Mobile Testing – Survival Knowledge – Part VII

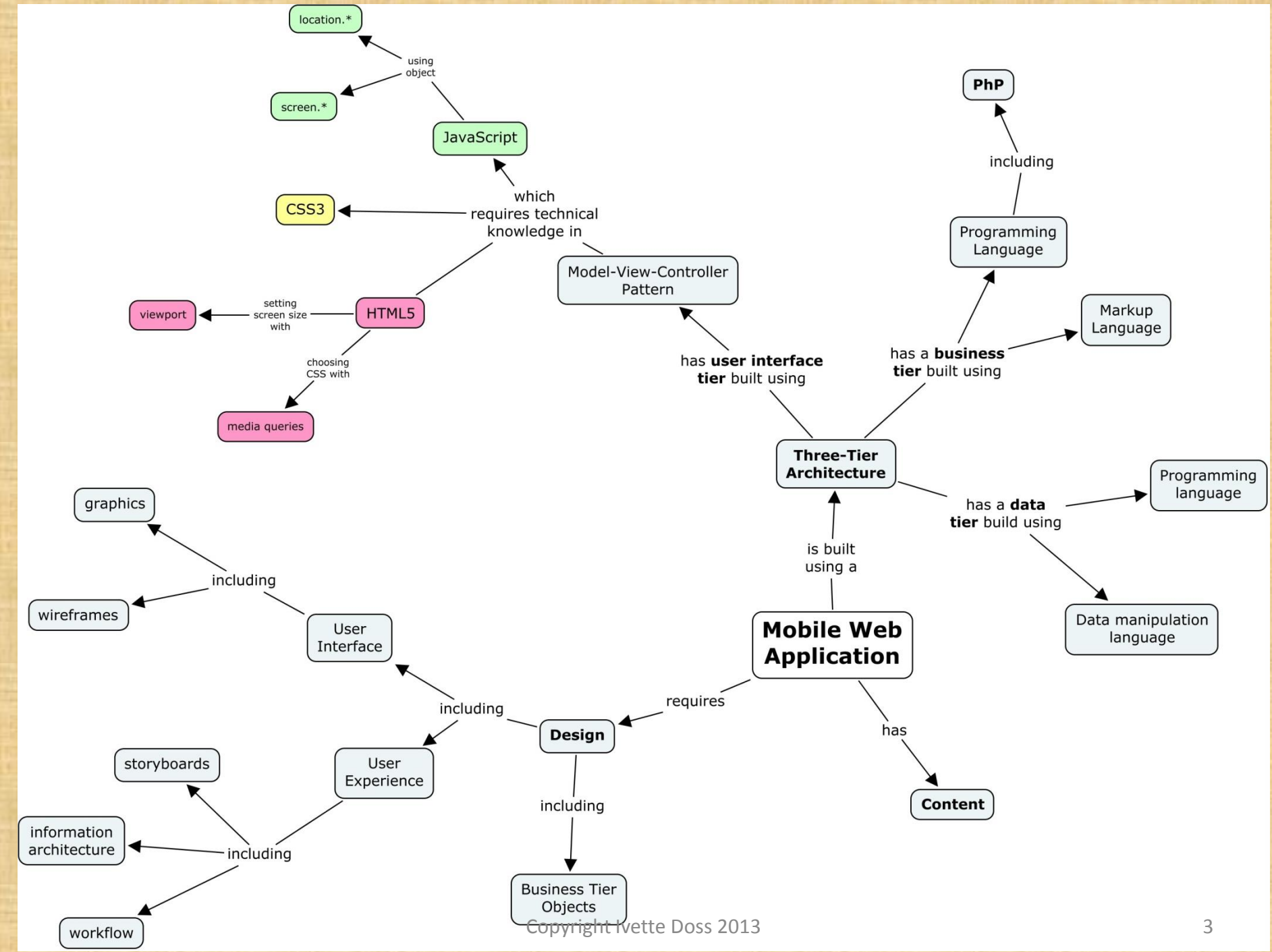
Created by Ivette Doss

Objective today:

6. Practicum 1 – Web Mobile Application: Introduction to Web Mobile Application – API: Web Design and Mobile Testing – specific of Web Mobile Application Testing – Interview Questions and Answers
Homework: Write 20 Test Cases that will provide the Full Test Coverage for the Mobile Web Application

https://mobile.dominos.com/mobile/wap_servlet?command_mobilehome

http://www.w3.org/2013/06/mobile-web-app-state/



The Difference between Mobile and Standard PC Website

- Reducing and simplifying of content
- Minimization of code
- Increasing of loading speed
- Simplification of text input and navigation

Advantages of the Web Mobile Application

- **1**. **Direct control over distribution:**
- You can update content on your web page yourself without involving a third party. Content update is fast; and its results are seen immediately.
- **2**. **Reducing of development costs.**
- Web sites are launched in browsers what makes their customization easier in terms of a developer due to creating a single web site for many platforms. However, optimization of mobile web sites for different operating systems also makes sense to enable their specific features.
- **3**. **Easy to find** in the Internet, e.g. through search results
- **4. Shorter time to market**
- However, an important **disadvantage** of mobile web site is the situation when your client has no access to the Internet or traffic is very expensive.

Mobile Web Applications

Furthermore, since these apps run on common browsers, device-specific customization/update delivery is much simpler. Cost benefit increases as hardware fragmentation increases.

