# Computer Science Principles

Web Programming

Unit 1: Internet

CHAPTER 1: REPRESENTING AND TRANSMITTING INFORMATION

DR. ERIC CHOU IEEE SENIOR MEMBER

## Objectives

- This unit explores the technical challenges and questions that arise from the need to represent digital information in computers and transfer it between people and computational devices.
- The unit then explores the structure and design of the internet and the implications of those design decisions.

## Topics

- The first chapter explores the challenges and questions that arise when representing information in a computer or sending it from one computer to another.
- It begins by investigating why on-off signals, also known as binary signals, are used to represent information in a computer. It then introduces the way common information types like text and numbers are represented using these binary signals.
- Finally, it illustrates the importance of establishing shared communication rules, or protocols, for successfully sending and receiving information.

## Topics (Internet)

**Course Motivation**

- Lesson 1: Personal Innovations

**Internet Transmission in Binary Format**

- Lesson 2: Sending Binary Messages
- HTML 2/3: How the Web Works and Web Overview
- Extra 1: Internet Simulator
- Lesson 3: Sending Binary Messages with the Internet Simulator

**Web-Page Format**

- HTML 4/5: Creating First Page and Basic Document Structure
- JavaScript 1: JavaScript Starter

## Topics

**Number System and File Transmission (Internet OSI MAC and Physical Layers)**

- Lesson 4: Number Systems
- Lesson 5: Binary Numbers
- JavaScript 2: JavaScript Data Types
- Lesson 6: Sending Numbers
- Extra 2: Encoding Numbers in the Real World
- Lesson 7: Sending Text

# Personal Innovations

LESSON 1

Welcome to Computer Science Principles! Groups make a “rapid” prototype of an innovative idea and share it. Students watch a brief video about computing innovations.

## Vocabulary

- Innovation - A new or improved idea, device, product, etc, or the development thereof
- Innovation, Renovation, Disruptive Technology, Emerging Technology, Evolutional Technology.

## Action Item

Draw a picture about the life at the year 2050 (30 years later).  Write it down on a paper and keep it safe.

Check your answer when time comes.

# Sending Binary Messages

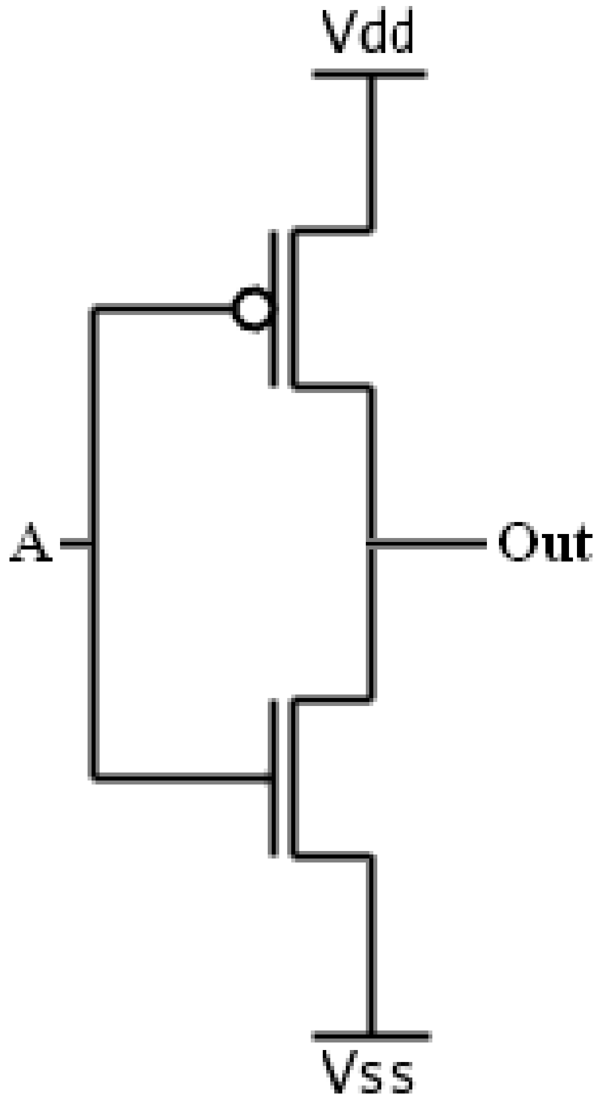
LESSON 2

## Overview

Students collaborate in an iterative design process to make a “bit sending device” using classroom supplies and everyday objects. They develop systems for encoding and sending simple binary messages over a physical distance.

## Vocabulary

- Binary - A way of representing information using only two options.
- Bit - A contraction of "Binary Digit"; the single unit of information in a computer, typically represented as a 0 or 1
- Byte, Word, Integer, Character, Character Set, Binary, Octal, Hexadecimal,
- Kilo, Mega, Giga, Tera



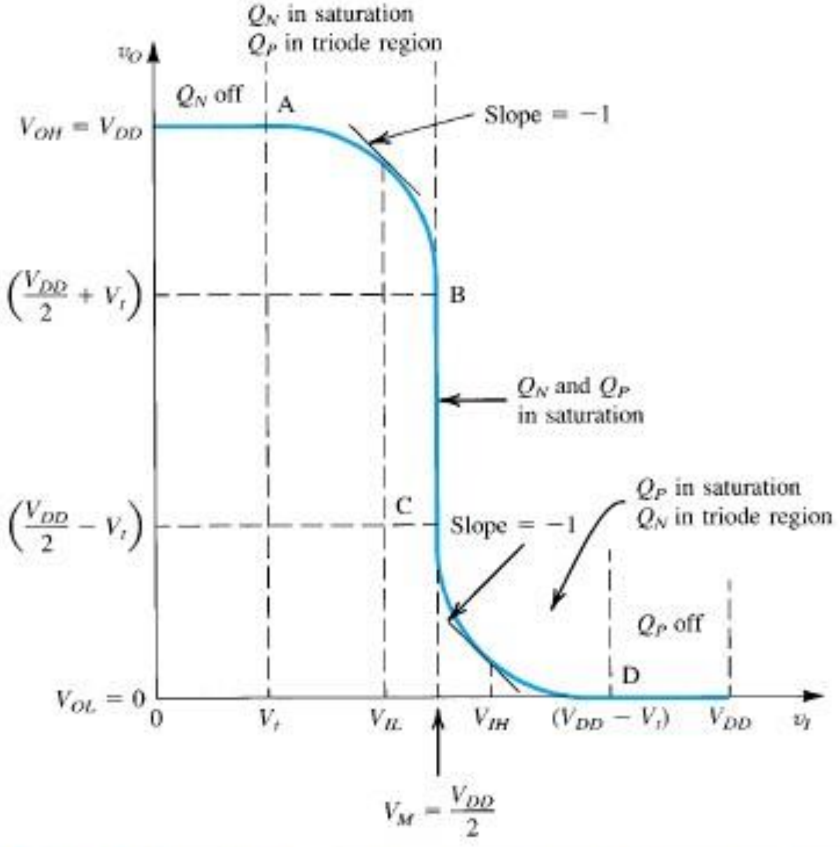
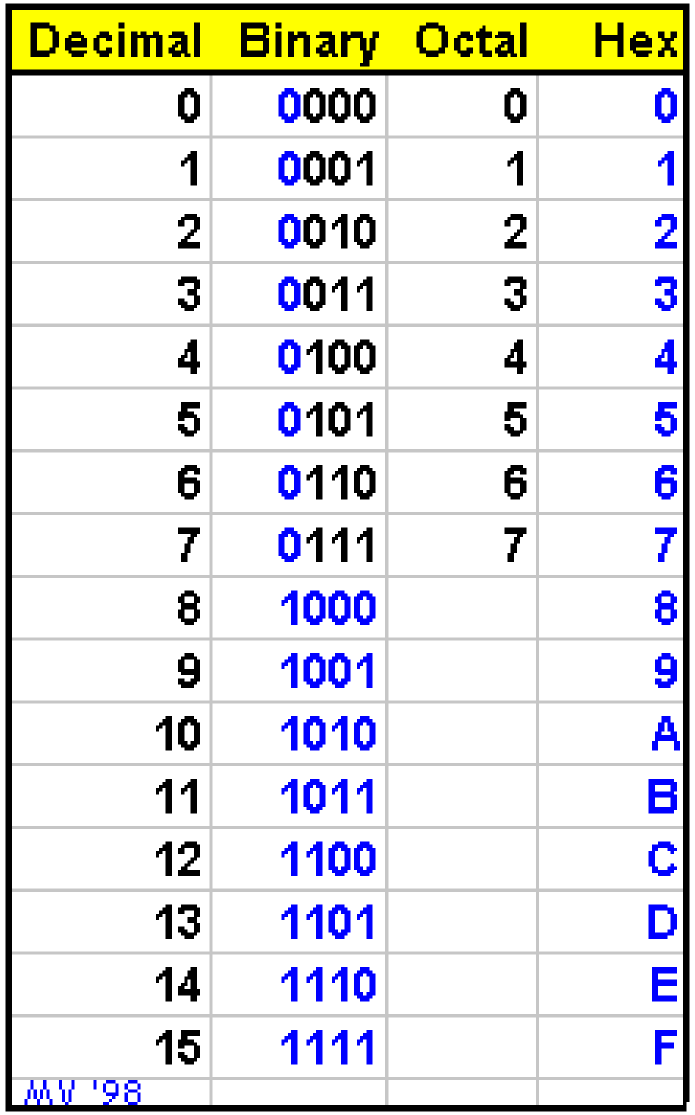
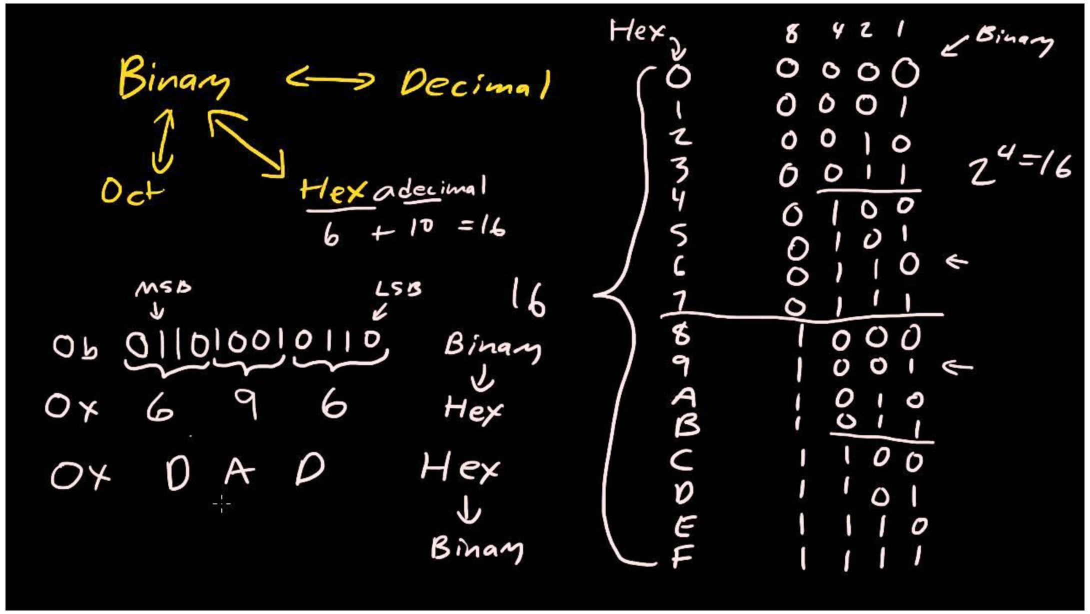


