding="UTF-8" ?>
<database>
 <person id="1">
  <name>Daniel</name>
 </person>
 <person id="2">
  <name>Fredrick</name>
  <owner />
 </person>
</database>
```

- It looks like HTML!!

XML

- XML is not HTML
- HTML is not XML
- XML is more strict than HTML
- HTML is a collection of specific tags.
- XML is generic. Designed to be used for creating your own document types.

XML is Strict

- Void tags look different
  -
  - Space and a trailing slash before closing brace
- Attributes are always wrapped in Quotes
  - <person id=”1”>
- Attributes must have values
  - <input type=”checkbox” checked=”true”>
  - versus <input type=”checkbox” checked>
- <?xml version="1.0" encoding="UTF-8" ?>

HTML and XHTML

- XHTML is a standard that bridges XML and HTML
- Everything is the same as HTML, but must now conform to the strictness of XML...
- Except the <?xml … ?> line is optional
- Also <!DOCTYPE html> never ends with a />

XHTML

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
  </head>
  <body>
    <p>Hey <strong>dog</strong>!</p>
    <img src="woof.jpeg" />
  </body>
</html>
```

Why bring up XHTML?

- Because HTML5 is (mostly) compatible with it
  - is okay!
  - Quoted attribute values everywhere too!
  - Just don't use the weird <?xml ?> thing.
- Any code you deal with will be either:
  - HTML style (relaxed)
  - XHTML style (strict)
  - Some combination of the two

CSS goes here

- Or it would if this was a regular HTML5 course

Raw Elements

- Surprise! <!DOCTYPE> isn't a Void Element.
- <! … > is different.
  - Everything inside doesn't have to follow the rules (though in DOCTYPE's case they do)
- `<!-- This is <u>commented</u> out -->`
- `<? … ?>` used by xml and php. `<?php code ?>`
- **`<script></script>`** is a standard HTML looking Raw Element. Everything contained within doesn't follow the rules.

JavaScript

- **FACT:** JavaScript is **NOT** Java
- It has *very little* to do with Java!
- JavaScript's real name is **EcmaScript**.
- **EcmaScript** is the result of a fight (“Browser War”) between Netscape (Mozilla) and Microsoft from 1996 to 1997, part of the standardization of HTML4 (1997).
- HTML people are trendy, and at the time Java was cool, so they stole the name. Marketing.

JavaScript

- JavaScript uses a C-like syntax
  - Java does too, but so does C++, Perl, PHP, C#, …
- Virtual Machine
- Garbage Collector
- No pointers, just References
- End of Similarities to Java

Inline JavaScript Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
    <script>
      document.write("Hello from JavaScript!");
    </script>
  </head>
  <body>
    <p>Hey <strong>dog</strong>!</p>
  </body>
</html>
```

The Script Tag

- **<script> </script>** is a Raw Element
- Once you enter a **<script>**, tags and HTML don't work until you reach **</script>**
- It's a completely different language now!
- **<script>** tags can be used anywhere, even inside the **<body>**
- JavaScript code executes immediately, as soon as **<script>** code is found.
  - This might not be ideal, but we can work around it

Inline JavaScript Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
    <script>
      function Message() {
        document.write("Hello from JavaScript!");
      }
    </script>
  </head>
  <body>
    <p>Hey <strong>dog</strong>!</p>
    <button onclick="Message()">Click Me!</button>
  </body>
</html>
```

Document Object Model (DOM)

- JavaScript Objects are different than HTML Objects.
- Inside JS, there is an object called **document**
- **document** contains features and functions for manipulating the current HTML document
- **WARNING! document.write** is a destructive function that, unless used carefully, rewrites the entire document!
- However, it makes a really easy JS demo.

