INTERNATIONAL ADVANCED LEVEL

INFORMATION TECHNOLOGY

SCHEME OF WORK
Unit 2

Pearson Edexcel International Advanced Subsidiary in Information Technology (XIT11) Pearson Edexcel International Advanced Level in Information Technology (YIT11)
First teaching September 2018
First examination from June 2019
First certification from August 2019 (International Advanced Subsidiary) and August 2020 (International Advanced Level)

INTRODUCTION

The following scheme of work provides an overview of the content of the 2018 International Advanced Level Information Technology and shows how the content could be taught as a guideline approach only.

It should be adapted by schools to fit their timetabling and staffing arrangements. It is based upon a two-year delivery model where all IAS content is being taught in the first year and the remaining IA2 content in the second year.

The scheme of work is broken up into units and topics, so that there is greater flexibility for moving topics around to meet planning needs.

It includes:

- recommended teaching time for topics, though of course this is adaptable according to individual teaching needs
- classroom activities, teaching points and suggested teaching resources
- objectives for students at the end of the topic area and integrated Transferable Skills* that are being developed.

The number of guided learning hours for Advanced Level is 360. Teachers should be aware that the estimated teaching hours are approximate and should be used as a guideline only.

Unit 2

(Refer also to the specification and the delivery and assessment guidance in the Getting Started Guide)

Topic 7: Understanding the functionality of HTML Topic 7 deals with the understanding of HyperText Markup Language (HTML). It is expected that students will carry out most of the styling within CSS. Audio and video content will use HTML only. Students may wish to specify the height and width of images, emphasising or specifying the importance of text and the start and/or type of lists within HTML. Styling will be intensively covered within the CSS content.			
Week	**Topic area / aims / learning outcomes**	**Exemplar classroom activities / teaching points / suggested teaching resources**	**Integrated Transferable Skills**
1	**7.1.1** Understand how HTML is used to structure web pages: a. doctypes b. elements c. tags. **7.1.2** Understand how to declare the language of an html document. **7.1.3** Understand how the head element is used to supply information about the document: a. document title. **7.1.4** Be able to write organised syntax: a. lower case letters within element names, values and attributes b. indenting nested elements c. double quotes. **7.1.5** Understand how global attributes are used to define elements: a. language.	**Activity 1: Webpage structure** Tutors to discuss/demonstrate the basic structure of a webpage including: • adding a doctype <!DOCTYPE html> • specifying the character set <meta charset="UTF-8"> • adding elements (encompass opening tag, closing tag and content) • adding tags (labels used to mark up the beginning and end of an element) • declaring the language of an html document <html lang="en"> • adding a document title <title>Example Title</title> • writing organised syntax. Tutors to provide a task that will enable the students to create the basic structure of a web page. Note: Students should include this basic structure in every web page they create. **Activity 2: Headings and paragraphs, refreshing pages** Tutors to introduce and demonstrate: • the use of headings and paragraphs etc, • refresh a page after a particular amount of time, e.g. <meta http-equiv="refresh" content="30"> Tutors to provide a task that will enable students to:	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	7.2.3 Be able to use some key elements to define the structure and formatting of text on a webpage: a. headings b. paragraphs. **7.2.4** Be able to create lists: a. unordered b. ordered (and use the start and type attributes) c. definition/ description d. nesting lists.	• create a web page around a topic • include the basic structure of a web page, headings and paragraphs.	
		Activity 3: Identifying HTML code errors Tutors to provide tasks that will enable the students to: • identify/explain/describe the basic structure of a web page • identify/explain the code/errors in the code • add to or amend code.	

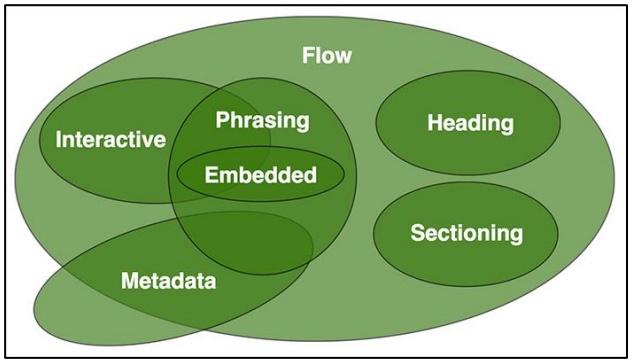
		Activity 4: Lists
		
		Tutors to discuss/demonstrate the difference between ordered, unordered, nested and definition/description lists including:
		
		• ordered lists
		  ○
		  ○
		  ○ using uppercase or lowercase letters in place of numbers (type)
		  ○ specifying a start number/letter (start)
		• unordered lists
		  ○
		  ○
		• nesting lists
		  ○ ordered list within ordered list
		  ○ unordered list within unordered list
		  ○ unordered list within ordered list
		  ○ ordered list within unordered list
		• definition/description lists
		  ○ <dl></dl> defines the description list
		  ○ <dt></dt> defines a data term
		  ○ <dd></dd> defines a data definition
		• double quotes
		  ○ &#34
		  ○ &quot
		
		Tutors to provide tasks that will enable the students to:
		
		• create a web page or pages to incorporate the lists above
		• identify/explain/discuss/describe the different types of lists	

		• identify/explain the code/errors in the code • add to or amend code.	
2	**7.1.4** Be able to write organised syntax: a. removing the forward slash at the end of self-closing elements. **7.2.5** Be able to create links: a. internal b. external c. email d. opening links in a new browser window or tab e. linking to a specific part of the same page f. linking to a specific part of another page. **7.3.2** Be able to represent information in a table: a. rows b. data c. headings, body and footer d. combining multiple cells.	**Activity 5: Links** Tutors to discuss/demonstrate links including: • an external link that opens in the same browser window • an external link that opens in a new browser window • linking to a specific part of the same page • linking to a specific part of another page • email links. Tutors to provide a two-page website that will enable students to: • add the links above • identify/explain/discuss/describe the different types of links • identify/explain the code/errors in the code • add to or amend code.	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts
		Activity 6: Tables Tutors to demonstrate/discuss: • <table></table> to define the table • <tr></tr> to define rows • <td></td> to define data • <th></th> to define headings • colspan/rowspan to combine multiple cells. Tutors to provide tasks that will enable the students to: • create a web page to present a table incorporating the above • identify/explain the code/errors in the code • add to or amend code.	
		Activity 7: Table grouping Tutors to demonstrate/discuss: • <thead></thead> used to group header content • <tbody></tbody> used to specify body content	

		• <tfoot></tfoot> used to group footer content • Browser can use the <thead><tbody> and <tfooter> to enable scrolling of the table body independently of the header and footer. Also allows the table header and footer to be printed at the top and bottom of each page for tables spanning multiple pages. Tutors to provide tasks that will enable the students to: • create a web page to present a table using table grouping incorporating the above • identify/explain the code/errors in the code • add to or amend code.	
3	**7.1.1** Understand how HTML is used to structure web pages: a. attributes. **7.1.4** Be able to write organised syntax: a. double quotes b. omit the values on Boolean attributes. **7.3.1** Be able to add images to web pages: a. file format b. image size c. resolution d. retaining original proportions f. alt tag. **7.3.5** Be able to prepare audio and video: a. multiple file formats b. embed in a web page c. controls d. customising controls e. adding attributes	**Activity 8: Adding images to web pages** Tutors to discuss/demonstrate images including: • different file formats (GIF, JPG and PNG) • specifying the size and height of an image (pixels, %) • adding alt tags • explain the purpose of image resolution within a web page (number of pixels within an image) and how to maintain proportions when scaling an image (e.g. specifying only one unit of length - either width or height) • removing slashes from self-closing elements (using slashes is also acceptable). Tutors to provide a task that will enable the students to: • create a two-page website on a topic of the tutor's choice that includes: ○ headings ○ paragraphs ○ an unordered list of images ▪ using different file formats ▪ with the heights and widths specified ▪ with alt tags ○ an external link that opens in a new browser window ○ a link to a specific part of another page ○ an email link • identify/explain the code/errors in the code • add to or amend code.	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	f. using the source element to specify multiple resources. **7.3.6** Understand how to use inline frames to add dynamic content from external websites.	Note: Images will be expanded upon in the topics for CSS and responsive page design. **Activity 9: Preparing audio and video** Tutors should discuss/demonstrate: ○ multiple file formats (wav, mpeg, mp3, mp4, webm, ogg, possible browser playback issues) ○ <audio></audio> ○ <video></video> ○ controls ○ <source> (can include multiple sound files in different formats to combat possible browser playback issues) ○ type ○ autoplay ○ loop For example: <audio controls autoplay loop> <source src="barkingDogs.mp3" type="audio/mpeg"> <source src="barkingDogs.wav type="audio/wav"> </audio> <video controls width="320" height="240"> <source src="barkingDogs/mp4"> <source src="barkingDogs/ogg "> </video> Tutors to provide tasks that will enable the students to: • embed audio (answers written on paper only) ○ with controls ○ without controls ○ setting height and width ○ specifying multiple file formats ○ specifying multiple resources ○ playing automatically ○ looping • embed video (answers on paper only) ○ with controls	

