# SWEN1005 Mobile Web Programming Lecture Notes 1

My Name : Chris Smith

Email: chris.smith.bim@gmail.com

## Course Content

- Utilise the latest web standards to create effective mobile web applications.
- Compare and contrast the experience of native applications with browser based / wrapped applications.
- Design mobile web pages and web applications.
- Evaluate mobile web pages / mobile web applications.
- Mobile web development frameworks.

# What is HTML5, CSS3, JAVASCRIPT?

## History of HTML

1991 ---------- HTML First Publish

1995 ---------- HTML 2.0

1997 ---------- HTML 3.2

1999 ---------- HTML 4.01

- After HTML 4.01 was released, focus shifted to XHTML and its stricter standards.

2000 ---------- XHTML 1.0

2002/2009----- XHTML 2.0

XHTML 2.0 had even stricter standards than 1.0, rejecting web pages that did not comply. It fell out of favor gradually and was abandoned completely in 2009.

2012 ---------- HTML5

HTML5 is much more tolerant and can handle markup from all prior versions.

Though HTML5 was published officially in 2012, it has been in development since 2004.

# What is HTML5?

- HTML5 is the newest version of HTML, only recently gaining partial support by the makers of web browsers.
- It incorporates all features from earlier versions of HTML, including the stricter XHTML.
- It adds a diverse set of new tools for the web developer to use.
- It is still a work in progress. No browsers have full HTML5 support. It will be many years – perhaps not until 2018 or later - before being fully defined and supported.

## Goals of HTML5

- Support all existing web pages. With HTML5, there is no requirement to go back and revise older websites
- Reduce the need for external plugins and scripts to show website content
- Improve the semantic definition (i.e. meaning and purpose) of page elements
- Make the rendering of web content universal and independent of the device being used
- Handle web document errors in a more consistent way

## Other New Features in HTML5

- Built-in audio and video support (without plugins)
- Enhanced form controls and attributes
- The Canvas (draw directly on a web page)
- Drag and Drop functionality
- Support for CSS3
- More advanced features: data storage and offline applications.

