Precursors to Computers

Precursor	**Something that comes before**. In this case: A machine that is not a computer, but does computer-like things.
Abacus	**The first calculating device.** An ancient tool, made of wood or metal, with movable beads that people use to help to count and do basic math.
Napier's Bones	17th century calculating device; a box containing bone rods with numbers on them, used to do basic math incl. addition, subtraction, multiplication and division.
Pascal's calculator	17th century calculating device; a box containing gears that displayed numbers; used for addition and subtraction. Modified by Leibniz (**Leibniz's calculator**) to do multiplication and division too.
Jacquard loom	A 19th century machine for weaving silk; operators used **punched cards** to tell the machine what patterns to weave.
Analytical engine	**Considered the first mechanical "computer"**. Made by Charles Babbage in the 19th century, it calculated using gears and cranks. Ada Lovelace (seen as **the first programmer**) used it to calculate coefficients of polynomial equations *(e.g. $x^2 + 2x + 1$).*

Generations of Computers

First gen.	1940-1956. Main technology: **Vacuum tubes**. Very large, expensive computers such as **ENIAC** and **UNIVAC**, which took up entire floors of a building and required large cooling systems. Only governments and large companies owned computers. Could only solve one problem at a time; programs took days or weeks to run. Input was through punched cards, output on printed paper, storage on magnetic drums.
Second gen.	1956-1963. Main technology: **Transistors**. Since transistors were smaller, computers became smaller too, as well as faster, cheaper and more energy-efficient. Computers became more common in universities and businesses. Input on punched cards, output on printed paper, storage on magnetic core memory, tape, or disks.
Third gen.	1964-1971. Main technology: **Integrated circuits**. Transistors were made smaller and placed on silicon chips (semiconductors), making them much faster. Computers became easier to use & more popular; people began to use them in small businesses and at home as a hobby. Input on keyboards, output on monitors/screens, storage on magnetic tape or disks. Operating systems developed, increasing computer's power.

Generations of Computers (cont)

Fourth gen.	1971-present. Main technology: **Very large scale integration (VLSI)** which made integrated circuits more complex and powerful, leading to **microprocessors**. Computers could now be very small (phones, portable devices). IBM and Apple made desktop computers and personal computers (PCs). These became extremely popular—anyone could have a computer now! Input on keyboard, mouse, touchscreen; output on monitor; storage on HDD/SSD. Graphical user interfaces (GUIs) and networking became common, and the Internet appeared.
Fifth gen.	Present+. Main technology: **Artificial intelligence**. Parallel processing (with multiple CPUs) and superconductors increased speed & efficiency. Voice and face recognition, natural language interfaces, quantum computing and nanotechnology appeared.

Representing Information

We can **represent information** in different ways. Using different languages, we can write the number nine as **9**, **IX** (Roman), ٩ (Arabic/Persian), ९ (Hindi), ๙ (Thai), 九 (Chinese/Japanese/...), 구 (Korean Hangul), ☐ , ☐, and so on. We can use other codes and visual systems too, like Braille, Morse code, hand signals and semaphore. To represent information (**data**) in computers, we use a number system called **binary** Binary is a *base-2* system, which means it uses only *two* digits, 0 and 1. This is different from the regular *base-10* **decimal** (or denary) system, which uses the ten digits from 0 to 9. As well, we sometimes use a *base-16* **hexadecimal** system, which has *sixteen* digits. Hexadecimal digits include the letters A, B, C, D, E, and F to represent the values 10, 11, 12, 13, 14, and 15. We can write the same value in different ways using different number systems..

Representing Information (cont)

Along with these number systems, we can use two kinds of **character sets** called **ASCII** and **Unicode**. These allow us to represent written characters using numbers. For example, the letter A is represented in ASCII by the decimal number 65, the binary number 01000001, and the hexadecimal number 41. The character 万 is represented in Unicode by the decimal number 19975, the binary number 0100111000000111, and the hexadecimal number 4E07.

ASCII is an 8-bit code. Its great advantage is that it takes up **very little space** to store data—only 8 bits (or one **byte**) for one character. On the other hand, its disadvantage is that it can only represent **a total of 255 characters**. The characters it displays mostly come from Latin languages like English, French, and Spanish.

Unicode has 16-bit and 32-bit varieties. Its great advantage is that it can represent **characters from many languages**, including Arabic, Chinese, Japanese, Korean, Russian, Thai, and many others. On the other hand, its disadvantage is that **each character takes up a lot of space** to store and send over a network.

Numbers in Different Systems

Binary (base 2)	**Decimal** (base 10)	**Hexadecimal** (base 16)
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

The Digital World

What does it mean that **information is digital**? It means that all the information we have can be represented as data on computers, using the *bi*nary digi*ts* (**bits**) **0** and **1**. Any computer can use this data, display it, or send it to someone else. That means you can **store** everything you know on a computer, and you can **share** it with anyone else who uses a computer. Using the Internet and the Web, you can share many types of **media** like words, pictures, video or music clips. *A digital world is a world where it's easy to connect!* One of the ways we can display and share information is using a **website**. If you know how to use tools like **HTML** and **CSS**, it is easy to create your own website to share with the world.