		○ without controls ○ setting height and width ○ specifying multiple file formats ○ specifying multiple resources ○ playing automatically • identify/explain the code/errors in the code (answers written on paper only) • add to or amend code (answers written on paper only).	
		Activity 10: Inline frames Tutors to discuss/demonstrate how inline frames can be used to add dynamic content from external websites: • <iframe></iframe> • src • width • height Tutors to provide a task that will enable the students to: • create a web page incorporating the above • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://www.w3schools.com/tags/tag_iframe.asp • https://www.w3schools.com/html/html_iframe.asp	
4	**7.2.1** Understand what is meant by block-level elements and inline elements. **7.2.2** Understand that elements can belong to different content models which follow certain rules: a. flow b. sectioning c. heading d. phrasing	**Activity 11: Block-level elements, inline elements and content models** Tutors to lead discussion or students to work in small groups to research block-level elements, inline elements and content models: • prior to modern HTML specification, HTML elements were either block-level or inline elements ○ block level elements ▪ take up the entire width of their parent ▪ always begin on a new line in the flow of the document ○ inline elements	• Co-operation • Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	e. embedded f. interactive. **7.2.3** Be able to use some key elements to define the structure and formatting of text on a webpage: a. article b. section c. headings d. paragraphs e. thematic breaks f. emphasis g. importance. **11.1.1** Understand how to add semantic markup to web pages that describes the content of a web page and how this is used by browsers and assistive technologies. **11.1.2** Be able to use elements that define what the content is on the web page: a. headers and footers b. navigation c. articles d. asides e. sections f. main. **11.1.3** Understand that you can still group elements together even if there is not a relevant semantic tag: a. div	▪ are on the same line (could have many inline elements, one after another, and they will all still be displayed on the same line ▪ restricted to only contain other inline elements ▪ take up only as much space as necessary • modern content models  ○ a content model refers to the set of rules that define what type of content each element can have ○ they can still loosely be thought of as falling into either block-level or inline elements ○ flow content ▪ roughly translates into the block-level category. ▪ most elements used in the body of documents are flow content (e.g. table, video, embed, article, etc.) ▪ elements can wrap (almost) all other elements ○ sectioning content ▪ content that defines the scope of headings and footers (e.g. article, nav, section, etc.) ○ heading content ▪ defines the header of a section (e.g. h1, h2, etc.) ▪ they convey a meaning – the content they wrap is to be treated as heading content ○ phrasing content ▪ roughly translates into the traditional inline category ▪ defines the text and the markup it contains (e.g. abbr, cite, em, etc.) ○ embedded content ▪ imports another resource (e.g. audio, embed, iframe, img, video, object, etc.)	

	b. span. **11.1.5** Be able to use semantic markup to add textual meaning: a. importance b. emphasis.	○ interactive content ▪ elements that are specifically designed for user interaction (e.g. button, embed, iframe, select etc.). Tutors to ensure tasks that will enable the students to: • identify/explain/discuss/describe inline/block level elements • identify/explain/discuss/describe content models. **Resources** • https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html • https://clearlydecoded.com/html-content-models • http://w3c.github.io/html-reference/common-models.html	
		Activity 12: The Semantic Web Tutors to use a pre-created web page to discuss/demonstrate: Semantic web • gives content on the web page meaning and structure by using the correct HTML element • describes the content rather than how content should look • enables computers, screen readers, search engines and other devices to understand the content • always uses semantic elements if they exist. Elements • used for layout only – semantically meaningless • <header></header> • <footer></footer> • <nav></nav> • <article></article> • <section></section> • <hr> used to signify and thematic break in content • emphasis ○ while both are typically (but not always) styled to display as italic, is used when you want to put more stress on a word or phrase rather than just display a word or phrase using italics. is semantic mark up • importance	

		○ both will embolden a word or phrase so they will look the same in a browser; however, only is of use to a screen reader. is semantic mark up. • <aside></aside> • <main></main> Tutors to provide two pre-created webpages that will enable the students to: • identify/explain the code/errors in the code • add to or amend the code. **Resources** • http://web-accessibility.carnegiemuseums.org/foundations/semantic/ • https://www.w3schools.com/html/html5_semantic_elements.asp	

Topic 8 Understanding the functionality of CSS **CSS documents contain styling rules that describe how HTML elements are displayed. Understanding CSS sizing, alignment, spacing and responsiveness will allow you to control the layout of a web page and create consistent and visually appealing web pages that can adapt to different device sizes.**			
Week	**Topic area / aims / learning outcomes**	**Exemplar classroom activities / teaching points / suggested teaching resources**	**Integrated Transferable Skills**
5-6	**7.1.3.** Understand how the head element is used to supply information about the document: a. styles b. links to external files. **7.1.5** Understand how global attributes are used to define elements: a. ID b. style. **8.1.1** Understand the purpose of CSS: a. controlling page layout b. consistent page design. **8.1.2** Be able to reference CSS within the HTML code: a. inline styles b. internal style sheets c. external style sheets (i) folder structure (ii) naming conventions (iii) relative path. **8.1.3** Understand how to write CSS rules:	**Activity 13: CSS fundamentals** Tutors to discuss/demonstrate: • the purpose of CSS ○ controlling page design ○ giving an entire website a consistent appearance ○ improve navigation and readability of websites • inheritance ○ an element's style values are copied from its parent element ○ it allows consistent styles throughout a website without having to repeat code • cascading ○ children inherit from their parents unless they override the parents' style ○ when multiple style sheets are used, they cascade ○ general rule is that the last style sheet overrides the ones specified before ○ order depends on other factors such as weight, specificity, use of special keywords, and browsers • how it is included within an HTML document ○ <link rel="stylesheet"> (external stylesheet) ○ <style type="text/css> (Internal in <head>) ○ style="color: red" (inline) • external CSS folder structure and naming conventions ○ styles folder ○ relative paths (../style.css) ○ absolute paths (C:/www/styles/style.css) • what classes and IDs are used for, and why they are different ○ IDs ▪ unique ▪ used to identify one element ▪ allow styling of one HTML element ▪ each element can have only one ID ▪ each page can have only one element with that ID	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	a. selectors b. properties c. values. **8.1.4** Understand CSS selectors: a. type b. class c. ID d. universal e. attribute f. child g. descendant h. adjacent sibling i. general sibling j. multiple selectors. **8.1.5** Understand how to write efficient style sheets through general rules that apply to most elements and applying specific rules to individual elements: a. cascade b. inheritance. **8.1.7** Be able to create rules using CSS attribute selectors that apply elements that have an attribute with a specific value: a. existence b. equality c. space d. prefix e. substring f. suffix. **8.2.4** Be able to style elements: a. text layout	○ classes ▪ not unique ▪ used to identify more than one element ▪ allow styling for many HTML elements with the same class ▪ can use the same class on multiple elements ▪ can use multiple classes on the same element ○ you can use an ID and class on a single element • CSS selectors ○ properties of those selectors ○ values of those properties ○ types of selectors ▪ universal (*) – applies to all elements on the page ▪ element (e.g. div) – applies to specific elements ▪ class (.myclass) – applies to any html element with that class ▪ ID (#myid) – applies to any html element with that id ▪ attribute (e.g. input[type="text"]) – targets elements based on the presence or value of html attributes ▪ pseudo class (e.g. a:hover) – targets an element in a specific pseudo state, e.g. a link being hovered ▪ pseudo element (e.g. .title:before, #header:after) – inserts and targets a pseudo element either before or after an actual element. Can be used for e.g. inserting an icon before a link, or the word "tel:" before a telephone number. ▪ multiple selectors: (e.g. p, div, #header) – targets every element in a comma separated list and applies the same styles to all • style elements (Note: This may be done anywhere the tutors feel most appropriate within the CSS/Design websites sections.) ○ text layout (color, alignment, text-decoration) ○ font (font-family, font-style, font-size (em, px, %, font-weight) ○ links (color, link, visited, hover, active, text-decoration, background-color) ○ lists (list-style-position, list-style-type:none) ○ tables (border, border-collapse, height, width, text-align, padding, background-color, nth-child(even), nth-child(odd))	

