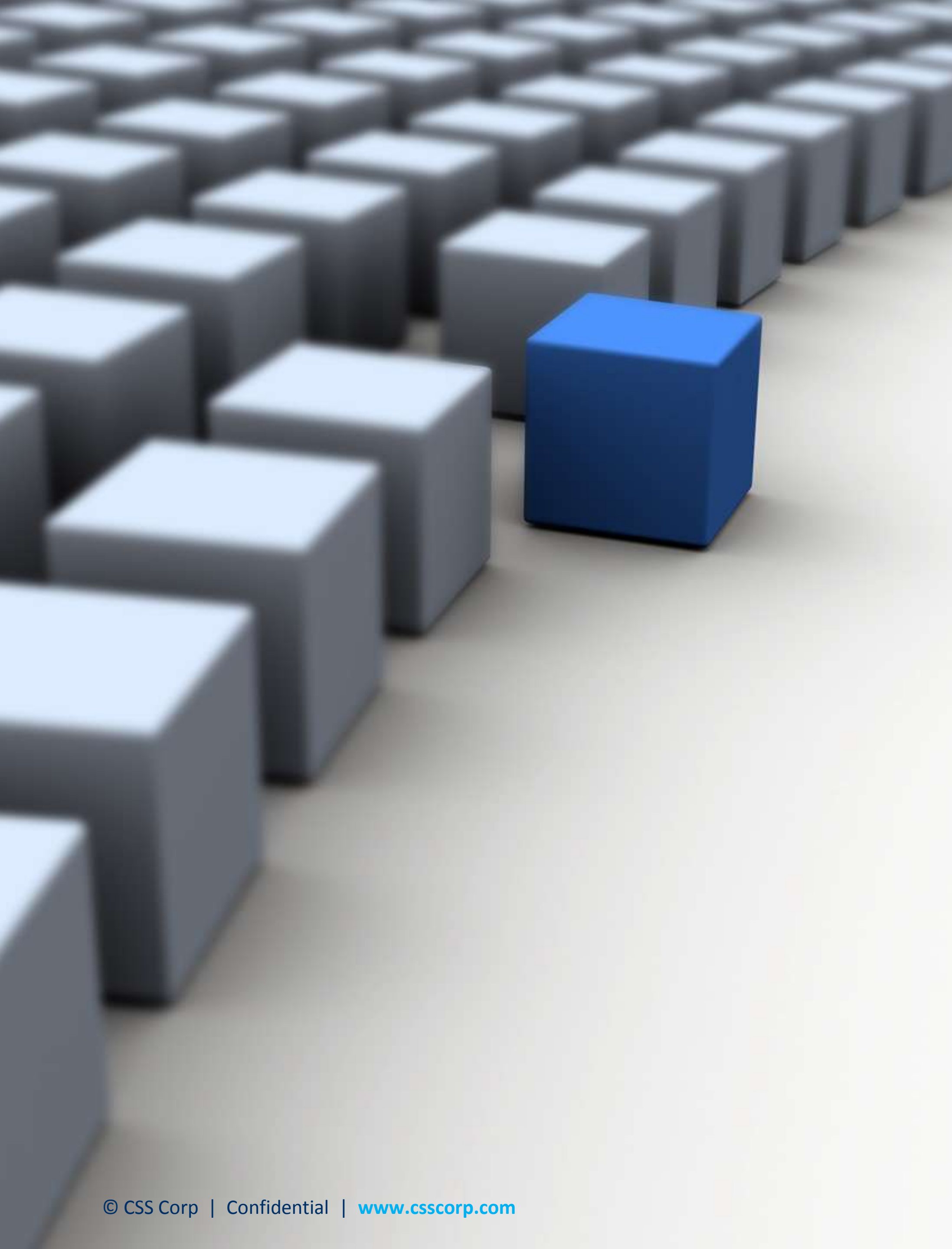


HTML5

Agenda

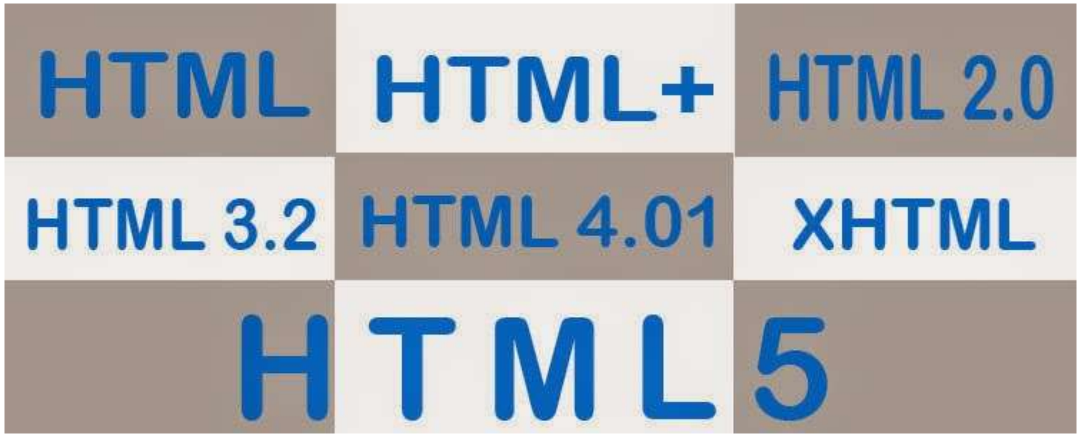
- What is HTML?
- HTML tags
  - Table
  - Form
  - List
- What is CSS?
- Why CSS?
- CSS3!!!
- HTML5 – New features!
  - Semantics
  - Form tags
  - Media
  - SVG
  - Geo Location
  - Web Storage

What is HTML?

- HTML = **Hypertext Markup Language**
- Language of The **Browser**

- HTML is not a programming language
- The **World Wide Web Consortium** creates the standards of HTML
- The tags **describe** document content

HTML Versions



HTML Uses Tag Elements

<p></p> - Paragraph

<h1></h1> - Heading

<i></i> - Italic

<b></b> - Bold

Content is wrapped in an open and closing tag

<h1>This is a heading**</h1>**

<p>This is a paragraph**</p>**

HTML Document Layout

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
        <head>
                <title>Page Title</title>
        </head>
        <body>
                <p>Hello World!</p>
        </body>
</html>
```

Block Elements

Block Level Element

- Creates a large block of content
- New lines before and after element
- Consumes the whole width available

Examples

- **<p>** - Paragraph
- **<h1>** - **<h6>** Headings
- **<form>** - Forms
- **<div>** - div tags

Inline Elements

Inline Level Element

- No new lines
- Can be placed aside other elements
- Can not define width

Examples

- **<a>** - Links
- **<strong>** and **<b>** - Bold
- **<input />** - Input
- **<span>** - Span tags

Element Attributes

- Most attributes are **Name-Value pairs** separated with an “=“
- Attributes provide **additional information** about an element
- Attributes are always specified in **the start tag**

“href” is the name of the attribute of the **<a> (link)** tag

$\downarrow$

```
<a href=“http://google.com”>Google</a>
```

$\uparrow$

“http://google.com” is the value of the **“href”** attribute of the <a> (link)tag

Other Common Attributes

style - Styling can be done via “style” attribute

<p style=“color:red”>This paragraph will be red</p>

Id & class – Specifies identification to an element

<p id=“myparagraph”>This paragraph has an id</p>

title – Adds extra info about the element. Also displayed in a tooltip in some browsers

<a href=“http://somesite.com” title=“Click to goto somesite”> This paragraph has an id</a>

Singleton Tags

- **Singleton tags** are tags with **no closing tag**. Also called **void elements**
- **<img>** is a singleton tag : <img></img> is wrong. Only the opening <img> is needed
- Used with **“attribute only”** tags such as <img>
- Can use a trailing slash **(<br />)**

No Trailing Slash	With Trailing Slash
<br>	<br />
<hr>	<hr />
<img>	<img />
<command>	<command />

You can leave off the trailing slash with HTML/HTML5. The trailing slash **is required for XHTML**

Using Images In HTML

- Images are defined in html by the **<img>** tag
- The **<img>** tag is empty aside from its attributes
- There is no closing tag