## Basic HTML5 Web Page

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <title>My     First    HTML5    Page</title>
   <link   rel="stylesheet" href="style.css">
</head>
<body>
   <p>HTML5 is fun!</p>
</body>
</html>
```

## What is CSS?

- CSS (Cascading Style Sheets) allows us to apply formatting and styling to the HTML that builds our web pages.
- CSS can control many elements of our web pages: colors, fonts, alignment, borders, backgrounds, spacing, margins, and much more
- CSS works in conjunction with HTML.
- An HTML file (or multiple files) links to a CSS file (or multiple CSS files) and when the web browser displays the page, it references the CSS file(s) to determine how to display the content.

## How does CSS work?

- HTML elements are assigned “ID” and “class” attributes that are defined in the CSS file
- This is how the browser knows which styles belong where.
- Each element type (<h1>, <img>, <p>, <li>, etc.) can also be styled with CSS.
    - IDs and classes are defined by the person writing the code – there are no default IDs and classes.

## What is CSS3?

- The newest version of CSS designed to compliment HTML5
- What's new:
    - Selectors – selects elements to be presented
    - Pseudo-elements / classes
    - Properties and property groups
    - Animation, Transition, Transform
    - Background
    - Etc.

## What is JavaScript?

- A programming (scripting) language
    - Developed by Netscape
    - Relatively easy to learn
- JavaScript is most often used for client-side web development
- Facilitates interactivity through browser/page manipulation
    - Reacting to user actions
- Also increasingly used for server-side web application development
- JavaScript is an implementation of the ECMAScript standard: https://www.ecma-international.org/publications/standards/Ecma-262.htm

ECMAScript only defines the syntax/characteristics of the language and a basic set of commonly used objects such as Number, Date, Regular Expression, etc.

- The JavaScript implemented by browsers typically support additional objects.
  e.g., Window, Frame, Form, DOM object, etc.

### How Does It Work?

- Embedded within an HTML page
- Executes on the client
- Fast, no connection needed once loaded
- Programming statements can be combined with HTML tags
- Interpreted (not compiled)
- No special tools required

## NATIVE APPS, MOBILE WEB APPS AND HYBRID APPS

### Choosing a Platform

- Native apps:
  - Native apps live on the device and are accessed through icons on the device home screen.
  - Native apps are installed through an application store (such as Google Play or Apple's App Store).
  - They are developed specifically for one platform, and can take full advantage of all the device features — they can

use the camera, the GPS, the accelerometer, the compass, the list of contacts, and so on.
    - They can also incorporate gestures (either standard operating-system gestures or new, app-defined gestures).
    - Native apps can use the device's notification system and can work offline.

- Mobile Web apps: Uses standards-based technologies (HTML, CSS, JavaScript)
    - Mobile-specific site: Optimized for mobile devices
    - Responsive site: re-orients or arranges itself for mobile devices
    - Web apps are not real applications
    - They are really websites that, in many ways, *look and feel* like native applications.
    - They are not *implemented* as real apps are.
    - They are run by a browser and typically written in HTML5.

- Hybrid apps:
    - Loads content from the web but provides users with an "app-like" interface
    - Are part native apps, part web apps. (Because of that, many people incorrectly call them "web apps").
    - Like native apps, they live in an app store and can take advantage of the many device features available.
    - Like web apps, they rely on HTML being rendered in a browser, with the caveat that the browser is embedded within the app.
    - They allow crossplatform development and thus significantly reduce development costs: that is, the same HTML code components can be reused on different mobile operating systems.

| Consideration | Mobile | Responsive | Native | Hybrid | Commments |
|---|---|---|---|---|---|
| Tailored to user Priorities | *** | ** | *** | *** | A mobile-first approach can improve responsive design's rating to three stars |
| Content delivery | ** | *** | ** | *** | Responsive sites and hybrid apps are more easily indexed in search engines |
| Functionality | ** | ** | *** | *** | Native apps provide access to device features (e.g GPS,Camera) Allowing for a more engaging experience |
| Compatibility | ** | *** | * | * | Responsive design is easily viewed on any screen. Mobile sites and applications are dependent on the device for which they're designed. |
| Development costs | ** | *** | * | ** | Dependent on if you are developing a whole site from scratch. Responsive design incurs extra build time, but not as much mobile + conventional design. |
| Maintenance costs | ** | *** | * | ** | Individual native apps require individual maintenance. However, responsive design results in a single site that needs to be maintained. |

# Native apps

- Use APIs
  - Application program interface (API) is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components.
- Involves the use of a programming language and SDK (Software Development Kit)
  - Windows: C#/Visual Basic and XAML
  - iOS: Objective C and Cocoa Touch
  - Android: Java and Android SDK 24.4.1

## Native apps: Pros

- Integrates with the user's data: calendar, contact list, etc.
- Enables the capture and storage of photos and video via the device's camera
- Uses sensor data from the gyroscope, compass, GPS, etc.
- Accesses device diagnostics such as the battery or network status
- Supports graphic intensive applications
- Functions without network connectivity

## Native apps: Cons

- App deployment time consuming with potential publication delays
- Multiple platform deployment requires larger or multiple teams
    - Implies larger investment of time and money
    - Targeting one platform potentially limits app reach and adoption

## Mobile-specific site

- Created using HTML, CSS and JavaScript like all web sites.
- HTML – gives structure to the content (tables, headers, titles, paragraphs, etc.)
- CSS – Presents the content (font style, text colour, background colour, etc.)
- JavaScript – provides function via the web browser
- The server detects the user's device and redirects to the mobile web site

## Mobile-specific site: Pros

- Allows for a more suitable UI
- Smaller file sizes render faster on mobile devices
- Mobile templates are more economical

## Mobile-specific site: Cons

- More difficult to maintain two sites
- The simplified mobile version may not meet all the client needs

## Responsive site

- Created using HTML, CSS and JavaScript like all web sites.
- HTML – gives structure to the content
- CSS – Presents the content
- JavaScript – provides function via the web browser
- Serves the same code to both types of devices
- CSS and JavaScript reformat the web pages on the client side

## Responsive site - Pros

- More elegant and streamlined
- Future-proof web site development
- Easier to maintain

## Responsive site - Cons

- Requires a larger budget and more work on UI/UX design
- Large files will be slow on mobile devices

## Hybrid Solutions

- Consider a natively built and deployed app whose sole interface is a web view control that takes up the entire screen of the device.
- All of the user interface and interactions can be built using traditional web development practices.
- The device-specific features that are not normally available to web apps, such as the microphone or the notification center can be made available through JavaScript.
- This is possible because many devices allow the JavaScript executing in a web view control to communicate with the native host app
- Some of the flexibility of this approach relates to where your web assets are stored.
    - They may be embedded in the native app itself, or they may be retrieved from the web.
    - Images, markup, style sheets, and scripts that aren't likely to change can often be bundled with the app to improve load times.
- Other assets (those that will likely change) can be downloaded as needed from remote servers.

- Hybrid apps: benefit from app storefront deployment with a smaller investment than native solutions
- Not perfect for all scenarios
  - Share the same deployment constraints as native solutions
  - More time consuming to publish new features or fixes compared to web-only solutions.
- Reach is broader than for a native app because the codebase remains more consistent across the targeted platforms

## How Do We Choose?

- Consider the apps requirements now and for the future
- Determine the impact it will have on user experience
- Evaluate the skill level of the development team
- Three major factors to consider:
  - Investment
  - Features
  - Reach
- Investment. Both the time and money required to build, deploy, and maintain the app must be considered. This includes team salaries, hosting, and maintenance costs.

- Features. The features your app needs will play an important role in your decision.
- Reach. The number of users you can reach will influence which approach you take.