	b. font c. links d. lists e. tables g. images.	○ images (border-radius, border, padding, width, height, opacity, visibility – also see positioning. Tutors to provide a range of tasks that will enable the students to: • describe the purpose of CSS • explain cascading • explain inheritance • explain/describe/discuss the difference between external and internal style sheets • use external and internal style sheets • name/identify/explain/describe/discuss and use these selectors: ○ universal ○ element ○ class ○ ID ○ multiple selectors • use these selectors: ○ pseudo class ○ pseudo state ○ pseudo element • style elements ○ text layout ○ font ○ links ○ lists ○ tables ○ images • identify/explain errors in the code. **Resources** • https://css-tricks.com/the-difference-between-id-and-class/ • https://www.w3schools.com/cssref/css_selectors.asp • https://www.w3schools.com/css/css_text.asp • https://www.w3schools.com/css/css_font.asp • https://www.w3schools.com/css/css_link.asp • https://www.w3schools.com/css/css_list.asp • https://www.w3schools.com/css/css_table.asp • https://www.w3schools.com/css/css3_images.asp	

		Activity 14: Further CSS selectors Tutors to discuss/demonstrate these selectors: • descendant (eg #header .title) – applies to a more specific selection, e.g., targets every element with the .title class within the #header element. Does not have to be an immediate child of #header • child (eg #header > .title) – same as descendant but only targets elements with the .title class that are a direct child of the #header element • general sibling (eg h2 ~ p) – targets every <p> which is a sibling of a <h2>. Does not have to be an immediate sibling • adjacent sibling (eg h2 + p) – same as general sibling but targets only <p> elements that are adjacent siblings to <h2> elements. Tutors to provide a range of tasks that will enable the students to name/identify/explain/discuss/describe the selectors above.	

		Activity 15: Attribute selector syntax
		
		Tutors to discuss/demonstrate:
		
		• [attr] – selects element with specific attribute
		• [attr=value] – selects element with specific attribute and specific value for that attribute
		• [attr~=value] – same as previous but can specify multiple, whitespace-separated values to match against
		• [attr|=value] – selects element with a specific value, or element beginning with a specific value immediately followed by a hyphen (often used for language subcode matches)
		• [attr^=value] – selects element with a prefix of a specific value
		• [attr$=value] – selects element with a suffix of a specific value
		• [attr*value] – selects element with at least one occurrence of specific value.
		
		Tutors to provide a range of tasks that will enable the students to:
		
		• [attr] and [attr=value]
		  ○ use both
		  ○ describe how code examples work
		  ○ identify/explain the code/errors in the code	

		○ add to or amend code • the rest ○ identify/explain them.	
7-8	**8.1.6** Understand the CSS Box Model. **8.2.1** Be able to specify colours: a. colour names b. hexadecimal notation c. RGB values. **8.2.2** Understand how to manipulate colour: a. opacity b. gradients c. HSL values. **8.2.5** Be able to use the box model to add backgrounds (background images, gradients, CSS sprites) and borders to elements. **8.3.2** Be able to use the box model to control the appearance of boxes: a. display b. width c. height d. borders e. margins and padding. **8.2.3** Be able to specify length values:	**Activity 16: CSS box model** Tutors should discuss/demonstrate: • the CSS box model ○ every element of a web page is made up of a rectangular box consisting of four layers – content, padding, border and margin, in that order ○ content box – the area in which content is displayed, e.g. text or images. The CSS width and height properties set the width and height of the content box ○ padding – the inner margin of a CSS box, like the margin of an A4 piece of paper ○ border – the border between the padding and margin of a content box. Has 0 width by default and is invisible, but can be displayed with a variety of styles ○ margin – the outermost part of a CSS box, surrounding the border, padding and content box. The margin of one CSS box touches against the margin of other CSS boxes, and can be used to create space between them • block level elements take up the entire width of their parent • inline elements take up only as much space as necessary. Tutors to provide a range of tasks that will enable the students to: • describe the CSS box model • use the CSS box model • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Box_model	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	a. absolute lengths b. relative lengths.	**Activity 17: Lengths and CSS units** Tutors to discuss/demonstrate: • absolute lengths units are a fixed length and will appear exactly at the size specified. Length examples include px and pts • relative lengths are a length relative to another length property (generally font size), e.g. a div with a width of 10em will appear 10 times the size of the font size. Tutors to provide a range of tasks that will enable the students to: • name/identify/explain/discuss/describe the lengths above • use the lengths above • identify/explain the code/errors in the code • add to or amend the code.	

		Activity 18: Colour and backgrounds
		
		Tutors to discuss/demonstrate:
		
		• colour names
		  ○ there are a number of colours specified by name in CSS, e.g. red, black, etc.
		• RGB
		  ○ red, green, blue values (e.g. rgb(255, 99, 70)
		• HSL
		  ○ hue, saturation and lightness (e.g. hsl(0, 100%, 50%)
		• hexadecimal
		  ○ hexadecimal value for colours (e.g. #24ba13)
		• opacity
		  ○ cannot be used with Hex, can only be used with RGB or HSL (e.g. opacity: 0.5)
		• gradients
		  ○ transiting between two or more specified colours
		• backgrounds can be transparent, an image or colour. For example:
		  ○ background-image: url(image.jpg)
		  ○ background-position:center
		  ○ background-color: #000000
		  ○ background-image: linear-gradient (top, red, black)
		• image sprites may be used to decrease memory usage and the number of http requests made by combining all images into one.	

		When an image sprite is set as a background, you are able target the specific area of the combined image.
		
		Tutors to provide a range of tasks that will enable the students to:
		
		• name/identify/explain/discuss/describe/use the different ways of specifying a colour
		• use opacity
		• use backgrounds
		• identify/explain the code/errors in the code in terms of
		  ○ specifying a colour
		  ○ using opacity
		  ○ using backgrounds
		• add to or amend code in terms of
		  ○ specifying a colour
		  ○ using opacity
		  ○ using backgrounds
		• describe how images sprites can be used as a background.	

9-10	**7.3.1** Be able to add images to web pages:
	a. positioning images.
	