Most suitable for Mobile Web Applications are:

- retail and shopping,
- communications
- news
- weather publishers,
- applications where iterative design and capturing user analytics are essential.

Main reasons for organizations to choose: 1

NATIVE

- Complies with company's own technological expertise (Java, C++ and/or Objective C are mastered better than HTML)
- uses makes creating of an excellent
- contains relevant content (e.g. an interactive game)
- performs complex calculations (bank payments, etc.)

WEB

- Suits best for chosen model of business management and promotion
- complies with company's own technological expertise (HTML is mastered better than C + + or Objective C).

BOTH: Native and Web

- Opportunity to develop both options
- Desire to test what works best
- Expansion of contacts with users
- Other (order of company management, integrated solution)

Tendencies that let organizations to choose: 2

Develop Web Apps

- If your goals are firstly focused on marketing
- if you offer timely and quickly updated information
- if you want to be easily found through search engines
- if you want to save your money

Develop Native Apps

- Communication and interaction with users
- creating an application working like a computer program
- ensuring an instant access to information about your company, regardless of accessibility of the Internet
- creation of a creative and catchy user interface

YOUR personal opinion regarding the best option is:

•starting with mobile version of web site,

•finding out places where users are most active (using Analytics),

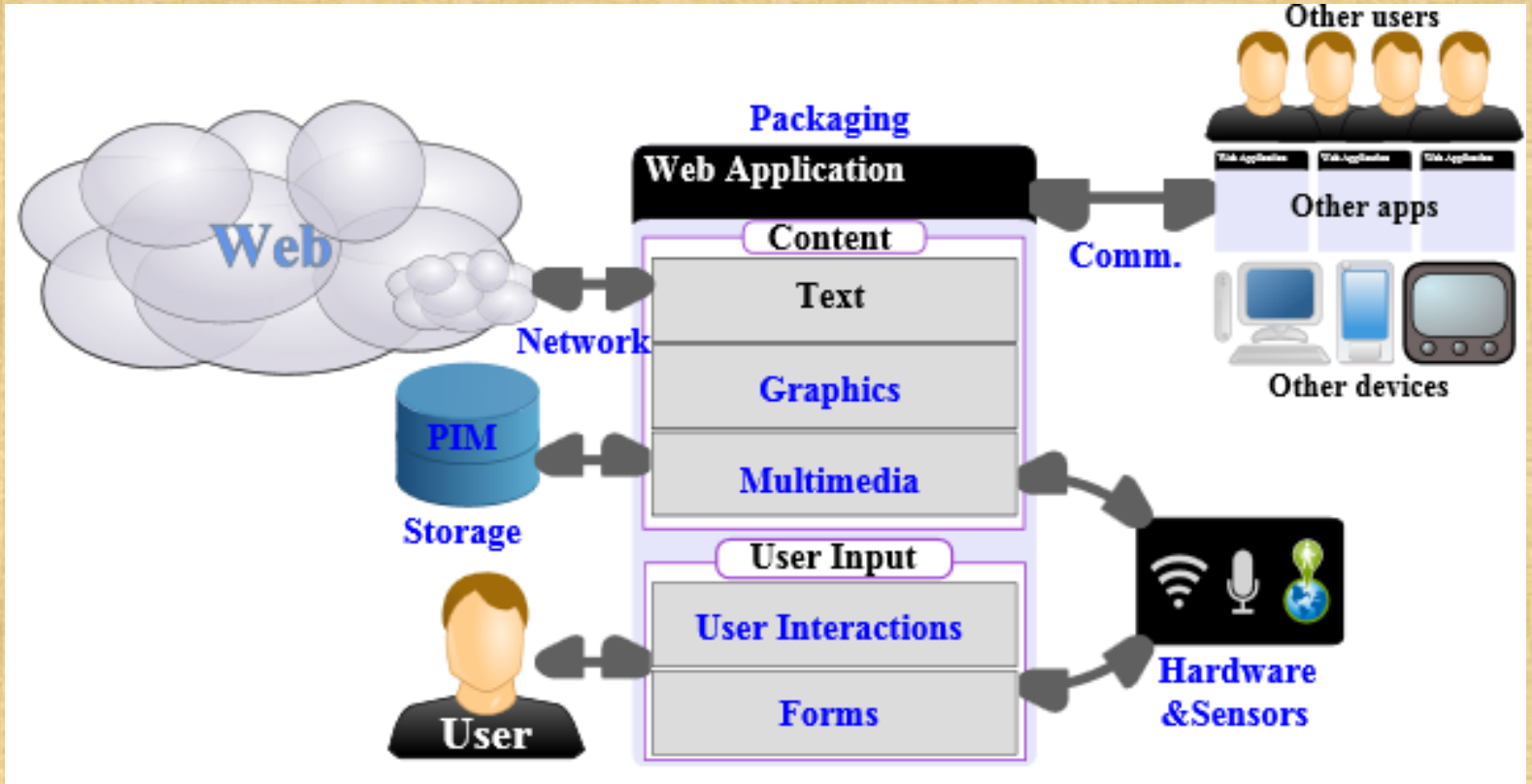
•creating a native mobile application based on this information.

Useful tips for last:

•If most people find company through search results, it is better to create a mobile version of your web site.

•If company already have a vast audience, it is better develop native mobile application extending the functionality of company website.

Web Mobile technology



Web platform technologies are organized under the following categories: graphics, multimedia, device adaptation, forms, user interactions, data storage, personal information management, sensors and hardware integration, network, communication and discovery, packaging, performance & optimization.

