ENG1003 Mobile Apps

Web application components

An individual web page is made up of multiple files:
> HTML file describing the content and structure of the page
> CSS files that describe the presentation or style of page elements
> JavaScript files that specify dynamic, interactive behaviour
> Media files, e.g., images, audio or video

Internet is based on request/response model. Web addresses send requests to web servers who send back HTML for that page (address). Web browsers receive HTML and interpret it.

Commented [tr1]: HTML specifies CSS, JavaScript or media files, your browser makes further requests to the web server(s) to obtain these

The browser uses HTML to construct a representation of the page known as DOM. This DOM holds info about the layout/style of everything on the web page, which is then used to render (display) the page for the user.

Commented [tr2]: Document Object Model (DOM)

A file linked to an HTML page is a Cascading Style Sheet (CSS). This contains style rules that select subsets of elements within the HTML page and imposes presentational styles on their display.

Another file linked to an HTML page is a JavaScript file. This contains code that responds to DOM events and then manipulates the DOM to change the displayed page. In short JavaScript brings interactivity to the web page. Both CSS and JavaScript can also be embedded in an HTML document rather than be in a linked external file.

Commented [tr3]: E.g. a button was clicked

- HyperText Markup Language (HTML)

  The content and structure of web sites is expressed in a format called HTML. HTML marks-up the content of the page to specify structure and semantic meaning

```
...
<p>This is a paragraph with <b>bold</b> and <i>italic</i> text,
and a <a href="http://www.google.com/">link to Google.com</a>.</p>
<p>Here is another paragraph following the first.</p>
...
```

  HTML specifies start/end tags that enclose content. In the example, is a start tag and is the corresponding end tag. The start/end tag and its content is called an element. Here, bold is a bold element that tell the browser that the content “bold” should be displayed bold

- Minimal Web App E.g

Line 2 of JavaScript code, method getElementById() in invoked on document object. This method returns a reference to the DOM node associated w/ HTML element w/ an id that matches its parameter ("outputArea").

Can use this reference to set/get properties of DOM node object and invoke methods on it.

app.html

```
1  <!DOCTYPE html>
2  <html xmlns="http://www.w3.org/1999/xhtml">
3      <head>
4          <title>ENG1003 Example</title>
5      </head>
6
7      <body>
8          <div id="outputArea"></div>
9
10         <script type="text/javascript" src="code.js"></script>
11     </body>
12 </html>
```

code.js

```
1  // Output a hello message:
2  var outputAreaRef = document.getElementById("outputArea");
3  outputAreaRef.innerHTML = "Hello ENG1003 students!";
```

Commented [tr32]: Here is a minimal example of a JavaScript app that can run in a web browser. It consists of two files `app.html` and `code.js`.

outputAreaRef changes displayed content on page (console.log just outputs private info in developer tools)

Commented [tr33]: These affect the corresponding HTML element. In this case the HTML element is an empty `div` element. Setting a DOM node's `innerHTML` property sets HTML content for that element as if it had been originally specified between the element's start and end tags. In this case we've just inserted plain text but plain text is valid HTML. I could have inserted any String that contained a valid HTML fragment.

- Launching web apps on a phone of tablet

There are several special HTML instructions that can be added to a web page to inform a web browser that the page is intended to be used as a web app.

The instructions required for both Android and iOS are listed below. The app icon should be a 196×196 PNG named appicon.png. These tags should be placed in the `<head>` tag of the web page.

```
1  <meta name="mobile-web-app-capable" content="yes" />
2  <link rel="shortcut icon" sizes="196x196" href="appicon.png" />
3  <meta name="apple-mobile-web-app-capable" content="yes" />
4  <link rel="apple-touch-icon-precomposed" href="appicon.png" />
```

Commented [tr34]: These instructions specify an icon for the app and allow the app to be added to the home screen of the device. When the user taps the app icon, the web page will be displayed full-screen so that it no longer looks as though it is running in a web browser.