```
<img src="http://www.somesite.com/images/imagename.jpg"
alt="A name for my image" />
```

alt is the "alternate" attribute
And is required

Notice, no closing tag,
Just an extra "/"

src is the image tag <img>
source (location) attribute

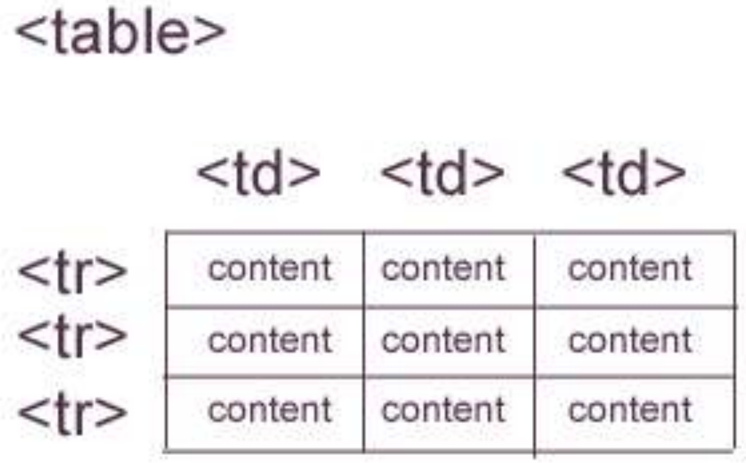
HTML Tables

What Are Tables Used For?

- Displaying tabular data
- Compare & Contrast
- Design & layout of a webpage (Bad Practice)

Simple Table Syntax

```
<table>
<tr>
<td>Content</td>
<td>Content</td>
<td>Content</td>
</tr>
<tr>
<td>Content</td>
<td>Content</td>
<td>Content</td>
</tr>
<tr>
<td>Content</td>
<td>Content</td>
<td>Content</td>
</tr>
</table>
```



content	content	content
content	content	content
content	content	content

HTML Lists

What Are Lists Used For?

- Group together related pieces of information
- Ranking Data
- Design & Layout Formatting

List Syntax

Unordered List

```
<ul>
<li>Content</li>
<li>Content</li>
<li>Content</li>
</ul>
```

- Content
- Content
- Content

Ordered List

```
<ol>
<li>Content</li>
<li>Content</li>
<li>Content</li>
</ol>
```

1. Content
2. Content
3. Content

Web Forms 101

- HTML web forms **let websites become interactive** by allowing user input
- Forms have front-end and back-end processing. **HTML takes care of the front end** and **another language takes care of the back-end** such as PHP or ASP
- HTML can create forms but HTML alone is not enough to process forms
- Forms usually include a series of html input tags and other elements

Simple Form Syntax

```
<form>
        Name: <input type="text>
         Email: <input type="text>
        Message:<textarea></textarea>
</form>
```

Common HTML tags found in forms:

<form>	<input>
<select>	<option>
<textarea>	<label>
<button>	<fieldset>

<form> Tag

The <form> tag wraps around the entire form

Common Attributes:

- **action:** This specifies the page or script used to process the form on the server side
- **method:** How the form gets processed (get, post)

<input> Tag

The <input> specifies an input field where a user can enter data.

Common Attributes:

- **type:** This specifies the type of input (text, password, etc
- **name:** Specifies a name for the element. Used in server-side and/or client-side processing
- **size:** Specifies the width in characters
- **max:** The maximum amount of characters

<textarea> Tag

The <textarea> tag is for large blocks of text

Common Attributes:

- **name:** Specifies a name for the element. Used in server-side and/or client-side processing
- **rows:** Specifies the visible number of lines
- **cols:** Specifies the visible width
- **maxlength:** Specifies the maximum number of characters allowed in the text area

<select> Tag

The <select> tag is used for dropdowns (single and multiple select

Common Attributes:

- **name:** Specifies a name for the element. Used in server-side and/or client-side processing
- **multiple:** Specifies if the list field can contain multiple values
- **size:** Specifies the maximum number of visible options

<label> Tag

The <label> tag is used to associate a fields heading with the input field

Common Attributes:

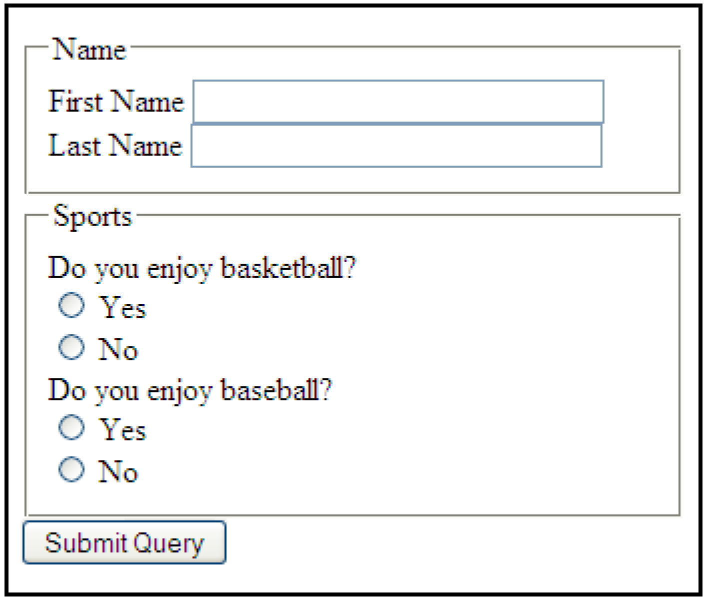
- **for:** Specifies which form element the label is bound to

<fieldset> Tag

The <fieldset> tag is used to group together certain fields. It draws a box around the elements

Common Attributes:

- name
- form
- disabled



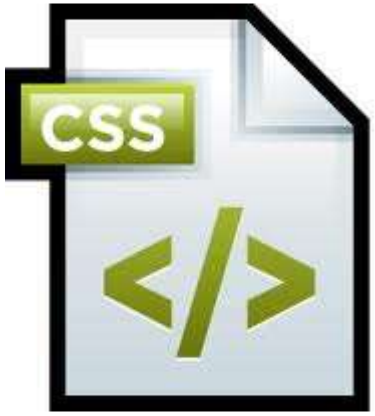
Name
First Name
Last Name
Sports
Do you enjoy basketball?
○ Yes
○ No
Do you enjoy baseball?
○ Yes
○ No
Submit Query

What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

Cascading Style Sheets

CSS is a **“stylesheet” language** that is used to describe the presentation semantics of a document that is written in a **“markup” language** such as **HTML**

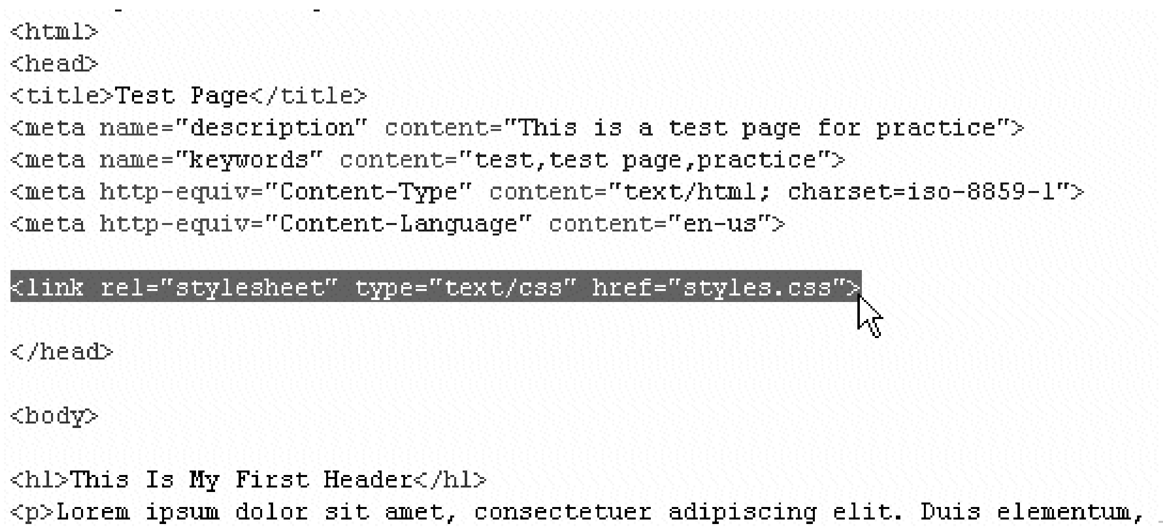


Linking An External Stylesheet

The following syntax is used to link a stylesheet to an html document

<**link** rel="stylesheet" href="path/to/file.css" type=“text/css”>

- This must be placed in the <head> of the HTML document
- The attributes can be in any order
- The “type” attribute is not required



```
<html>
<head>
<title>Test Page</title>
<meta name="description" content="This is a test page for practice">
<meta name="keywords" content="test,test page,practice">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv="Content-Language" content="en-us">

<link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

<h1>This Is My First Header</h1>
<p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis elementum,
```

<div> and <span> Elements

<span> and <**div>** elements are used for generic organizational or stylistic applications

- **No specific meaning**
- **<div> is block-level**
- **<span> is inline-level**

Id and **class** attributes

<div> and <span> most commonly are used with class or is attributes. Syntax for this is…

```
<div id="header" class="space anotherclass">
        <!—your code for the header div→
</div>
```

CSS Syntax

From the html file we define our id and class…

```
<div id="header" class="space anotherclass">
        <!—your code for the header div→
</div>
```

From the css we define the styling for the id and class…

```
#header{
        width:800px;
        height: 100px;
        background:#000;
}
```