Building Blocks of the Web

HTML	**Hypertext Markup Language**. Called the "building blocks of the Web" because we use it to give structure to a web page.
HTML tag	A code that **explains what something is** on a web page. HTML tags are always surrounded by angle brackets (as in). Most tags must be paired with a matching end tag; between these is the text or object that the tag applies to. For example, **hello** says that 'hello' is bold text.
HTML attribute	An **extra piece of information** about an HTML tag. For example, in the tag <p align="right">, the 'align' attribute says that the paragraph should be aligned to the right. The part between the quotes ("") is called the attribute's **value**.
CSS	**Cascading Style Sheets**. Where HTML acts like building blocks giving structure, CSS acts like paint and decorations giving a web page a beautiful appearance. CSS can be *inline* in an HTML document, or it can exist in its own file (a *stylesheet*).

Building Blocks of the Web (cont)

Inline CSS	A style that can be added **directly to an HTML tag** to change its appearance. For example, in the tag , the 'style' attribute says that the text enclosed by the span tag must be coloured red.
Stylesheet	A separate **file** (with extension .css) **that contains CSS rules**. Writing CSS in a stylesheet makes it easier to update the appearance of large HTML files, because making one change to a rule will update many HTML tags.
CSS rule	A **set of CSS styles** that apply to one or more objects. For example, the rule h1 { color:blue } will make the <h1> tag blue.
CSS declaration	A **single style** in a CSS rule. For example, in the rule h1 { color:#9c9; font-size:120% }, 'color:#9c9' and 'font-size:120%' are both declarations. If there is more than one declaration in a rule, they must be separated by a semicolon (;).
CSS selector	The **object** that a CSS rule applies to. For example, in the rule h1 { text-decoration:underline }, the selector is 'h1'.
CSS class	A kind of CSS selector that applies to HTML tags with the matching 'class' attribute. A selector that starts with a dot (.) represents a CSS class. For example, the rule .purp { color:purple } will match the HTML tag <p class="purp">.
CSS property	The **kind of formatting** a declaration changes. For example, in the rule h1 { font-size:120% }, the property is 'font-size'. The part after the colon (:) is the property's **value**.

Common HTML Formatting Tags

<b>,	Bold text.	<i>,	Italic text.
<u>	Underlined text.		List item.
	Unordered list (bullets).		Ordered list (numbers).
<h1>	Heading, level 1.	<h2>	Heading, level 2.
<h[3-6]>	Heading, levels 3–6.	<sub>	Subscript (as in H_2O).
<sup>	Superscript (as in x^2).	<code>	Code snippet (`monospaced`).

Other Common HTML tags

<p>	Paragraph.		Span of inline text (e.g. a few words inside a paragraph)
	Division, or section, of text.		Image. The 'src' tag is the URL of the image to be shown.
<a href="...">	Hyperlink. The 'href' attribute is the URL of the page where you will go when clicking the link.	 	Line break.

Common HTML Attributes

align	Alignment of an object. ex: *left, right*.
style	Inline CSS that applies to an object. ex: *color:red*.
class	CSS class name that applies to an object. ex: *highlight, red-text, student-name*.
height	How tall an object is. ex: *256px, 2em, 100%*.
width	How wide an object is. ex: *150px, 2em, 75%*.

Common CSS Formatting Properties

color	Colour of text. ex: *red, #000, #ff0000.*
background	Background or highlight. ex: *yellow, #0e0, #cccc99.*
font-size	Size of text. ex: *16pt, 12px, 1em, 120%.*
font-weight	How bold the text is. ex: *bold, normal.*
font-style	Style of the text. ex: *italic.*
font-family	Name or type of font used. ex: *Helvetica, Arial, serif.*
line-height	Space between lines. ex: *24pt, 36px, 2em, 140%.*
text-align	Alignment of text in an object. ex: left, right, center.
text-transform	Case of text. ex: *lowercase, uppercase, capitalized.*

Common CSS Layout Properties

border	Border around an object. ex: *4px solid blue, 1px dotted #fffddff*.
padding	Blank space *inside* the object's border. ex: *4px, .5em*. Multiple values can be added for top, right, bottom, left padding. ex: *2px 0px 4px 0px*.
margin	Blank space *outside* the object's border. ex: *4px, .5em*. Multiple values can be added for top, right, bottom, left margin. ex: *2px 0px 4px 0px*.
height	How tall an object is. ex: *256px, 2em, 100%*.
width	How wide an object is. ex: *150px, 2em, 75%*.