	8.3.1 Be able to control the position of elements:
	a. normal flow
	b. relative positioning
	c. absolute positioning
	d. fixed positioning
	e. floating elements
	f. overlapping elements.
		Activity 19: Positioning
		
		Tutors to discuss/demonstrate:
		
		• positioning of CSS boxes
		  ○ normal flow (position: static) - one CSS box follows another
		  ○ relative positioning (position: relative) – CSS box positioned in relation to its default (static) positioning, e.g. adding "top: 10px" to a relatively positioned element will move it 10px down from its normal starting position
		  ○ absolute positioning (position: absolute) – CSS box positioned exactly where specified. Unless otherwise configured, will position elements in relation to the <html> element, e.g. "top: 0px" will move it to the top of the browser window, "bottom: 0px" will move it to the bottom. If an absolutely positioned element is the descendant of a box which has absolute or relative positioning itself, it will be in relation to that box, rather than the <html> element
		  ○ fixed positioning (position: fixed) – is relative to the viewport, or the browser window itself. The viewport does not change when the page is scrolled, so a CSS box with "top: 10px" will always appear 10 pixels from the top of the
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		browser, no matter where the user scrolls (it will appear to "float" above the rest of the page) ○ floating elements (float: left) place an element on the left or right side of its container, allowing text and inline elements to wrap around it. The element is removed from the normal flow of the page. Absolutely positioned elements cannot be floated ○ images may be positioned using floats ○ overlapping elements – z-index applies to any element that has a position other than static and controls the stacking of elements. Elements with a higher z-index appear stacked on top of an element with a lower z-index ○ clearfix to clear floats ○ img.hover, e.g. to change opacity. Tutors to provide a range of tasks that will enable the students to: • use the above • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://developer.mozilla.org/en-US/docs/Web/CSS/position • https://developer.mozilla.org/en-US/docs/Web/CSS/z-index • https://developer.mozilla.org/en-US/docs/Web/CSS/float • https://www.w3schools.com/howto/howto_css_clearfix.asp	
11	**8.4.1** Be able to use CSS transitions and transforms to create animations: a. transition properties b. transform properties c. 3D transforms d. cubic-bezier.	**Activity 20: Animations** Tutors to discuss/demonstrate: • transition properties ○ allows you to change property/properties values smoothly (from one value to another) over a give duration ○ need to specify CSS property to add effect to and the duration of the effect • transform ○ applies a 2D or 3D transformation to an element ○ allows you to rotate, scale, move, etc. • transition and transform combined ○ you can use a transition to add an effect to a transformation	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		• shapes with a transition effect applied to:
		  ○ width
		  ○ height
		  ○ width and height
		  ○ cubic-bezier – variable speed from start to end
		• shapes with a rotate transform effect
		• shape with a combined transition and 2D rotate transform effect.
		
		Tutors to provide a range of tasks that will enable the students to:
		
		• identify/explain the code/errors in the code
		• add to or amend code.
		
		Resources
		• https://www.w3schools.com/css/css3_transitions.asp
		• https://www.w3schools.com/cssref/func_cubic-bezier.asp
		• https://www.w3schools.com/css/tryit.asp?filename=trycss3_transition_transform
		• https://thoughtbot.com/blog/transitions-and-transforms	

Topic 10: Designing web pages **Adopting a design, implement, test and iterate approach to coding for the web enables web designers and developers to build functional web pages for specific audiences and purposes. Effective page layout and design is essential to guiding the user through a web product. Students will understand how to incorporate the principles of design, accessibility and usability within their coding to be best placed to create visually appealing page layouts that enhance the user experience.** Topic 11: The semantic web **Semantic code describes the content rather than how the content should look. Semantic HTML adds functionality to web pages, works better on mobile devices and supports search engine optimisation (SEO). Semantic markup also makes the code easier to write and maintain, as it shows what each piece of content is about, as well as enabling students to take advantage of default styles and functionality.**			
Week	**Topic area / aims / learning outcomes**	**Exemplar classroom activities / teaching points / suggested teaching resources**	**Integrated Transferable Skills**

12-13	**8.3.3** Understand how to design for differently sized screens:
	a. fixed width layouts
	b. liquid layouts
	c. layout grids
	d. CSS frameworks.
	
	10.1.1 Understand how to design web pages:
	a. wireframes
	b. mock-ups
	c. web-design style guide
	d. prototyping.
	
	10.1.3 Understand how to use design principles to create effective page layout and design:
	a. visual hierarchy
	b. flow
	c. colour theory
	d. balance and contrast.
		Activity 21 : Different sized screens
		
		Tutors to lead discussion or students to work in small groups to research designing for different sized screens including:
		
		• fixed width layouts
		  ○ width of the entire page is set with a specific numerical value
		  ○ remain this width regardless of the size of the browser window viewing the page
		  ○ allows the designer to build pages that will look identical no matter who is looking at them
		  ○ can cause horizontal scrolling in smaller browser windows
		  ○ can result in a lot of unused space and more scrolling vertically
		• liquid layouts
		  ○ based on percentages of the current browser window's size
		  ○ expands and contracts to fill the available space
		  ○ do not allow for precise control over the width of the various elements
		• layout grids
		  ○ made up of rows and columns
		  ○ vertical line of grid items is a column
		  ○ horizontal line of grid items is a row
		  ○ spaces between each column/row are called gaps
		  ○ lines between rows are called row lines
		• CSS frameworks
		  ○ software framework that is meant to make it easier to make sure a web page/web site complies to standards using CSS language
			• Co-operation
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		○ bootstrap is a widely used example. Tutors to provide tasks that will enable the students to: • name/identify/explain/describe the layouts and CSS framework. Note: Students will create a fixed layout web page in Activity 25. **Resources** • https://www.lifewire.com/fixed-width-vs-liquid-layouts-3468947 • https://www.sitepoint.com/resizing-fixed-fluid-or-responsive-layouts/ • https://visme.co/blog/layout-design/ • https://www.w3schools.com/css/css_grid.asp • https://en.wikipedia.org/wiki/CSS_framework	
		Activity 22 : Designing web pages Tutors to lead discussion or students to work in small groups to research designing web pages including: • wireframes ○ sketch of a web page/web site before any design or development takes place • mock-ups ○ look more like a finished product or prototype but is not interactive and not clickable; it is still a graphical representation • web-design style guide ○ document that details the elements and patterns of a web page/site, e.g. headers, links, buttons, etc. • prototyping ○ close to the finished product ○ processes can be simulated, and user interaction tested ○ looks very similar to the finished product. Tutors to provide tasks that will enable the students to: • name/identify/explain/discuss/describe/use a wireframe • name/identify/explain/discuss/describe/use a mock-up • name/identify/explain/discuss/describe/use a web-design guide • name/identify/explain/discuss/describe/use prototyping.	

		Note: Students will create a prototype of a web page using a wireframe and web-design style guide in Activity 26. **Resources** • https://www.youtube.com/watch?v=T0vt3nLZKks • https://uxplanet.org/basic-ui-ux-design-concept-difference-between-wireframe-prototype-e38cd3580543 • https://premium.wpmudev.org/blog/web-design-style-guide/ • https://www.mockplus.com/blog/post/wireframe-mockup-prototype-selection-of-prototyping-tools	

		Activity 23: Effective page layout and design
		
		Tutors to lead discussion or students to work in small groups to research designing web pages including:
		
		• an effective web page/site design should meet its intended function by getting across its message while engaging the user by considering:
		  ○ visual hierarchy
		  ▪ the order in which a user processes information of a page
		  ▪ taken into account so that users can understand information easily
		  ▪ F pattern is a reading pattern that roughly resembles the letter F
		  • Users first read in a horizontal movement, usually across the upper part of the content area. This initial element forms the F's top bar.
		  • Next, users move down the page a bit and then read across in a second horizontal movement that typically covers a shorter area than the previous movement. This additional element forms the F's lower bar.
		  • Finally, users scan the content's left side in a vertical movement. This last element forms the F's stem.
		  ○ flow
		  ▪ guiding the user through the page
		  ○ colour theory	

		▪ colour wheel – circle of colour hues showing relationships between primary, secondary and tertiary colours ▪ monochromatic – uses a single base colour and any number of hues of this colour ▪ analogous – use any three colours that are side by side on a 12-part colour wheel; usually one of the three colours is predominant ▪ complementary – uses colours that are located opposite each other on the colour wheel ▪ triadic – use three colours equally spaced around the colour wheel ▪ compound – nearly the same as complementary except it uses colour on both sides of the opposite colour ○ balance and contrast ▪ balance – the distribution of elements of the design • symmetrical – mirrored balance of elements across a centre line • asymmetrical – no mirrored balance of elements across a centre line • radial – elements focussed around a central point rather than line ▪ contrast • can be through colours or use of shapes, space, textures, etc. Elements that are the opposite of each other. Tutors to provide tasks that will enable the students to: • name/identify/explain/discuss/describe ○ visual hierarchy ○ flow ○ colour theory ○ balance and contrast • assess the effectives of a page layout and design • analyse a page layout and design and provide recommendations. **Resources** • https://www.youtube.com/watch?v=7wvQd3-nDCs • https://www.youtube.com/watch?v=oztFP1eBjY8 • https://www.interaction-design.org/literature/topics/visual-hierarchy	

		• https://tomkenny.design/articles/the-principles-of-good-web-design-part-1-layout/ • https://www.w3schools.com/colors/colors_monochromatic.asp • https://www.w3schools.com/colors/colors_analogous.asp • https://www.w3schools.com/colors/colors_complementary.asp • https://www.w3schools.com/colors/colors_compound.asp • https://www.smashingmagazine.com/2015/06/design-principles-compositional-balance-symmetry-asymmetry/ • https://www.wix.com/blog/2018/07/7-principles-of-design-websites/	

14-15	**10.1.2** Be able to use web typography:
	a. web-safe fonts
	b. embedding web fonts.
	
