WEB PROGRAMMING

Overview

- Computing fundamentals
- Client-server applications
- Programming environment
  - Backend or server (machine) requirements and specifications
  - Tools and utilities i.e. PL, IDE, API, DB, middleware
- Architecture

Computing fundamentals

- Encoding systems
  - ASCII
  - EBCDIC
  - UNICODE
- Processors
  - x86
  - x64
  - Multicore

Client-server applications

- Web-based, use of browser (e.g. I.E., Safari, Firefox et al) and a web server that facilitates data/resources (e.g. Apache, IIS)
- WAN- or private network-based, IBM Notes (a productivity suite including email client) and IBM Domino (email server)
- Remote access, SSH and Telnet have their own client and server implementations

Programming environment

- Backend or server requirements and specification
  - Capacity
  - Operating systems
  - Web server
  - Database server

Programming environment cont'n

- Tools and utilities
  - PL e.g. HTML, PHP, C#, Java
  - IDE e.g. Visual Studio, Eclipse, Dreamweaver, Notepad++, BlueJ

Architecture

- Data
- Computing and infrastructure

WDE build, localhost

- WAMP
- XAMPP
- IIS/MWP
  - PHP and MySQL not installed by default

HTML 5

HTML

- Has always been the WWW markup language
- Primarily designed as a language for semantically describing scientific documents
- Web applications – is the main area that has not been adequately addressed as also called a vague subject, however, v5 specification attempts to rectify this

HTML resource

- Supported, when referring to whether a user agent has an implementation capable of decoding the semantics of an external resource
  - If the implementation can process an external resource of that format or type without critical aspects being ignored
- Examples
  - PNG image, supported if its pixel data could be decoded and rendered
  - MPEG-4 video, if the compression format use was not supported, even if the implementation could determine the dimensions of the movie from the file's metadata

HTML elements

- Refers to any element in that of xhtml namespace, for the purposes of DOM and CSS, and thus refers to both XML and XHTML
- Element type, a term used to refer to the set of elements that have a given local name and namespace
- Elements, the structures that describe the parts of the HTML document
  - 3 parts: start, content and end tags
- DOM trees, root element

HTML declaration

- `<!DOCTYPE html>` - doctype
- `<meta charset=“UTF-8”>` - character encoding

HTML example code

```
<!DOCTYPE html>
<html>

  <head>
   <meta charset="UTF-8">
    <title>Facula Industries</title>
  </head>

  <body>
   About Facula Industries and its nature of business.
  </body>

</html>
```

HTML new semantic / structural elements

- <article> Defines an article in the document
- <aside> Defines content aside from the page content
- <bdi> Defines a part of text that might be formatted in a different direction from other text
- <details> Defines additional details that the user can view or hide
- <dialog> Defines a dialog box or window
- <figcaption> Defines a caption for a <figure> element
- <figure> Defines self-contained content, like illustrations, diagrams, photos, code listings, etc.

HTML new semantic / structural elements continuation

- <footer> Defines a footer for the document or a section
- <header> Defines a header for the document or a section
- <main> Defines the main content of a document
- <mark> Defines marked or highlighted text
- <menuitem> Defines a command/menu item that the user can invoke from a popup menu
- <meter> Defines a scalar measurement within a known range (a gauge)
- <nav> Defines navigation links in the document

HTML new semantic / structural elements continuation

- <progress> Defines the progress of a task
- <rp> Defines what to show in browsers that do not support ruby annotations
- <rt> Defines an explanation/pronunciation of characters (for East Asian typography)
- <ruby> Defines a ruby annotation (for East Asian typography)
- <section> Defines a section in the document
- <summary> Defines a visible heading for a <details> element
- <time> Defines a date/time
- <wbr> Defines a possible line-break

HTML new semantic / structural elements example

```
<article>
  <header>
    <h1>Facula Industries</h1>
    <p><time pubdate datetime="2011-03-15"></time></p>
  </header>
  <p>The company has released its new business platform in the public
on 2 September 2007 at 21:00 PDT.</p>
</article>
```

Document Object Model

Is a cross-platform and language-independent *convention* for representing and interacting with objects in HTML, XHTML and XML documents. The nodes of every document are organized in a tree structure, called the *DOM tree*. Objects in the DOM tree may be addressed and manipulated by using methods on the objects. The public interface of a DOM is specified in its application programming interface (API).

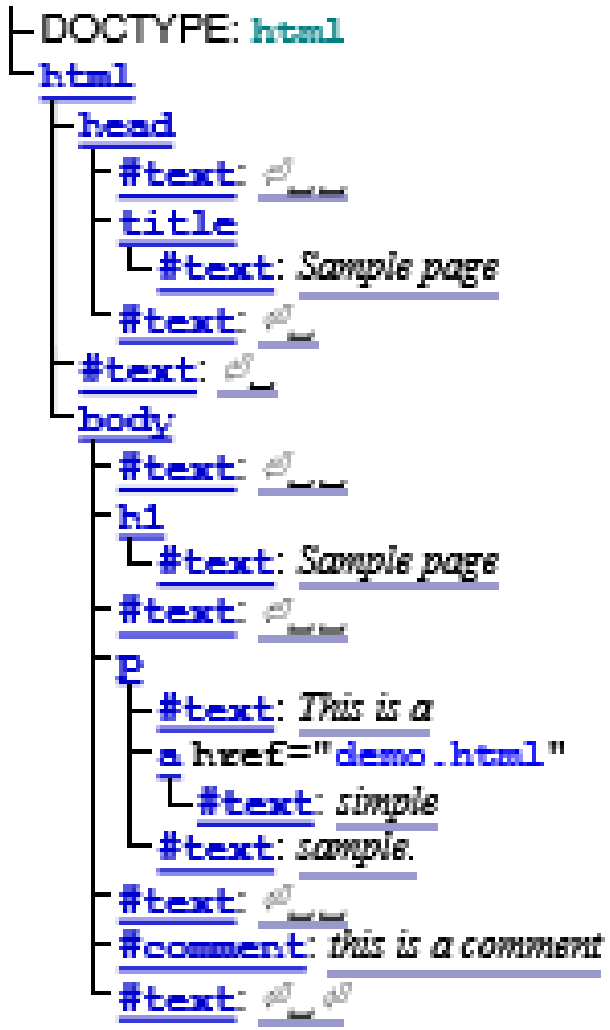
DOM history

Is intertwined with the history of the "browser wars" of the late 1990s between Netscape Navigator and Microsoft Internet Explorer, as well as with that of JavaScript and JScript, the first scripting languages to be widely implemented in the layout engines of web browsers.