```
.space{
        margin-bottom:10px;
}
```

Why CSS?

- HTML was intended to define the content of a document
- <font> and color attributes were added to the HTML 3.2 specification
- Development of large web sites became a long and expensive process
- Styles Solved this Big Problem
- In HTML 4.0, all formatting could be stored in a separate CSS file.

CSS defines HOW HTML elements are to be displayed

CSS3?

- CSS3 is the latest standard for CSS.
- CSS3 is completely backwards-compatible with earlier versions of CSS.

HTML5!!!!

What is HTML5?

- HTML5 is the new standard
- HTML5 is a cooperation between the W3C and the Web Hypertext Application Technology Working Group (WHATWG).
- HTML5 offers some great new features that will make the web more dynamic

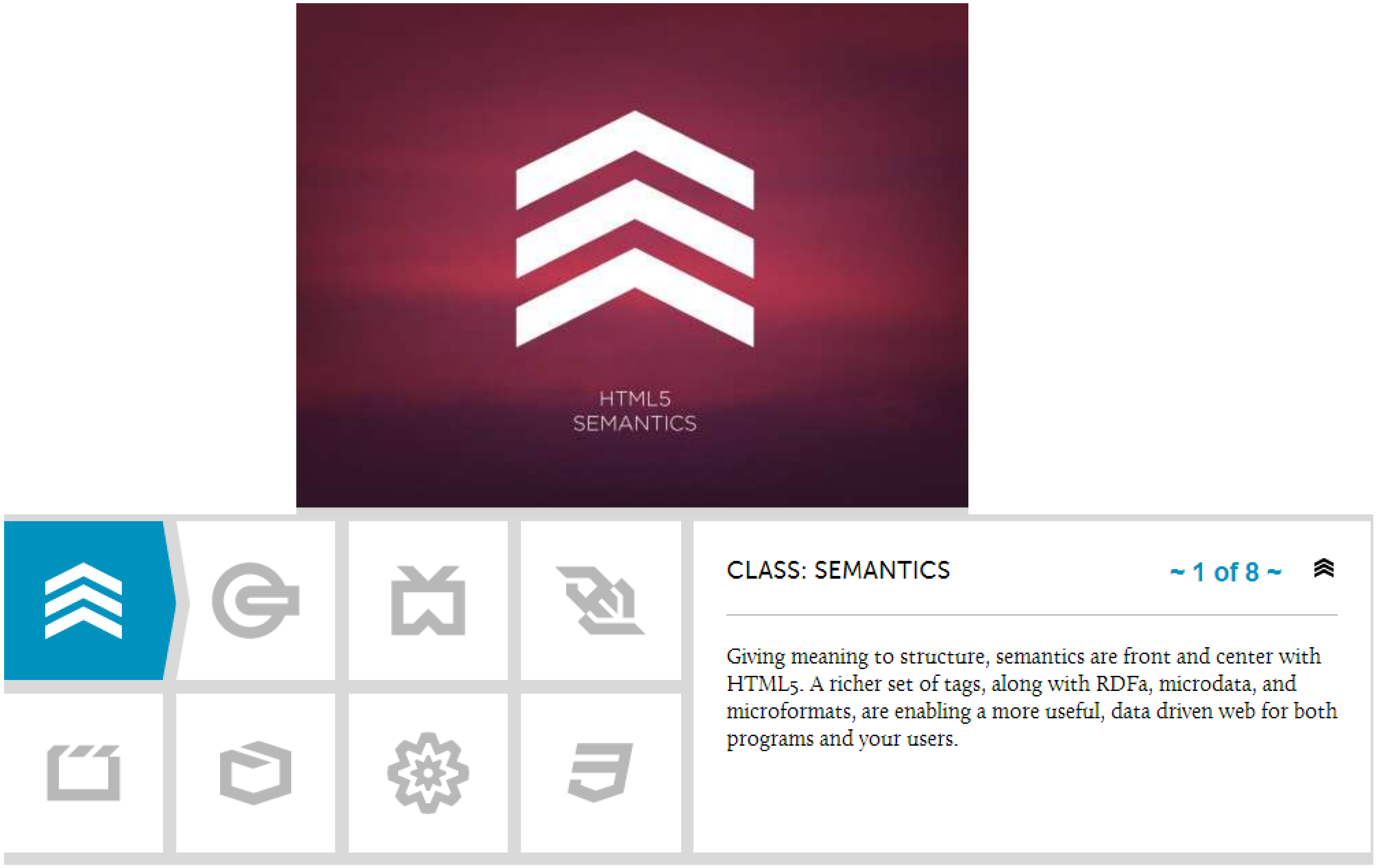
New Features

- New sets of tags such as <header> ,<aside> and <section>
- <canvas> element for 2d drawing
- Local storage
- New form controls like calendar, date and time
- New media functionality
- Geolocation

HTML5 Browser Support

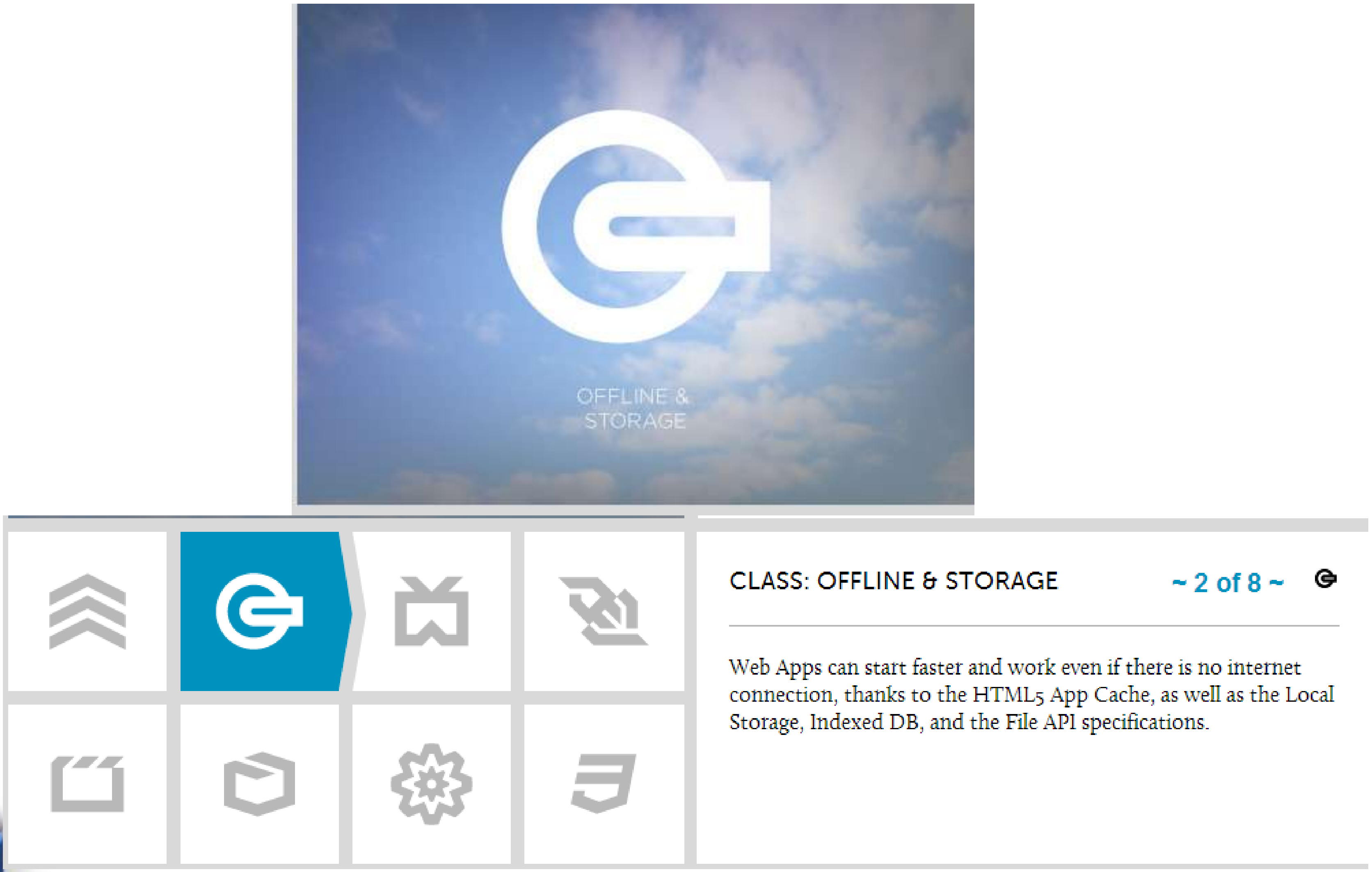
HTML5 : Is not an official standard yet and not all browsers support HTML5 or at least some of the features of HTML5

New Features



CLASS: SEMANTICS ~ 1 of 8 ~

Giving meaning to structure, semantics are front and center with HTML5. A richer set of tags, along with RDFa, microdata, and microformats, are enabling a more useful, data driven web for both programs and your users.



CLASS: OFFLINE & STORAGE ~ 2 of 8 ~

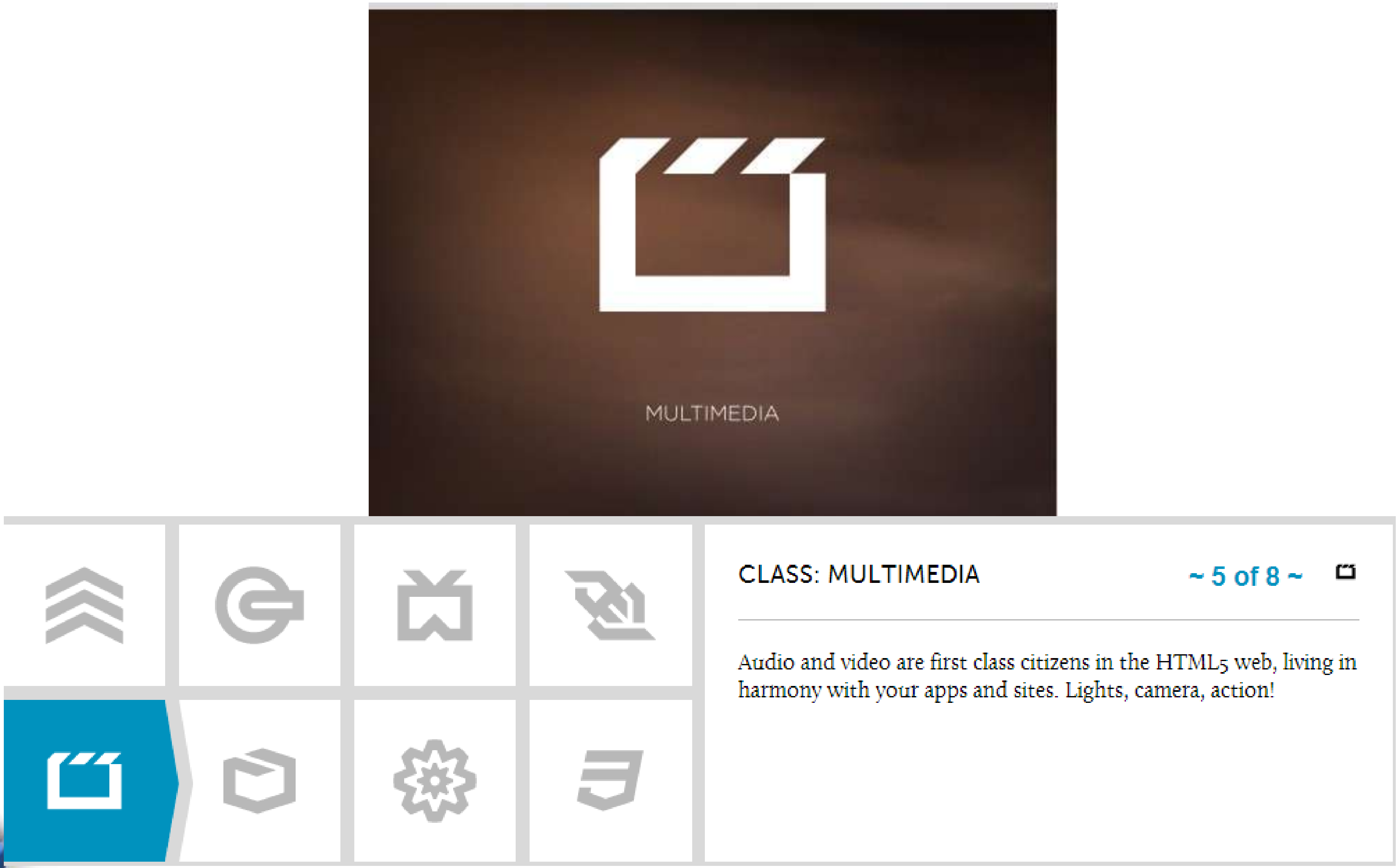
Web Apps can start faster and work even if there is no internet connection, thanks to the HTML5 App Cache, as well as the Local Storage, Indexed DB, and the File API specifications.

CLASS: DEVICE ACCESS ~ 3 of 8 ~