What is API?

- **An Application Programming interface** (API) is a set of functions, classes, libraries, or packages (a.k.a. frameworks) that allowing the programmer to access an application's services by using the programming languages.
- An API may include specifications for routines, data structures, object classes, and variables.

Generally speaking, an application programming interface (API) is a specification of how some software components should interact with each other.

API examples: Google Maps, Twitter, Instagram

http://blog.gmapify.fr/

Frameworks

Web development is about getting stuff done, not figuring out *how* to get it done. **Frameworks and libraries help the web developers focus on creating rather than figuring stuff out.**
Rather than reinventing the wheel, Developers can use a framework or library to delegate brunt, non-creative and repetitive work, freeing up their time and energy to create the actual website or application.

http://speckyboy.com/2011/03/07/20-new-frameworks-for-web-and-mobile-app-developers/

http://www.rhomobile.com/webinars-old/rhodes-testing-framework/

Framework- definition

- ***A software framework*** is an **abstraction in which software providing generic functionality can be selectively changed by additional User Written Code, thus providing application specific software**.
- A software framework is a universal, reusable software platform used to develop applications, products and solutions.
- Software frameworks include support programs, compilers, code libraries, an application programming interface (API) and tool sets that bring together all the different components to enable development of a project or solution**.**

For example:

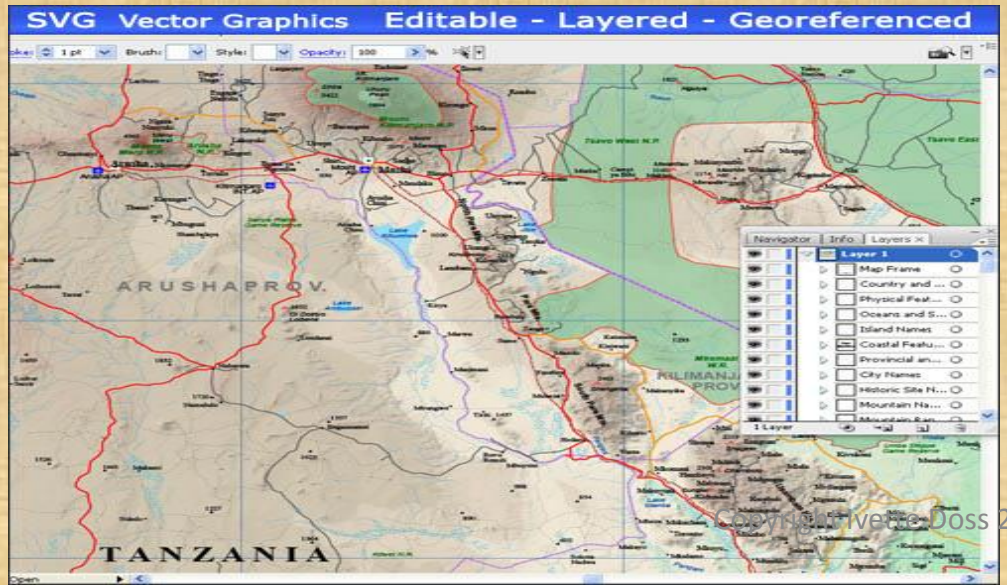
- **Platform**: Windows CE;
- **OS:** Windows Phone;
- **Framework:** .NET