DOM trees

- Tree order, means a pre-order, depth first traversal of DOM nodes involved
- Root element, the document first element child, means the furthest ancestor element node of whatever node is being discussed or the node itself if it has no ancestors
- Orphaned, node not currently part of the document tree
- Can be manipulated from scripts i.e. JavaScript in the page and these are all programs that can be embedded using `script` element that sets the value of the form's output, see example in the DOM trees examples

DOM tree example



Descriptions

`html` – is the root element of this tree, which is the element ways found at the root of HTML elements.

`head` and `body` – contained under the html element.

`text` – contained in between head and body. Also the source contains a number of *spaces* and *line breaks* which all end up as `text` nodes in the DOM tree.

DOM trees example continuation

```
<form name="main">
 Result: <output name="result"></output>
 <script>
  document.forms.main.elements.result.value = 'Hello World';
 </script>
</form>
```

Each element in this DOM tree is represented by an object and these have APIs so that they can be manipulated. For instance a link with the `a` element in the tree above can have its `href` attributes change in several ways like

```
var a = document . links [0];          // obtain the first link in the document
a.href = 'sample.html';                // change the destination URL of the link
a. protocol = 'https';                 // change just the scheme part of the URL
a.setAttribute('href', 'http://example.com/'); // change the content attribute directly
```

Conformance classes and requirements

- Web browsers and other interactive user agent
- Non interactive presentation user agents
- Visual user agent that support the suggested the default rending
- User agent with no scripting support
- Conformance checkers
- Data mining tool
- Authoring tools and markup generators

Dependencies

Encoding, XML, URLs, Cookies, CORS, WebIDL, JavaScript, DOM, Type arrays, File API, XmlHttpRequest, Progress Events, Server Sent events, Media Queries, CSS modules, SVG, WebGL, WebVTT, Web Socket Protocol, Web Workers, 2D Content, Web Messaging, ARIA

- More about this in the succeeding slides.

Extensibility

- Vendor specific proprietary user agent extensions to this specification are strongly discouraged.
- Documents must not use extensions, as doing so reduces interoperability and fragments the user base, allowing only users of specific user agents to access the content in question.
- For DOM extensions e.g. new methods and IDL attributes, the new members should be pre-fixed by vendor specific strings to prevent clashes with future version of HTML, particularly this specification with version 5.

Interface Descriptor Language

- Not to be confused with Interactive Data Language
- Any computer language used to describe a software component's interface
- 3 IDL sources
  - IDL specification language created by Lamb, Wulf, Nestor at Queen's University, Canada
  - OMG IDL standardized and selected by W3C for exposing the DOM of XML, HTML, CSS and SVG documents
  - MS IDL an extension of OMG IDL for MS DCOM services

IDL by Lamb, Wulf and Nestor

- Defined interfaces in a language- and machine- independent way, allowing the specification of interfaces between components written in different languages, and possibly executing on different machines using remote procedure calls.
- RPC is an interprocess communication that allows computer program to cause a subroutine or procedure to execute in another address space without the programmer explicitly coding the details of this remote interaction.

OMG IDL

- Specifications include for
  - Ada
  - C/C++
  - COBOL
  - CORBA scripting language
  - IDL to C++11
  - IDL to Java
  - Java to IDL
  - MOF2 to IDL
  - etcetera

OMG IDL on enabling distribution

- *Encapsulation* is the principle of object-orientation that enables seamless distribution. By encapsulating the internal structure and mechanism of the objects inside a boundary that the client is not allowed to penetrate, it gain not only flexibility but also a simplicity of architecture that could not be achieved any other way.

OMG IDL types, 2 levels

- 1st, every CORBA object has a type name, which is the same as the **interface** name assigned in its IDL declaration.
  - The operations that it can perform, and the variables (and their types!) that it understands, are all part of its type. Every time one define a new kind of CORBA object as it write OMG IDL (by declaring a new **interface**), it create a new CORBA type.
- 2nd, IDL variables are typed but not in the same way as CORBA objects. One don't create types for variables; instead it has to choose from the official list of types allowed by the IDL specification. It's a full list; base types include three different precisions of integers and floating-point numbers plus fixed-point; standard and wide characters and strings; boolean; valuetype (we'll write a separate page about the valuetype soon); and octet for when you absolutely have to send a binary value that only an application understands.

OMG IDL definition of scopes

- Delimits where type definitions may be used. The namespaces used in C++ may be similar to its concept.
- Global scope encompasses the entire compilation unit.

OMG IDL interface denitions

- IDL interface definitions are elegant and (usually!) easy to write and understand. An interface from programming example in the book CORBA 3 Fundamentals and Programming

```
interface salestax  {
   float  calculate_tax ( in float taxable_amount );
}
```

HTML attributes

- A modifier of an element.
- It generally appear as name value pairs, separated by “=” and are written within the start tag of an element after element’s name:

```
<tag attribute="value">(content to be modified by the tag)</tag>
```

- Where *tag* names the HTML element, *attribute* is the name of the attribute, set to the provided *value*.

Common attribute, id

- Provides a unique-wide common identifier for an element.
- Can be used as CSS selector to provide presentational properties, by browsers to focus attention on the specific element, or by scripts to alter the contents or presentation of an element
- Appended to the URL of the page, the URL directly targets the specific element within the document, typically a sub-section of the page. For example, the ID "Attributes" in http://en.wikipedia.org/wiki/HTML#Attributes

Common attribute, class

- Provides a way of classifying similar elements. This can be used for semantic or presentation purposes.
- Semantically are used in microformats (more for it in the succeeding slides), which seeks to re-use existing HTML/XHTML tags to convey metadata and other attributes in web pages and other contexts that support (X)HTML such as RSS. This approach allows software to process information intended for end-users (such as contact information, geographic coordinates, calendar events, and similar information) automatically.

Common attribute, class with μF

In this example, the contact information is presented as follows:

```
<ul>
  <li>Joe Doe</li>
  <li>Facula Industries</li>
  <li>63-921-3333191</li>
  <li><a href="http://facula.in/">http://facula.in/</a></li>
</ul>
```

With hCard (short for HTML vCard) microformat markup, that becomes:

```
<ul class="vcard">
  <li class="fn">Joe Doe</li>
  <li class="org">Facula Industries</li>
  <li class="tel">63-921-3333191</li>
  <li><a class="url" href="http://facula.in/">http://facula.in/</a></li>
</ul>
```

Common attribute, title

- Used to attach subtextual explanation to an element. In most browsers this attribute is displayed as what is often referred to as a tooltip.