	11.1.5 Be able to use semantic markup to add textual meaning:
	d. quotations
	e. abbreviations and acronyms
	f. citations and definitions
	g. address
	h. mark.
	
	11.1.6 Be able to semantically mark up self-contained content:
	a. figure
	b. caption.
		Activity 24: Web typography
		
		Tutors to discuss/demonstrate:
		
		• web-safe fonts
		  ○ each device comes with its own pre-installed font selection based largely on its operating system
		  ○ problem is systems can differ
		  ○ if this is not taken into account, then the font a user sees when they view a web page may not be the font that was intended
		  ○ web-safe fonts are fonts that will appear across all operating systems
		  ○ designers should specify fonts to fall back to if the font they use is not recognised by a particular operating system
		  ○ common web safe fonts include Arial, Helvetica, Times New Roman, etc.
		• embedding web fonts.
		
		Tutors to provide tasks that will enable the students to:
		
		• identify/explain/discuss/describe/use web-safe fonts
		• describe how to embed web fonts.
		
		Activity 25: Semantic mark up
		
		Tutors to discuss/demonstrate:
		• semantic markup that adds textual meaning
		  ○ quotations
		  ▪ <blockquote></blockquote> long section that is quoted from another source
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		▪ <q></q> short quotation ○ abbreviations and acronyms ▪ <abbr></abbr> ○ citations and definitions ▪ <cite></cite> ▪ <dfn></dfn> ○ address ▪ <address></address> ○ mark ▪ <mark></mark> • semantic markup for self-contained content ○ figure ▪ <figure></figure> ○ caption ▪ <caption></caption> Tutors to provide tasks that will enable the students to: • use semantic markup • identify/explain the code/errors in the code • add to or amend the code.	
		Activity 26: Create web-page using a fixed layout Tutors to provide a wireframe, style guide and assets for students to use in order to create a fixed layout web page. The sample and additional sample material should be used as a general guide as to the type of information they will need in order to complete the task.	
16-17	**10.3.1** Be able to design intuitive navigation systems: a. horizontal scroll menu b. vertical menu c. dropdown menus d. breadcrumb navigation e. button groups.	**Activity 27: Intuitive navigation** Tutors to discuss/demonstrate: • horizontal scroll menus ○ menus that are placed across a screen ○ the navigation is generally always visible although some of the items may not be ○ can be more natural to use on devices where you 'swipe' ○ can be good to use on responsive pages as may not require any changes between desktop and mobile versions ○ forces users to scan horizontally, which may not be a bad thing as that is what people do when they read	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		○ can be more visible than vertical menus ○ can be easier to find that vertical menus ○ may not be so good for those using a mouse ○ some of the content may be missed as users don't expect further menu items to be there • vertical menus ○ menus that are placed down the screen either left or right ○ can be faster and more efficient for users to scan ○ can be less visible than horizontal menus as sometimes there are too many items to fit on the screen ○ can be better than horizontal for those using a mouse ○ may not be suitable for tablets or mobile phones, etc. as those can take up too much space unless the designer provides methods of hiding/viewing it • dropdown menus ○ list of items that appear when clicking a button, icon or text selection ○ good for grouping pages into categories ○ good for showing a large list of choices without taking up much space ○ good for big sites with many sections as can improve usability ○ would only be on screen as and when needed ○ can mean users skip top-level pages ○ can be difficult to scroll, e.g. with mouse ○ users have to click to see what options are available • breadcrumb navigation ○ should use if there is a large amount of content in a strict linear or hierarchical structure with definite categories ○ should not be used for single-level websites that have no logical hierarchy or grouping ○ navigation that allows a user to see where the current page is in relation to the web site's hierarchy ○ easily shows where a user can go on the site ○ users may not use this design element, but they understand what they are showing and can use them • button groups ○ series of buttons grouped together on a single line in a button group. Tutors to provide tasks that will enable the students to:	

		• identify/explain/discuss/describe the different types of navigation systems • describe how code examples work • identify/explain the code/errors in the code • add to or amend code • assess the suitability of navigation systems • analyse navigation and make recommendations as to how it could be improved. **Resources** • https://www.web-designlondon.co.uk/horizontal-scrolling/ (talks about horizontal scrolling in general but still useful • https://codepen.io/mahish/pen/RajmQw demonstration of a horizontal scroll menu • https://www.w3schools.com/howto/howto_css_menu_horizontal_scroll.asp demonstration of a horizontal scroll menu • https://www.wix.com/blog/2015/07/vertical-navigation-menus-should-you-use-them/ • http://www.grayboxpdx.com/blog/post/making-a-comeback-the-vertical-menu • https://designshack.net/articles/navigation/side-navigation-trend/ • https://www.w3schools.com/howto/howto_css_vertical_menu.asp demonstration of a vertical menu • https://baymard.com/blog/drop-down-usability • https://uxplanet.org/breadcrumbs-for-web-sites-what-when-and-how-9273dacf1960	
18	**8.3.4** Understand responsive design techniques.	**Activity 28: Responsive design techniques** Tutors to discuss/demonstrate: • a responsive web page is a page that adapts to any screen size so it is as easy to use on a mobile as it is on a tablet or desktop • the design instinctively adapts to the device it is being viewed on, so it looks just as good on any device • can use a fluid layout • uses HTML and CSS to automatically resize, hide, shrink or enlarge elements • in terms of this unit creating a responsive web page, responsive will include ○ <meta name="viewport" content="width=device-width, initial-scale=1.0"> needs to be added to the head element	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		○ specifying the design for mobiles first in CSS ○ adding a breakpoint for devices 600px or over (@media only screen and (min-width: 600px) ○ specifying the design for these larger devices in CSS. Tutors to provide: • a mock-up of a mobile web page • a mock-up of how the page should look when the browser reaches 600px • assets • a style guide detailing the design below 600px and the differences at 600px and over. **Resources** • a responsive layout template has been included • https://www.youtube.com/watch?v=snQp757_Rr0 • https://www.w3schools.com/css/css_rwd_mediaqueries.asp	
19	**10.3.2** Understand how to target specific devices and browsers: a. cross-browser compatibility b. functionality and usability testing c. code validation d. browser development tools.	**Activity 29: Targeting specific devices and browsers** Tutors to discuss/demonstrate targeting specific devices and browsers: • cross-browser compatibility ○ is the manner in which your web page, web site or web applications work properly across all browsers ○ the aim is to provide all users with the same experience across all browsers • functionality and usability testing ○ functionality testing ▪ assesses whether the web page, web site or web applications works the way it should (behaves according to the functional requirements) without taking design principles into consideration ○ usability testing ▪ focuses on how well the customer can use the web page, web site or web app to complete the required task ▪ tests overall structure, navigational flow, layout of elements on a page, clarity of content and overall behaviour ○ code validation	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		▪ the process of checking that the coding of an HTML or XHTML web page follows the standards and recommendations set by the World Wide Web Consortium (W3C) for the web ▪ can validate HTML, CSS and links • HTML validators check that HTML coding complies with the HTML standards set by the W3C • CSS validators check that CSS coding complies with the CSS standards set by W3C • link validators check that the hyperlinks work correctly ○ browser development tools ▪ most popular browsers have built-in tools to help web developers and many additional plugins can be added. Tutors to provide tasks that will enable the students to: • explain/describe/discuss cross-browser compatibility • explain/describe/discuss functionality and usability testing • explain/describe/discuss code validation • explain/describe/discuss browser development tools • carry out functionality and usability testing • use code validators • use browser development tools. **Resources** • https://medium.com/@sarahelson81/what-is-cross-browser-compatibility-and-why-we-need-it-b41423c3501a - what is cross browser compatibility and similar concepts • https://www.qualitestgroup.com/white-papers/functional-testing-vs-usability-testing/ - difference between functional and usability testing • https://chatbotslife.com/9-ways-to-avoid-cross-browser-compatibility-issues-ada192ef47bf - cross browser issues • http://www.htmlbasictutors.ca/code-validation-clean-code.htm - code validation • http://validator.w3.org/ - HTML validator • http://jigsaw.w3.org/css-validator/ - CSS validator • http://validator.w3.org/checklink - link validator • https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_are_browser_developer_tools - what are browser development tools	