Commenting

- To disable code, or to add a comment use either of the following.

```
// This is a comment that ends
// at the end of the line
```

or

```
/* This comment ends here */
```

Inline JavaScript Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
    <script>
      function Message() {
        document.getElementById(“Me”).innerHTML += "Hello";
      }
    </script>
  </head>
  <body id=”Me”>
    <p>Hey you!</p>
    <button onclick="Message()">Click Me!</button>
  </body>
</html>
```

Inline JavaScript Code

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
    <script>
      function ReturnMe() {
        return document.getElementbyId(“Me”);
      }

      function Message() {
        ReturnMe().innerHTML += "Hello";
      }
    </script>
  </head>
  <body id=”Me”>
    <p>Hey you!</p>
    <button onclick="Message()">Click Me!</button>
  </body>
</html>
```

Variables

- Variables are created using the **var** keyword.

```
var MyNumber = 10; // or 3.14

var MyBoolean = true; // or false

var MyString = “Hello World”;

var MyNull = null;

var MyFunction = Message;
```

- Variable types can be changed after the fact. Not limited to just what they start as.

Global and Local Sample

```
var Global = 1;

function Message() {
  var Local = 1;
  var Msg = “”;
  Msg += Global.toString() + “, ”;
  Msg += Local.toString() + “ | ”;

  document.getElementById("Me").innerHTML += Msg;

  Global++;
  Local++;
}
```

Global and Local Sample

```
var Global = 1;

function Message() {
    var Local = 1;
    var Msg = “”;
    Msg += Global.toString() + “, ”;
    Msg += Local.toString() + “ | ”;

    document.getElementById("Me").innerHTML += Msg;

    Global++;
    Local++;
}
```

Global and Local Sample

```
var Global = 1;

function Message() {
   var Local = 1;
   var Msg = “”;
   Msg += Global.toString() + “, ”;
   Msg += Local.toString() + “ | ”;

   document.getElementById("Me").innerHTML += Msg;

   Global++;
   Local++;
}
```

Arithmetic Operators

- Numbers
  - `var Value = 10 + 2;`
  - `Value += 12; // Same as Value = Value + 12;`
  - Subtraction -, Multiplication *, Division /
  - Increment ++ (by one), Decrement -- (by one)
  - `Value++; // Same as Value = Value + 1;`
  - `Value = -Value; // Negative`
  - And more!
  - **Math** library for even more.

Global and Local Sample

```
var Global = 1;

function Message() {
  var Local = 1;
  var Msg = “”;
  Msg += Global.toString() + “, ”;
  Msg += Local.toString() + “ | ”;

  document.getElementById("Me").innerHTML += Msg;

  Global++;
  Local++;
}
```

String Operations

- Strings
  - Use **`.toString()`** to convert Numbers, Booleans, and Objects to strings.
  - `var MyText = “Hey” + “You”; // “HeyYou”`
  - `MyText += “More”; // Same as Text = Text + “More”;`
  - `var MyNumber = 5 + +“5”; // Any String to Number`
  - `var MyText = “Hey” + 5; // “Hey5”. Anything added to a string is automatically converted so it can be added.`
  - **`String`** library for more string operations.

Function Sample

```
function Out( Index, Words ) {
  var Msg = Index.toString() + " " + Words;
  document.getElementById("Me").innerHTML += Msg;
}

function Message() {
  Out( 10, "Do it" );
}
```

Conditional Sample

```
var Global = 1;

function Message() {
  var Local = 4;
  var Msg = “No ”;
  if ( Global == Local ) {
    Msg = “Yes “;
  }

  document.getElementById("Me").innerHTML += Msg;

  Global++;
}
```

Conditional Sample

```
var Global = 1;

function Message() {
  var Local = 4;
  var Msg = “No ”;
  if ( Global == Local ) {
    Msg = “Yes “;
  }

  document.getElementById("Me").innerHTML += Msg;

  Global++;
}
```

Boolean Operations

- "**if**" statements do { } code if true
- **A == B** returns a boolean **true** or **false**

```
var MyBoolean = A == B;
```

- = is for setting variables, == is for checking equality
- A > B (greater than), A < B (less than), A >= B (greater than or equal to), A <= B (yep)
- A != B (not equal to)
- MyBoolean = !MyBoolean; // the opposite

Boolean Operations

- You can say:

```
var MyBoolean = true;
if ( MyBoolean ) {
  // Do Something //
}
```

- This is common.