Attributes with abbr element

```
<abbr id="anId" class="aClass"
style="color:blue;"
title="Hypertext Markup
Language">HTML</abbr>
```

HTML attribute classes

- Required attributes
- Optional attributes
  - Both modify specific HTML elements
- Standard attributes
  - Can be applied to most HTML elements
- Event attributes
  - Added in HTML 4, allow an element to specific scripts to be run under specific circumstances

Attributes, required and optional

- Used by one tag
- Used by two tag
- Used by multiple tag

Attribute, used by one tag, examples

- `<applet>`: *`code`*, *`object`*
- `<area>`: *`nohref`*
- `<body>`: *`alink`*, *`background`*, *`link`*, *`text`*, *`vlink`*

Attribute, used by two tags, examples

- <a> and <area>: coords – coordinates of an area or a link within it.
  - shape – shape of an area or a link within it. Values: default, rect, circle, poly.

- <a> and <link>: hreflang – Language code of the linked document. (a, link)
  - rel – Nature of the linked document (relative to the page currently displayed). Free text for a, but link uses a set of terms (alternate, appendix, bookmark, chapter, contents, copyright, glossary, help, home, index, next, prev, section, start, stylesheet, subsection).
  - rev – Nature of the currently displayed page (relative to the linked document). Varies for a and link as for rel.

Attribute, used by multiple tags, example

- alt – applet, area, img, input
- bgcolor – body, table, td, th, bgcolor
- border – img, object, table
- char – char, <colgroup>, <tbody>, <td>, <tfoot>, <th>, <thead>, <tr>

Specific µF

- Several microformats have been developed to enable semantic markup of particular types of information. However, only hCard and hCalendar have been ratified, the others remaining as drafts:
- ##hAtom – for marking up Atom feeds from within standard HTML
- ##hCalendar – for events
- ##hCard – for contact information; includes: ##adr – for postal addresses
- ##geo – for geographical coordinates (latitude, longitude)

Specific µF continuation

- ##hMedia - for audio/video content [7][8]
- ##hAudio – for audio content
- ##hNews - for news content
- ##hProduct – for products
- ##hRecipe - for recipes and foodstuffs.
- ##hResume – for resumes or CVs
- ##hReview – for reviews
- ##rel-directory – for distributed directory creation and inclusion[9]

Specific μF continuation

- ##rel-enclosure – for multimedia attachments to web pages[10]
- ##rel-license – specification of copyright license[11]
- ##rel-nofollow, an attempt to discourage third-party content spam (e.g. spam in blogs)
- ##rel-tag – for decentralized tagging (Folksonomy)[12]
- ##xFolk – for tagged links
- ##XHTML Friends Network (XFN) – for social relationships
- ##XOXO – for lists and outlines

µF arguments

- Accordingly, Alex Faaborg summarizes the arguments for putting the responsibility for microformat user interfaces in the web browser rather than making more complicated HTML:
  - Only the web browser knows what applications are accessible to the user and what the user's preferences are
  - It lowers the barrier to entry for web site developers if they only need to do the markup and not handle "appearance" or "action" issues
  - Retains backwards compatibility with web browsers that don't support microformats
  - The web browser presents a single point of entry from the web to the user's computer, which simplifies security issues

WEB BROWSERS AND ENGINES

Before anything else

Major parsing and rendering engines

- Trident
- KHTML
- Gecko
- Blink
- Presto

Trident

- Also known as MSHTML.
- The layout engine for the Microsoft Windows version of Internet Explorer.
- It was first introduced with the release of Internet Explorer version 4.0 in October 1997; it has been steadily upgraded and remains in use today.
- For versions 7 and 8 of Internet Explorer, Microsoft made significant changes to the Trident layout engine to improve compliance with web standards and add support for new technologies.

Trident versions and improvement

Trident version	MSHTML.dll version	Internet Explorer version	Internet Explorer Mobile version	Notes
No ver	4.0.x	4.0	N/A	Initial version.
	5.0.x	5.0	N/A	Improved CSS 1 support and had sweeping changes in CSS 2 rendering.
	5.5.x	5.5	N/A	Corrected issues with CSS handling.
	6.0.x	6.0	N/A	Corrected the box model and added quirks mode with DTD switching.
	7.0.x	7.0	N/A	Fixed many CSS rendering issues and added partial PNG alpha support.
	N/A	N/A	6.0	IEMobile 6 combines many features of IE 6, 7, and 8.
3.1	7.0	N/A	7.0	Second port on a mobile system of Trident. IE Mobile version for Windows Phone 7.
4.0	8.0.x	8.0	N/A	First version to pass the Acid 2 test. Added full support for CSS 2.1.
5.0	9.0.x	9.0	9.0	Added support for SVG, XHTML, HTML5, and CSS 3. Added a new hardware-accelerated JScript engine named Chakra. Scores 100/100 on the Acid3 test. Included with IE 9 Mobile in Windows Phone 7.5 "Mango".
6.0	10.0.x	10.0	10.0	More support for CSS 3, HTML5 and ES5. Included in Windows Phone 8.
7.0	11.0.x	11.0	11.0	Support for WebGL and SPDY. Improved support for HTML5. Speed improvement. Included in Windows Phone 8.1.

Trident and standards compliance

- As of Internet Explorer 9 have introduced support for CSS 3, HTML5, and SVG, as well as other modern web standards.
- Web standards compliance was gradually improved with the evolution of Trident.
- Each version of IE has improved standards support, including the introduction of a "standards-compliant mode" in version 6, the core standards that are used to build web pages (HTML and CSS) were sometimes implemented in an incomplete fashion.
  - For example, there was no support for the <abbr> element which is part of the HTML 4.01 standard prior to IE 8. There were also some CSS attributes missing from Trident, like min-height, etc. as of IE 6.
- As of Internet Explorer 8 CSS 2.1 is fully supported as well as some CSS 3.0 attributes. This lack of standards compliance has been known to cause rendering bugs and lack of support for modern web technologies, which often increases development time for web pages. Still rendering differences of HTML between standards-compliant browsers are not completely resolved yet.

Microsoft alternative engines

- Tasman, used in IE5 for MAC, development halted in 2003.
  - Was later included in Office 2004 for MAC.
- Webkit, an open source and used by Office for MAC 2011

KHTML

- A layout engine developed by the KDE project.
- It is the engine used by the Konqueror web browser.
- A forked version of KHTML called WebKit is used by several web browsers, among them Safari.
- It has supports to the latest web standards such as HTML5, Javascript, CSS3, and others.
- Built on the KPart framework (see next slide) and written in C++, KHTML has relatively good support for Web standards. To render as many pages as possible, some extra abilities and quirks from Internet Explorer are also supported, even though those are non-standard.