- **Platform**: OS X;
- **OS**: iOS;
- **Framework**: Cocoa Touch

http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks

Graphics

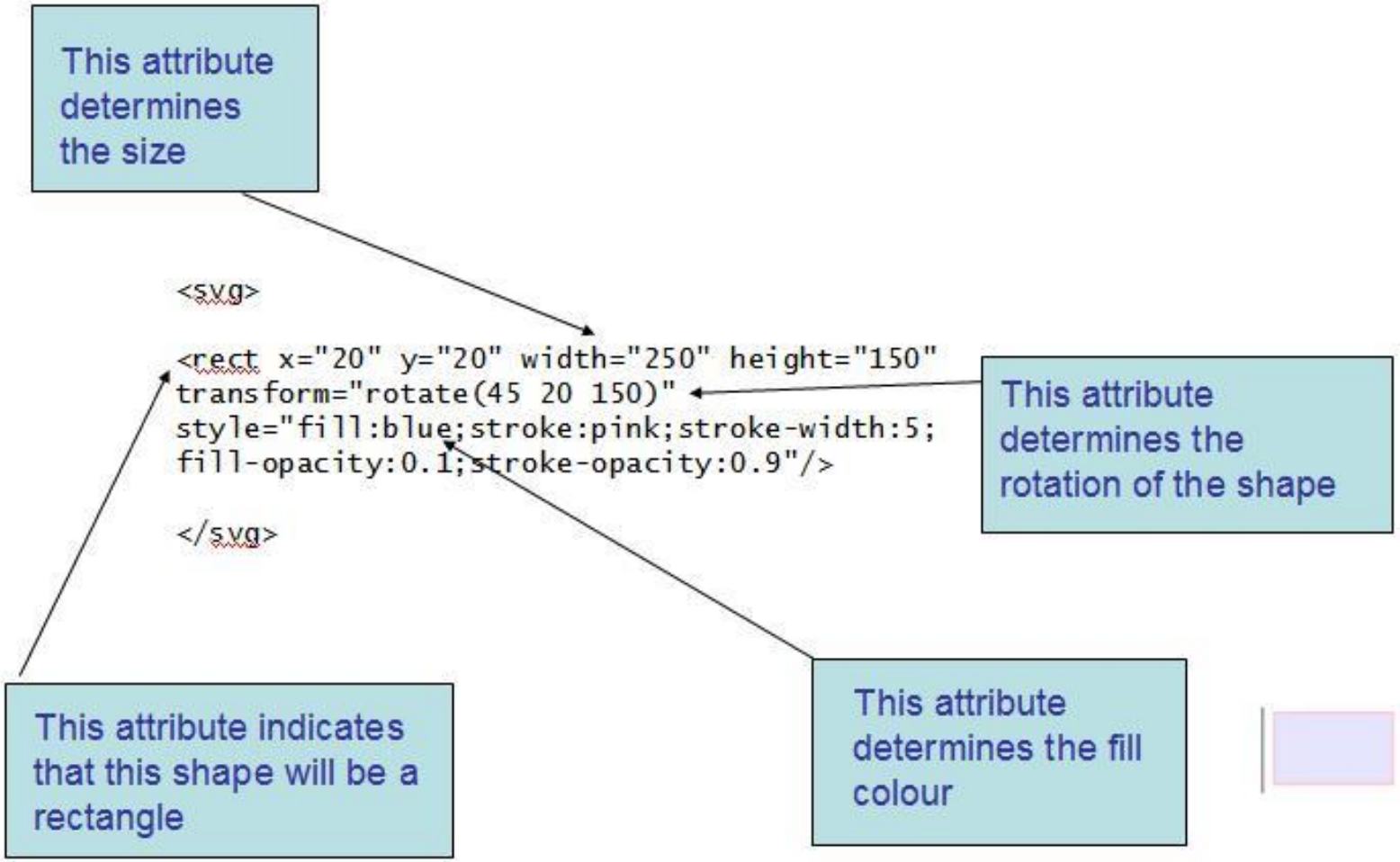
- **SVG**, **Scalable Vector Graphics**, provides an XML-based markup language to describe two-dimensions vector graphics.
- Since these graphics are described as a set of geometric shapes, they can be zoomed at the user request, which makes them well-suited to create graphics on mobile devices where screen space is limited. They can also be easily animated, enabling the creation of very advanced and slick user interfaces.
- The integration of SVG in HTML5 opens up new possibilities, for instance applying advanced graphic filters (through SVG filters) to multimedia content, including videos.



Scalable Vector Graphics (SVG)

SVG is a format for encoding vector graphics designed to appear in Web pages.

Text (rather than pixels) is used to describe the attributes of a shape, as shown:

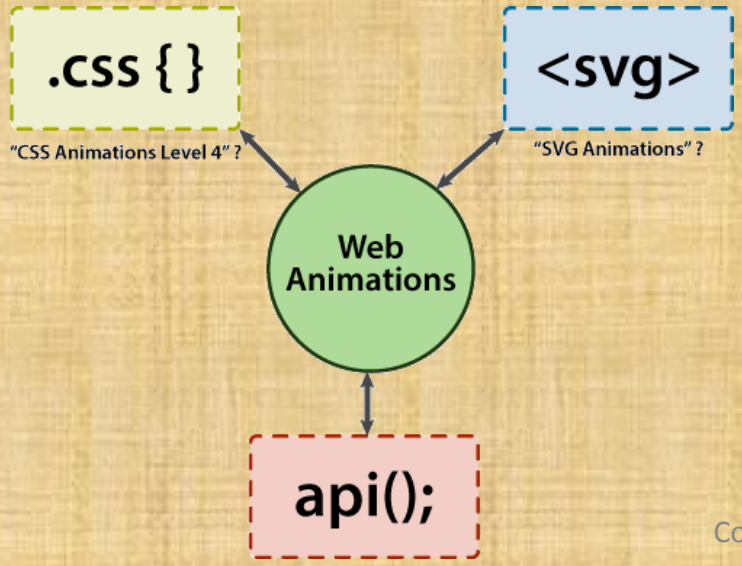


Cascading Style Sheets

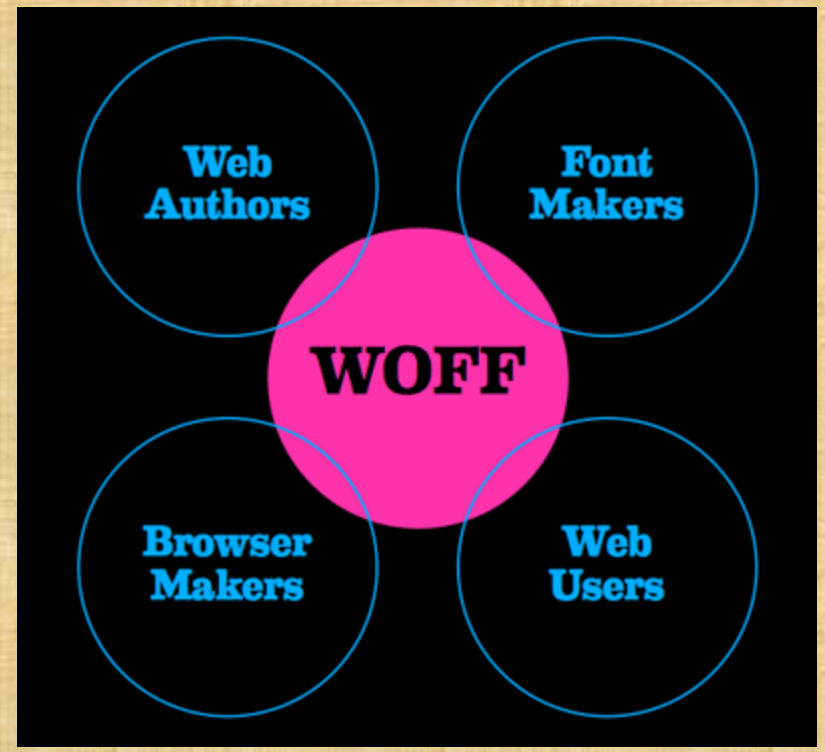
- Both SVG and HTML can be styled using **CSS.**
- CSS3 (the third level of the specification) is built as a collection of specifications set to offer a large number of new features that make it simple to create graphical effects, such as rounded corners, complex background images, shadow effects (*CSS Backgrounds and Borders*), rotated content (*CSS Transforms*, including with 3D effects), animations (*CSS Animations*, and *CSS Transitions*).