- Web Components of Native Apps

Functions

Function is a named code block that can be called (executed) any time to perform a specific task. Functions can be parameterised (passed values to change their behaviour) and may return a value.

- Why use functions?

  Code of a task can be placed in a function and called (executed) anywhere.
  // The fnc provides single point of guaranteed consistent update.
  // If the fnc is parameterised each call does not have to be identical, but can be given different parameter values to get different behaviour
  // Way of protecting variables from changing/clashing (encapsulates local variables as they arnt seen by anything else)

- Function that do not return values

  Syntax for defining and calling a function, and the execution sequence resulting from a call to such a function:

```
myFunction(argument1, argument2, ...); // Function call

// Function declaration:
function myFunction(parameter1, parameter2, ...)
{
    // Body of the function
    // ...

    return; // Optional, function will return at end of body.
}
```

When called, arguments are evaluated to single value. These values initialise the corresponding parameter of each argument. Then body of function is executed until a `return` statement or function closing brace(`}`). At this point execution continues in calling code w/ the statement after the function call statement

Below the Function performs an addition and date stamps it. Formatted result is displayed in an alert pop-up dialogue. Function declaration and 2 calls to the function are shown in E.g.

```
var aVariable = 7;

addAndTimeStamp(1, 2);
addAndTimeStamp(Math.sin(0.5), aVariable + 10);

function addAndTimeStamp(number1, number2)
{
    var now = new Date();
    var output = "";

    var sum = number1 + number2;
    output = sum.toFixed(2) + " (calculated at: " + now.toLocaleTimeString() + ")";

    alert(output);
}
```

3.00 (calculated at: 17:32:21) displayed in alert pop up
17.48 (calculated at: 17:32:30) displayed in alert pop up

Commented [tr44]: Arguments = Literal, variable, expression
Parameters = variable name

Arguments and parameters are matched by having the same position in the respective argument and parameter lists of the function call and declaration.

Commented [tr45]: In this example, the function is named addAndTimeStamp. It has two parameters, number1 and number2.The block following line 6 is the body of the function.

The code within the function will NOT execute unless the function is called, which it is here (twice) by code outside the function.

- First Class Functions

  JavaScript fnc's are first class which means they can be assigned to a variable, and like any variable can be passes as a parameter to a fnc.

```
var outputAreaRef = document.getElementById("outputArea");
var output = "";

function addAndTimeStamp(number1, number2)
{
    var now = new Date();

    var sum = number1 + number2;
    var result = sum.toFixed(2) + " (calculated at: " + now.toLocaleTimeString()  + ")";

    return result;
}

// Declare a variable that references the function  above:
var function1 = addAndTimeStamp;

// RHS is an anonymous function declaration:
var function2 = function(number1, number2) {
    var now = new Date();

    var sum = number1 * number2;
    var result = sum.toFixed(2) + " (calculated at: " + now.toLocaleTimeString()  + ")";

    return result;
}

// Call the addAndTimeStamp function
output += addAndTimeStamp(1, 2) + "<br />";
// Call the function referenced by function1 variable
output += function1(3, 4) + "<br />";
// Call the function referenced by function2 variable
output += function2(5, 6) + "<br />";

outputAreaRef.innerHTML = output;
```

```
3.00 (calculated at: 1:50:36 PM AEDT)
7.00 (calculated at: 1:50:36 PM AEDT)
30.00 (calculated at: 1:50:36 PM AEDT)
```

E.g.

```
var outputAreaRef = document.getElementById("outputArea");
var output = "";

function flexible(fOperation, operand1, operand2)
{
    var result = fOperation(operand1, operand2);

    return result;
}

output += flexible(function(num1, num2) {return num1 + num2;}, 3, 5) + "<br />";
output += flexible(function(num1, num2) {return num1 * num2;}, 3, 5) + "<br />";

outputAreaRef.innerHTML = output;
```

```
8
15
```