KPart Framework (where KHTML was built)

- It is the component framework for the KDE desktop environment. An individual component is called a KPart. KParts are analogous to Bonobo components in GNOME and ActiveX controls in Microsoft's Component Object Model. Konsole is available as a KPart and is used in applications like Konqueror and Kate.
- Example uses of KParts:
  - Konqueror uses the Okular part to display documents
  - Konqueror uses the Dragon Player part to play multimedia
  - Kontact embeds kdepim applications
  - Kate and other editors use the katepart editor component
  - Several applications use the Konsole KPart to embed a terminal

Gecko

- The open source web browser layout engine used in many applications developed by the Mozilla Foundation and Mozilla Corporation, as well many other open source software projects.
- Notably the Firefox web browser including its mobile version and their e-mail client Thunderbird.
- It is designed to support open Internet standards, and is used by different applications to display web pages and, in some cases, an application's user interface itself (by rendering XUL, see next slide).
- Offers a rich programming API that makes it suitable for a wide variety of roles in Internet-enabled applications, such as web browsers, content presentation, and client/server.

XML user interface language (Gecko support to open Internet standards)

- It is a user interface markup language that is developed by Mozilla. XUL is implemented as an XML dialect; it allows for graphical user interfaces to be written in a similar manner to Web pages.
- XUL can be used to write cross-platform applications such as Mozilla Firefox, where it is interpreted by the layout engine known as Gecko which renders Firefox's user interface and Web page display.

XUL adaptations and applications

- The Ample SDK open-source GUI framework provides a cross-browser implementation of XUL in JavaScript.
- The ActiveState Komodo IDE uses XUL as well as the Open Komodo project announced in 2007.
- The Songbird music-player and Miro video-player both use built-in XUL.
- The Elixon WCMS/XUL Content management system uses exclusively remote XUL, thus overcoming some of the aforementioned limits of remote unprivileged XUL documents.
- The developers of the Celtx media pre-production application used XUL.
- The Flickr Uploader was built using XUL, and source code is available under GPLv2.
- Kiwix, the offline Wikipedia-viewer.

Blink

- is a web browser engine developed as part of the Chromium project by Google with contributions from Opera Software ASA, Intel, Samsung and others. It was first announced in April 2013. It is a fork of the WebCore component of WebKit and is used in Chrome starting at version 28, Opera (15+), Amazon Silk and other Chromium based browsers as well as Android's (4.4+) WebView and Qt's upcoming WebEngine.

Presto

- The layout engine of the Opera web browser for a decade.
- A dynamic engine.
- It was released on 28 January 2003 in Opera 7 for Windows, after several public betas and technical previews. Opera continued to use Presto until version 15, at which point the browser was rewritten with a Chromium backend, containing the Blink layout engine and V8 JavaScript engine.

SEMANTICS, STRUCTURE AND APIs of HTML DOCUMENTS

Document’s address

- It is an absolute URL that is initially set when the Document is created but that can change during the lifetime of the Document for example when the user navigates to a fragment identifier on the page or when the pushState() method is called with a new URL .
  - When a Document is created by a script using the createDocument() or createHTMLDocument() APIs, the document's address is the same as the document's address of the responsible document specified by the script's settings object, and the Document is both ready for post-load tasks and completely loaded immediately.

Warning! Interactive user agents typically expose the document's address in their user interface. This is the primary mechanism by which a user can tell if a site is attempting to impersonate another.

Document’s referrer

- It is an absolute URL that can be set when the Document is created. If it is not explicitly set, then its value is the empty string.

Reload override flag

- Each Document object has this that is originally unset.
- The flag is set by the document.open() and document.write() methods in certain situations. When the flag is set, the Document also has a **reload override buffer** which is a Unicode string that is used as the source of the document when it is reloaded.

Reload override flag continuation

- When the user agent is to perform an overridden reload, given a source browsing context, it must act as follows:
  - Let source be the value of the browsing context's active document's reload override buffer.
  - Let address be the browsing context's active document's address .
  - Navigate the browsing context to a resource whose source is source , with replacement enabled and exceptions enabled. The source browsing context is that given to the overridden reload algorithm. When the navigate algorithm creates a Document object for this purpose, set that Document's reload override flag and set its reload override buffer to source .
- When it comes time to set the document's address in the navigation algorithm, use address as the override URL .

Document interface in DOM specification with

- Resource data management
- DOM tree accessors
- Dynamic markup insertion
- User interaction
- Special event handler IDL attributes that only apply to Document objects

Document interface, resource data management

- Document.referrer
  - Returns the address of the Document from which the user navigated to this one, unless it was blocked or there was no such document, in which case it returns the empty string.
  - The noreferrer link type can be used to block the referrer.
  - The referrer attribute must return the document's referrer.
- Document.cookie[=value]
  - Returns the HTTP cookies that apply to the Document . If there are no cookies or cookies can't be applied to this resource, the empty string will be returned.
  - Can be set, to add a new cookie to the element's set of HTTP cookies.
  - If the contents are sandboxed into a unique origin (e.g. in an iframe with the sandbox attribute), a SecurityError exception will be thrown on getting and setting.
  - The cookie attribute represents the cookies of the resource identified by the document's address .

Document interface, resource data management continuation

- A Document object that falls into one of the following conditions is a cookie-averse Document object:
  - A Document that has no browsing context.
  - A Document whose address does not use a server-based naming authority.
- Document.lastModified
  - Returns the date of the last modification to the document, as reported by the server, in the form "MM/DD/YYYY hh:mm:ss", in the user's local time zone.
  - If the last modification date is not known, the current time is returned instead.

Document interface, resource data management, Document.lastModified

- The lastModified attribute, on getting, must return the date and time of the Document's source file's last modification, in the user's local time zone, in the following format:

1. The month component of the date.
2. A "/" (U+002F) character.
3. The day component of the date.
4. A "/" (U+002F) character.
5. The year component of the date.
6. A U+0020 SPACE character.
7. The hours component of the time.
8. A ":" (U+003A) character.
9. The minutes component of the time.
10. A ":" (U+003A) character.
11. The seconds component of the time.

Document interface, resource data management continuation

- Document.readyState
  - Returns "loading" while the Document is loading, "interactive" once it is finished parsing but still loading sub-resources, and " complete" once it has loaded.
  - The readystatechange event fires on the Document object when this value changes.
- Each document has a current document readiness. When a Document object is created, it must have its current document readiness set to the string "loading" if the document is associated with an HTML parser, an XML parser, or an XSLT processor, and to the string "complete" otherwise. Various algorithms during page loading affect this value. When the value is set, the user agent must fire a simple event named readystatechange at the Document object.
- A Document is said to have an active parser if it is associated with an HTML parser or an XML parser that has not yet been stopped or aborted .
- The readyState IDL attribute must, on getting, return the current document readiness.

DOM tree accessors

- The html element of a document is the document's root element, if there is one and it's an html element, or null otherwise.
- Document.head
  - Returns the head element .
  - The head element of a document is the first head element that is a child of the html element, if there is one, or null otherwise.
  - The head attribute, on getting, must return the head element of the document (a head element or null).