Web Animation

- Animations, which can also be managed via scripting through the API exposed in Web Animations, can be resource intensive — the possibility offered by the *Timing control for script-based animations API* to manage the rate of updates to animations can help keep them under control.



Fonts



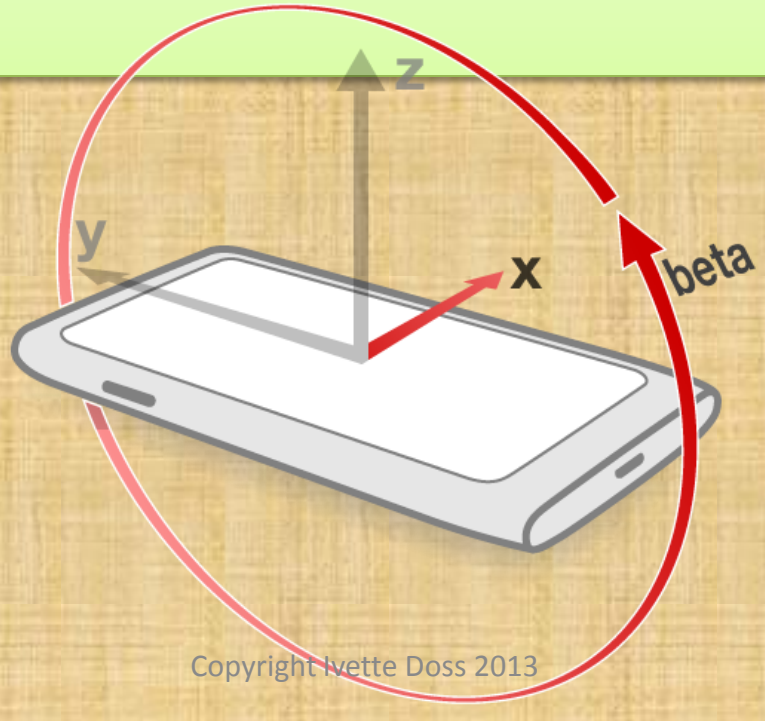
- Fonts play also an important role in building appealing graphical interfaces, but mobile devices are in general distributed with only a limited set of fonts.
- **WOFF** (*Web Open Font Format*) addresses that limitation by making it easy to use fonts that are automatically downloaded through style sheets, while keeping the size of the downloaded fonts limited to what is actually needed to render the interface.

Fullscreen

- Another important aspect in graphics-intensive applications (e.g. games) is the possibility to use the entire screen to display the said graphics; the ***Fullscreen API*** lets a Web application requests and detects full screen display.

Screen Orientation

- Likewise, in these scenarios, it is often useful to be able to **lock the orientation of the screen**; the *Screen Orientation API* allows not only to detect orientation change, but also to lock the orientation in a specific state.



3D graphic API for HTML5 canvas, called WebGL

Multimedia

- HTML5 adds two tags that dramatically improve the integration of multimedia content on the Web: the **<video>** and **<audio>** tags.
- Respectively, these tags allow embedding video and audio content, and make it possible for Web developers to interact much more freely with that content than they would through plug-ins.
- The playback content can be augmented and completed via Media Source Extensions that lets developers generate media content in JavaScript.

Media Intent

- The *Pick Media Intent* offers a Web-intent based approach to search and retrieve **locally or remotely stored media content**, while the *Networked Service Discovery* API opens the door for integrating DLNA-hosted content into Web applications.

Multimedia capture and record

- While the new HTML5 tags allow to play multimedia content, the *HTML Media Capture* defines a **markup-based mechanism to access captured multimedia content** using attached camera and microphones, a very common feature on mobile devices.
- The Web Real-Time Communications Working Group and the Device APIs Working Group are building together an API (getUserMedia) to directly manipulate **streams from camera and microphones**, as well as an API to record these streams into files.
- Beyond capturing and recording, two additional APIs add multimedia manipulation capabilities to the Web platform:
  - *Canvas 2D Context* API: it enables modifying images, which in turn opens up the possibility of **video editing**.
  - Audio Working Group is working on an API that that makes it possible to modify audio content, as well as **analyze, modify and synthesize sounds**, the Web Audio API.