**Figure 13.20** The voltage-transfer characteristic of the CMOS inverter when $Q_N$ and $Q_P$ are matched.



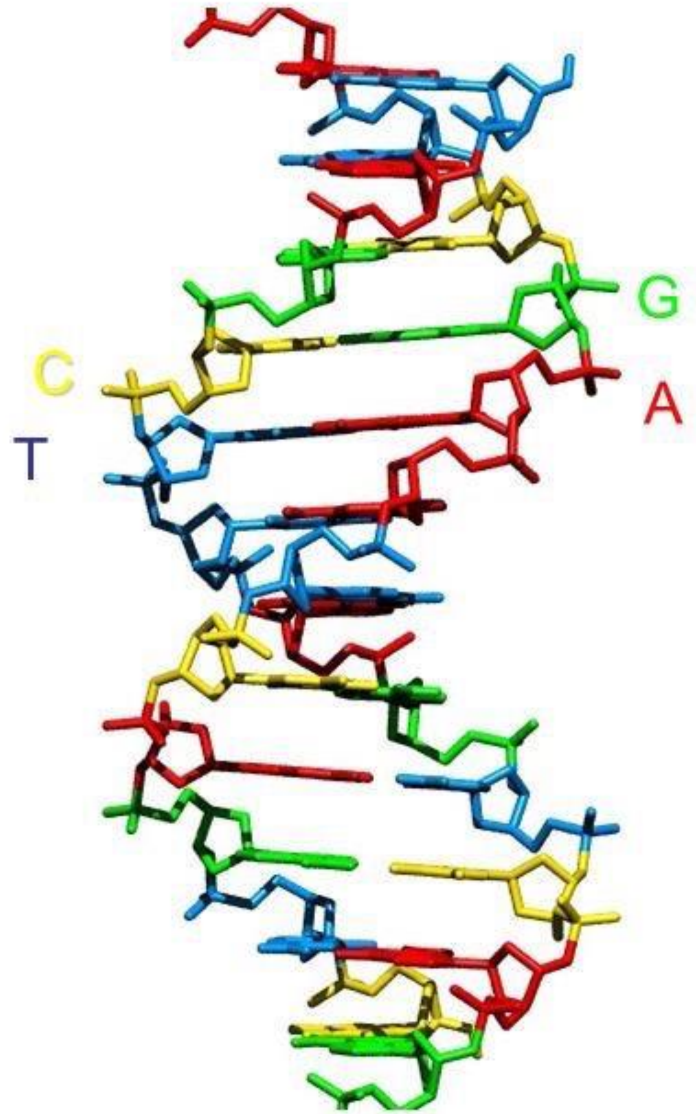
| Decimal | Binary | Octal | Hex |
|---|---|---|---|
| 0 | 0000 | 0 | 0 |
| 1 | 0001 | 1 | 1 |
| 2 | 0010 | 2 | 2 |
| 3 | 0011 | 3 | 3 |
| 4 | 0100 | 4 | 4 |
| 5 | 0101 | 5 | 5 |
| 6 | 0110 | 6 | 6 |
| 7 | 0111 | 7 | 7 |
| 8 | 1000 | | 8 |
| 9 | 1001 | | 9 |
| 10 | 1010 | | A |
| 11 | 1011 | | B |
|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

MV '98

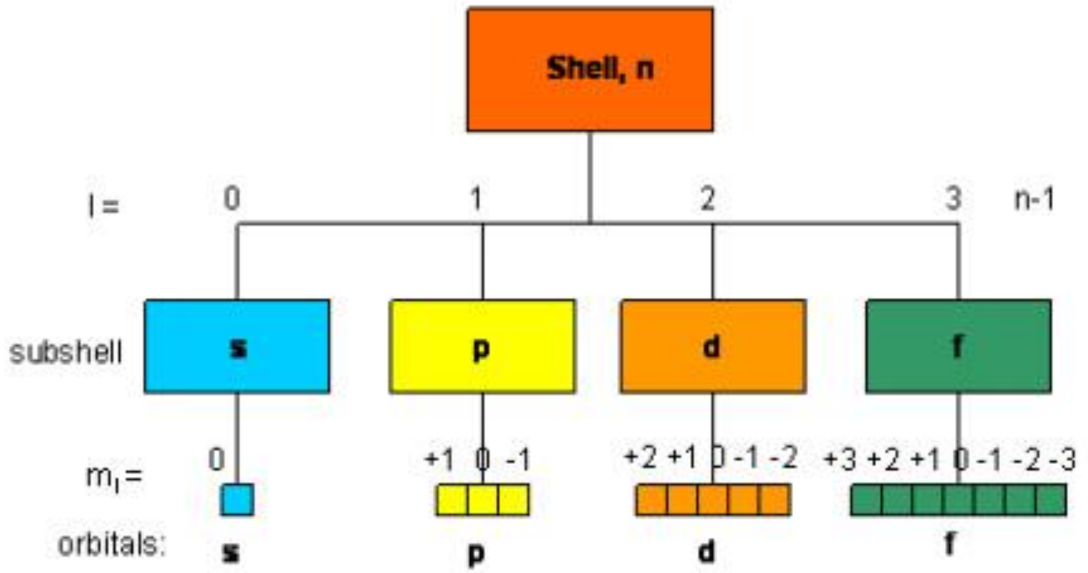


## DNA

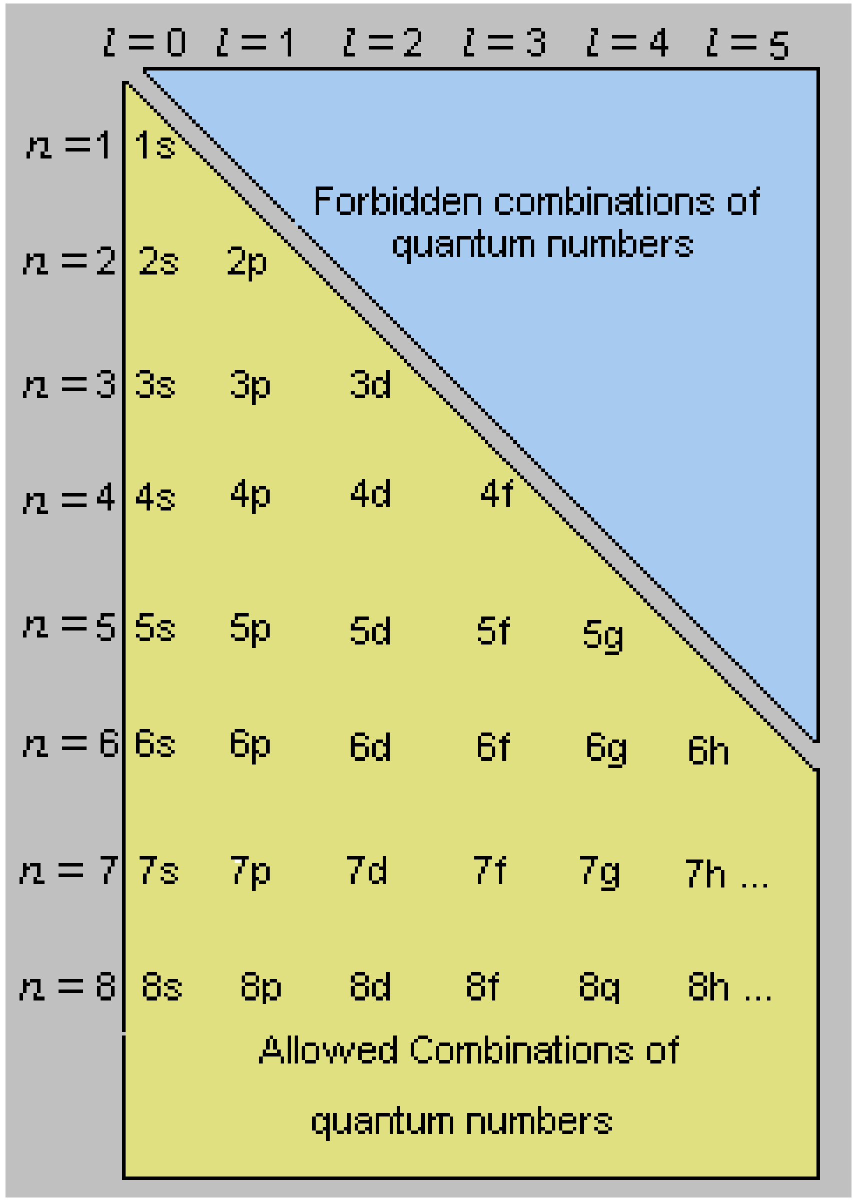
- Sequence of nucleotides: adenine (A), guanine (G), cytosine (C), and thymine (T)
- Two strands, A must attach to T and G must attach to C

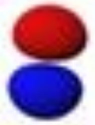
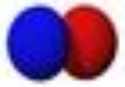
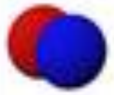
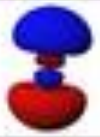
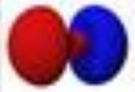
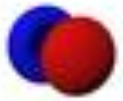
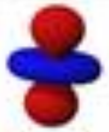
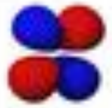
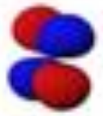
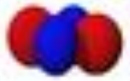
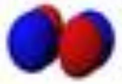
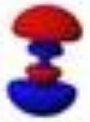
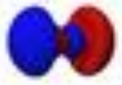
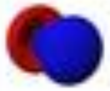
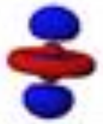
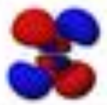
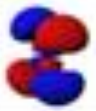
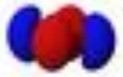
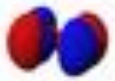
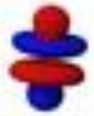
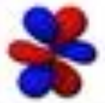
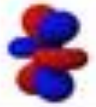
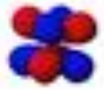
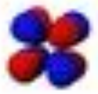
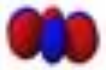
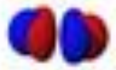
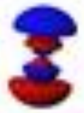
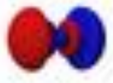
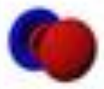
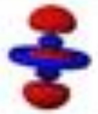
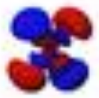
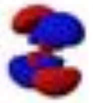
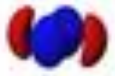
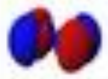
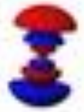
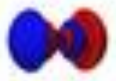
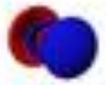


# Quantum Numbers



| Quantum Number | Sym bol | Values | Meaning |
|---|---|---|---|
| principal | n | 1,2,.... | Determine size and energy level. For H, energy depend only on n. (Shell) |
| angular mom. | l | 0,1,... ,n -1 | Determine 3D shape of orbitals: 0 =s, 1=p, 2=d, 3=f (Subshell l = 0,1,2,3,.) |
| magnetic | $m_l$ | -l, l-1, ... -l | Spatial orientation of orbitals. |
| spin magnetic | $m_s$ | +1/2; -1/2 | spin state of electron |