For Loop Sample

```
function Message() {
  var Msg = "";

  for ( var Loop = 0; Loop < 4; Loop++ ) {
    Msg += Loop.toString() + " ";
  }

  document.getElementById("Me").innerHTML += Msg;
}
```

For Loop Sample

```
function Message() {
  var Msg = "";

  for ( var Loop = 0; Loop < 4; Loop++ ) {
    Msg += Loop.toString() + " ";
  }

  document.getElementById("Me").innerHTML += Msg;
}
```

Boolean

While Loop Sample

```
function Message() {
  var Msg = "";

  {
    var Loop = 0; // FOR //
    while ( Loop < 4 ) { // FOR //
      Msg += Loop.toString() + " ";
      Loop++; // FOR //
    }
  }

  document.getElementById("Me").innerHTML += Msg;
}
```

While Loops

- “while” works similar to if.
- While the condition is true, the code within { } repeats.
- Before the code starts, the condition is checked
- To run a loop before checking, use a do-while.

```
do {
   // This will be run before checking //
} while( Loop < 4 );
```

Arrays

```
var MyArray = [];
```

- Creates an Empty Array

```
MyArray.push("Element Zero");
MyArray.push("Element One");
```

- Add two elements to the array.

```
var FirstValue = MyArray[0];
var TheLength = MyArray.length;
```

Array Loop Sample

```
var Text = ["Hello", "this", "IS", "text."];

function Message() {
  var Msg = "";

  for ( var Loop = 0; Loop < Text.length; Loop++ ) {
    Msg += Text[Loop] + " ";
  }

  document.getElementById("Me").innerHTML += Msg;
}
```

Objects

```
var MyObject = {};
```

- Name:Slot (Key/Value)

```
var MyObj2 = {
   Name:”Frank”,
   Color:”Green”,
   Age:14,
   Friendly:true
};

document.write( MyObj2.Name );

document.write( MyObj2[“Color”] );
```

Object Loop

```
var Info = {
  Name:"Theodore",
  Age:14,
  Nice:true
};

function Message() {
  var Out = document.getElementById("Me");

  for ( var Value in Info ) {
    var Msg = "<p>";
    Msg += Value + " = " + Info[Value];
    Msg += "</p>";

    Out.innerHTML += Msg;
  }
}
```

Combining Arrays and Objects

```
var Database = {
  “People”:[
    {
      "Name":"Jonathan",
      "Favorites":[ 1 ]
    },
    {
      "Name":"Walrus",
      "Favorites":[ 2, 7, 18 ]
    },
    {
      "Name":"MrCool",
      "Favorites":[]
    }
  ]
};
```

JSON

```
{
  “People”:[
    {
      "Name":"Jonathan",
      "Favorites":[ 1 ]
    },
    {
      "Name":"Walrus",
      "Favorites":[ 2, 7, 18 ]
    },
    {
      "Name":"MrCool",
      "Favorites":[]
    }
  ]
}
```

JSON

- JavaScript Object Notation
- Can take any JSON file and copy+paste it in to JavaScript code.
- JSON is slightly stricter than Objects and Arrays in JavaScript.
  - All names and strings are in “quotes”
  - All “**Name:Slot**” pairs and Array Elements must have a value.
  - No semicolons