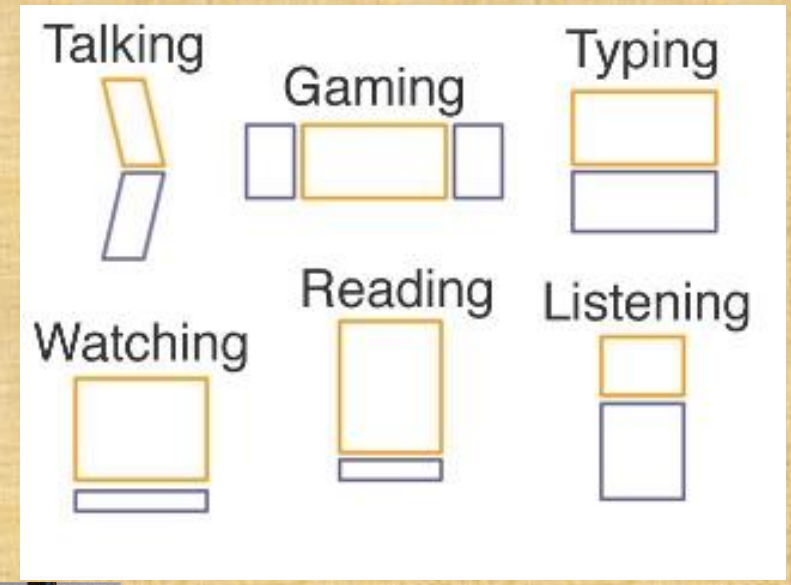
3. Device Adaptation

- Mobile devices not only differ widely from traditional computers, but they also have a lot of variations among themselves, in term of screen size, resolution, type of keyboard, media recording capabilities, etc.
- The **Device Description Repository API** is a unified server-side API that allows Web developers to retrieve data on the devices that are accessing their pages on a variety of device information database.
- The **Media Capture Streams API** exposes some specific information on capabilities of camera and microphones to make it possible to take advantage of the large variety of media capturing devices provided on mobile phones.
- **CSS Media Queries** offer a mechanism that allows adapting the layout and behavior of a Web page based on some of the characteristics of the device, including the screen resolution — to which **Media Queries Level 4** proposes to add the availability and type of a pointing device, the ability to hover over elements, and the ambient luminosity.
- **CSS Device Adaptation** defines a set of CSS directives to define the size on which this layout should be based, relatively to the size of the underlying device — specifying what has been implemented using the <meta name="viewport"> element so far.

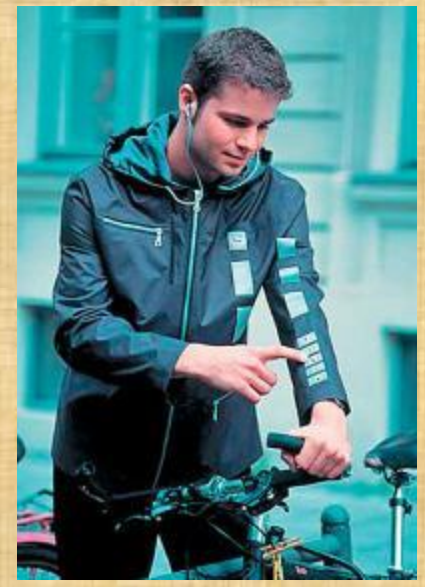
4. Forms

- The ability to build rich forms with HTML is the basis for user input in most Web-based applications.
- Due to their limited keyboards, text input on mobile devices remains a difficult task for most users; *HTML5* address parts of this problem by offering new type of form controls that optimize the way users will enter data.

Good to know: Mobile Form Factor



- **Brick**
- **Bar**
- **Touchscreen**
- **Taco**
- **Flip**
- **Slider**
- **Swivel**



http://conversations.nokia.com/2012/01/30/why-you-could-soon-be-wearing-your-smartphone/

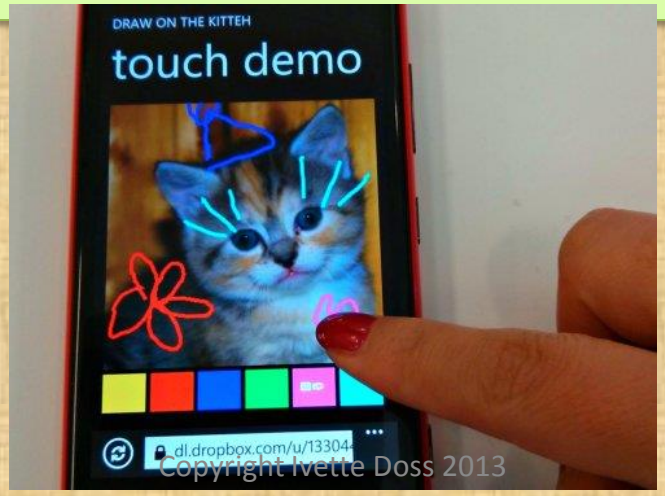
http://knstrct.com/2012/09/20/activate-your-wardrobe-9-ways-technology-can-enhance-your-style/

Types of form controls

- **date and time entries** can take advantage of a number of dedicated form controls (e.g. <input type="date">) where the user can use a native calendar control;
- the <input type="**email**">, <input type="**tel**"> and <input type="**url**"> can be used to optimize the ways user enter these often-difficult to type data, e.g. through dedicated virtual keyboards, or by accessing on-device records for these data (from the address book, bookmarks, etc.);
- the **inputmode** attribute (proposed in HTML 5.1) defines the type of textual input expected in a text entry;
- the **pattern** attribute allows both to guide user input as well as to avoid server-side validation (which requires a network round-trip) or JavaScript-based validation (which takes up more resources);
- the **placeholder** attribute allows to guide user input by inserting hints as to what type of content is expected in a text-entry control;
- the <datalist> element allows creating free-text input controls coming with **pre-defined values** the user can select from;
- HTML 5.1 defines a mechanism for the autocomplete attribute to automatically fill input fields based on **well-known data** for the user

5. User interactions

- An increasing share of mobile devices relies on touch-based interactions. While the traditional interactions recognized in the Web platform (keyboard, mouse input) can still be applied in this context, a more specific handling of touch-based input is a critical aspect of creating well-adapted user experiences, which **Touch Events in the DOM** (Document Object Model) enable.