Beginning with the Geolocation API, Web Applications can present rich, device-aware features and experiences. Incredible device access innovations are being developed and implemented, from audio/video input access to microphones and cameras, to local data such as contacts & events, and even tilt orientation.

CLASS: CONNECTIVITY ~ 4 of 8 ~

More efficient connectivity means more real-time chats, faster games, and better communication. Web Sockets and Server-Sent Events are pushing (pun intended) data between client and server more efficiently than ever before.

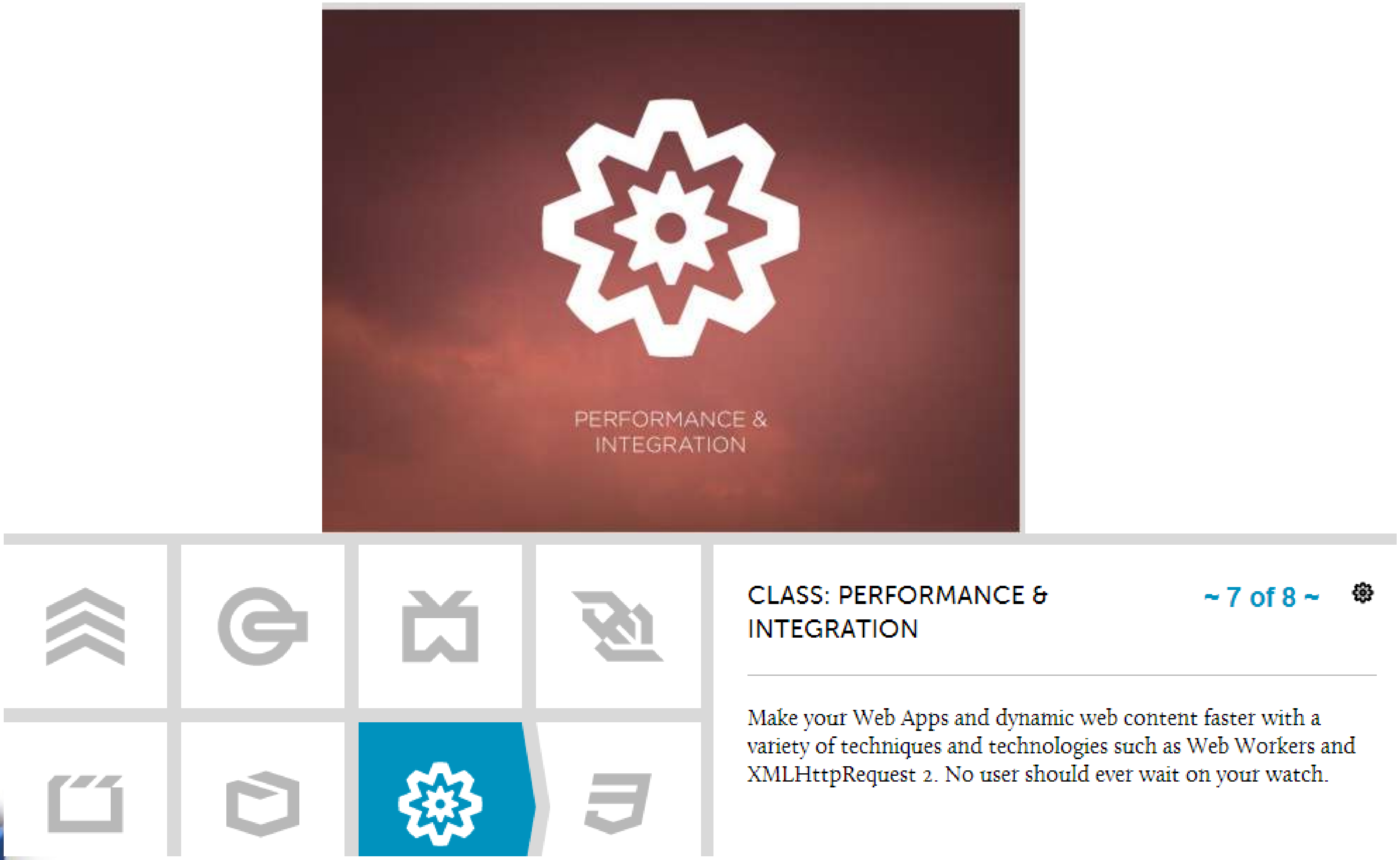


CLASS: MULTIMEDIA

Audio and video are first class citizens in the HTML5 web, living in harmony with your apps and sites. Lights, camera, action!

CLASS: 3D, GRAPHICS & EFFECTS ~ 6 of 8 ~

Between SVG, Canvas, WebGL, and CSS3 3D features, you're sure to amaze your users with stunning visuals natively rendered in the browser.



CLASS: PERFORMANCE & INTEGRATION

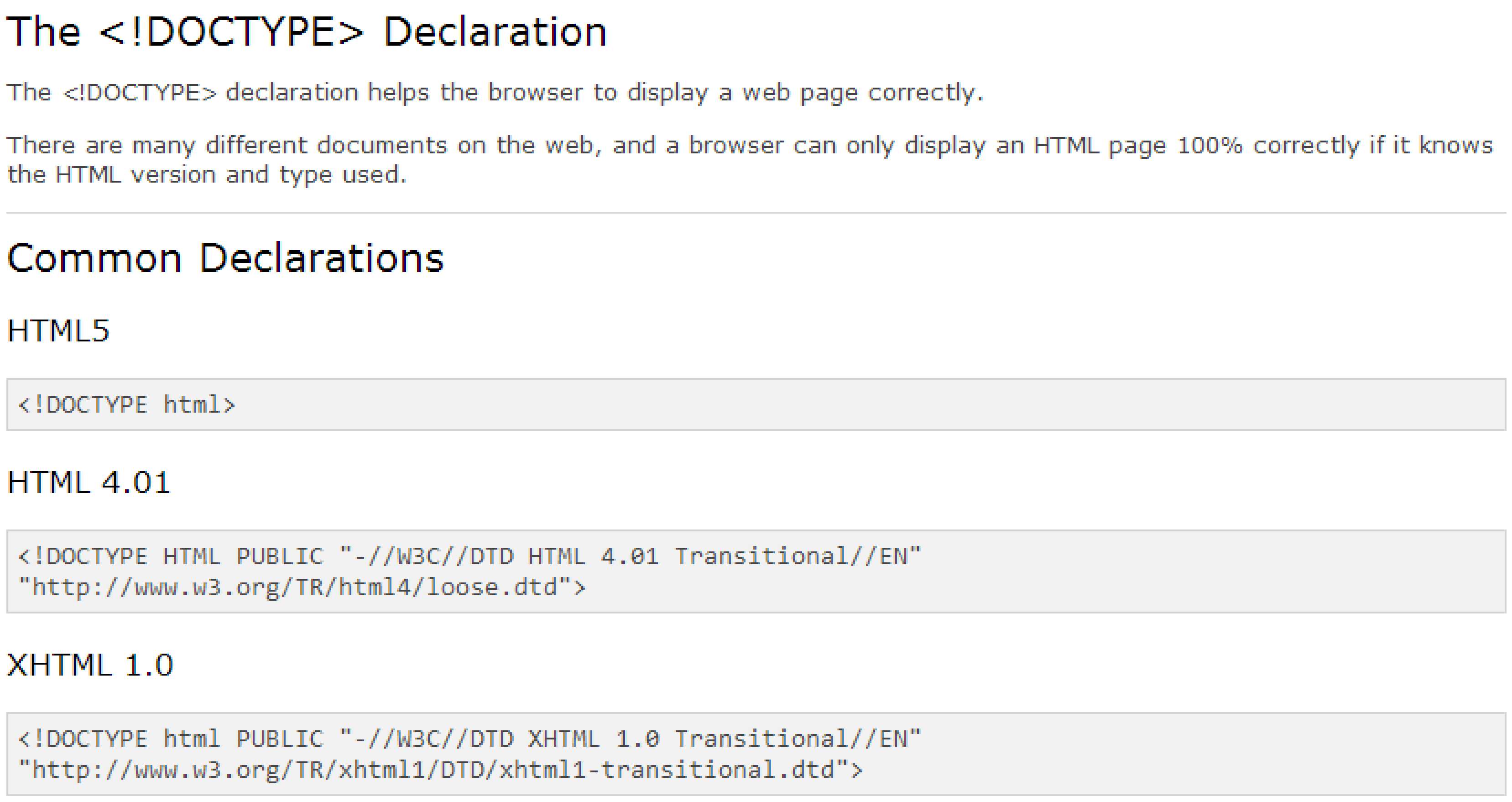
~ 7 of 8 ~

Make your Web Apps and dynamic web content faster with a variety of techniques and technologies such as Web Workers and XMLHttpRequest 2. No user should ever wait on your watch.

CLASS: CSS3 ~ 8 of 8 ~

CSS3 delivers a wide range of stylization and effects, enhancing the web app without sacrificing your semantic structure or performance. Additionally Web Open Font Format (WOFF) provides typographic flexibility and control far beyond anything the web has offered before.

HTML Vs XHTML vs HTML5



The <!DOCTYPE> Declaration

The <!DOCTYPE> declaration helps the browser to display a web page correctly.

There are many different documents on the web, and a browser can only display an HTML page 100% correctly if it knows the HTML version and type used.

Common Declarations

HTML5