		• https://developers.google.com/web/tools/chrome-devtools/?hl=en – Chrome developer tools • https://getfirebug.com/ - Firebug developer tools • https://developer.mozilla.org/en-US/docs/Tools - Firefox developer tools • https://developer.apple.com/safari/tools/ - Safari developer tools • https://docs.microsoft.com/en-gb/microsoft-edge/devtools-guide - Microsoft Edge developer tools	

20	**7.1.3** Understand how the head element is used to supply information about the document:
	a. metadata.
	
	7.1.5 Understand how global attributes are used to define elements:
	a. tabindex
	b. data
	c. hidden.
	
	10.2.1 Understand how to make websites accessible to the widest possible audience referring to the Web Content Accessibility Guidelines (WCAG).
	
	11.1.4 Understand how semantic code is used by search engines (search engine optimisation (SEO)).
	
	11.1.7 Know how Web Accessibility Initiative – Access Rich Internet Applications (WAI-ARIA) can be used to provide additional semantics and improve accessibility:
		Activity 30: Accessibility
		
		Tutors to demonstrate/discuss:
		
		• Web accessibility is the inclusive practice of ensuring there are no barriers that prevent interaction with, or access to, websites on the World Wide Web by people with disabilities
		• All users should have equal access to information and functionality
		• Web accessibility aims to include
		  ○ visual impairments including blindness, various common types of low vision and poor eyesight and various types of colour blindness
		  ○ mobility issues, e.g. difficulty or inability to use hands
		  ○ auditory (hearing) issues, e.g. deaf or hard of hearing
		  ○ seizures, e.g. people who suffer from epileptic seizures caused by flashing effects, etc.
		  ○ cognitive and intellectual issues, e.g. developmental or learning difficulties, etc.
		• WCAG
		  ○ guidelines
		  ▪ perceivable – information cannot be invisible to all of user's senses
		  ▪ operable – interface cannot require interaction that a user cannot perform
		  ▪ understandable – content or operation cannot be beyond their understanding
		  ▪ robust – interpreted reliably by a wide variety of user agents, including assistive technologies
		• WAI-ARIA
		  ○ technical specification published by W3C that specifies how to increase the accessibility of web pages for assistive technologies such as screen readers
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	a. functionality through roles b. states and properties c. live regions for dynamic content d. enhanced keyboard navigation.	○ set of attributes for plugging the accessibility gaps in HTML5 semantics ○ has no effect on how page elements are displayed or behave in browsers ○ only use where HTML5 semantics are not available or not sufficient ○ functionality through roles – enables the classification of otherwise meaningless tags ○ states and properties – information on how to interact with a particular widget ○ live regions for dynamic content – allow notifications whenever there are changes in that particular part of the page ○ enhanced keyboard navigation – allows every HTML element to receive keyboard focus ○ things to think about ▪ images ▪ audio and video ▪ colours ▪ text ▪ links ▪ forms ▪ navigation and site structure (see checklist 1 below) ▪ page regions ▪ labelling regions ▪ headings ▪ content structure (see checklist 2 below) ○ tabindex, data, hidden ▪ tabindex – can be used to set the focus on elements in a particular order when the user uses the tab key; tabindex=0 is default order, -1 means no tab stop. Only use them where absolutely necessary, e.g. custom elements that would not ordinarily receive focus from tab ▪ data – used to store custom data (data-*) ▪ hidden – Boolean attribute, when present specifies whether an element should be seen/heard; useful for hiding elements that people with disabilities are not required to see or hear, e.g. people with visual impairments would not need to see asterisks for compulsory fields.	

		• SEO
		  ○ name given to activity that attempts to improve search engine rankings
		  ○ should ensure a website can be found in search engine for words and phrases relevant to what the site is offering
		  ○ some of most important techniques to use
		  ▪ remove anything that slows down the site
		  ▪ link to other websites with relevant content
		  ▪ write for humans first, search engines second
		  ▪ encourage other trustworthy sites to link to you
		  ▪ use web analytics to see what is working and what is not
		  ▪ write unique and relevant meta descriptions for every page
		  ▪ use readable and meaningful URLs only
		  ▪ use the right keywords in your images
		• metadata
		  ○ data about the HTML document, e.g.
		  ▪ meta name=”description” content=”Free CSS Lessons”>
		  ▪ meta name=”keywords” content=”HTML, CSS”>
		
		Tutors to provide tasks that will enable the students to:
		
		• explain/describe/discuss web accessibility
		• explain/describe/discuss WCAG
		• explain/describe/discuss HTML5 semantics
		• explain/describe/discuss WAI-ARIA
		• explain/describe/discuss SEO (including metadata)
		• use metadata
		• assess how well particular web pages incorporate accessibility through the use of semantics, WCAG and WAI-ARIA
		• analyse the use of semantics, WCAG and WAI-ARIA employed on particular pages and make recommendations where appropriate
		• assess how well particular web pages incorporate SEO techniques
		• analyse the use of SEO techniques used on a particular web page and recommend improvements.
		
		Resources
		
		• https://www.w3.org/TR/UNDERSTANDING-WCAG20/intro.html#introduction-fourprincs-head	

		• http://www.disability.wa.gov.au/Global/Publications/Understanding%20disability/Built%20environment/Accessible%20websites%20checklist.pdf – checklist 1 • https://www.w3.org/WAI/tutorsials/page-structure/ - checklist 2 • http://heydonworks.com/practical_aria_examples/ - examples of ARIA in action • https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/tabindex - tab index • https://www.w3schools.com/tags/att_global_data.asp - data • https://www.w3schools.com/tags/att_hidden.asp - hidden • https://www.youtube.com/watch?v=B4IqW-5a16o – SEO • https://www.youtube.com/watch?v=sd0ypO9MTWY - SEO	

Topic 9: Understanding the functions of JavaScript **JavaScript is the programming language that, when applied to a HTML document, provides dynamic interactivity, such as image sliders, galleries and fluctuating layouts. To make web pages interactive, the JavaScript selects elements on the page. Students will select an element through the Document Object Model (DOM), and affect it to make it behave a certain way when a user interacts with it. They will develop an understanding of fundamental programming concepts and the syntax of the language to enable them to use JavaScript to create interactive web pages that respond to user actions.**			
Week	**Topic area / aims / learning outcomes**	**Exemplar classroom activities / teaching points / suggested teaching resources**	**Integrated Transferable Skills**

21-22	**7.1.3.** Understand how the head element is used to supply information about the document:
	a. scripts.
	