DOM tree accessors

- Document.title[=value]
  - Returns the document's title, as given by the title element .
  - Can be set, to update the document's title. If there is no head element, the new value is ignored.
  - In SVG (more of this in succeeding slides) documents, the SVGDocument interface's title attribute takes precedence.
- The title element of a document is the first title element in the document (in tree order), if there is one, or null otherwise.

DOM tree accessors , Document.title

- The title attribute must, on getting, run the following algorithm:

1. If the root element is an svg element in the "http://www.w3.org/2000/svg" namespace, and the user agent supports SVG, then return the value that would have been returned by the IDL attribute of the same name on the SVGDocument interface.

2. Otherwise, let value be a concatenation of the data of all the child Text nodes of the title element, in tree order, or the empty string if the title element is null.

3. Strip and collapse whitespace in value .

4. Return value .

DOM tree accessors, Document.title continuation

On setting, the following algorithm must be run. Mutation events must be fired as appropriate.

1. If the root element is an svg element in the "http://www.w3.org/2000/svg" namespace, and the user agent supports SVG, then the setter must act as if it was the setter for the IDL attribute of the same name on the Document interface defined by the SVG specification. Stop the algorithm here.

2. If the title element is null and the head element is null, then the attribute must do nothing. Stop the algorithm here.

3. If the title element is null, then a new title element must be created and appended to the head element. Let element be that element. Otherwise, let element be the title element .

4. The children of element (if any) must all be removed.

5. A single Text node whose data is the new value being assigned must be appended to element .

DOM tree accessors, Document.title continuation

- The title IDL attribute defined above must replace the attribute of the same name on the Document interface defined by the SVG specification when the user agent supports both HTML and SVG.

Scalable Vector Graphics

- *(http://www.w3.org/TR/SVG/Overview.html)*
- A modularized language for describing two-dimensional vector and mixed vector/raster graphics in XML.
- SVG allows for three types of graphic objects:
  - Vector graphic shapes (e.g., paths consisting of straight lines and curves),
  - Images,
  - Text.

SVG feature set

- Nested transformations,
- Clipping paths,
- Alpha masks,
- Filter effects and
- Template objects.

SVG drawings and animations

- Can be interactive and dynamic.
- Can be defined and triggered either declaratively (i.e., by embedding SVG animation elements in SVG content) or via scripting.

HTML 5 DEPENDENCIES

e.g. JavaScript, CSS etcetera

Caveat: Not everything is included here. Refer to HTML5 specification for more.

Unicode and encoding

- The Unicode character set is used to represent textual data, and the Encoding standard defines requirements around character encodings.
  - This specification introduces terminology based on the terms defined in those specifications, as described earlier.
- The following terms are used as defined in the Encoding specification.
- Getting an encoding
  - The encoder and decoder algorithms for various encodings, including the UTF-8 encoder and UTF-8 decoder. The generic decode algorithm which takes a byte stream and an encoding and returns a character stream. The UTF-8 decode algorithm which takes a byte stream and returns a character stream, additionally stripping one leading UTF-8 Byte Order Mark (BOM), if any
  - The UTF-8 decoder is distinct from the UTF-8 decode algorithm. The latter first strips a Byte Order Mark (BOM), if any, and then invokes the former.
- For readability, character encodings are sometimes referenced in this specification with a case that differs from the canonical case given in the encoding standard. (For example, "UTF-16LE" instead of "utf-16le".)

Extensible Markup Language

- Implementations that support the XHTML syntax must support some version of XML, as well as its corresponding namespaces specification, because that syntax uses an XML serialization with namespaces.

Uniform Resource Locator

- The following terms are defined in the URL standard:

• URL • Absolute URL • Relative URL • Relative schemes • The URL parser • Parsed URL • The scheme component of a parsed URL • The scheme data component of a parsed URL • The username component of a parsed URL • The password component of a parsed URL • The host component of a parsed URL • The port component of a parsed URL • The path component of a parsed URL • The query component of a parsed URL • The fragment component of a parsed URL • Parse errors from the URL parser • The URL serializer • Default encode set • Percent encode • UTF-8 percent encode • Percent decode • Decoder error • URLUtils interface • href attribute • protocol attribute • The get the base hook for URLUtils • The update steps hook for URLUtils • The set the input algorithm for URLUtils • The query encoding of an URLUtils object • The input of an URLUtils object • The url of an URLUtils object

Typed arrays

- The ArrayBuffer and ArrayBufferView interfaces and underlying concepts from the Typed Array Specification are used for several features in this specification.
- The Uint8ClampedArray interface type is specifically used in the definition of the canvas element's 2D API.

File API

- This specification uses the following features defined in the File API specification:
  - Blob
  - File
  - FileList
  - Blob.close()
  - Blob.type
  - The concept of read errors

Media Queries

- Extend the functionality of media types by allowing more precise labeling of style sheets.
- Consists of a media type and zero or more expressions that check for the conditions of particular media features. Among the media features that can be used in media queries are ‘width’, ‘height’, and ‘color’.
- By using it, presentations can be tailored to a specific range of output devices without changing the content itself.

Cascading Style Sheet

- Some features require that a string be parsed as a CSS <color> value. When parsing a CSS value, user agents are required by the CSS specifications to apply some error handling rules.
- For example, user agents are required to close all open constructs upon finding the end of a style sheet unexpectedly. Thus, when parsing the string "rgb(0,0,0" (with a missing close-parenthesis) for a color value, the close parenthesis is implied by this error handling rule, and a value is obtained (the color 'black'). However, the similar construct "rgb(0,0, " (with both a missing parenthesis and a missing "blue" value) cannot be parsed, as closing the open construct does not result in a viable value.

CSS continuation

- CSS element reference identifier is used as defined in the CSS Image Values and Replaced Content specification to define the API that declares identifiers for use with the CSS 'element()' function.
- Similarly, the term provides a paint source is used as defined in the CSS Image Values and Replaced Content specification to define the interaction of certain HTML elements with the CSS 'element()' function.
- The term default object size is also defined in the CSS Image Values and Replaced Content specification.

CSSOM specifications, required for scripting

- Screen
- LinkStyle
- CSSStyleDeclaration
- cssText attribute of CSSStyleDeclaration
- StyleSheet
- sheet
- disabled
- Alternative style sheet sets and the preferred style sheet set
- Serializing a CSS value
- Scroll an element into view
- Scroll to the beginning of the document

NAMESPACES

Definition

- An abstract container for various items; each item within a namespace has a unique name, but the namespace allows disambiguation of items with the same name that are in different namespaces.
- The mechanism for managing names in a distributed way that greatly reduces the likelihood that two independent parties will create the same name for different purposes.