Functions and Objects

```
var Info = {
  "Name":"Theodore",  "Age":14,  “Nice”:true
};

function InfoMessage() {
  var Msg = "<p>";
  for ( var Value in Info ) {
    Msg += typeof(Info[Value]) + " $$$ ";
    Msg += Value + " = " + Info[Value];
    Msg += "<br>";
  }
  Msg += "</p>";
  return Msg;
}

function Message() {
  var Out = document.getElementById("Me");
  Out.innerHTML += InfoMessage();
}
```

Adding Functions to Objects

```
var Info = {
  "Name":"Theodore",  "Age":14,  “Nice”:true
};

Info.Message = function() {
  var Msg = "<p>";
  for ( var Value in Info ) {
    Msg += typeof(Info[Value]) + " $$$ ";
    Msg += Value + " = " + Info[Value];
    Msg += "<br>";
  }
  Msg += "</p>";
  return Msg;
}

function Message() {
  var Out = document.getElementById("Me");
  Out.innerHTML += Info.Message();
}
```

Adding Functions to Objects: this

```
var Info = {
  "Name":"Theodore",  "Age":14,  “Nice”:true
};

Info.Message = function() {
  var Msg = "<p>";
  for ( var Value in this ) {
    Msg += typeof(this[Value]) + " $$$ ";
    Msg += Value + " = " + this[Value];
    Msg += "<br>";
  }
  Msg += "</p>";
  return Msg;
}

function Message() {
  var Out = document.getElementById("Me");
  Out.innerHTML += Info.Message();
}
```

Objects as Classes: Constructor

```
function Info() {
  this.Name = "Theodore";
  this.Age = 10 + Math.floor(Math.random()*10);
  this.Nice = true;
  this.Message = function() { return this.Age; };
}

var MyInfo = new Info();

function Message() {
  var Out = document.getElementById("Me");
  Out.innerHTML += MyInfo.Message();
}
```

Objects as Classes: Constructor

```
function Info() {
  this.Name = "Theodore";
  this.Age = 10 + Math.floor(Math.random()*10);
  this.Nice = true;
  this.Message = function() { return this.Age; };
}

var MyInfo = new Info();

function Message() {
  var Out = document.getElementById("Me");
  Out.innerHTML += MyInfo.Message();
}
```

new

- `var MyArray = new Array(10); // 10 elements`
- `var MyObj = new Object(); // Same as = {}`
- `var MyNumber = new Number(5); // same as = 5`
- `var MyString = new String(“”); // same as = “”`
- `var MyDate = new Date(“January 13th, 2013”);`

Numbers

- All Numbers in JavaScript are floating point
- Double precision floating point
- No concept of Integers

  ```
  var MyNumber = 5 / 2; // 2.5
  ```

- If you want it to be “Integer Like”, use Floor.

  ```
  var MyNumber = Math.floor( 5 / 2 ); // 2
  ```

JavaScript Libraries

- Math
  - Math.random() // Random number between 0 and 1
  - Math.floor(a) // Remove decimal
  - Math.round(a) // Round up or down
  - Math.min(a,b) // Smallest Number
  - Math.max(a,b) // Largest Number
  - Math.sqrt(9) // Square Root
  - Math.PI
  - Math.abs(a) // Positive Number
  - Math.sin(a) // Sine

JavaScript Libraries

- String
  - var MyString = “Hello World”;
  - alert(MyString.substr(6,5)); // “World”
  - alert(MyString.toLowerCase());
  - alert(MyString.toUpperCase());
  - if (MyString.search(“He”) != -1) { … }
  - MyString.replace(“World”,”Worm”);

Good References

- **w3schools.com** – Most Google searches lead here. Good Language documentation.
- **developer.mozilla.org** – Mozilla has comprehensive HTML5 documentation. Where to learn about Canvas, WebSockets, etc.
- **Wikipedia.com** – More “to the point” than some references. Other references are good for details.
- **StackOverflow.com** – Have a question? Somebody probably already asked it.