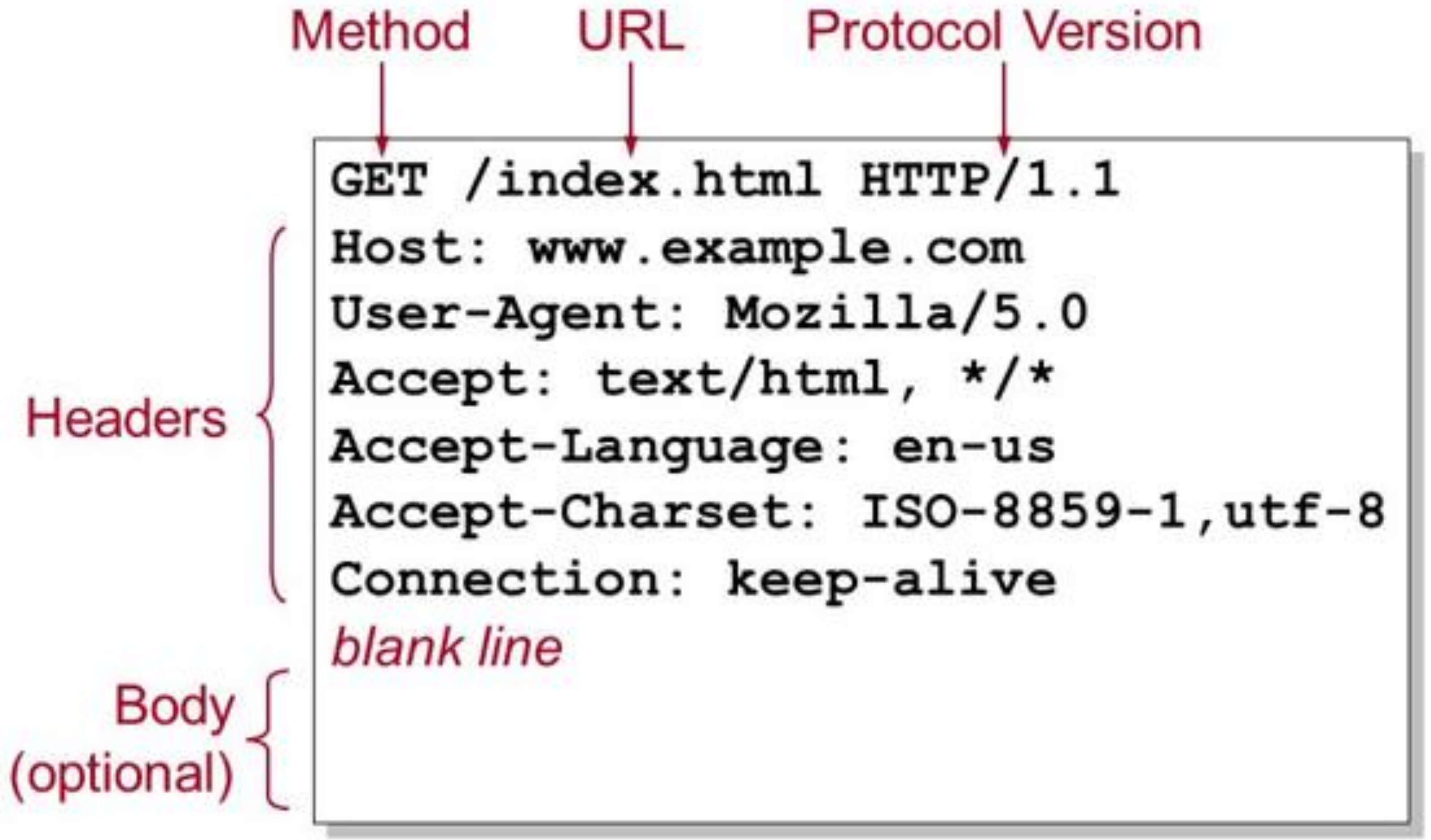

l = 0 l = 1 l = 2 l = 3 l = 4 l = 5
n = 1 1s
n = 2 2s 2p
n = 3 3s 3p 3d
n = 4 4s 4p 4d 4f
n = 5 5s 5p 5d 5f 5g
n = 6 6s 6p 6d 6f 6g 6h
n = 7 7s 7p 7d 7f 7g 7h ...
n = 8 8s 8p 8d 8f 8g 8h ...
Forbidden combinations of quantum numbers
Allowed Combinations of quantum numbers

| | s (ℓ = 0) | p (ℓ = 1) | | | d (ℓ = 2) | | | | | f (ℓ = 3) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $m = 0$ | $m = 0$ | $m = \pm1$ | | $m = 0$ | $m = \pm1$ | | $m = \pm2$ | | $m = 0$ | $m = \pm1$ | | $m = \pm2$ | | $m = \pm3$ | |
| | s | $p_z$ | $p_x$ | $p_y$ | $d_{z^2}$ | $d_{xz}$ | $d_{yz}$ | $d_{xy}$ | $d_{x^2-y^2}$ | $f_{z^3}$ | $f_{xz^2}$ | $f_{yz^2}$ | $f_{xyz}$ | $f_{z(x^2-y^2)}$ | $f_{x(x^2-3y^2)}$ | $f_{y(3x^2-y^2)}$ |
| $n = 1$ | | | | | | | | | | | | | | | | |
| $n = 2$ | |  |  |  | | | | | | | | | | | | |
| $n = 3$ | |  |  |  |  |  |  |  |  | | | | | | | |
| $n = 4$ | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| $n = 5$ | |  |  |  |  |  |  |  |  | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| $n = 6$ | |  |  |  | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| $n = 7$ | | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |

## HTML Character Set

- For HTML5, the default character encoding is **UTF-8**.
- This has not always been the case. The character encoding for the early web was **ASCII**. In Windows: **Windows-1252**
- Later, from HTML 2.0 to HTML 4.01, **ISO-8859-1** was considered the standard.
- With XML and HTML5, UTF-8 finally arrived and solved a lot of character encoding problems.