Uniform Resource Identifier

- A namespace name and a set of local names.
- It leverages the URI allocation mechanism [with the Disposition of Names in an XML Namespace]
- A string of characters used to identify a name of a resource.
  - Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols.
  - Schemes specifying a concrete syntax and associated protocols define each URI. The most common form of URI is the uniform resource locator (URL), frequently referred to informally as a *web address*.
  - More rarely seen in usage is the uniform resource name (URN), which was designed to complement URLs by providing a mechanism for the identification of resources in particular namespaces.

URI syntax

- It consists of a URI scheme name (such as "http", "ftp", "mailto", "crid" or "file") followed by a colon character, and then by a scheme-specific part. The specifications that govern the schemes determine the syntax and semantics of the scheme-specific part.
- It does require all schemes to adhere to a general syntax that (among other things) reserves certain characters for special purposes (without always identifying those purposes).
- It also enforces restrictions on the scheme-specific part in order to (for example) provide for a degree of consistency when the part has a hierarchical structure.

URI, subclasses and their relationships

- A uniform resource name (URN) functions like a person's name, while
- A uniform resource locator (URL) resembles that person's street address. In other words: the URN defines an item's identity, while the URL provides a method for finding it.

URL

- A URL is a URI that, in addition to identifying a web resource, specifies the means of acting upon or obtaining the representation, specifying both its primary access mechanism and network location.
- For example, the URL http://example.org/wiki/Main_Page refers to a resource identified as /wiki/Main_Page whose representation, in the form of HTML and related code, obtainable via HyperText Transfer Protocol (http) from a network host whose domain name is example.org.

URN

- A URN is a URI that identifies a resource by name in a particular namespace. A URN can be used to talk about a resource without implying its location or how to access it.
- The International Standard Book Number (ISBN) system for uniquely identifying books provides a typical example of the use of URNs. ISBN 0-486-27557-4 cites unambiguously a specific edition of Shakespeare's play *Romeo and Juliet*. The URN for that edition would be *urn:isbn:0-486-27557-4*. To gain access to this object and read the book, its location is needed, for which a URL would have to be specified.

Digital Object Identifier

- A platform independent and created to an agreed international standard formally specified in ISO 26324.
- Identify content permanently.
- They are coupled with metadata.
- They can be modified over time to keep track of the locations and characteristics of the objects they identify, both for your and your users.
- It benefits efficient management and accurate tracking, as well as gaining the ability to more easily automate processes and collaborate with partners.
- Adds value to content.

DOI

- It is a Digital Identifier of an Object. Not Identifier of Digital Object.
- Example
  - 10.1234/456-mydoc-456584893489
- It has Prefix which always begins with 10, indicating this is a DOI name.
  - The 2nd part of the prefix is typically 4 digits and is allocated to the DOI assigner.
- The Suffix is created by the assigner, and can be any length and incorporate other number schemes including an existing identifier if desired

DOI handle system (handle.net)

- Unique and persistent identifiers for Internet resources.
- Provides efficient, extensible and secure resolution services for unique and persistent identifiers of digital objects and is a component of Corporation for National Research Initiatives' Digital Object Architecture.

DOA

- Provides a means of managing digital information in a network environment.
- A digital object has a machine and platform independent structure that allows it to be identified, accessed and protected, as appropriate.
- A digital object may incorporate not only informational elements, i.e., a digitized version of a paper, movie or sound recording, but also the unique identifier of the digital object and other metadata about the digital object.
- The metadata may include restrictions on access to digital objects, notices of ownership, and identifiers for licensing agreements, if appropriate.

DOI System significant difference from other identifiers

- Purpose: The purpose of an identifier registry is to manage a given collection of identifiers; the primary purpose of the DOI System, on the other hand, is to make a collection of identifiers actionable and interoperable, where that collection can include identifiers from many other controlled collections.
- Coverage: A registry aims to be definitive and comprehensive across its content type; the DOI System is not intended to be a comprehensive identifier registry of all items falling potentially within its scope, but any such item may be registered as a DOI name.
- Scope: The scope of a standard registry is defined and fixed by content type (e.g. books and ISBN). The scope of the DOI System is potentially any resource involved in an intellectual property transaction; hence the coverage of DOI names is extensible (actual use expands continually as new areas of application are created): "The scope of the DOI system is not defined by reference to the type of content (format, etc) of the referent, but by reference to the functionalities it provides and the context of use". (ISO 26324 FDIS, Introduction)
- Granularity: The granularity of a content registry is typically defined and fixed by content type. The DOI System may be applied at any desired level of functional granularity, which may be modified by creating supersets or sub sets (including related types).

WRITING SECURE APPLICATIONS WITH HTML

Security model of the Web

- The security model of the Web is based on the concept of “origins”.
- Potential attacks on the Web involve “cross origin attacks”.
- When accepting untrusted input like for example user generated content such as
  - Text comments,
  - Values in URL parameters
  - Messages from third party sites
    - It is imperative that the data be validated before use, and properly escaped when displayed.

Filtering and validating user input

- Categorize into
  - Whitelist-based
    - Allowing known-safe constructs and disallowing all other input.
  - Blacklist-based
    - Disallow known bad inputs and prevent everything else.
    - Caveat – not everything that is bad is known (they might still be invented in the future)

Writing whitelist/blacklist filters

- When allowing harmless-seeming elements like `img`, it is important to whitelist any provided attributes as well. If one allowed all attributes then an attacker could, for instance, use the `onload` attribute to run arbitrary script.
- When allowing URLs to be provided (e.g. for links), the scheme of each URL also needs to be explicitly whitelisted, as there are many schemes that can be abused. The most prominent example is "javascript: ", but user agents can implement (and indeed, have historically implemented) others.
- Allowing a base element to be inserted means any script elements in the page with relative links can be hijacked, and similarly that any form submissions can get redirected to a hostile site

Web security attack, cross-site scripting

- Suppose a page looked at its URL's query string to determine what to display, and the site then redirected the user to that page to display a message, as in:

```
<ul>
<li><a href="message.cgi?say=Hello">Say Hello</a>
<li><a href="message.cgi?say=Thank You">Say Thank You</a>
<li><a href="message.cgi?say=Welcom">Say Welcome</a>
</ul>
```

- If the message was just displayed to the user without escaping, a hostile attacker could then craft a URL that contained a script element:

```
http://example.com/message.cgi?say=%3Cscript%3Ealert%28%27Oh%20no%21%27%29%3C/script%3E
```

- If the attacker then convinced a victim user to visit this page, a script of the attacker's choosing would run on the page. Such a script could do any number of hostile actions, limited only by what the site offers: if the site is an e-commerce shop, for instance, such a script could cause the user to unknowingly make arbitrarily many unwanted purchases.