3rd Party JavaScript Libraries

- JQuery
  - The most popular JavaScript Libarary
  - Provides streamlined ways to do almost everything document.
    - Some are simpler too, but to make the most of JQuery you do need to understand much of what we learned
  - Visit jquery.com/download/
  - Download the uncompressed, development Jquery
  - Save alongside your **.html** file

Document with JQuery

- document element finding replaced by `$("pattern")`

```
document.getElementById("Me").innerHTML
$("#Me").html() // Get
$("#Me").html( "Hello" ) // Set
document.getElementsByTagName("body")[0].innerHTML
$("body")[0].html() // Get
$("body")[0].html( "World" ) // Set
```

- Sorry, we had to do it the long way to appreciate JQuery.

JQuery Simple Sample

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
    <script src="jquery-1.9.1.js"></script>
    <script>
      function Message() {
        var Me = $("#Me");
        Me.html( Me.html() + "Hello!" );
      }
    </script>
  </head>
  <body id="Me">
    <p>Hey Wurld!</p>
    <button onclick="Message()">Click Me!</button>
  </body>
</html>
```

JSON: syk-country.appspot.com


```
{
  "CountryCode" : "CA",
  "IP" : "69.165.123.142",
  "Latitude" : 43.856099999999998,
  "Longitude" : -79.337018999999998
}
```


JSONP: syk-country.appspot.com

```
MyFunc({
  "CountryCode" : "CA",
  "IP" : "69.165.123.142",
  "Latitude" : 43.856099999999998,
  "Longitude" : -79.337018999999998
});
```

- We do this to work around JavaScript's typical cross domain resource sharing limitations.
- i.e. Relative Files Only (but not on our HDD)

JSONP Sample

```
<!DOCTYPE html>
<html>
  <head>
    <title>The Title</title>
    <script src="jquery-1.9.1.js"></script>
    <script>
      function Out( data ) {
        var Msg = "";
        for ( var Value in data ) {
          Msg += Value + " = " + data[Value] + "\n";
        }
        alert( Msg );
      }
      function Message() {
        $.getJSON("http://syk-country.appspot.com/?jsonp=?", Out );
      }
    </script>
  </head>
  <body id="Me">
    <p>Hey Wurld!</p>
    <button onclick="Message()">Click Me!</button>
  </body>
</html>
```

http://query.yahooapis.com/v1/public/yql?q=select%20item%20from%20weather.forecast%20where%20location%3D%2248907%22&format=json

{ "query" : { "count" : 1,
"created" : "2013-03-06T06:28:37Z",
"lang" : "en-US",
"results" : { "channel" : { "item" : { "condition" : { "code" : "26",
"date" : "Tue, 05 Mar 2013 11:52 pm EST", "temp" : "27", "text" : "Cloudy"
},
"description" : "\n
\nCurrent Conditions:
\nCloudy, 27 F
\n
Forecast:
\nTue - Few Snow Showers. High: 33 Low: 26
\nWed - Flurries. High: 38 Low: 28
\n
\nFull Forecast at Yahoo! Weather

\n(provided by The Weather Channel)
\n",
"forecast" : [{ "code" : "14",
"date" : "5 Mar 2013", "day" : "Tue", "high" : "33", "low" : "26", "text" : "Few Snow Showers"
},
{ "code" : "13",
"date" : "6 Mar 2013", "day" : "Wed", "high" : "38", "low" : "28", "text" : "Flurries"
}
],
"guid" : { "content" : "USMI0477_2013_03_06_7_00_EST", "isPermaLink" : "false"
},
"lat" : "42.73",
"link" : "http://us.rd.yahoo.com/dailynews/rss/weather/Lansing__MI/*http://weather.yahoo.com/forecast/USMI0477_f.html",
"long" : "-84.56",
"pubDate" : "Tue, 05 Mar 2013 11:52 pm EST", "title" : "Conditions for Lansing, MI at 11:52 pm EST"
} } }
} }

The End

Mike Kasprzak
Sykhronics Entertainment
www.sykhronics.com