SWEN1005

MOBILE WEB PROGRAMMING

LECTURE 002

Course Content

- Section 1: Utilise the latest web standards to create effective mobile web applications.
- Section 2: Compare and contrast the experience of native applications with browser based / wrapped applications.
- Section 3: Design mobile web pages and web applications.
- Section 4: Evaluate mobile web pages / mobile web applications.
- Section 5: Mobile web development frameworks.

Section One

- Identify mobile-specific site, responsive site, native apps, and hybrid apps
- Describe mobile-specific site, responsive site, native apps, and hybrid apps
- Identify the most common web standards
- Outline the need for web standards
- Analyse mobile web pages and content

Section Two

- Distinguish among mobile-specific sites, responsive sites, native apps, and hybrid apps
- Compare mobile-specific sites, responsive sites, native apps, and hybrid apps

Section Three – mainly during labs

- Use HTML and CSS to create mobile web applications
- Use Offline API
- Use Geolocation
- Use JQuery/JavaScript to create web applications
- Use wrappers prepare HTML applications for the app stores
- Create mobile design patterns e.g. lists, feedback, hierarchy, hub & spoke navigation, etc.
- Create web pages using HTML/CSS/JavaScript

Section Four

- Research and assess established mobile web page and web application standards
- Assess mobile-friendly web content
- Compare and contrast the content of any two popular mobile web sites, or applications, based on the established criteria
- Testing mobile web site and applications

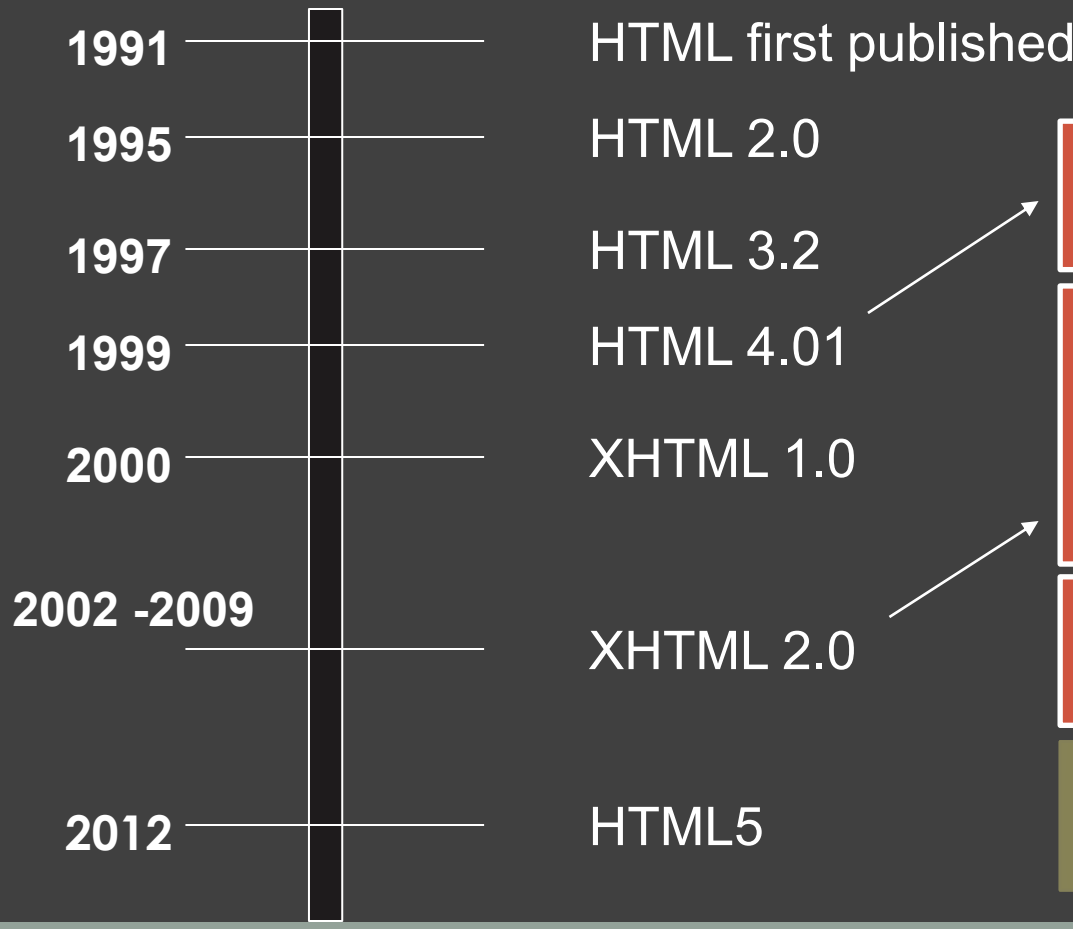
Section Five

- Critically assess popular mobile web development frameworks:
  - React Native
  - Flutter
  - Apache Cordova (PhoneGap)
  - jQuery Mobile
  - Sencha Ext JS
  - Ionic
- Create a simple mobile site using one of the most common frameworks

Section One

WHAT IS HTML5, CSS3, JAVASCRIPT

History of HTML



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

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.