Web security attack, cross-site request forgery

- If a site allows a user to make form submissions with user-specific side-effects, for example posting messages on a forum under the user's name, making purchases, or applying for a passport, it is important to verify that the request was made by the user intentionally, rather than by another site tricking the user into making the request unknowingly.
- This problem exists because HTML forms can be submitted to other origins.
- Sites can prevent such attacks by populating forms with user-specific hidden tokens, or by checking Origin headers on all requests.

Web security attack, clickjacking

- A page that provides users with an interface to perform actions that the user might not wish to perform needs to be designed so as to avoid the possibility that users can be tricked into activating the interface.
- One way that a user could be so tricked is if a hostile site places the victim site in a small `iframe` and then convinces the user to click, for instance by having the user play a reaction game. Once the user is playing the game, the hostile site can quickly position the iframe under the mouse cursor just as the user is about to click, thus tricking the user into clicking the victim site's interface.
- To avoid this, sites that do not expect to be used in frames are encouraged to only enable their interface if they detect that they are not in a frame (e.g. by comparing the `window` object to the value of the `top` attribute).

Pitfalls to avoid when scripting APIs

- Run-to-completion semantics are scripts in HTML have.
  - Meaning that the browser will generally run the script uninterrupted before doing anything else, such as firing further events or continuing to parse the document.
- On the other hand, parsing of HTML files happens asynchronously and incrementally, meaning that the parser can pause at any point to let scripts run. This is generally a good thing, but it does mean that authors need to be careful to avoid hooking event handlers after the events could have possibly fired.

Scripting APIs reliable techniques

- Use event handler content attributes, or
- Create the element and add the event handlers in the same script.
- The latter is safe because, as mentioned earlier, scripts are run to completion before further events can fire.

Scripting APIs code sample

- One way this could manifest itself is with img elements and the load event. The event could fire as soon as the element has been parsed, especially if the image has already been cached (which is common).
- Here, the author uses the onload handler on an img element to catch the load event:

```
<img src="games.png" alt="Games" onload="gamesLogoHasLoaded(event)">
```

- If the element is being added by script, then so long as the event handlers are added in the same script, the event will still not be missed:

```
<script>
var img = new Image();
img.src = 'games.png';
img.alt = 'Games';  img.onload = gamesLogoHasLoaded;
// img.addEventListener('load', gamesLogoHasLoaded, false); //would work
</script>
```

Scripting APIs code sample continuation

- However, if the author first created the img element and then in a separate script added the event listeners, there's a chance that the load event would be fired in between, leading it to be missed:

```
<!—Don’t use this style, it has a race condition!-->
<img id="games" src="games.png" alt="Games">
<!--the 'load' event might fire here while the parser is taking a break, in which case you will not see it!-->
<script>
var img = document.getElementById('games');
img.onload = gamesLogoHasLoaded; // might never fire!
</script>
```

PHP

Setup PHP development environment

- Ways to do it in various platforms can be found here:
  - http://php.net/manual/en/install.php
- In particular to Windows follow the instruction provided earlier using
  - Internet Information Server from your workstation (Windows 7)
  - Web Platform Installer, handles installation of
    - PHP v6
    - Acquia Drupal 7
    - SQL Server 2008 R2 Service Pack 2

PHP Hypertext Preprocessor

- It is a scripting language designed to fill the gap between SSI (Server Side Includes) and Perl, intended for the web environment.
- Its principal application is the implementation of web pages having dynamic content.
- It has gained quite a following in recent times, and it is one of the frontrunners in the Open Source software movement.
- Its popularity derives from its C-like syntax, and its simplicity.

Preprocessor

- Means that PHP makes changes before the HTML page is created. This enables developers to create powerful applications which can publish a blog, remotely control hardware, or run a powerful website.

PHP uses and implementations

- Commercial web hosting
- Drupal
- Moodle

PHP default and new platforms compatibility

- MySQL, Apache
- MS SQL, IIS, Azure

phpinfo()

- Reflects or used to check PHP configuration settings and for available predefined variables on a given system.
- A valuable debugging tool, contains all EGPCS (environment, GET, POST, Cookie, Server) data.

phpinfo() constant and options

Name (constant)	Value	Description
INFO_GENERAL	1	The configuration line, *php.ini* location, build date, Web Server, System and more.
INFO_CREDITS	2	PHP Credits. See also phpcredits().
INFO_CONFIGURATION	4	Current Local and Master values for PHP directives. See also ini_get().
INFO_MODULES	8	Loaded modules and their respective settings. See also get_loaded_extensions().
INFO_ENVIRONMENT	16	Environment Variable information that's also available in *$_ENV*.
INFO_VARIABLES	32	Shows all predefined variables from EGPCS (Environment, GET, POST, Cookie, Server).
INFO_LICENSE	64	PHP License information. See also the » license FAQ.
INFO_ALL	-1	Shows all of the above.

phpinfo() examples

- The following shows all information and it default to INFO_ALL constant.

```
<?php
 phpinfo()
?>
```

- The following shows the predefined variables.

```
<?php
 phpinfo(INFO_VARIABLES)
?>
```

- In versions of PHP before 5.5, parts of the information displayed are disabled when the expose_php configuration setting is set to *off*. This includes the PHP and Zend logos, and the credits.

Language reference

- PHP tags
  - When PHP parses a file, it looks for opening and closing tags, which are *<?php* and *?>* which tell PHP to start and stop interpreting the code between them. Parsing in this manner allows PHP to be embedded in all sorts of different documents, as everything outside of a pair of opening and closing tags is ignored by the PHP parser.
  - PHP also allows for short open tags *<?* and *?>* (which are discouraged because they are only available if enabled with short_open_tag *php.ini* configuration file directive, or if PHP was configured with the **--enable-short-tags** option.
  - If a file is pure PHP code, it is preferable to omit the PHP closing tag at the end of the file. This prevents accidental whitespace or new lines being added after the PHP closing tag, which may cause unwanted effects because PHP will start output buffering when there is no intention from the programmer to send any output at that point in the script.

Parse

- Everything outside of a pair of opening and closing tags is ignored by the PHP parser which allows PHP files to have mixed content. This allows PHP to be embedded in HTML documents, for example to create templates.

```
<p>PHP ignored and displayed by the browser.</p>
<?php echo ‘Parsed by PHP.'; ?>
<p>PHP ignored and displayed by the browser.</p>
```

Comment

```
/*
This is a comment within PHP code spanning multiple
lines. This comment style is also being used by
other P.L.
*/


//Comment in single line within PHP.
```

Instruction separation

- As in C and Perl, PHP requires instructions to be terminated with a semicolon at the end of each statement. The closing tag of a block of PHP code automatically implies a semicolon; you do not need to have a semicolon terminating the last line of a PHP block. The closing tag for the block will include the immediately trailing newline if one is present.

```
<?php
    echo 'This is a test';
?>

<?php echo 'This is a test' ?>
```

Instruction separation

- The closing tag of a PHP block at the end of a file is optional (only for pure PHP code and nothing mixed on it e.g. HTML etcetera), and in some cases omitting it is helpful when using `include` or `require`, so unwanted whitespace will not occur at the end of files, and you will still be able to add headers to the response later. It is also handy if you use output buffering, and would not like to see added unwanted whitespace at the end of the parts generated by the included files.

```
<?php echo 'We omitted the last closing tag';
```

PHP operators

- Arithmetic
- Assignment
- Compare
- String
- Increment/decrement
- Array
- Logical
- Array

PHP operators, arithmetic

- Use to calculate given values.