## HTTP Request

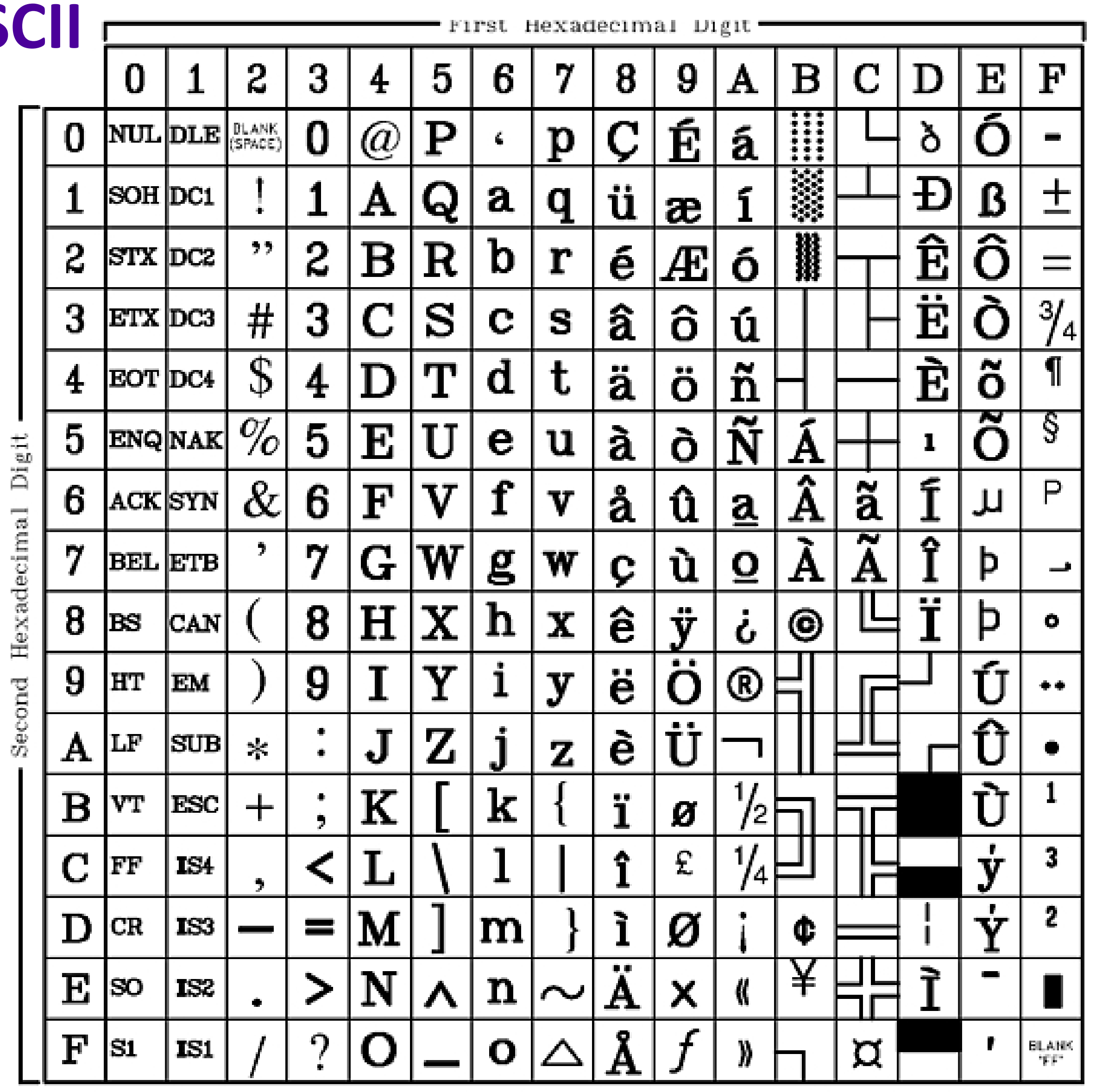


```
GET /index.html HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html, */*
Accept-Language: en-us
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

## HTML Charsets

- HTML ASCII
- HTML WIN-1252
- HTML ISO-8859
- HTML Symbols
- HTML UTF-8

# ASCII



First Hexadecimal Digit

Second Hexadecimal Digit

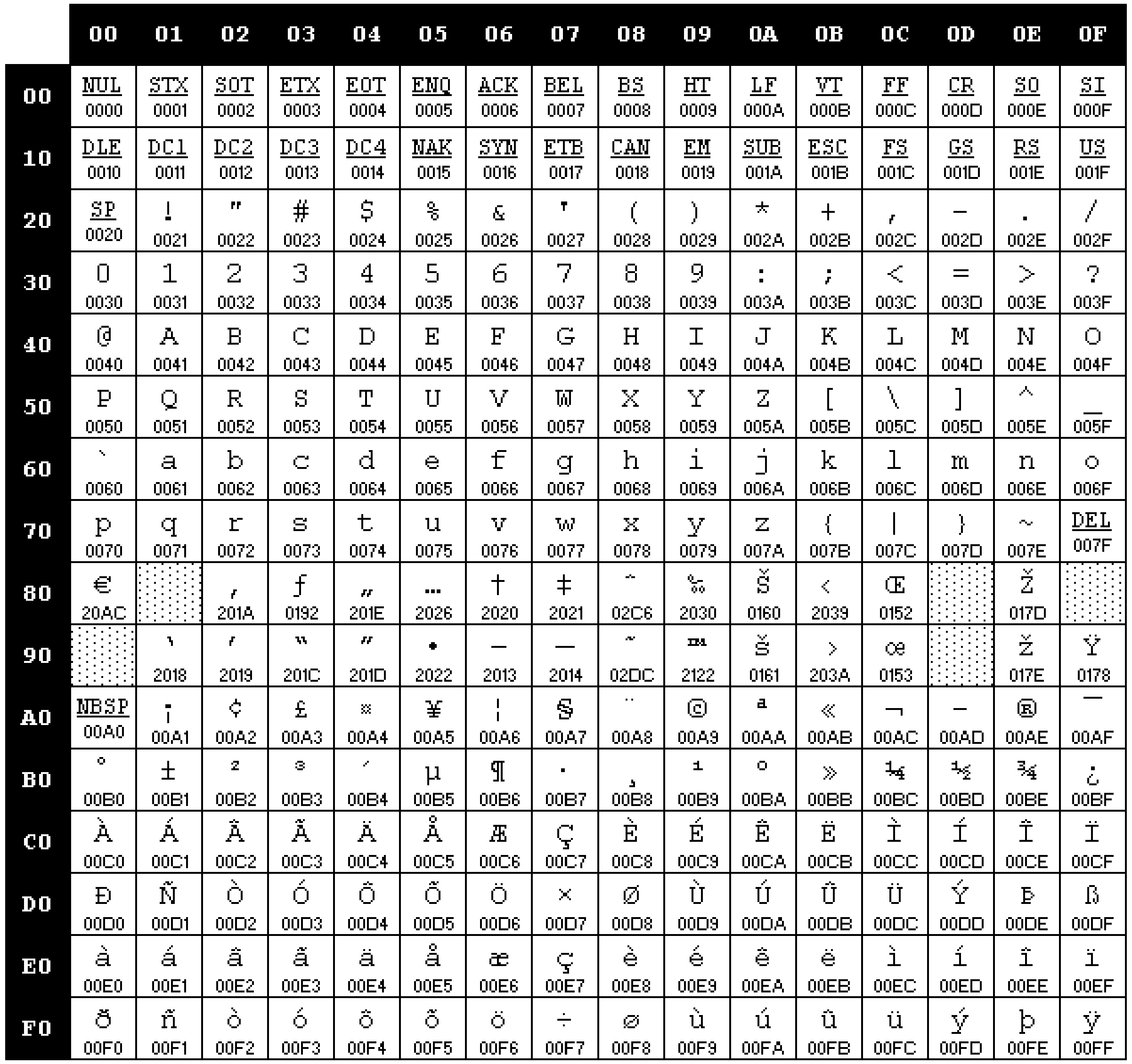
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | DLE | BLANK (SPACE) | 0 | @ | P | ‘ | p | Ç | É | á | ░ | └ | ð | Ó | - |
| 1 | SOH | DC1 | ! | 1 | A | Q | a | q | ü | æ | í | ▒ | ┴ | Ð | ß | ± |
| 2 | STX | DC2 | ” | 2 | B | R | b | r | é | Æ | ó | ▓ | ┬ | Ê | Ô | = |
| 3 | ETX | DC3 | # | 3 | C | S | c | s | â | ô | ú | │ | ├ | Ë | Ò | ¾ |
| 4 | EOT | DC4 | $ | 4 | D | T | d | t | ä | ö | ñ | ┤ | ─ | È | õ | ¶ |
| 5 | ENQ | NAK | % | 5 | E | U | e | u | à | ò | Ñ | Á | ┼ | ı | Õ | § |
| 6 | ACK | SYN | & | 6 | F | V | f | v | å | û | ª | Â | ã | Í | µ | ÷ |
| 7 | BEL | ETB | ’ | 7 | G | W | g | w | ç | ù | º | À | Ã | Î | þ | ¸ |
| 8 | BS | CAN | ( | 8 | H | X | h | x | ê | ÿ | ¿ | © | ╚ | Ï | Þ | ° |
| 9 | HT | EM | ) | 9 | I | Y | i | y | ë | Ö | ® | ╣ | ╔ | ┘ | Ú | ¨ |
| A | LF | SUB | * | : | J | Z | j | z | è | Ü | ¬ | ║ | ╩ | ┌ | Û | · |
| B | VT | ESC | + | ; | K | [ | k | { | ï | ø | ½ | ╗ | ╦ | █ | Ù | ¹ |
| C | FF | IS4 | , | < | L | \ | l | \| | î | £ | ¼ | ╝ | ╠ | ▄ | ý | ³ |
| D | CR | IS3 | - | = | M | ] | m | } | ì | Ø | ¡ | ¢ | ═ | ¦ | Ý | ² |
| E | SO | IS2 | . | > | N | ^ | n | ~ | Ä | × | « | ¥ | ╬ | Ì | ¯ | ■ |
| F | SI | IS1 | / | ? | O | _ | o | △ | Å | ƒ | » | ┐ | ¤ | ▀ | ´ | BLANK 'FF' |

Code Set IBM-850

# More character sets

- US-ASCII (basic English)
- ISO-8859-1 (Western Europe)
- ISO-8859-2 (Central Europe)
- ISO-8859-3 (Southern Europe)
- ISO-8859-4 (Baltic)
- ISO-8859-5 (Cyrillic)
- ISO-8859-6 (Arabic)
- ISO-8859-7 (Greek)
- ISO-8859-8 (Hebrew)
- ISO-8859-9 (Turkish)
- ISO-8859-15 (Latin 9)
- SHIFT_JIS (Japanese, Win/Mac)
- Windows-1250 (legacy, Central Europe)
- Windows-1251 (legacy, Cyrillic)
- Windows-1252 (legacy, Western Europe)
- Windows-1253 (legacy, Greek)
- Windows-1254 (legacy, Turkish)
- Windows-1255 (legacy, Hebrew)
- Windows-1256 (legacy, Arabic)
- Windows-1257 (legacy, Baltic Rim)
- Windows-1258 (legacy, Vietnam)

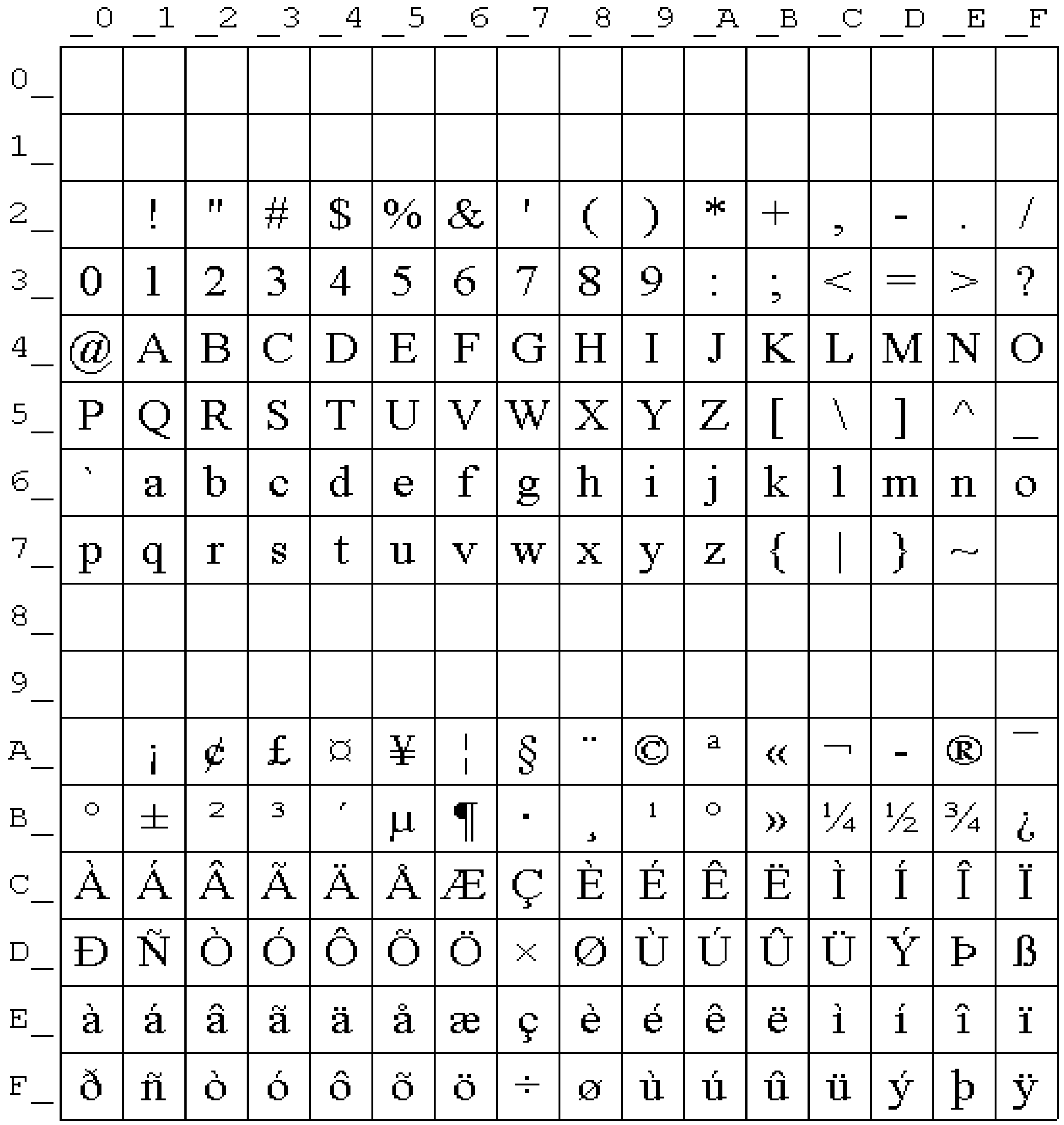
# Windows-1252



| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | NUL 0000 | STX 0001 | SOT 0002 | ETX 0003 | EOT 0004 | ENQ 0005 | ACK 0006 | BEL 0007 | BS 0008 | HT 0009 | LF 000A | VT 000B | FF 000C | CR 000D | SO 000E | SI 000F |
| 10 | DLE 0010 | DC1 0011 | DC2 0012 | DC3 0013 | DC4 0014 | NAK 0015 | SYN 0016 | ETB 0017 | CAN 0018 | EM 0019 | SUB 001A | ESC 001B | FS 001C | GS 001D | RS 001E | US 001F |
| 20 | SP 0020 | ! 0021 | " 0022 | # 0023 | $ 0024 | % 0025 | & 0026 | ' 0027 | ( 0028 | ) 0029 | * 002A | + 002B | , 002C | - 002D | . 002E | / 002F |
| 30 | 0 0030 | 1 0031 | 2 0032 | 3 0033 | 4 0034 | 5 0035 | 6 0036 | 7 0037 | 8 0038 | 9 0039 | : 003A | ; 003B | < 003C | = 003D | > 003E | ? 003F |
| 40 | @ 0040 | A 0041 | B 0042 | C 0043 | D 0044 | E 0045 | F 0046 | G 0047 | H 0048 | I 0049 | J 004A | K 004B | L 004C | M 004D | N 004E | O 004F |
| 50 | P 0050 | Q 0051 | R 0052 | S 0053 | T 0054 | U 0055 | V 0056 | W 0057 | X 0058 | Y 0059 | Z 005A | [ 005B | \ 005C | ] 005D | ^ 005E | _ 005F |
| 60 | ` 0060 | a 0061 | b 0062 | c 0063 | d 0064 | e 0065 | f 0066 | g 0067 | h 0068 | i 0069 | j 006A | k 006B | l 006C | m 006D | n 006E | o 006F |
| 70 | p 0070 | q 0071 | r 0072 | s 0073 | t 0074 | u 0075 | v 0076 | w 0077 | x 0078 | y 0079 | z 007A | { 007B | \| 007C | } 007D | ~ 007E | DEL 007F |
| 80 | € 20AC | ▨ | ‚ 201A | ƒ 0192 | „ 201E | … 2026 | † 2020 | ‡ 2021 | ˆ 02C6 | ‰ 2030 | Š 0160 | ‹ 2039 | Œ 0152 | ▨ | Ž 017D | ▨ |