E.g.

```
var outputAreaRef = document.getElementById("outputArea");
var output = "";

function flexible(fOperation, operand1, operand2)
{
    var result = fOperation(operand1, operand2);

    return result;
}

output += flexible(function(num1, num2) {return num1 + num2;}, 3, 5) + "<br />";
output += flexible(function(num1, num2) {return num1 * num2;}, 3, 5) + "<br />";

outputAreaRef.innerHTML = output;
```

```
0.5,1,1.5,2,2.5
```

Commented [tr54]: A variable that references a fnc (which is another object) can be used to call the fnc by following the variable name w/ parentheses enclosing an argument (list)

RHS of assignment statement for function2 is an anonymous fnc expression

. When passing a fnc as a parameter you can pass a reference to the function (e.g. function2) or just define the fnc anonymously in place of the parameter

Functions can be passed as parameters to other functions

. Then called within these other functions

i.e you create a fnc then give that fnc as a argument to another fnc, then you can call that w/in the fnc

Storing data persistently

Sandboxed; documents are isolated from rest of the platform they run on for security reasons i.e browsers don't have access to you computers file system to store/retrieve documents

- Local Storage

HTML and modern browsers support a stage API called Local storage. Its limitations and characteristics include;

- Browsers can allocate upto 5MB of sandboxed data per web domain
- Only web apps. from same domain can access their shared Local Storage data
- Unlike cookies, local Storage data is never transferred to a server
- Only supports saving key/value String pairs
- Object data cannot be stored in Local Storage directly, it must first be converted to a String

Commented [tr64]: LOCAL STORAGE
LS allows us to store & retrieve strings
LS key/value store (when you have key you can simply find value that corresponds to key)
When we save; need info (string) and key to save it
To retrieve we just need key

Commented [tr65]: Cookies; website can keep info based on browser instance that was accessing website
. But cookies expire, time limited and disappear after time
. Can only store small amount of text

To use Local Storage, first check your browser supports is:

```
if (typeof(Storage) !== "undefined")
{
    console.log("localStorage is available.");
}
else
{
    console.log("localStorage is not supported by current browser.");
}
```

localStorage is available.

You can store a String value together with its String key in local storage by calling the setItem() method of the localStorage object:

```
localStorage.setItem("name", "John");
```

You can retrieve String value from Local Storage by calling getItem() method and giving it as an argument the String key of the item you want to retrieve:

```
console.log("The user's name is " + localStorage.getItem("name"));
```

The user's name is John

You can clear a key/value pair from Local Storage by calling removeItem() method and giving appropriate String key as argument:

```
localStorage.removeItem("name");
```

Or,

use localStorage w/ dot-notation like a normal object, so long as the key fits the variable naming requirement. E.g.

```
localStorage.name = "Eliza";

console.log("The user's name is " + localStorage.name);
```

The user's name is Eliza

NOTE* an object called sessionStorage with the same interface (i.e., methods) as localStorage. Data in sessionStorage persists across page reloads, but only until the browser is next closed, and so cannot be used for permanent storage.

What about Object Data?

Commented [tr66]: If the app. has been designed following the OO paradigm most of our persistent data will reside in JavaScript objects. . How can we save this data persistently? In the case of Local Storage we must convert these objects to Strings bcas that is the only data format Local Storage accepts. These Strings must encode the data and the structure of the objects so that they can be retrieved and reconstituted later. In general the process of converting object data to a format that can be stored or transmitted (commonly a String) is called serialisation or stringifying and the reverse process is called de-serialisation orparsing.
As you can imagine stringifying objects to strings and and parsing these strings back to objects are common tasks as web application data is stored and retrieved.

Software Engineering Processes

- ## What is Software Engineering?

Software engineering is the design, development and maintenance of software via the application of engineering principles and practices.

Unconstrained by Physical Laws

Constraint imposed by environment (gravity). Software has a blank slate where gravity needs to be programme into system giving flexibility.

Many ways to achieve same aim

Many ways to solve the same problem, but simple problems have a body of knowledge to do it, certain properties can achieve this quickly.