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.

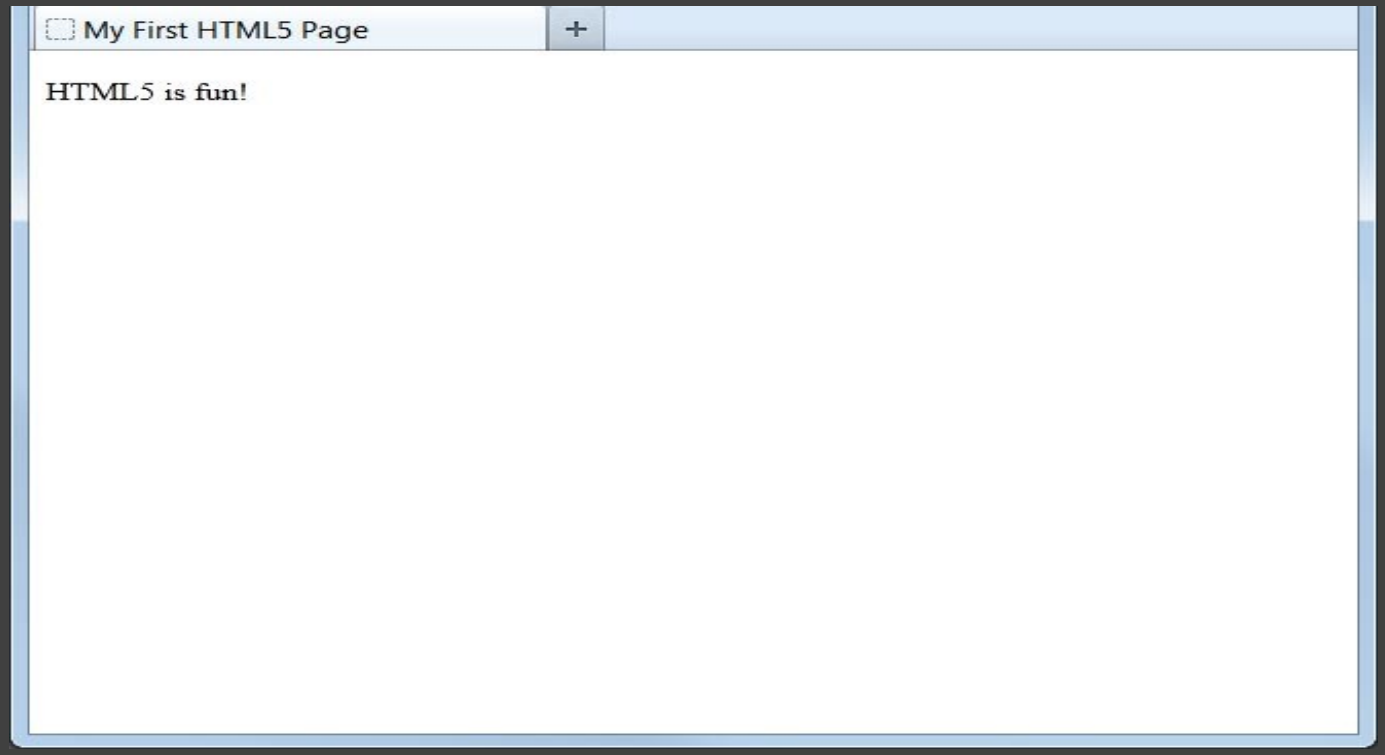
Basic HTML5 Web Page

```
<!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>
```

Putting the prior sections together, and now adding the <body> section and closing tags, we have our first complete web page in HTML5:

Let's open this page in a web browser to see how it looks…

Viewing the HTML5 Web Page



Even though we used HTML5, the page looks exactly the same in a web browser as it would in XHTML. Without looking at the source code, web visitors will not know which version of HTML used to create the page

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?