	9.3.1 Understand how to add JavaScript to web pages.
	
	9.3.2 Be able to program functionality:
	a. comments
	b. assignment
	c. selection
	d. variables
	e. subprograms.
	
	9.3.5 Understand how to use error handling and debug JavaScript.
	
	9.3.4 Understand how to combine JavaScript with HTML and CSS to create page components:
	a. pop-ups.
		Activity 31: Programming functionality
		
		Tutors to discuss/demonstrate:
		
		• writing single line comments
		• writing multi line comments
		• using alerts to display as pop ups
		• declaring variables and constants
		• assigning values to variables and constants
		• carrying out calculations
		  ○ addition
		  ○ subtraction
		  ○ division
		  ○ multiplication
		  ○ use BODMAS/BIDMAS effectively
		• using selection including
		  ○ IF…THEN
		  ○ IF…THEN…ELSE
		  ○ IF…THEN…ELSEIF
		  ○ IF…THEN…ELSEIF…ELSE
		  ○ SWITCH
		• using comparison operators including
		  ○ == equal to
		  ○ === equal value and equal type
		  ○ != not equal
		  ○ !== not equal value or not equal type
		  ○ > greater than
		  ○ < less than
		  ○ >= greater than or equal to
		  ○ <= less than or equal to
		  ○ IsNaN is not a number
		• Using logical operators including
		  ○ && and
		  ○ || or
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		○ ! not • Using string methods ○ length ○ substring ○ substr ○ replace ○ toUpperCase ○ toLowerCase ○ split ○ charAt • subprograms ○ function declarations ○ calling functions ○ passing parameters ○ executing functions ○ returning values (when required). Tutors may use 'prompt' and 'alert' pop up boxes for input at this stage. Form input and similar concepts are covered in later weeks. Tutors should explain the purpose of *try*, *catch*, *throw* and finally *in error handling* though it does not need to be coded. There is no preference as to whether students terminate lines using semi-colons. Tutors to provide tasks that will enable the students to: • create code to cover all the above • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://www.w3schools.com/js/js_string_methods.asp • https://www.w3schools.com/jsref/jsref_try_catch.asp	
23	**9.3.2** Be able to program functionality: a. repetition b. iteration.	**Activity 32: Repetition and iteration** Tutors to explain that: • repetition is an instruction or instructions that need(s) repeating • iteration is the carrying out of this/these instruction(s).	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new

		Tutors to discuss/demonstrate:
		
		• definite iteration
		  ○ loops where the exact number of iterations can be defined
		  ○ for loop
		• indefinite iteration
		  ○ the exact number of iterations cannot be defined
		  ○ while loop
		  ▪ condition will be tested at beginning of loop
		  ▪ code within loop may or may not be executed
		  ○ do…while loop
		  ▪ condition will be tested at end of loop
		  ▪ code within loop will be executed at least once
		  ○ increment
		  ○ decrement.
		
		Tutors to provide tasks that will enable the students to:
		
		• create code to cover all the above
		• describe how code examples work
		• identify/explain the code/errors in the code
		• add to or amend code.
			situations/contexts

24	**9.3.2** Be able to program functionality:
	a. 1D and 2D data structures.
		Activity 33 : Arrays
		
		Tutors to explain the difference between 1D and 2D arrays:
		
		• 1D array can be thought of as a single column, multiple row table
		• 2D array can be thought of as a multiple column, multiple row table
		• both are used to store multiple values in a single variable.
		
		Tutors to discuss/demonstrate 1D arrays including:
		
		• declaring and initialising an empty array
		  ○ var names = []
		• declaring an array and initialising it with values, e.g.
		  ○ var names = [“Simon”,“Steve”,“Mobin”]
		  ○ var ages = [12,16,19,21,45]
		• adding values to an array, e.g.
		  ○ ages[1] = 12
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		○ names[2]= "Julie" ○ age.push(10) • sorting an array alphabetically ○ names.sort() • setting the length of an array, e.g. ○ age.length=10 • finding the length of an array using array.length • looping through arrays, e.g. loop to find a value within the array. Tutors to provide tasks that will enable the students to: • create code to cover all the above • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://www.quora.com/What-is-the-difference-between-one-dimensional-area-and-a-multidimensional-array • https://www.w3schools.com/js/js_arrays.asp	
25-26	**7.3.3** Be able to create a form on a web page: a. form structure b. form elements c. form controls d. form buttons e. organising and grouping for elements f. input types g. text areas h. drop-down lists. **7.3.4** Understand how web forms work: a. how information is sent from the browser to the server b. form validation.	**Activity 34: DOM** Tutors to lead discussion or students to work in small groups to research the HTML Document Object Model (DOM) and how JavaScript works with it including: • the HTML DOM is an Object Model for HTML and a programming interface for JavaScript • the HTML DOM defines ○ HTML elements as objects ○ properties of all HTML elements ○ methods to access all HTML elements ○ events for all HTML elements • JavaScript can ○ add/change/remove HTML elements ○ add/change/remove HTML attributes ○ add/change/remove CSS styles ○ react to HTML events ○ add/change/remove HTML events.	• Co-operation • Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

	8.2.4 Be able to styles to elements: a. forms. **9.1.1** Understand how the Document Object Model (DOM) allows JavaScript to access and update the contents of a web page while it is in the browser window. **9.2.1** Understand regular expressions. **9.2.2** Understand regular expressions used for validation check. Search for matching: a. letters and sequences of upper/lower case characters b. numbers c. punctuation and other symbols. **9.2.3** Be able to interpret and construct patterns consisting of repeating characters and digits. **9.3.3** Understand how events can be used to trigger a function: a. form events.	Tutors to explain that the 'document' object represents the web page in the browser and that to access any element in the HTML page you would always start with accessing the document object. Tutors to provide tasks that will enable the students to: • name/identify/explain/discuss/describe the HTML DOM and how JavaScript works with it. **Resources** • https://www.w3schools.com/whatis/whatis_htmldom.asp • https://www.w3schools.com/js/js_htmldom.asp **Activity 35: Creating forms** Tutors to discuss/demonstrate forms including: • <form></form> to define the form • these form elements/attributes/types/properties ○ <input> ▪ type • button • checkbox • radio • submit • text (tutors could include number, tel, password, etc. if they want to) • url ○ <select></select> (drop-down list) ▪ <option></option> (options that can be selected) ▪ selected ○ <textarea></textarea> (multi-line input field) ▪ rows ▪ columns ○ <label></label> (label for an <input> element) ○ <fieldset></fieldset> (to group related elements in a form) ○ <legend></legend> (caption for a <fieldset> element) ○ placeholder (hint to describe what to enter) ○ CSS styling (e.g. input field widths, padding, margins, box-sizing, borders) Tutors to provide tasks that will enable the students to:	

		• create code to cover all the above • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://www.w3schools.com/css/css_form.asp • https://developer.mozilla.org/en-US/docs/Learn/HTML/Forms/Styling_HTML_forms	
		Activity 36: Events and validation Tutors to provide short introduction to events and their use in form validation: • event - action a browser or user does that can be detected • event handler – attribute added to an HTML element used to trigger functions in JavaScript code • can be used to trigger validation checks (e.g. **clicking** submit button). Tutors to discuss/demonstrate validating forms using JavaScript including: • presence checks (including making sure input is present, items selected from drop down boxes, checkboxes, etc.) • type checks (including isNaN()) • range checks (between min and max numeric values) • length checks (input is a length). Format checks to be covered in regular expressions. Tutors to provide tasks that will enable the students to: • create code to cover all the above • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. **Resources** • https://www.geeksforgeeks.org/form-validation-using-html-javascript/	

		Activity 37: Regular expressions and validation Tutors to explain that a regular expression is a sequence of characters that define a search pattern. Input is then checked against this pattern to determine whether it is valid. Tutors to explain: • [ ] finds range of characters • ^ look at the beginning of string • | or - pattern can be one thing **or** another • + at least one character • * zero or more character • ? zero or one character • $ looks at end of string • { } how many occurrences • /s a whitespace. Tutors to provide written tasks for students to define particular patterns. Tutors to discuss/demonstrate declaring and defining regular expressions in Javascript such as: • **var regExpExample =/^[A-Z]{7}[a-z]+$/** / start of expression ^ match beginning of string [A-Z] uppercase letters {1} 7 characters only [a-z] lowercase letters + at least one of $ end of string / end of expression This regular expression searches the beginning of the string to make sure there are exactly seven uppercase letters and then checks to make sure the rest of the string is made up of at least one lowercase letter. SUNSETTing – acceptable sunsetTTing – not acceptable SUNSET – not acceptable	

		SUNSETT3ng – not acceptable
		
		• **var regExpExample2 = /[0-9]{3}/**
		
		/start of expression
		[0-9] digits 0 to 9
		{3} 3 digits only
		/end of expression
		
		this regular expression searches for exactly three digits
		
		054 – acceptable
		2 – not acceptable
		a23 – not acceptable
		
		• **var regExpExample3 = /[A-Z]{2}[0-9]{1}\s[0-9]{1}[A-Z]{2}**
		
		/start of expression
		[A-Z]{2} two uppercase letters
		[0-9]{1} one digit
		\s a space
		[0-9]{1} one digit
		[A-Z]{2} two uppercase letters
		
		This regular expression searches for two uppercase letters, followed by one digit, followed by a white space, followed by one digit, followed by two uppercase letters.
		