Mobile Devices response

- Many mobile devices use haptic feedback (such as vibration) to create new form of interactions (e.g. in games); work on a **vibration API** in the Device APIs Working Group is making good progress.
- But as the Web reaches new devices, and as devices gain new user interactions mechanisms, it also becomes important to allow Web developers to react to a more abstract set of user interactions:
- instead of having to work in terms of “click”, “key press”, or “touch event”, being able to react to an “undo” command, or a “next page” command independently of how the user instructed it to the device will prove beneficial to the development of device-independent Web applications.

Mobile Device response

- Mobile devices follow their users everywhere, and many mobile users rely on them to remind them or notify them of events, such as messages: the ***Web Notifications*** specification proposes to add that feature to the Web environment.
- Mobile devices, and mobile phones in particular, are also in many cases well-suited to be used through voice-interactions; the **Speech API Community Group** is exploring the opportunity of starting standardization work around a JavaScript API that would make it possible for users to interact with a Web page through spoken commands.

Mobile Device response

- How Web Content Accessibility Guidelines (WCAG) and User Agent Accessibility Guidelines (UAAG) provide guidance on mobile accessibility — that is, making websites and applications more accessible to people with disabilities when they are using mobile phones and a broad range of other devices.

6. Data storage

- A critical component of many applications is the ability to save state, export content, as well as integrate data from other files and services on the system.
- For simple data storage, the ***Web Storage*** specification offers two basic mechanisms, localStorage and sessionStorage, that can preserve data respectively indefinitely, or on a browser-session basis.
- For richer interactions, the Web platform has a growing number of APIs to interact with a device filesystem: the ***File Reader API*** makes it possible to load the content of a file, the ***File Writer API*** allows saving or modifying a file, while the nascent **FileSystems API** give access to more general file operations, including directory management. Discussions have started on whether these two latter APIs would better be implemented on top of IndexedDB.

Data Storage

- On top of this file-based access, the ***Indexed Database API*** (IndexedDB) defines a database of values and hierarchical objects that integrates naturally with JavaScript, and can be queried and updated very efficiently. Note that the work around a client-side SQL-based database, which had been started in 2009, has been abandoned in favor of this new system.
- As more and more data need to be stored by the browser (e.g. for offline usage), it becomes critical for developers to get reliable storage space, which the proposed **Quota Management API** will offer to Web applications.
- Likewise, as some of this data need to be encrypted, the emerging Web Cryptography API

7. Personal Information Management

- Applications can benefit from integrating with existing data records; on mobile devices, the address book and calendar are particularly useful source of information, which the ***Contacts API*** and the ***Calendar API*** bring access to.
- For integration in browser-based Web Apps, Web Intents based approaches have been suggested, but are now stalled due to the lack of progress around that technology.
- For Web apps outside of the browser, a purely programmatic approach is part of the System Applications Working Group, with work on a Contacts Manager API in progress.

8. Sensors and hardware integration

- Mobile devices are packed with sensors, making them a great bridge between the real and virtual worlds: GPS, accelerometer, ambient light detector, microphone, camera, thermometer, etc.
- To take full advantage of these sensors in mobile Web applications, Web developers need to be provided with hooks to interact with them.
- The ***Geolocation API*** provides a common interface for locating the device, independently of the underlying technology (GPS, WIFI networks identification, triangulation in cellular networks, etc.)
- Web applications can also now access **orientation and acceleration** data via the *DeviceOrientation Event Specification*.
- The work on a generic *Sensor API* has been put on hold in favor to designing APIs for specific sensors, such as the *Proximity Events* API, the *Ambient Light Events* API or the proposed *Ambient Humidity Events* API

Sensors and hardware integration

- As already mentioned in the section on multimedia, there is ongoing work on APIs to open up access to camera and microphone streams.
- The opportunity for Web applications to use **Near-Field Communications (NFC)** mechanisms have led to the chartering of the NFC Working Group to develop a Web NFC API.

9. Network

- Network connectivity represents a major asset for mobile devices: the Web is an immense store of content, as well as an almost endless source of processing power, overcoming two of the limitations of mobile devices.
- The Web platform is growing a number of APIs that facilitate establishing network connectivity in different contexts.
- ***XMLHttpRequest*** (the “X” in AJAX) is a widely deployed API to load content from Web servers using the HTTP and HTTPs protocol: the W3C specification (formerly known as *XMLHttpRequest Level 2*) completes the existing deployed API with the ability to make requests on servers in a different domain, programmatic feedback on the progress of the network operations, and more efficient handling of binary content. The work on documenting the currently deployed API (XMLHttpRequest Level 1) has been abandoned in favor of getting the new API developed more quickly

Network

- By default, browsers do not allow to make request across different domains (or more specifically, across different *origins*, a combination of the protocol, domain and port) from a single Web page; this rule protects the user from having a Web site abusing their credentials and stealing their data on another Web site. Sites can opt-out of that rule by making use of the ***Cross-Origin Resource Sharing*** mechanism, opening up much wider cooperation across Web applications and services.
- XMLHttpRequest is useful for client-initiated network requests, but mobile devices with their limited network capabilities and the cost that network requests induce on their battery (and sometimes on their users bill) can often make better use of server-initiated requests. The ***Server-Sent Events*** API allows triggering DOM events based on push notifications (via HTTP and other protocols.)