| 90 | ▨ | ‘ 2018 | ’ 2019 | “ 201C | ” 201D | • 2022 | – 2013 | — 2014 | ˜ 02DC | ™ 2122 | š 0161 | › 203A | œ 0153 | ▨ | ž 017E | Ÿ 0178 |
| A0 | NBSP 00A0 | ¡ 00A1 | ¢ 00A2 | £ 00A3 | ¤ 00A4 | ¥ 00A5 | ¦ 00A6 | § 00A7 | ¨ 00A8 | © 00A9 | ª 00AA | « 00AB | ¬ 00AC | - 00AD | ® 00AE | ¯ 00AF |
| B0 | ° 00B0 | ± 00B1 | ² 00B2 | ³ 00B3 | ´ 00B4 | µ 00B5 | ¶ 00B6 | · 00B7 | ¸ 00B8 | ¹ 00B9 | º 00BA | » 00BB | ¼ 00BC | ½ 00BD | ¾ 00BE | ¿ 00BF |
| C0 | À 00C0 | Á 00C1 | Â 00C2 | Ã 00C3 | Ä 00C4 | Å 00C5 | Æ 00C6 | Ç 00C7 | È 00C8 | É 00C9 | Ê 00CA | Ë 00CB | Ì 00CC | Í 00CD | Î 00CE | Ï 00CF |
| D0 | Ð 00D0 | Ñ 00D1 | Ò 00D2 | Ó 00D3 | Ô 00D4 | Õ 00D5 | Ö 00D6 | × 00D7 | Ø 00D8 | Ù 00D9 | Ú 00DA | Û 00DB | Ü 00DC | Ý 00DD | Þ 00DE | ß 00DF |
| E0 | à 00E0 | á 00E1 | â 00E2 | ã 00E3 | ä 00E4 | å 00E5 | æ 00E6 | ç 00E7 | è 00E8 | é 00E9 | ê 00EA | ë 00EB | ì 00EC | í 00ED | î 00EE | ï 00EF |
| F0 | ð 00F0 | ñ 00F1 | ò 00F2 | ó 00F3 | ô 00F4 | õ 00F5 | ö 00F6 | ÷ 00F7 | ø 00F8 | ù 00F9 | ú 00FA | û 00FB | ü 00FC | ý 00FD | þ 00FE | ÿ 00FF |

## ISO-8859-1

- ISO-8859-1 was the default character in HTML 4.01.
- ISO (The International Standards Organization) defines the standard character sets for different alphabets/languages.
- The different variants of ISO-8859 are listed at the bottom of this page.



| | _0 | _1 | _2 | _3 | _4 | _5 | _6 | _7 | _8 | _9 | _A | _B | _C | _D | _E | _F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0_ | | | | | | | | | | | | | | | | |
| 1_ | | | | | | | | | | | | | | | | |
| 2_ | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3_ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4_ | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5_ | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6_ | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7_ | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | |
| 8_ | | | | | | | | | | | | | | | | |
| 9_ | | | | | | | | | | | | | | | | |
| A_ | | ¡ | ¢ | £ | ¤ | ¥ | ¦ | § | ¨ | © | ª | « | ¬ | - | ® | ¯ |
| B_ | ° | ± | ² | ³ | ´ | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| C_ | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| D_ | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| E_ | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| F_ | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

Learning Channel

ISO-8859-1

## UTF-8

| Bits of code point | First code point | Last code point | Bytes in sequence | Byte 1 | Byte 2 | Byte 3 | Byte 4 | Byte 5 | Byte 6 |
|---|---|---|---|---|---|---|---|---|---|
| **7** | U+0000 | U+007F | 1 | `0xxxxxxx` | | | | | |
| **11** | U+0080 | U+07FF | 2 | `110xxxxx` | `10xxxxxx` | | | | |
| **16** | U+0800 | U+FFFF | 3 | `1110xxxx` | `10xxxxxx` | `10xxxxxx` | | | |
| **21** | U+10000 | U+1FFFFF | 4 | `11110xxx` | `10xxxxxx` | `10xxxxxx` | `10xxxxxx` | | |
| The following sequences are not part of the UTF-8 standard, only part of the original proposal | | | | | | | | | |
| **26** | U+200000 | U+3FFFFFF | 5 | `111110xx` | `10xxxxxx` | `10xxxxxx` | `10xxxxxx` | `10xxxxxx` | |
| **31** | U+4000000 | U+7FFFFFFF | 6 | `1111110x` | `10xxxxxx` | `10xxxxxx` | `10xxxxxx` | `10xxxxxx` | `10xxxxxx` |

|  | a | é | 🐱 |
|---|---|---|---|
| ANSI | 0x61 | 0xE9 | - |
| UTF-8 | 0x61 | 0xC3 0xA9 | 0xF0 0x9F 0x90 0xB1 |
| UTF-16 (big) | 0x00 0x61 | 0x00 0xE9 | 0xD8 0x3D 0xDC 0x31 |
| UTF-16 (little) | 0x61 0x00 | 0xE9 0x00 | 0x3D 0xD8 0x31 0xDC |
| UTF-32 (big) | 0x00 0x00 0x00 0x61 | 0x00 0x00 0x00 0xE9 | 0xD8 0x3D 0xDC 0x31 |
| UTF-32 (little) | 0x61 0x00 0x00 0x00 | 0xE9 0x00 0x00 0x00 | 0x31 0xDC 0x3D 0xD8 |

## In HTML5: Unicode UTF-8

### Example

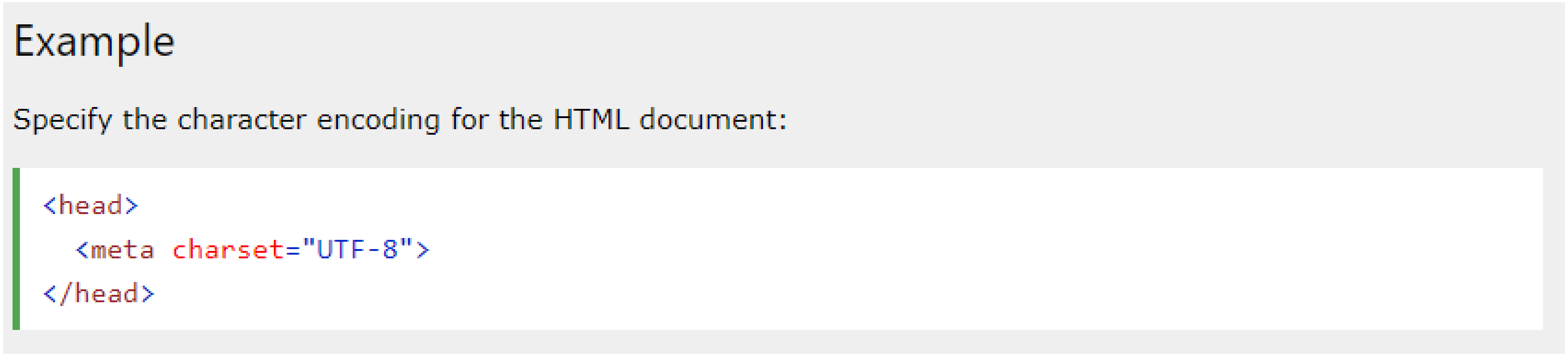
```
<meta http-equiv="Content-Type"
content="text/html;charset=ISO-8859-8">
```

All HTML 4 processors also support UTF-8:

### Example