```
<!DOCTYPE html>
```

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Your Old HTML4 Code Is Fine

- You don’t need to throw away or re-do your old webpages
- The base layout for an html document is the exact same

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

Expanding HTML

One of the biggest reasons for HTML5 is to prevent users from needing to download and install multiple plugins such as Flash and Silverlight

HTML5 Stack:

- HTML5
- CSS3
- Javascript

Some say HTML5 stack will replace flash

Separate & Style Your HTML

- You can style your html with the **“style”** attribute or by using CSS(Cascading Style Sheets). CSS is preferred.
- Use the **<div>** or **<span>** tag to single out blocks of html to apply style to

```
<div style=“padding:5px;background-color:black;color:white;”>
        <h1>Your Heading</h1>
        <p>This is your paragraph</p>
</div>
```

Style Attribute Selectors

The syntax…

<p style=“**selector:value;**”>Some text</p>

Common Selectors and values…

color:red;
background-color:blue;
background-image:some-image.jpg
padding:5px
margin:5px
display:block;
border-style:solid;
border-color:black
border-width:1px;

Shorthand way…

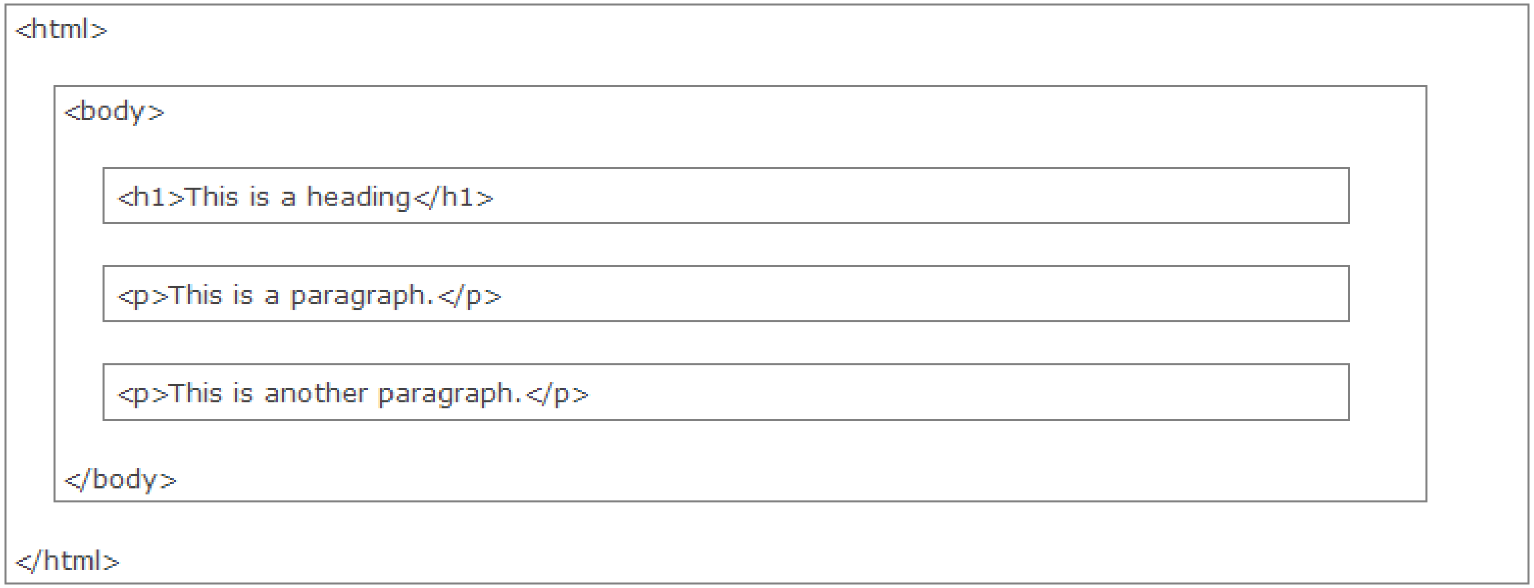
border:solid black 1px;

HTML5 new features

- Semantics
- New Input Types
- New Tags
- New List Elements

HTML5 Semantic Tags

HTML Page Structure



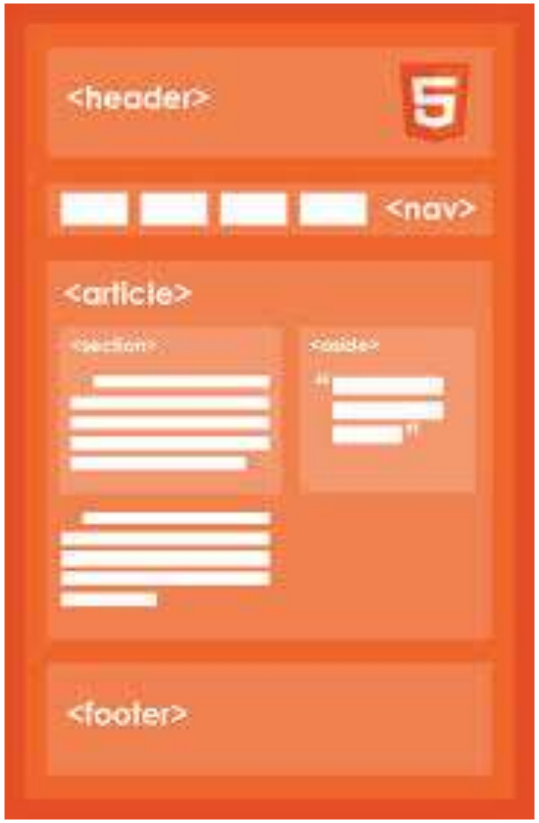
```
<html>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