Solutions can be correct and incorrect; there are generally better and worse approaches for different problems

Complexity

Software has many logical paths that make it hard to reason about behaviour

Code can have unexpected interactions between components (using APIs) that give unexpected behaviour. Consider amount of choice for road networks (impossible to test every combination of choices)

Same thing can happen to software

Code is not just individual choices, you carry states (values of variables) as you execute code you modify state.

Software is living

Software is living (rarely finished) keeps evolving; don't just plan for initial purpose, plan and design a foundation you can build on

Reliability & stability

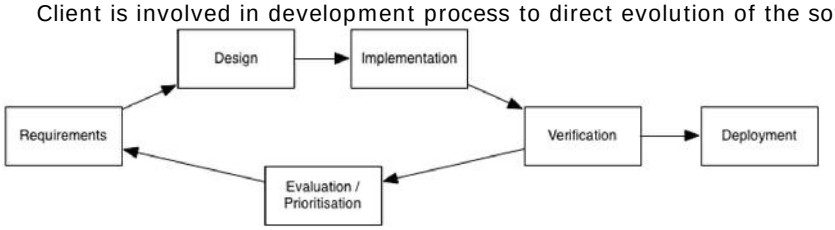
Since software is written by humans, its rarely free from errors, bugs at all levels (i.e in Materials science you can depend on the properties of your materials).

- Software Development Methodologies

Agile Development

Lightweight interactive methodology in practice today

Client is involved in development process to direct evolution of the software.



Requirements involves figuring out what needs to be done

Design deals w/ how to get things done

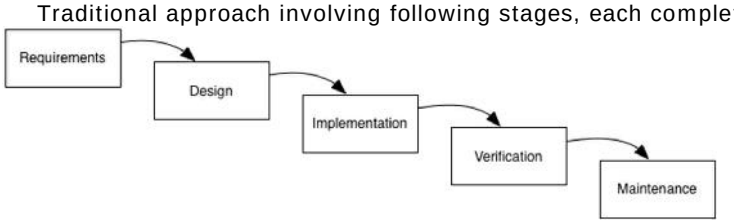
Implementation involves writing code to do the task

Verification deals w/ quality assurances (testing something)

PROS	CONS
Delivers something useful immediately (code, documentation)	Requires involvement (time and interest) of client, which is not always possible
Requirements can be adapted and not nailed down at beginning	Since requirements are refined each iteration, they can become overly ambitious (scope creep)

Waterfall Model

Traditional approach involving following stages, each completed before progressing



This style follows a well-defined linear process, requirements are clear and less chance of changes.

Good for large (and expensive) or mission-critical projects.

.

Software Engineering Processes

Requirements give formal description of what we are working towards, it's the first step for any software project.

Besides just finding what features are needed, requirements determine correctness and completeness of software (tests originate from requirements)

- Requirements Elicitation (drawing out requirements)

  Requirements are a description of a piece of functionality

  When collecting requirements there can be problems ie. The client has a different technical knowledge and might not know exactly what they want

  Waterfall

  Requirements are collected at beginning and set before starting design the danger w/ this is if requirements can change during development

  Agile

  Requirements are gradually done (iterated), client gains better understanding of project ∴ client understands what's possible

- Capturing Requirements in Agile

  User stories

  User stories are short concise description of actions a user wants w/ a system

  As a <persona>, I want <action> [so that <outcome>]

  User story must capture; need/effect, components/people involved and motivation for the feature (should be short and phrased in clients language)

  >User stories are phrased in terms of need ∴ easy to write tests for them

  A way to produce/check user stories is INVEST

  Independent; self-contained
  Negotiable; easily revised/replaced
  Valuable; beneficial to end users
  Estimable;
  Small; small enough to be scheduled independently
  Testable; verity it's been implements satisfactorily