```
<meta http-equiv="Content-Type"
content="text/html;charset=UTF-8">
```

## HTML <meta> charset Attribute



### Example

Specify the character encoding for the HTML document:

```
<head>
  <meta charset="UTF-8">
</head>
```

## <br>

- **Line breaker tag:** <br>
- **Space: &nbsp;**
- **Space: &#20;   <!-- ASCII 20 -->**
- **Try:** Alt+Numbers (ASCII)

## Demo Program

- character_set_Chinese.html
- character_set_Chinese2.html

## Practice

Try Code.org Lesson 2 Activities.

# How the Web Works and Web Overview

WEB-DESIGN CHAPTER 2/3

## Sending Binary Messages with the Internet Simulator

LESSON 3

## Overview

- Students use the Internet simulator for the first time in this lesson. It is configured to represent two parties connected on a single shared wire that only hold one of two possible states.
- Students invent a binary call-response protocol that can overcome the coordination, timing and synchronization problems that arise when forced to use such a truly binary system.

## Vocabulary

- Bandwidth - Transmission capacity measure by bit rate
- Bit - A contraction of "Binary Digit"; the single unit of information in a computer, typically represented as a 0 or 1
- Bit rate - (sometimes written bitrate) the number of bits that are conveyed or processed per unit of time. e.g. 8 bits/sec.
- Latency - Time it takes for a bit to travel from its sender to its receiver.
- Protocol - A set of rules governing the exchange or transmission of data between devices.
- ISO-7 layers architecture, turn-around time, delay, latency, wait-time, response time.

# Internet Simulator

UNIT 1 EXTRA LESSON 1/2

## Internet Simulator Activity

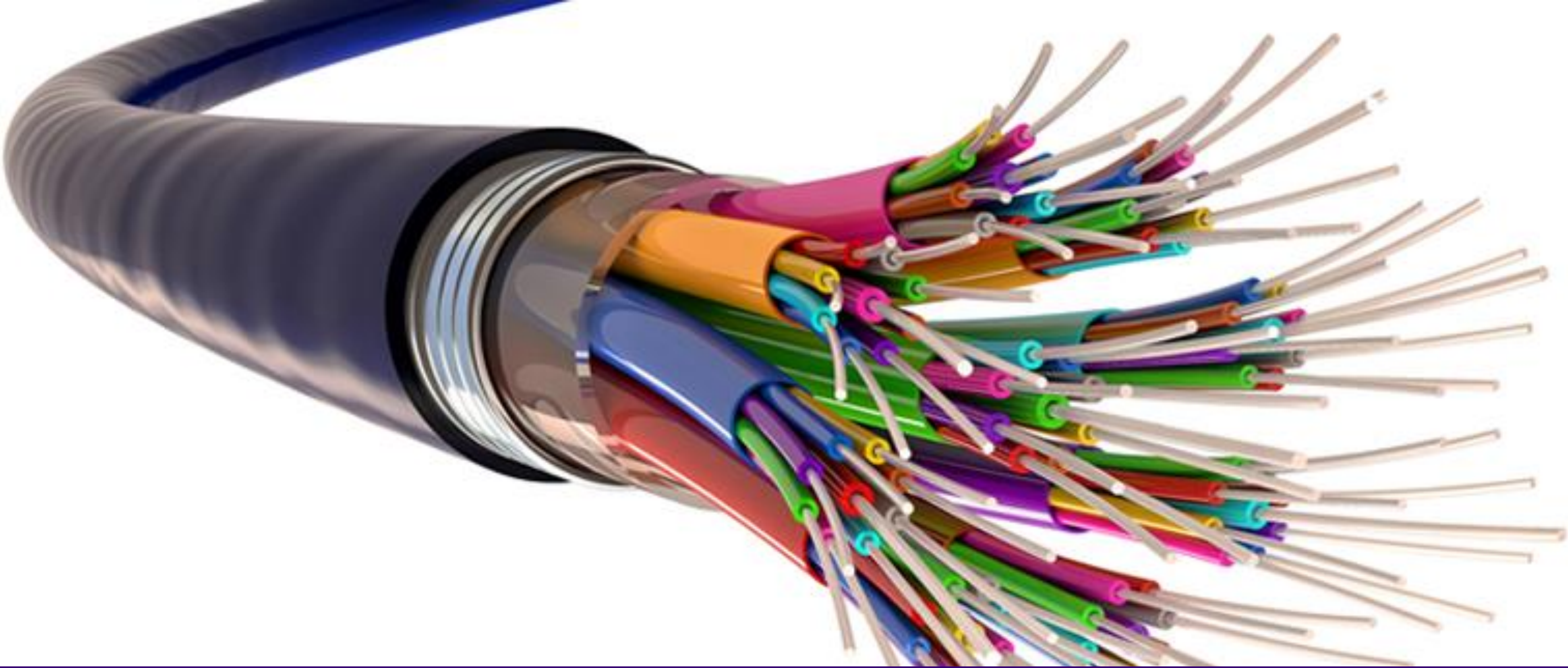
**Go Code.org!**

## Video

**The Internet: Wires, Cables & Wifi**

# Internet

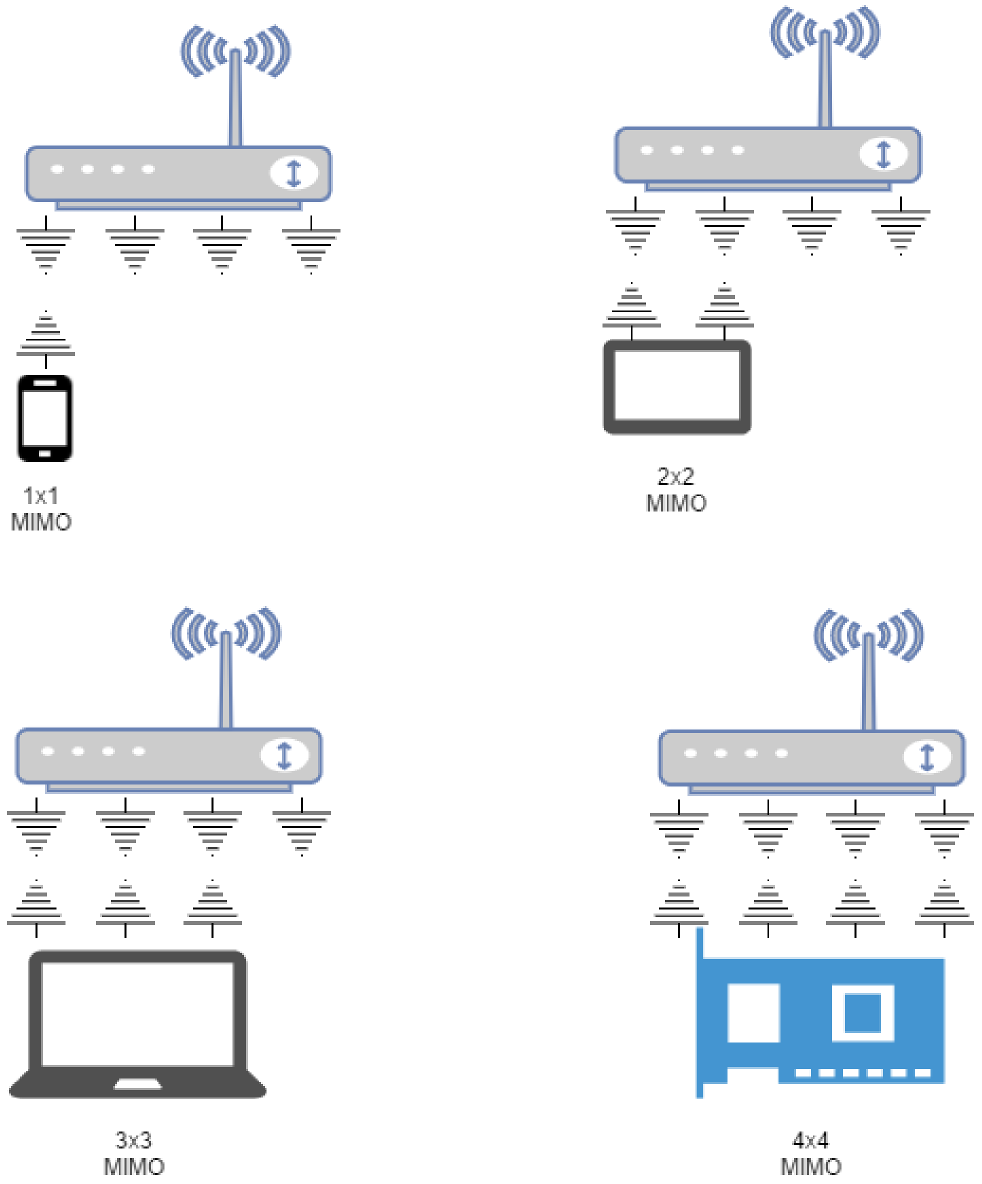
## Wired(LAN)



Internet

## Fiber

## Internet Co-Axial Cable



## Internet
## WIFI

5G

## Practice

Try Code.org Lesson 3 Activities.

# Creating First Page and Basic Document Structure

WEB-DESIGN CHAPTER 4/5

## Practice

- Create a HTML page with bullet points, ordered list.
- Down download a text file from internet:
  - Declaration of independence.
  - I have a dream.
- Then, format it with HTML

# JavaScript

JAVASCRIPT CHAPTER 1

## Sending Bits in the Real World

OPTIONAL LESSON

## Objectives

- In this lesson, students will be introduced to how bits are transmitted over the most common mediums (copper wire, fiber-optic cable, and radio waves) used to connect devices on the Internet.
- They then chose a device that transmits bits and research that device and the system it uses.
- Students create a poster presenting their findings, and the lesson concludes with a gallery walk of the posters.

## Work Sheet

- Worksheet - Video Guide for Wires, Cables & WiFi

## Practice

Activity Guide - Sending Bits in the Real World

# Number Systems

LESSON 4

## Overview

Students will explore the properties of number systems by inventing their own number system using only three shapes: a circle, triangle and a square.

This lesson is a precursor to looking at several other number systems important to computing, especially binary and hexadecimal.

## Video

**Tips & Tricks - Number Systems**

## Practice

Try Code.org Lesson 4 Activities.

## Different number systems used with computers:

- Decimal (base 10) – 1, 2, 3 or 45.6, etc.
- Binary (base 2) – state “a” or state “B”
- hexadecimal (base 16)

## Different number systems used with computers:

- Numbers themselves (quantities) are laws of nature.
- The symbols we use to represent the numbers are arbitrary, man-made abstraction.

# Purpose for today:

## you will invent your own base-3 number system.

**Rules:**

you must have

1. a set of distinct symbols
2. an agreement about how those symbols should be ordered

- How Many ways can you represent “7”?

## How Many ways can you represent "7"?

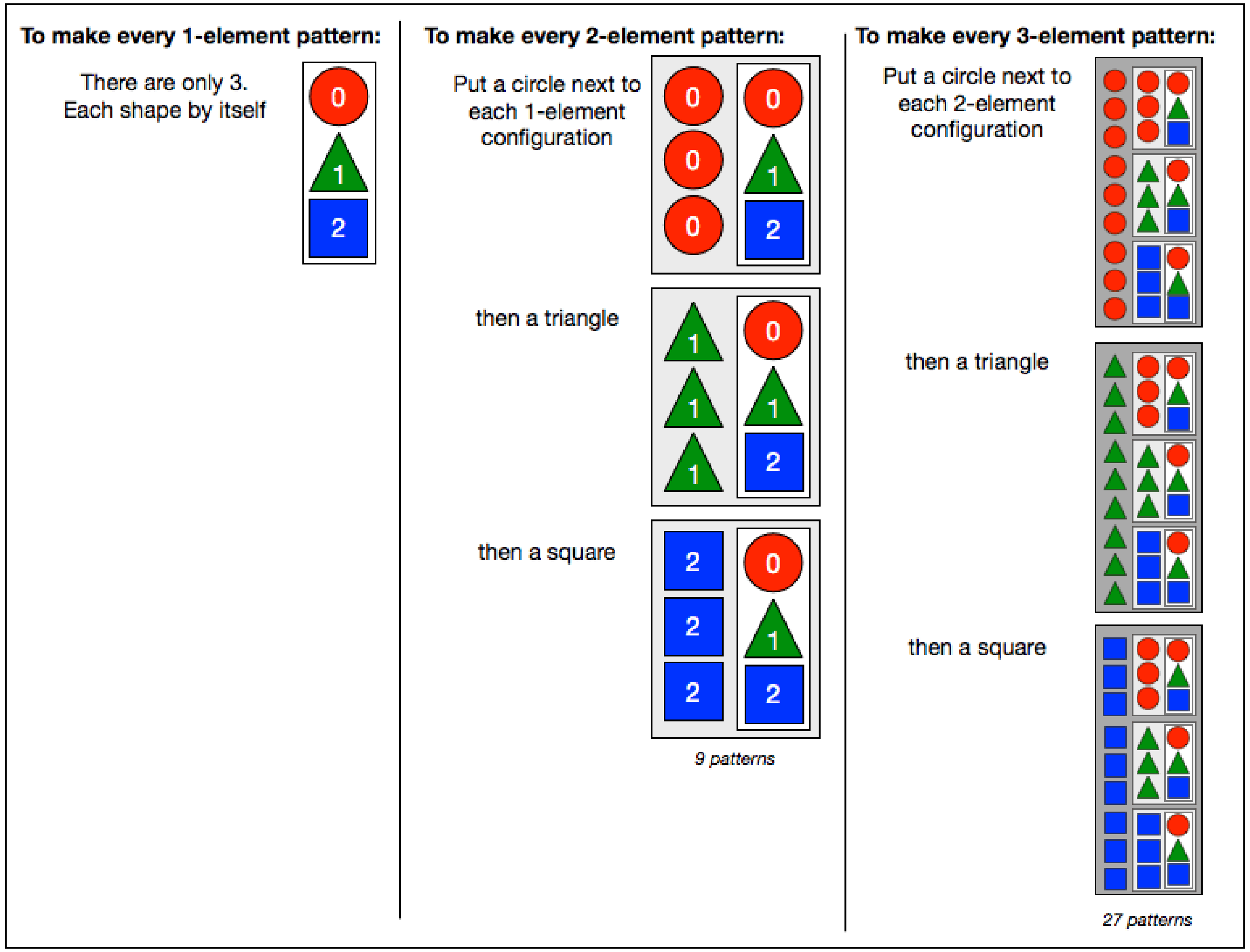
- Use the shapes you cut out.
- Use the activity guide.
- Have one partner move the shapes into patterns, and the other partner write the unique 3-place patterns down.
- Order matters!  So circle, triangle, square is different from triangle, circle, square
- do challenge 1 and challenge 2 – write your rules

## Number System

- YOU JUST MADE A NUMBER SYSTEM!