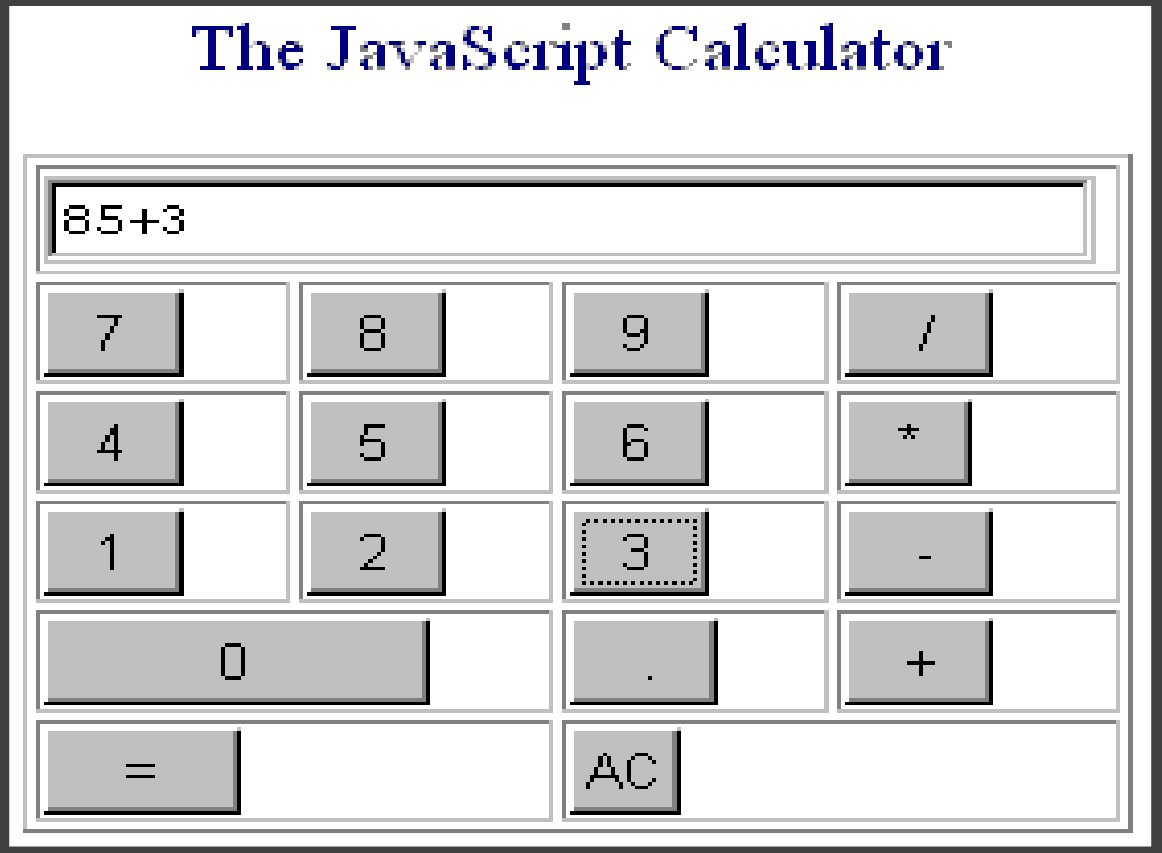
- A programming (scripting) language
  - Developed by Netscape
  - Relatively easy to learn
- JavaScript is most often used for client-side web development
- Facilitates interactivity through browser/page manipulation
  - Reacting to user actions
- Also increasingly used for server-side web application development

What is JavaScript?

- 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.

JavaScript and Interactivity

- Improve appearance - especially graphics with visual feedback
- Site navigation
- Perform calculations
- Validation



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

SWEN1005

MOBILE WEB PROGRAMMING

Section One

NATIVE APPS, WEB APPS AND HYBRID APPS

First things first – Choosing a Platform

- Native app: Language specific
- Web solution: Uses standards-based technologies (HTML, CSS, JavaScript)
  - Mobile-specific site: Optimized for mobile devices
  - Responsive site: re-orients or arranges itself for mobile devices
- Hybrid app: Loads content from the web but provides users with an “app-like” interface

Pros and cons

Consideration	Mobile	Responsive	Native	Hybrid	Comments
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 used across more devices.
Functionality	★★	★★	★★★	★★★	Native apps provide access to device features (e.g. GPS, Camera), allowing more engaging experiences.

Pros and cons

Consideration	Mobile	Responsive	Native	Hybrid	Comments
Compatibility	★★	★★★	★	★	Responsive design is easily viewed on any screen. Mobile sites and applications are dependent on the device for which they're designed.

Pros and cons

Consideration	Mobile	Responsive	Native	Hybrid	Comments
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.

SWEN1005

MOBILE WEB PROGRAMMING

Section One

NATIVE APPS, WEB APPS AND HYBRID APPS

Native apps

- Use APIs
  - Application program interface (API) is a set of routines, protocols, and tools for building software applications.
- Involves the use of a programming language and SDK (Software Development Kit)
  - Windows: C#/Visual Basic and XAML
  - iOS: Objective C, Swift and Cocoa Touch
  - Android: Java and Android SDK 26.1.1 (September 2017)

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 is 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 the user 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 clients 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
  - Search engine optimization covers all bases

Responsive site - Cons

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

Hybrid Solutions

- A natively built and deployed app with a full screen web view control
- The user interface can be built using web development practices
- Device-specific features, such as the microphone, is accessible using JavaScript
- JavaScript can communicate with the native host app when in a web view control

Hybrid solutions

- Some of the flexibility is due to web app asset storage
  - Embedded in app itself
  - Retrieved from the web
  - Bundled with the app to improve load times
- Other assets can be downloaded as needed from remote servers.

Hybrid solutions

- Benefit from app storefront deployment
- Not perfect for all scenarios
  - Same deployment constraints as native apps
  - More time consuming to maintain that web-only solutions
- Reach is broader than a native app
  - Codebase is more consistent across 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

How Do We Choose?

- **Investment**. Both the time and money to build the app
- **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.

Summary

- Understand the advantages and disadvantages of each platform
- Choice is determined mainly target audience usage
- Third-party frameworks can be very useful
  - Examine the advantages and disadvantages