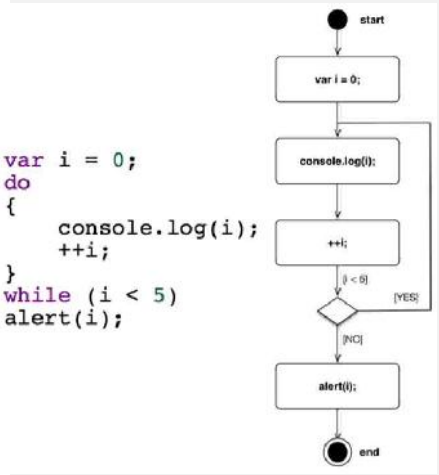
Commented [tr73]: Where do software requirements come from?
. Clients
. Government; legislation laws you have to obey
. Security; storage of credit card no. (can overlap w/ legislation) follow standards not . . to store card details in entirety
. Hardware; software runs on hardware, so must fit limitations/features of hardware
. Software; Java can restricts was can be accomplished
. Competitive pressure; competitors are doing something so you must do the same . . thing to be on the same level
. Standards; source of requirements

Commented [tr74]: Is a user stories good/well written? Check INVEST (user stories are tasks or bugs to fix, small pieces we can assign to someone)

- Activity diagram

Used to document program flow (i.e flow charts for software)

Shows how components interacting simultaneously



- Prototyping and UI design

Another aspect of design is User Interface (UI) and User Experience (UX)

To plan the layout and behaviour of the user interface, we use prototyping

simple representation to demonstrate interaction

>Useful for communicating ideas as mock ups provide mechanisms for testing feasibility of ideas

>Allows users/clients to react to design and suggest changes

Wireframing

Sketches of user interfaces showing mock data

Skeletal view like a building frame; gives idea of layout

>Give sense of user interface before code is written

>Wire frame sketches are made to look scratchy so clients are more willing to critique

Storyboards

Sequence of stills used for simulating interaction (may be wireframe sketches)

Serve to show progression through a task,

Gives understanding of navigation and flow of an app, can be discussed w/ clients and iterated between designers and engineers

- Approaches to Testing

Black box testing

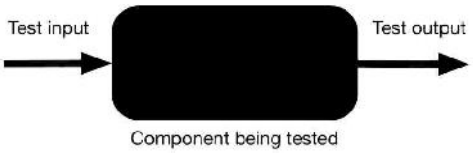
Involves testing code while knowing what it is supposed to do, but not how it does it.

This approach tests the behaviour of code

For each requirement, we need to know what the right behaviour of the software

∴ we consult an oracles (such as a software specification determined during requirements analysis) to determine the correct behaviour.

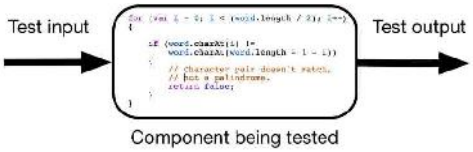
This sort of testing is done without seeing source code and considers the software component being tested as a black box that we can't see inside. Black box testing involves giving the black box a set of input and verifying that it produces a particular set of output.



White box testing/clear box

Code-based approach where tests are written knowing how the code being tested works

Such testing might test individual code paths, ranges of variables or data types used.



- Levels of testing

Software testing can happen at different levels

Unit testing

Test small units (methods, fncs etc.) of code

Each unit test generally tests a function, or a method of a class in the case of O-O code

Usually tested while implementing that part of code

Integration Testing

This testing checks behaviour of components vs. documentation

Bugs may emerge as a result of interaction ∴ It's these bugs that integration testing targets

Units need to interact and test that is works

Systems Testing

Testing the complete application/overall system in order to check that it satisfies requirements

Such tests might be created from user stories, or activity diagrams created during design

As well as checking for the correct behaviour, these test also check the application doesn't do anything unwanted, like crash or lose user data

Commented [tr78]: To use a car metaphor, black box testing would be like testing a car by sitting inside it, without ever seeing the engine or other mechanical components.