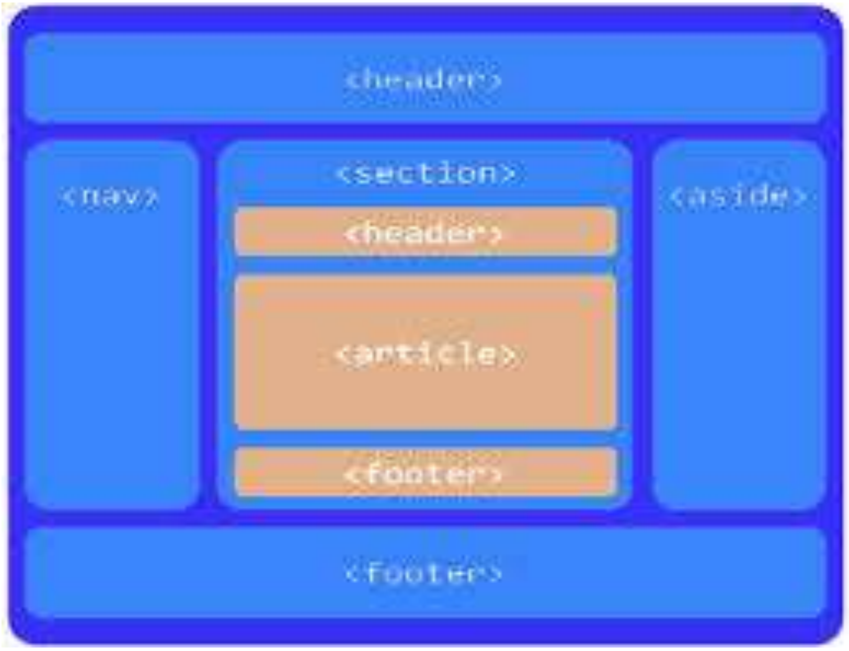
Semantic Tags

- Describes its meaning to both the browser and the developer
  - Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.
  - Examples of **semantic** elements: <article>, <footer>, and <aside> - Clearly defines its content.

- <div id="nav">
- <div class="header">
- <div id="footer">

PageLayout





Semantic tags

- HTML5 offers new semantic elements to clearly define different parts of a web page:

  <header>

  <nav>

  <section>

  <article>

  <aside>

  <figure>

  <figcaption>

  <footer>

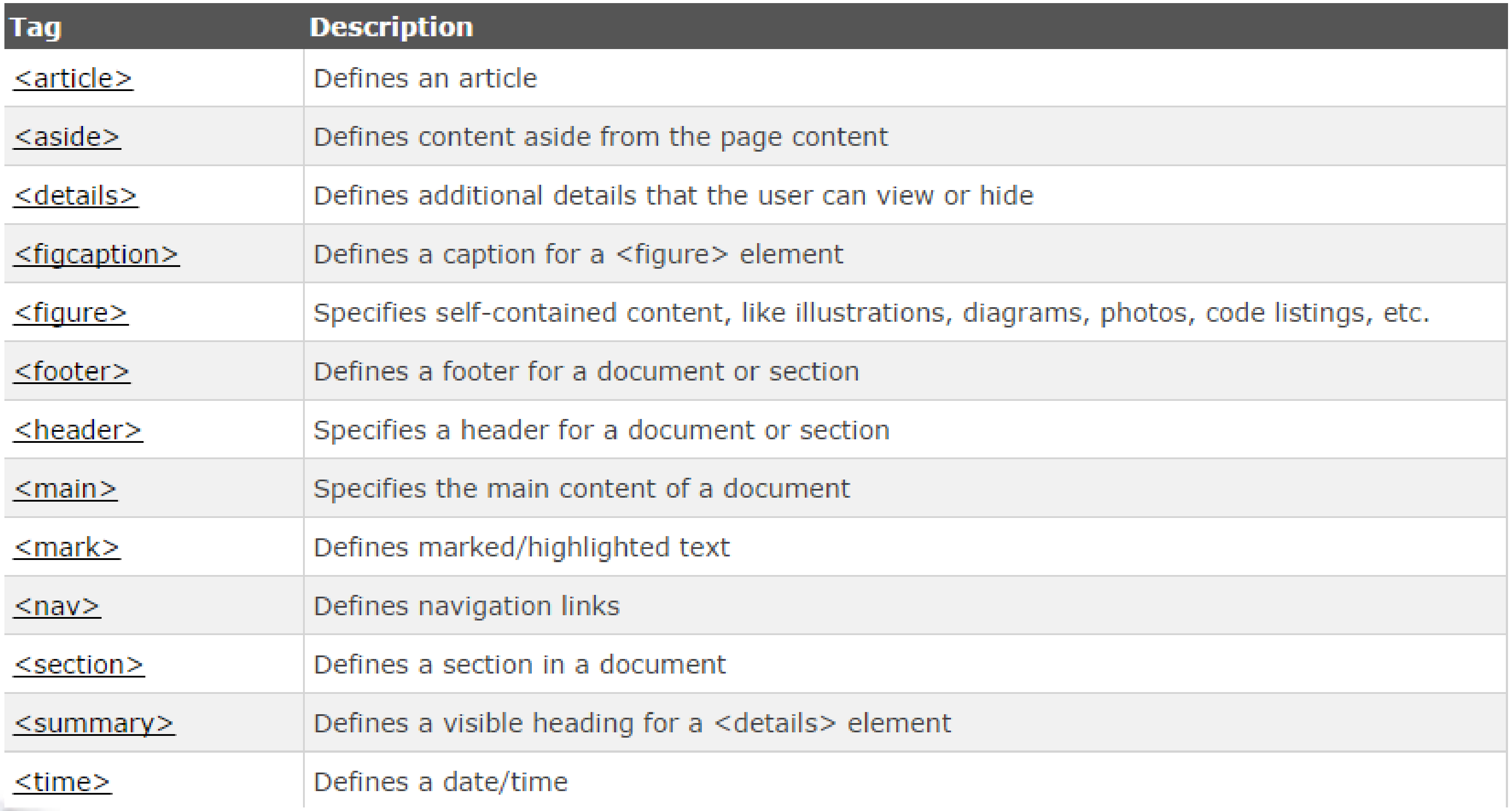
  <details>

  <summary>

  <mark>

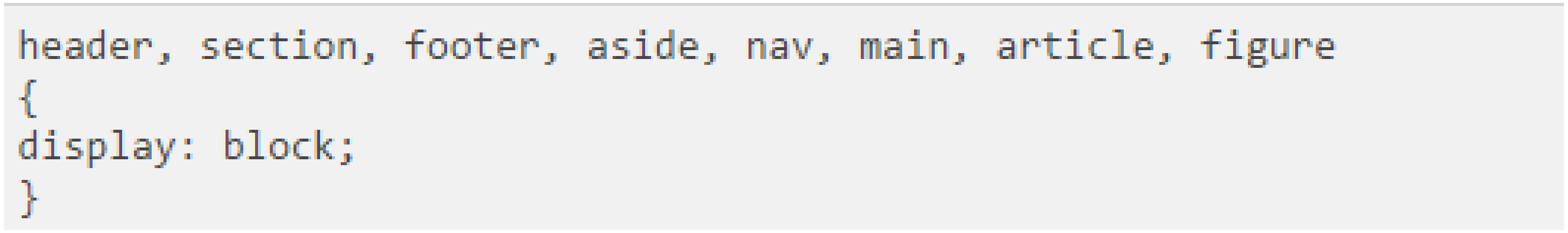
  <time>

Semantic tags



Tag	Description
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

- The elements are all block elements (except <figcaption>).
- To get these elements to work properly in older browsers



```
header, section, footer, aside, nav, main, article, figure
{
display: block;
}
```

Let’s do some coding!!!