Networks

- Early work on a **Push API** would allow Web applications to receive server-sent messages whether or not the said Web app is active in a browser window. As patents have been disclosed on that API, a Patent Advisory Group is being formed to assess how to make further progress possible on this work item.
- The ***WebSocket API***, built on top of the IETF *WebSocket protocol*, offers a bidirectional, more flexible, and less resource intensive network connectivity than XMLHttpRequest.

Network

- The work on Web Real-Time Communications will also provide direct **peer-to-peer data connections** between browsers with real-time characteristics, opening the way to collaborative multi-devices Web applications.
- Of course, an important part of using network connectivity relies on being able to determine if such connectivity exists, and the type of network available. The HTML5 **onLine DOM flag** (and its associated change event, ononline) signals when network connectivity is available to the Web environment.
- The **network-information API** addresses discovery of the network characteristics, allowing to determine for instance the rough bandwidth of the current connection. The **Resource Timing API** offers to measure precisely the impact of the network on the time needed to load various resources, offering another approach to adapt a Web app to its network environment.

10. Communication and Discovery

Beyond connection to on-line services, allowing communications between users, but also between devices and between applications is an important aspect of a good mobile development platform. To communicate with unknown devices or pre-existing services, a discovery component is critical.

- The ***Messaging API*** completes the existing ability to create and send message through links (with sms:, mms: and mailto: URI schemes) with more control on adding attachments and the success of the message sending. For browsers, this API would preferably be replaced by an approach based on *Web Intents*. For Web apps not in a browser, the System Applications Working Group is working on more complete Messaging API.

10. Communication and Discovery

- The **postMessage** API of *HTML5 Web Messaging* allows for Web Applications to communicate between each other.
- A joint task force of the Device APIs and Web Apps Working Groups had been looking at a mechanism called Web Intents: it aimed at closer integration of Web applications, as well as of Web applications with native applications. Some of the initial use cases for Web Intents have proved hard to expose through the regular Web browser UI, and discussions on how to properly scope that technology are on-going. In the mean time, progress on Web Intents and derived APIs has **stalled.**

10. Communication and Discovery

- The Networked Service Discovery API offers to discover services on the local network (such as the ones offered via DLNA), enabling mobile Web applications to integrate seamlessly with these services. An alternative approach based on Web Intents has also been under exploration.
- The **Web Real-Time Communications Working Group** is the host of specifications for a wider set of communication opportunities:
- **Peer-to-peer connection** across devices,
- **P2P Audio and video streams** allowing for real-time communications between users

11. Packaging

- An important aspect of the user experience of applications is linked to how the user perceives the said application is available permanently (even when off-line, which is particularly important on mobile devices), as well as how it can be shared and distributed, typically through purchases via applications stores — this is adequately addressed by packaging the application.

11. Packaging

- The Web platform offers two complementary approaches to packaging Web applications:
- HTML5's **ApplicationCache** enables access to Web applications off-line through the definition of a manifest of files that the browser is expected to keep in its cache; while relatively well deployed, the current approach has shown some strong limitations and the HTML and Web Applications Working Groups are considering a potentially major overhaul of the technology, likely based on NavigationController.
- a JSON-based manifest format in development by the Web Apps Working Group (as replacement for the W3C Widgets family of specifications). The System Applications Working Group is building a runtime and security model on top of that packaging

12. Performance & Optimization

- Due to their limited CPU, and more importantly to their limited battery, mobile devices require a lot of attention in terms of performance.
- The work started by the Web Performance Working Group on ***Navigation Timing***, ***Resource Timing***, ***Performance Timeline*** and ***User Timing***, gives tools to Web developers for optimizing their Web applications.
- The proposed work on Efficient Script Yielding offers the opportunity to Web developers to use more efficiently asynchronous programming, but has some far gained very limited traction.
- The API to determine whether a Web page is being displayed (***Page Visibility API***) can also be used to adapt the usage of resources to the need of the Web application, for instance by reducing network activity when the page is minimized. Likewise, the Timing control for script-based animations API can help reduce the usage of resources needed for playing animations.

12. Performance & Optimization

- The battery API allows adjusting the use of resources to the current level of power available in the battery of a mobile device.
- Beyond optimization of resources, the perceived reactivity of an application is also a critical aspect of the mobile user experience. The **thread-like mechanism** made possible via *Web Workers* allows keeping the user interface responsive by offloading the most resource-intensive operations into a background process.
- The *Mobile Web Application Best Practices* provide general advice on how to build Web applications that work well on mobile devices, taking into account in particular the needs for optimization.

Helpful Resources

Template for HP QC Test Cases

Sr#	Test Case Name/Title	Step Description	Step Expected	Step Actual	Status	Attachments
1	[1]Availabilty Calendar - Components				No Run	

Pass
Failed
Blocked
Not Completed
N/A
No Run

Mobile Web App - Interaction with Mobile Device

Site Map – Structure of the website