Commented [tr79]: i.e software specifications, existing algorithm or previous versions of code

Commented [tr80]: An analogy for white box testing would be testing a car while being able to see under the car and bonnet in order to determine the components being used and how they are connected.

Commented [tr81]: Imagine you have a data layer of a mobile app that communicates with Twitter and stores the local version of a set of information from the Twitter website. You could have integration tests that test this code by creating code that returns sample data in the same format as the Twitter web service.

- Code Complexity

Complexity analysis looks at how optimal algorithms are, particularly how much time, memory or other resources they require

Complexity depends on the size of the input

e.g. the same algorithm takes longer to sort a million items when compared to ten items

Hence complexity can be represented as a function of input size n.

When considering execution time, time complexity is expressed as O(..)

Deals w/ the fnc bounding the run time

Unconcerned w/ constant & unchanging factors which are unrelated to input size

i.e complexity gives us an idea of how slow an algorithm gets proportional to the input

Some examples of tasks we perform in code and the time complexities they fit into:

Constant time: O(1)

Direct access to an element of an array e.g. array[15] is constant regardless of length

Bcas arrays allow position of a value to be accessed from its index regardless of size of array

Any code that runs in a similar time regardless of size = constant

Logarithmic time: O(log(n))

Commented [tr84]: Algorithms that take time to run proportional to the logarithm of the input size are said to run in logarithmic time.

An e.g. is binary search

Finds position of a value is an array, it works by splitting the search space in half each iteration

Selects middle of array, if middle === value algorithm stops

if middle > value, algorithm repeats w/ first half of array etc.

In this way binary search requires 9 steps for an array of 500, or roughly $\log_2(500)$

So an array of size n, steps needed for binary search = $\log_2(n)$

Linear time: O(n)

Linear time algorithms take time directly proportional to the size of the input

To find the lowest/highest value of an array is a linear time operation

You look at each element once, so its dependant on size of array

Quadratic time: O(n^2)

The selectionSort() fnc runs in quadratic time.

Takes time proportional to the square of the input size

selectionSort() outer loop looks at every element in array, then inner loop looks at all remaining

∴ n*n operation, hence why its slower than Array.sort()

- Good Design Principles

Design principles are guides/rules to abide by when designing mobile interfaces

E.g.

Donald Normans Principle

Commented [tr90]: https://www.alexandriarepository.org/reader/eng1003/86923

Proposed a few important principles of user-centred design

1. Visibility
   Relevant elements should be obvious what they are for even before user initiates it
2. Affordance
   Appearance of objects should indicate how it is used, e.g., buttons “invite” pressing
   Interface hints at functionality
3. Constraints
   Limitations of the possible actions of an object, limiting errors
   e.g swipe to unlock prevents unlocking by mistake
4. Cognitive Aids
   Offering user additional help with state of an element e.g fuel gauge arrow indicates the side of the pump if you forget
5. Transfer effects
   Benefit from making things in a way similar to what the user has seen, utilizing user experience
6. Natural Mapping
   Can we make computer interfaces similar to something the user has been seen, giving the user an idea of how to interact w/ an interface before they even try

or,

Ben Schniederman’s 8 Golden Rules of Interface Design

Commented [tr91]: Idea of consistency,

1. Strive for Consistency
   Consistent interfaces make it easier to learn and use, utilizing user experience of an OS
2. Cater to universal usability
   Cater to the needs of a wide range of users
3. Offer informative feedback
   Gives users feedback on actions e.g sounds, highlights, lets user know action happened
4. Design dialog to yield closure
   Provide feedback on the end of a transaction
5. Prevent errors
   Detect errors and correct them. Dynamic system
6. Permit easy reversal of actions
   Offer an easy way out, offer an undo e.g. quit to home screen
7. Support Internal locus of control
   Allow users to initiate actions, giving a sense of control
   Makes user feel in control, interface need to corresponds to user action
8. Reduce short term memory load
   Don’t make tasks complex, hint to the user what is needed
   make the interface remember