HTML5 Input Tags

New Input Types

HTML5 has several new input types.

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

Some browsers might not be able to render the new types YET but the good news is, you can still use them. They will just be read as standard text input

New Input Field Types: Email

The new **email** input type allows the form to validate the email address format without the use of Javascript

Usage:

```
<input type=“email” name=“email”>
```

New Input Field Types: Color

Allows you to select a color from a color picker

Usage:

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

New Input Field Types: Date

Allows the user to select a date.

Usage:

```
<input type="date" name="bday">
```

New Input Field Types: Date

Used for input fields that should contain a numeric value.

You can also set restrictions on what numbers are accepted

Usage:

```
<input type="number" name="quantity"
min="1" max="5">
```

New Input Field Types: Range

Gives us a control for entering a number whose exact value is not important (like a slider control)

Usage:

```
<input type="range" name="points"
min="1" max="10">
```

New Input Field Types: Tel

Used to define a field for entering a telephone number:

Usage:

```
<input type="tel" name="usrtel">
```

New Input Field Types: Time

Used to define a field for entering a time

Usage:

```
<input type="time" name="usr_time">
```

New Input Field Types: url

Used to define a field for entering a url/link

Usage:

```
<input type="url" name="homepage">
```

New Form Elements: <datalist>

- Specifies a list of pre-defined options for an <input> element.
- Provides an "autocomplete" feature on <input> elements. Users will see a drop-down list of pre-defined options as they input data.
- Use the <input> element's list attribute to bind it together with a <datalist> element.

<datalist> Example

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
</datalist>
```

New Form Elements: <keygen>

- Provides a secure way to authenticate users.
- The <keygen> tag specifies a key-pair generator field in a form.
- When the form is submitted, two keys are generated, one private and one public.
- The private key is stored locally, and the public key is sent to the server. The public key could be used to generate a client certificate to authenticate the user in the future.

<keygen> Example

```
<form action="demo.php" method="get">

User:       <input type="text" name="usr_name">
Encryption: <keygen name="security">

<input type="submit">

</form>
```

New Form Elements: <output>

- Perform a calculation and show the result in an <output> element:

```
<form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
<input type="range" id="a" value="50">100  +
<input type="number" id="b" value="50">=
<output name="x" for="a b"></output>
</form>
```

Let’s dive back into some coding!!!

HTML5 Media - <video> & <audio>

Finally!

We finally have a simple way to include videos and audio clips in web pages

Until now, we always needed to download plugins such as Flash to display videos in a web page. Now it is almost as simple as including an image

Browser Support

The following browsers support HTML5 video and audio

- IE9+
- Chrome 6+
- Firefox 3.6+
- Safari 5+
- Opera 10.6+

Types of Media (MIME Types)

VIDEO

- Mp4
- WebM
- Ogg

- Mp3
- Ogg
- Wav

Video Tags

<video> - Define a video

<source> - Defines multiple sources for the video element

<track> - Defines text tracks in media players

Note: *Javascript is needed for extra function and controls*

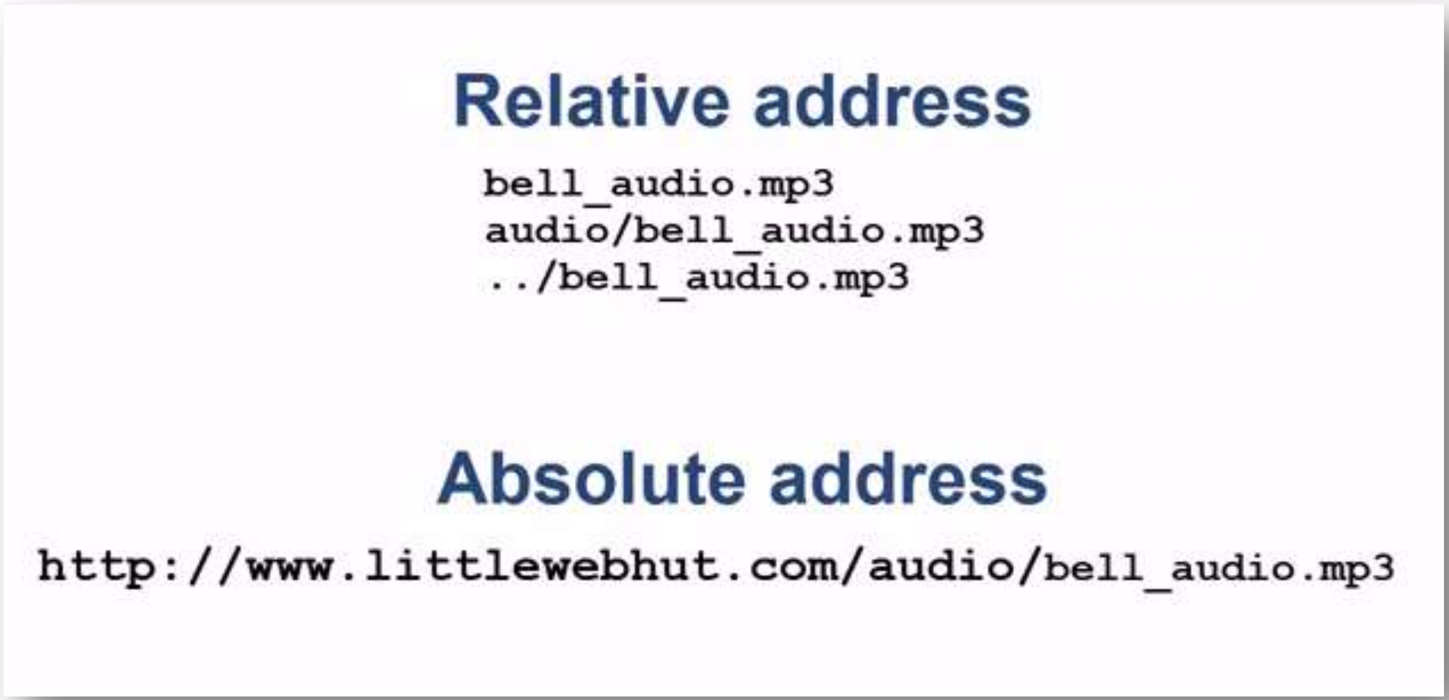
Audio Tags

<audio> - Define sound content

<source> - Defines multiple sources for the audio element

```
<audio src="bell_audio.mp3" controls>
  Audio element not supported by your browser.
</audio>
```

```
<audio controls>
  <source src="bell_audio.mp3" type="audio/mpeg">
  <source src="bell_audio.ogg" type="audio/ogg">
  Audio element not supported by your browser.
</audio>
```



Relative address

```
bell_audio.mp3
audio/bell_audio.mp3
../bell_audio.mp3
```

Absolute address

```
http://www.littlewebhut.com/audio/bell_audio.mp3
```

Javascript Audio/Video Methods

addTextTrack() – Adds a new track

canPlayType() – Checks if browser can play type

load() - Re-loads the audio/video element

play() - Starts playing the audio/video

pause() - Pauses the playing audio/video

Let’s do some coding!!!

SVG

Scalable Vector Graphics

- SVG is used to define vector-based graphics on a web page
- Graphics are defined in XML(Extensive Markup Language)
- Vector graphics **do not** lose any quality if zoomed in or resized
- Elements can be animated
- SVG is a W3C recommendation

More About SVG

SVG is mostly useful for vector type diagrams like Pie charts, Two-dimensional graphs in an X,Y coordinate system etc.

Most of the web browsers can display SVG just like they can display PNG, GIF, and JPG.

Embeding SVG

HTML5 allows embedding SVG directly using **<svg>...</svg>** tag which has following simple syntax:

```
<svg xmlns="http://www.w3.org/2000/svg">
        //Content Here
</svg>
```

Is SVG The Same As Canvas?

No it's not. SVG is a language for describing 2D graphics in XML while Canvas draws 2d graphics on the fly using JavaScript

Each shape in SVG is looked at as an object. If attributes are changed, the browser can automatically re-render the shape. Canvas is pixel by pixel. Once the graphic is drawn, the browser forgets it. If its position is changed, the entire scene needs to be redrawn.

SVG has support for event handlers, Canvas does not

SVG is not for graphic-intensive games and applications. Canvas is well suited for intense graphics

Advantages Over Raster-Based Images

SVG graphics are created using mathematical formulas that need far less data to be stored in the source file because you don't have to store the data for each individual pixel like standard images.

Vector images scale much better. Trying to scale a standard image up from its original size can result in distorted (or pixilated) images.

The source file for an SVG image is a text-based file, so it's both accessible and search engine friendly.

Some code samples!!!

SVG - Drawings