- Share your number system with another team.
- Can your team follow the other team’s rules to make their number system?
- Were some sets of rules easier to follow than others?
- Do you think there are limits to the number of symbols we could use to represent numbers?

## Number System

- What if we only had 2 symbols, and circle and a square? Could we still make a number system?
- What if we had 10 symbols:  a circle, square, star, etc. Could we still make a number system?


To make every 1-element pattern:
There are only 3. Each shape by itself
0
1
2
To make every 2-element pattern:
Put a circle next to each 1-element configuration
then a triangle
then a square
9 patterns
To make every 3-element pattern:
Put a circle next to each 2-element configuration
then a triangle
then a square
27 patterns

## Did your team have all the possibilities?

# Binary Numbers

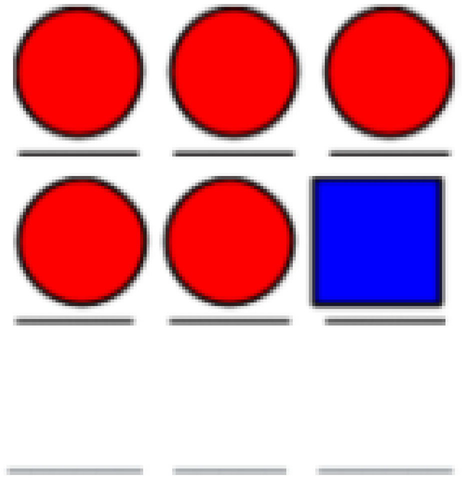
LESSON 5

## How Many ways can you represent "7"?

- Computers use the binary number system to represent numeric values.
- We communicated with a binary system before. We found that develop a number system is a little different.
- A number system is infinite, and also has rules for counting – or how to get from one value to the next.

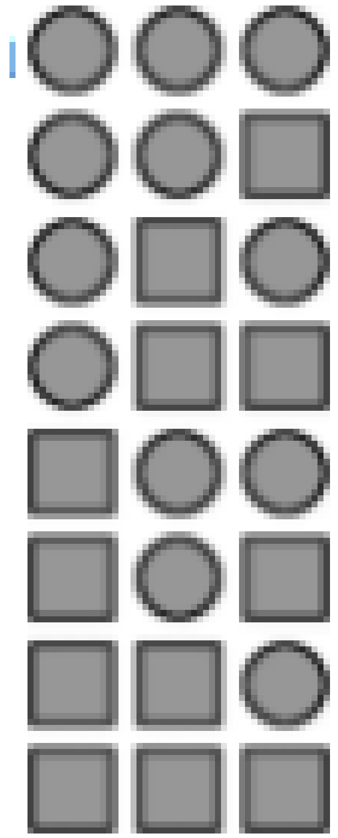
## ND-Space versus Permutation?

- What if you only had a circle and square? With only a circle and square, how many 3-place patterns are there?
- Let's start some for you below, you make the rest. How many are there?



# ND-Space versus Permutation?

- You have 8 possible combinations of 3-place patterns with 2 shapes.



## Number System

- Why might we want to create a number system that includes only two symbols?
- Then we can use bits as the building blocks of our number system. How large of numbers do you think this system can represent? How could we go higher?
- Adding more bits (increasing the size of our permutations) allows us to count as high as we wish.

## Video

**Circle Triangle Square to Binary**

**VIDEO ON PROMETHEAN BOARD** (10 MINUTES)

# Presentation

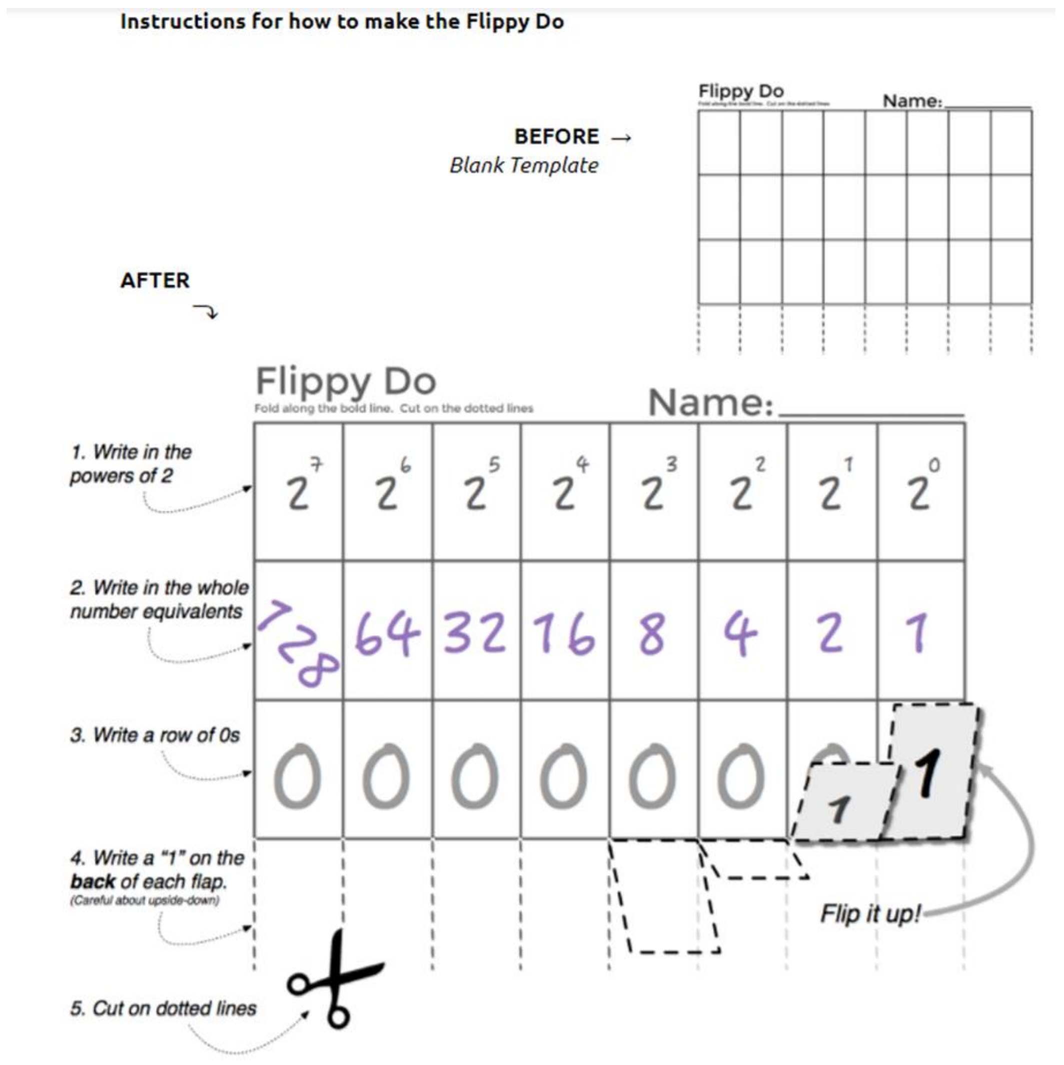
## Ternary System by Shape

U1C1L5e1 PPTX

## Number System

- The number system we have just been introduced to is called the binary number system and can be constructed entirely from bits.
- This is the number system implemented in almost every computer.
- While it may look different from our familiar number system, as we'll see in today's lesson, it can be used in the same way and shares many properties.

# Construct a Flippy Do (10 mins)



Instructions for how to make the Flippy Do

BEFORE → *Blank Template*

Flippy Do Name:

AFTER

Flippy Do Name:

Fold along the bold line. Cut on the dotted lines

1. Write in the powers of 2
2. Write in the whole number equivalents
3. Write a row of 0s
4. Write a "1" on the **back** of each flap. (Careful about upside-down)
5. Cut on dotted lines

|  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Flip it up!

## Binary Odometer

- We’ll keep filling in our Flippy Do’s but you should also know about a helpful widget -- It was used in the opening demonstration -- called the "Binary Odometer"
- Go to the **Binary Odometer** - Code Studio (linked in Code Studio)
- This tool allows you to watch how counting is related across multiple number systems, binary, decimal and other number bases as well.
- Get acquainted with the Binary Odometer.

## Code Studio

https://studio.code.org/s/csp1-2018/stage/5/puzzle/2

## Binary System

- Tasks/Prompts to consider:
  "What's the largest number you can make in binary with the binary odometer?"
  BIN: 1111111111 --> DEC: 1023
- "What happens when the odometer run out of numbers?"
  Overflow! The binary odometer rolls back over to all zeros but the other numbers keep going up.

## Video

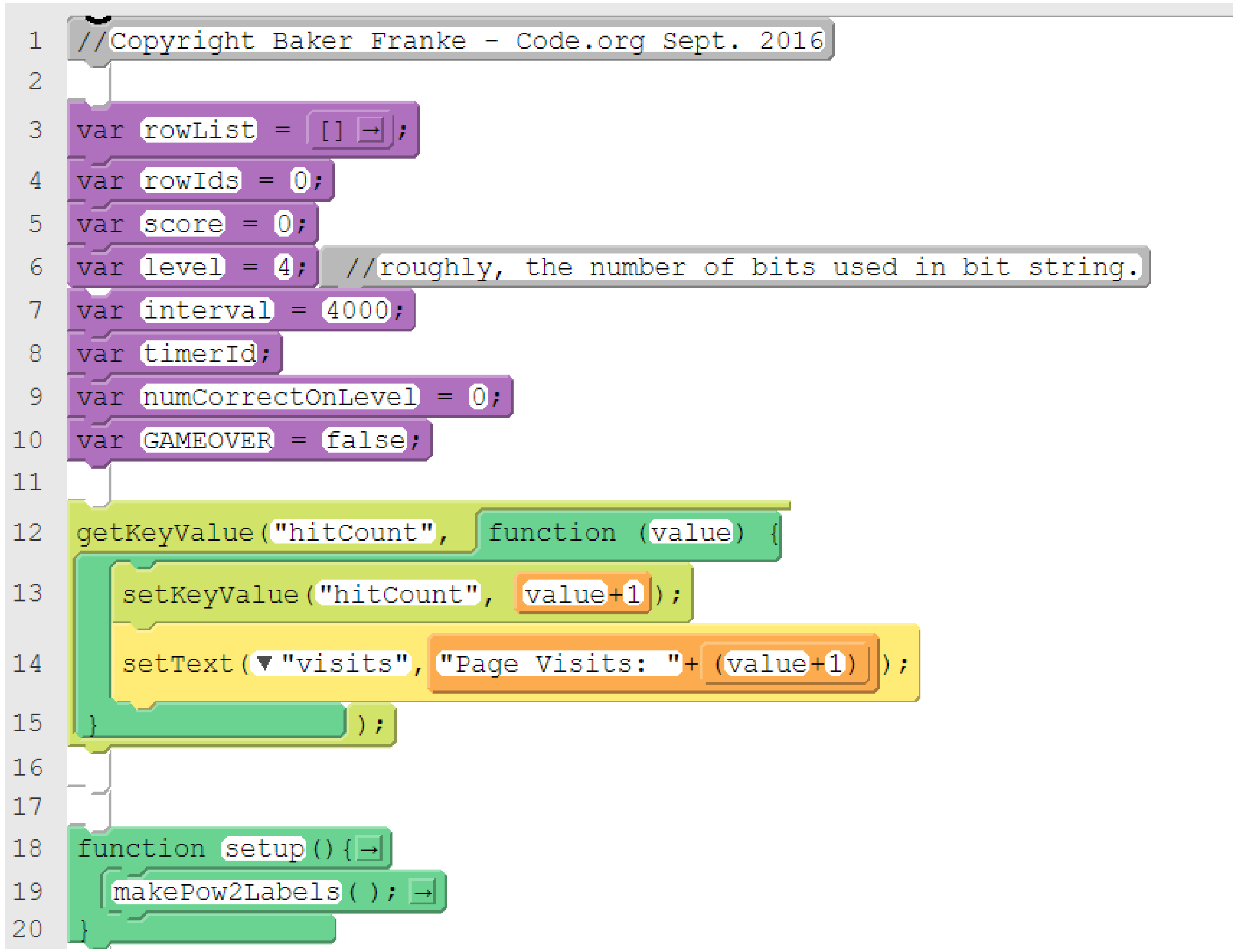
**Binary Numbers**

## Practice

- Complete the Binary Practice Activity Guide (15 mins)
- Complete the Binary Practice - Activity Guide individually (Work with a partner but each should complete your own sheet )
- Use your Flippy Do and Odometer as resources

## AppLab App - App

- Do the Binary Game AppLab App - App.
- Practice translating between bases using the Cisco Binary Game.
- https://studio.code.org/projects/applab/iukLbcDnzqgoxuu810unLw