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power (available in PHP 5.6)

PHP operators, arithmetic examples

```
<?php
$x=8;
$y=11;
echo ($x + $y); // outputs what?
echo ($x - $y); // outputs what?
echo ($x * $y); // outputs what?
echo ($x / $y); // outputs what?
echo ($x % $y); // outputs what?
?>
```

PHP operators, assignment

- Use to write a value to a variable.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP operators, assignment examples

```
<?php
$x=10;
echo $x; // outputs 10
$y=20;
$y += 100;
 echo $y; // outputs 120
$z=50;
$z -= 25;
echo $z; // outputs 25
// outputs 3
 ?>
```

PHP operators, strings and examples

- Use to manipulate texts and strings.

Operator	Name	Example	Result
.	Concatenation	$str1 = “Facula" $str2 = $str1 . " Industries!"	Now $txt2 contains “Facula Industries!"
.=	Concatenation assignment	$str1 = “Facula" $str1 .= " Industries!"	Now $str1 contains “Facula Industries!"

PHP operators, increment/decrement

Operator	Name	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP operators, comparison

Operator	Name	Example	Result
==	Equal	$x == $y	True if $x is equal to $y
===	Identical	$x === $y	True if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	True if $x is not equal to $y
<>	Not equal	$x <> $y	True if $x is not equal to $y
!==	Not identical	$x !== $y	True if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	True if $x is greater than $y
<	Less than	$x < $y	True if $x is less than $y
>=	Greater than or equal to	$x >= $y	True if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	True if $x is less than or equal to $y

PHP operators, logical

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP operators, array

- Use to compare array.

Operator	Name	Example	Result
+	Union	$x + $y	Union of $x and $y (but duplicate keys are not overwritten)
==	Equality	$x == $y	True if $x and $y have the same key/value pairs
===	Identity	$x === $y	True if $x and $y have the same key/value pairs in the same order and of the same types
!=	Inequality	$x != $y	True if $x is not equal to $y
<>	Inequality	$x <> $y	True if $x is not equal to $y
!==	Non-identity	$x !== $y	True if $x is not identical to $y

PHP types

- Support 8 primitive types
- Four scalar types:
  - boolean
  - integer
  - float (floating-point number, aka double)
  - string
- Two compound types:
  - array
  - object
- And finally two special types:
  - resource
  - NULL

boolean

- This is the simplest type. A boolean expresses a truth value. It can be either TRUE or FALSE.
- To specify a boolean literal, use the constants TRUE or FALSE. Both are case-insensitive.

boolean example

```
<?php
$x=TRUE;
$y=FALSE;
$z=$y & $x; //change it with arithmetic, logical,
            //array operators e.g. || et al
echo "This is the value of $z";
?>
```

Converting to boolean

- The following values are considered FALSE:
  - the boolean FALSE itself
  - the integer 0 (zero)
  - the float 0.0 (zero)
  - the empty string, and the string "0"
  - an array with zero elements
  - an object with zero member variables (PHP 4 only)
  - the special type NULL (including unset variables)
  - SimpleXML objects created from empty tags
- Every other value is considered TRUE (including any resource).

integer

- Can be specified in decimal (base 10), hexadecimal (base 16), octal (base 8) or binary (base 2) notation, optionally preceded by a sign (- or +).
- Binary integer literal was made available since PHP 5.4.0.
- To use octal notation, precede the number with a *0* (zero). To use hexadecimal notation precede the number with *0x*. To use binary notation precede the number with *0b*.
- To explicitly convert a value to integer, use either the *(int)* or *(integer)* casts. However, in most cases the cast is not needed, since a value will be automatically converted if an operator, function or control structure requires an integer argument. A value can also be converted to integer with the intval() function.

float

- Floating point numbers (also known as "floats", "doubles", or "real numbers") can be specified using any of the following syntaxes:

string

- It is series of characters, where a character is the same as a byte. This means that PHP only supports a 256-character set, and hence does not offer native Unicode support.
- A string literal can be specified in four different ways:
  - single quoted
  - double quoted
  - heredoc syntax
  - nowdoc syntax (since PHP 5.3.0)

string, escape characters

\n linefeed (LF or 0x0A (10) in ASCII)

\r carriage return (CR or 0x0D (13) in ASCII)

\t horizontal tab (HT or 0x09 (9) in ASCII)

\v vertical tab (VT or 0x0B (11) in ASCII) (since PHP 5.2.5)

\e escape (ESC or 0x1B (27) in ASCII) (since PHP 5.4.0)

\f form feed (FF or 0x0C (12) in ASCII) (since PHP 5.2.5)

\\ backslash

\$ dollar sign

\" double-quote

\[0-7]{1,3} the sequence of characters matching the regular expression is a character in octal notation

\x[0-9A-Fa-f]{1,2} the sequence of characters matching the regular expression is a character in hexadecimal notation

array

- In PHP it is actually an ordered map. A map is a type that associates values to keys.
- This type is optimized for several different uses; it can be treated as an array, list (vector), hash table (an implementation of a map), dictionary, collection, stack, queue, and probably more.
- As array values can be other arrays, trees and multidimensional arrays are also possible.

classes and objects

- Starting with PHP 5, the object model was rewritten to allow for better performance and more features. This was a major change from PHP 4. PHP 5 has a full object model.
- Among the features in PHP 5 are the inclusions of visibility, abstract and final classes and methods, additional magic methods, interfaces, cloning and typehinting.
- PHP treats objects in the same way as references or handles, meaning that each variable contains an object reference rather than a copy of the entire object.

resource

- It is a special variable, holding a reference to an external resource. Resources are created and used by special functions.
- As resource variables hold special handlers to opened files, database connections, image canvas areas and the like, converting to a resource makes no sense.
- The reference-counting system introduced with PHP 4's Zend Engine, a resource with no more references to it is detected automatically, and it is freed by the garbage collector. For this reason, it is rarely necessary to free the memory manually.
  - Persistent database links are an exception to this rule. They are *not* destroyed by the garbage collector.

null

- A special value represents a variable with no value. NULL is the only possible value of type null.
- A variable is considered to be null if:
  - it has been assigned the constant NULL.
  - it has not been set to any value yet.
  - it has been unset().