		BE1 1BB – acceptable
		Bb1 1BB – not acceptable
		0B1 2XY – not acceptable
		
		Tutors to discuss/demonstrate testing input using .test. For example:
		
		if (regExpExample.test(input)===false){
		alert("Must start with an uppercase letter then lowercase letters ")
		}
		
		Tutors to discuss/demonstrate testing whether a word can be found in a string using .match. For example:
		
		var str = "the cat and the dog"	

```
if (res = str.match(/the/g)){
        alert("a match was found")
 }else{
        alert("no match found")
 }
```

Tutors to provide tasks that will enable the students to:

- write a regular expression pattern
- interpret a regular expression pattern
- create code to cover all the above
- describe how code examples work
- identify/explain the code/errors in the code
- add to or amend code.

Tutors could also stress the importance of JavaScript validation by demonstrating how a page with a form can be inspected in the browser and HTML attributes overridden.

Resources

- https://regexr.com/
- https://www.w3schools.com/jsref/jsref_isnan.asp

Activity 38: Sending form information

Tutors to discuss the ***basic*** principles of how form information is sent from the browser to the server.

- At some point forms will normally require interactivity with a server; for example, adding records to a database.
- This interaction is called a *request*.
    1. The request is sent from a user through a client device to the server.
    2. The server acknowledges the request and replies to the client device with the results of the request.
    3. The client device displays the results to the user.

Students to produce an annotated diagram of the process.

27	**9.3.3** Understand how events can be used to trigger a function: a. User Interface (UI) events b. keyboard events c. mouse events d. focus and blur events.	**Activity 39: Events in more depth** Tutors to recap events and explain that the user interface can trigger events including keyboard, mouse, focus and blur events. Students could work in small groups to research these events or tutors could provide worksheets for students to complete. *Keyboard events* • keydown – triggered when any key is pressed down, triggered first, and always before the browser processes the key • keypress – triggered when a key that produces a character value is pressed down, triggered after keydown, and before the browser processes the key • keyup – triggered when any key is released, triggered last, and the browser processes the key *Mouse events* • mousedown – triggered when a user presses a mouse button over an element • mouseup – triggered when a user releases a mouse button over an element • onclick – triggered when a user clicks a button *Focus and blur events* • focus – triggered when an element gets focus • blur – triggered when an element loses focus *Other events* • load – triggered immediately after a page has been loaded • change – triggered when a user changes the selected option of a <select> element Tutors to provide tasks that will enable the students to: • create code to cover all the above (change will be covered in weeks 28 to 29) • describe how code examples work • identify/explain the code/errors in the code	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		• add to or amend code. **Resources** • https://www.mutuallyhuman.com/blog/2018/03/27/keydown-is-the-only-keyboard-event-we-need/ • https://www.w3schools.com/js/js_events.asp • https://www.w3schools.com/js/js_htmldom_events.asp	
28-29	**9.3.4** Understand how to combine JavaScript with HTML and CSS to create page components: a. slideshow b. modal boxes c. modal images d. filter list e. sort list f. pop-ups g. tabbed content.	**Activity 40: Slideshow** Tutors to discuss/demonstrate a slide show program that: • includes multiple images • includes a next and back button • displays the first slide when the page loads • moves to the next image when the next button is clicked • moves to the previous image when the back button is clicked • moves to the first image when the next button has been clicked and the last image has been viewed • moves to the last image when the back button has been clicked and the first image has been viewed. Tutors to provide tasks that will enable the students to: • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. Tutors could use the code in the resources for these tasks. **Resources** • https://www.makeuseof.com/tag/how-to-build-javascript-slideshow/ • https://www.w3schools.com/w3css/w3css_slideshow.asp • http://javascript-tutors.net/index.php/lesson-29-creating-slideshow-in-javascript/	• Communication • Adaptive learning • Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts
		Activity 41: Modal boxes Tutors to explain that modal disables the main window but keeps it visible and that users must interact with the modal window before they can return to the main window.	

		Tutors to discuss/demonstrate modal boxes including: • displaying the modal box • closing the modal box • textual content • action button(s), e.g. *yes*, *no* • drawing attention to the model by darkening the background of the page. Tutors to provide tasks that will enable the students to: • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. Tutors could use the code in the resources for these tasks. **Resources** • https://www.w3schools.com/howto/howto_css_modals.asp • https://sabe.io/tutorsials/how-to-create-modal-popup-box	
		Activity 42: Modal images Tutors to discuss/demonstrate modal images including: • displaying the modal image • closing the modal image. Tutors to provide tasks that will enable the students to: • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. Tutors could use the code in the resources for these tasks. **Resources** • https://www.w3schools.com/howto/howto_css_modal_images.asp • https://codepen.io/Muhnad/pen/dMbXNb	

		Activity 43: Filter list Tutors to discuss/demonstrate filter lists including: • a filter list used to narrow down a list to a specific search item • a search input field • search items • a method of filtering the search items depending on the contents of the search field. Tutors to provide tasks that will enable the students to: • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. Tutors could use the code in the resources for these tasks. **Resources** • https://www.w3schools.com/howto/howto_js_filter_lists.asp • https://www.w3schools.com/howto/howto_js_filter_table.asp	
		Activity 44: Sort list Tutors to discuss/demonstrate sort lists including: • an unsorted list • a button to trigger the process of sorting the list • the sorted list to be displayed in place of the unsorted list. Tutors to provide tasks that will enable the students to: • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. Tutors could use the code in the resources for these tasks: • sortingListExample.html • https://www.w3schools.com/howto/howto_js_sort_list.asp	

		Activity 45: Tabbed content Tutors to discuss/demonstrate tabbed content including: • tabs • different content depending on the tab selected. Tutors to provide tasks that will enable the students to: • describe how code examples work • identify/explain the code/errors in the code • add to or amend code. Tutors could use the code in the resources for these tasks. **Resources** • https://www.w3schools.com/howto/howto_js_tabs.asp • https://www.101computing.net/creating-tabs-in-html-css-js/	

30	**9.3.2** Be able to program functionality:
	a. object orientation.
		Activity 46: Object orientation
		
		Tutors to discuss/demonstrate basic object orientation including:
		
		• JavaScript is an object-based language based on prototypes
		• This allows you to create hierarchies of objects and to have inheritance of properties
		• user defined objects types
		  ○ can be used for complex kinds of variables
		  ○ can store multiple data items and functions
		  ○ allow you to group related data items into a single object
		  ○ allow you to create as many 'instances' of that particular object type as you want to
		  ○ can be thought of as a 'blueprint' for objects
		  ○ the 'blueprints' can be expanded.
		
		Tutors to provide tasks that will enable the students to:
		
		• explain/describe/discuss JavaScript object orientation
		• create an object type
		• create objects of that type.
			• Communication
			• Adaptive learning
			• Adapting prior knowledge, skills and experience of IT to deal with new situations/contexts

		Tutors could use the task provided (objectTutorResource.pdf and ooExample.html). **Resources** • objectTutorResource.pdf • ooExample.html • https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Details_of_the_Object_Model • https://www.tutorsialspoint.com/javascript/javascript_objects.htm • https://www.w3schools.com/js/js_object_definition.asp	

FOR INFORMATION ABOUT EDEXCEL, BTEC OR LCCI QUALIFICATIONS
VISIT QUALIFICATIONS.PEARSON.COM

EDEXCEL IS A REGISTERED TRADEMARK OF PEARSON EDUCATION LIMITED

PEARSON EDUCATION LIMITED. REGISTERED IN ENGLAND AND WALES NO. 872828
REGISTERED OFFICE: 80 STRAND, LONDON WC2R 0RL
VAT REG NO GB 278 537121

GETTY IMAGES: ALEX BELMONLINSKY