```
<!DOCTYPE html>
<html>
<head>
   <title>Inline SVG Sample</title>
</head>

<body>
<svg id="svg1" width="600" height="400" xmlns="http://www.w3.org/2000/svg" style="background-color: yellow;">

<rect id=rect1 width=300 height=100 x=10 y=10 fill="blue"></rect>

<rect id=square1 width=100 height=100 x=320 y=10 fill="magenta" style="stroke:red;stroke-width:5px;"></rect>

<circle id=circle1 cx=150 cy=150 r=50 fill="red"></circle>
<circle id=circle1 cx=420 cy=150 r=70 fill="pink" style="stroke:blue;stroke-width:5px;"></circle>

<line id=line1 x1=50 y1=350 x2=550 y2=30 style="stroke:black;stroke-width:10px;"></line>
<line id=line2 x1=550 y1=30 x2=550 y2=350 style="stroke:blue;stroke-width:10px;"></line>
<line id=line3 x1=550 y1=350 x2=50 y2=350 style="stroke:red;stroke-width:10px;"></line>

<ellipse id=ellipse1 cx=150 cy=250 rx=50 ry=105 fill="green" style="stroke:black;stroke-width:2px;"></ellipse>

<polygon id=poly1 points="250,200 250,375 150,400 150,240 120,200"  fill=red style="stroke:black;stroke-width:2px"> </polygon>

</svg>
</body>
</html>
```

Geo Location

What Is Geolocation?

Geolocation is an HTML5 API that allows us to get the geographical location of a website user

The user MUST approve before getting the users position. This usually happens via button and/or browser popup

All of the latest versions of Chrome, Firefox, IE, Safari and Opera can use the geolocation feature of HTML5

Some great uses of Geolocation

- Public transportation websites
- Taxi and other transportation websites
- Calculate shipping costs on an Ecommerce site
- Travel angency websites
- Real estate websites
- Movie theater websites can find movies playing nearby
- Online gaming
- Local headlines and weather on their front page.
- Job postings can automatically include commute times
- The possibilities are **endless!**

How It Works

Geolocation works by scanning common sources of location information which include the following…

- Global Positioning System (GPS) – Most Accurate
- Network Signals - IP address, RFID, WiFi and Bluetooth MAC addresses
- GSM/CDMA cell IDs
- User Input

Checking for Geolocation Support

The API offers a very handy function to detect for Geolocation support in browsers…

```
if (navigator.geolocation) {
        // do stuff
}
```

You can also check for Geolocation using the “Modernizr” script

getCurrentPosition( )

The getCurrentPosition API is the main method using geolocation. It retrieves the current geographic location of the users device. The location is expressed as a set of geographic coordinates together with information about heading and speed. The location information is returned in a Position object.

Syntax:

```
getCurrentPosition(showLocation, ErrorHandler, options);
```

getCurrentPosition( ) Parameters

- **showLocation** : Defines the callback method that retrieves location information.
- **ErrorHandler(Optional)** : Defines the callback method that is invoked when an error occurs in processing the asynchronous call.
- **options (Optional):** Defines a set of options for retrieving the location information.

getCurrentPosition( ) Return Values

coords.latitude	The latitude as a decimal number
coords.longitude	The longitude as a decimal number
coords.accuracy	The accuracy of position
coords.altitude	The altitude in meters above the mean sea level
coords.altitudeAccuracy	The altitude accuracy of position
coords.heading	The heading as degrees clockwise from North
coords.speed	The speed in meters per second
timestamp	The date/time of the respons

Some code samples!!!

latitude and longitude of the user's position

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

<p id="demo">Click the button to get your coordinates:</p>

<button onclick="getLocation()">Try It</button>

<script>
var x = document.getElementById("demo");

function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);
    } else {
        x.innerHTML = "Geolocation is not supported by this browser.";
    }
}

function showPosition(position) {
    x.innerHTML="Latitude: " + position.coords.latitude +
    "<br>Longitude: " + position.coords.longitude;
}
</script>

</body>
</html>
```

Handling errors

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

<p id="demo">Click the button to get your coordinates:</p>

<button onclick="getLocation()">Try It</button>

<script>
var x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {

      navigator.geolocation.getCurrentPosition(showPosition
      ,showError);
  } else {
     x.innerHTML = "Geolocation is not supported by this
      browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
```

```
function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for
     Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Location information is
     unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML = "The request to get user
     location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
  }
}
</script>

</body>
</html>
```

Displaying in a Map

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

<p id="demo">Click the button to get your position:</p>
<button onclick="getLocation()">Try It</button>
<div id="mapholder"></div>

<script>
var x = document.getElementById("demo");

function getLocation() {
   if (navigator.geolocation) {

        navigator.geolocation.getCurrentPosition(showPosition
        ,showError);
   } else {
      x.innerHTML = "Geolocation is not supported by this
        browser.";
   }
}

function showPosition(position) {
   var latlon =
        position.coords.latitude+","+position.coords.longitude;

 var img_url =
"http://maps.googleapis.com/maps/api/staticmap?center
="
   +latlon+"&zoom=14&size=400x300&sensor=false";
   document.getElementById("mapholder").innerHTML =
"<img src='"+img_url+"'>";
}

function showError(error) {
   switch(error.code) {
      case error.PERMISSION_DENIED:
         x.innerHTML = "User denied the request for
Geolocation."
         break;
      case error.POSITION_UNAVAILABLE:
         x.innerHTML = "Location information is unavailable."
         break;
      case error.TIMEOUT:
         x.innerHTML = "The request to get user location
timed out."
         break;
      case error.UNKNOWN_ERROR:
         x.innerHTML = "An unknown error occurred."
         break;
   }
}
</script>

</body>
</html>
```

Web Storage

What Is Web Storage?

With HTML5 web storage, websites can store data on a users local computer

No More Cookies

We had to use Javascript cookies in the past to achieve this functionality.

How It Differs From Cookies

- More secure
- Faster
- Stores a larger amount of data
- The stored data is not sent with every server request. It only is included when asked for. This gives a big advantage over cookies

Detecting HTML5 Storage Support

```
function supports_html5_storage() {
      try {
            return 'localStorage' in window && window['localStorage'] !== null;
      } catch (e) {
            return false;
      }
}
```

localStorage( ) & sessionStorage()

There are 2 types of web storage objects, local and session

Difference:

localStorage – Stores data with no expiration date

sessionStorage – Stores data for one session

How It Works

The localStorage and sessionStorage objects create a key = value pair

Example: key="Name", value="Bob"

They are stored as strings but can be converted after if needed by using JS functions like parseInt() and parseFloat()

Syntax for Web Storage

Storing a Value:

- localStorage.setItem("key1", "value1");
- localStorage["key1"] = "value1";

Getting a Value:

- alert(localStorage.getItem("key1"));
- alert(localStorage["key1"]);

Remove a Value:

- removeItem("key1");

Remove All Values:

- localStorage.clear();

Some code examples!!

Session Storage

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

  <script type="text/javascript">
    if( sessionStorage.hits ){
      sessionStorage.hits = Number(sessionStorage.hits) +1;
    }else{
      sessionStorage.hits = 1;
    }
    document.write("Total Hits :" + sessionStorage.hits );
  </script>
  <p>Refresh the page to increase number of hits.</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
   if(typeof(Storage) !== "undefined") {
      if (sessionStorage.clickcount) {
         sessionStorage.clickcount = Number(sessionStorage.clickcount)+1;
      } else {
         sessionStorage.clickcount = 1;
      }
      document.getElementById("result").innerHTML = "You have clicked the button " + sessionStorage.clickcount + " time(s) in this
        session.";
   } else {
      document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";
   }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter is reset.</p>
</body>
</html>
```

Local Storage

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

  <script type="text/javascript">
    if( localStorage.hits ){
       localStorage.hits = Number(localStorage.hits) +1;
    }else{
       localStorage.hits = 1;
    }
    document.write("Total Hits :" + localStorage.hits );
  </script>
  <p>Refresh the page to increase number of hits.</p>
  <p>Close the window and open it again and check the result.</p>

</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<script>
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (localStorage.clickcount) {
            localStorage.clickcount = Number(localStorage.clickcount)+1;
        } else {
            localStorage.clickcount = 1;
        }
        document.getElementById("result").innerHTML = "You have clicked the button " + localStorage.clickcount + " time(s).";
    } else {
        document.getElementById("result").innerHTML = "Sorry, your browser does not support web storage...";
    }
}
</script>
</head>
<body>
<p><button onclick="clickCounter()" type="button">Click me!</button></p>
<div id="result"></div>
<p>Click the button to see the counter increase.</p>
<p>Close the browser tab (or window), and try again, and the counter will continue to count (is not reset).</p>
</body>
</html>
```

Thank You