```
//Copyright Baker Franke - Code.org Sept. 2016

var rowList = [] →;
var rowIds = 0;
var score = 0;
var level = 4;  //roughly, the number of bits used in bit string.
var interval = 4000;
var timerId;
var numCorrectOnLevel = 0;
var GAMEOVER = false;

getKeyValue("hitCount", function (value) {
  setKeyValue("hitCount", value+1);
  setText(▼"visits", "Page Visits: "+ (value+1));
});


function setup(){→
  makePow2Labels(); →
}
```

## Block Programming

https://scratch.mit.edu/help/videos/

## Practice

**Convert:**

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 to binary

## Answer to HW question:

- Binary uses only zeroes and ones so that numbers look like this.
- Base 10 number = Base 2 (binary) number
  **0 = 0**
  **1 = 1**
  **2 = 10 (one 2 and zero 1s)**
  **3 = 11 (one 2 and one 1)**
  **4 = 100 (one 4 and zero 2s and zero 1s)**
  **5 = 101 (one 4 and zero 2s and one 1)**
  **6 = 110 (one 4 and one 2 and zero 1s)**
  **7 = 111**
  **8 = 1000**
  **9 = 1001**
- So there are really only two kinds of people in the statement, those who understand binary and those that don't and two in binary = 10

# Sending Numbers

LESSON 6

## Lesson Overview

- We will be using the Internet Simulator in this lesson.
- You will send a simple line drawing to a classmate (connecting 3 – 7 dots with straight lines).
- WinSCP and SmartFTP

## Lesson Overview

- You must develop a protocol which will allow you to send any image you can make on your grids.
- Pay attention to how many bits are used to represent each binary number.
- This gives you more practice encoding and decoding binary numbers.
- The lesson ends with testing your protocols using a teacher-supplied test image to transmit.

## Objective

- The main purpose of this lesson is to reveal that just because we now know how binary numbers work, there are still challenges of information representation and communication.
- We use the Internet Simulator to enforce some of the constraints imposed by real computers and systems.
- In particular we are interested in developing a more robust protocol for sending a list of numbers over the internet.

## Background Knowledge

- You will need to make choices about how to structure your messages in terms of number of bits to use for each number and even how to communicate the beginning and end of the list.
- While the binary number system can be used to represent any value we wish, in practice the range of values we are able to represent is limited by the number of bits we use.
- Thus, protocols for exchanging binary information must specify in advance how many bits will form a single number among other things.
- Without knowing this information the receivers of a message have no way of determining how to break up an incoming stream of bits into individual numbers; it will just appear to be a random string of 0s and 1s.

## Discussion

- "How many more numbers can be represented with 4 bits as opposed to 3?“
- Twice as many. Every time a bit is added the number of values we can represent doubles
- "What is the highest value I can count to using 3 bits? What about with 4?“
- With 3 bits, 7. With 4 bits, 15. In general the answer is 2^(number of bits) - 1

## Discussion

- "Justify the following claim: Regardless of the number of bits in our binary number system, the first value we represent is 0."
- This is actually the case in any number system that uses place value. The smallest value we can represent is composed of all 0s and is exactly 0.

## Activity

- In computer science, and when talking about transmitting information between computing devices, a protocol is a set of rules that tell us how to encode, communicate and exchange information.
- Today, you and a partner will invent a communication protocol that allows you to send a list of numbers to represent a drawing.

## Video

- **New Internet Simulator**

# Internet Simulator

UNIT 1 EXTRA LESSON 3

## Internet Simulator Activity

**Go Code.org!**

## Practice

### Sending Number for Graphing

- Code.org Activity.
- Get activity guide and graph worksheet.
- What order will the points be sent in?
- How does the recipient know when one number ends and the next begins?
- Test your protocols with the picture you each drew.
- Do the reflection and assessment in code studio for credit.

## Encoding Numbers in the Real World

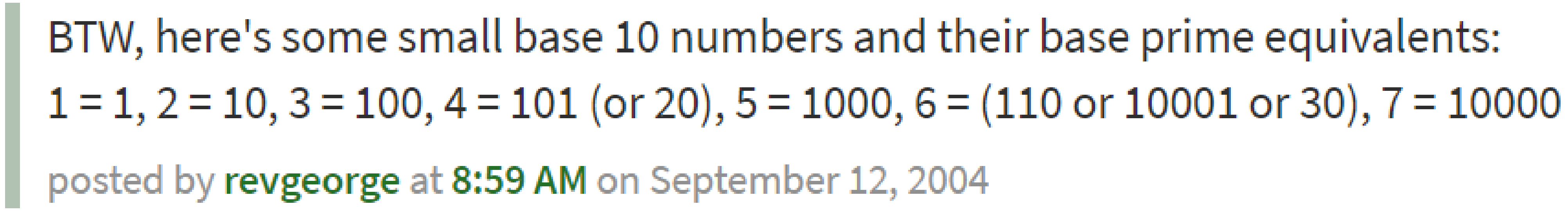
OPTIONAL LESSON

## Objective

- In this lesson, students explore some fascinating stories from the news and history (and the future) about number encodings in computers.
- These stories should serve to illuminate how the kinds of decisions students have been making about number encodings are the same kinds of things that real scientists in the world have to worry about, sometimes with disastrous consequences.
- While this lesson has the possibility of running long, it is meant only as a short excursion into real-world application and should be limited to one class period.

# Base Prime Number System?

https://ask.metafilter.com/10114/Base-prime-number-system-possible



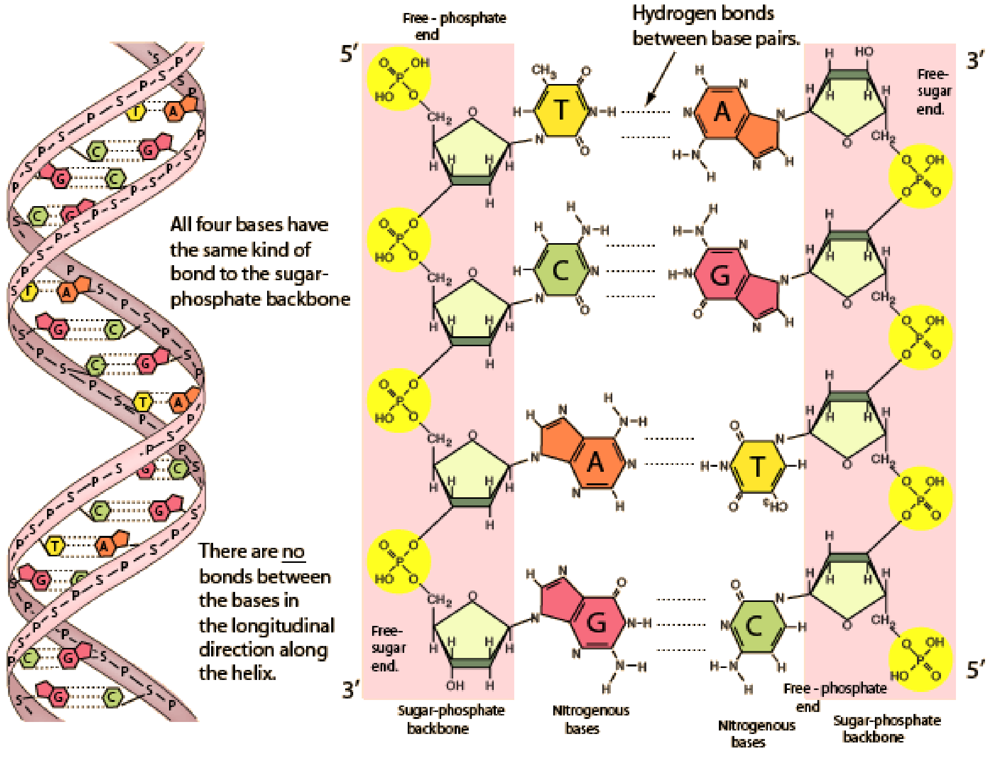
BTW, here's some small base 10 numbers and their base prime equivalents:

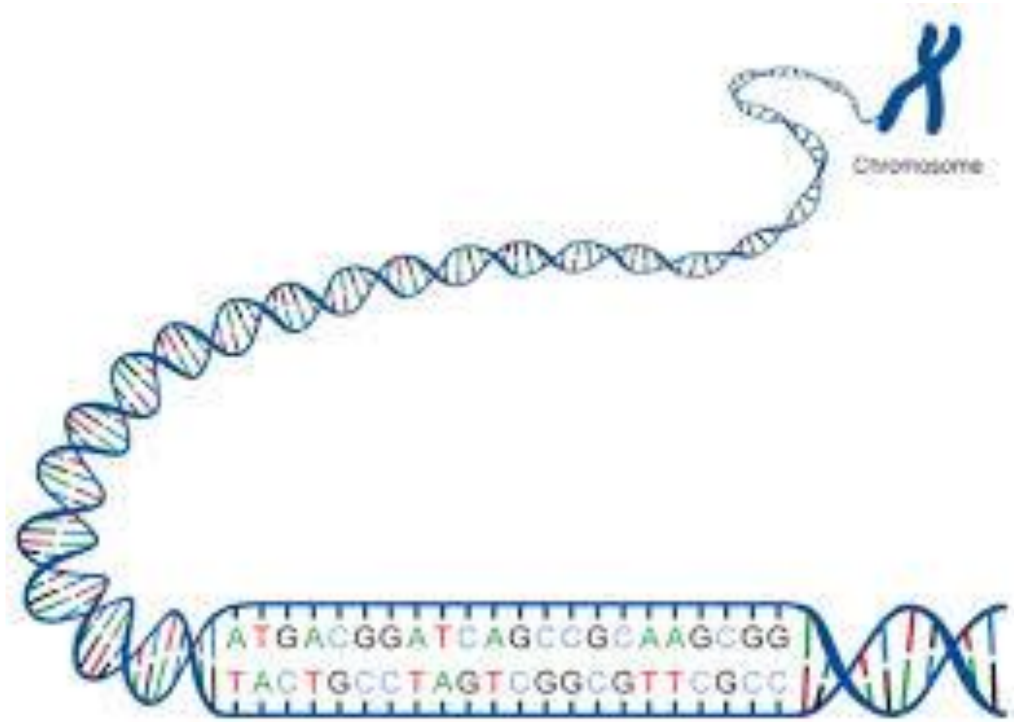
1 = 1, 2 = 10, 3 = 100, 4 = 101 (or 20), 5 = 1000, 6 = (110 or 10001 or 30), 7 = 10000

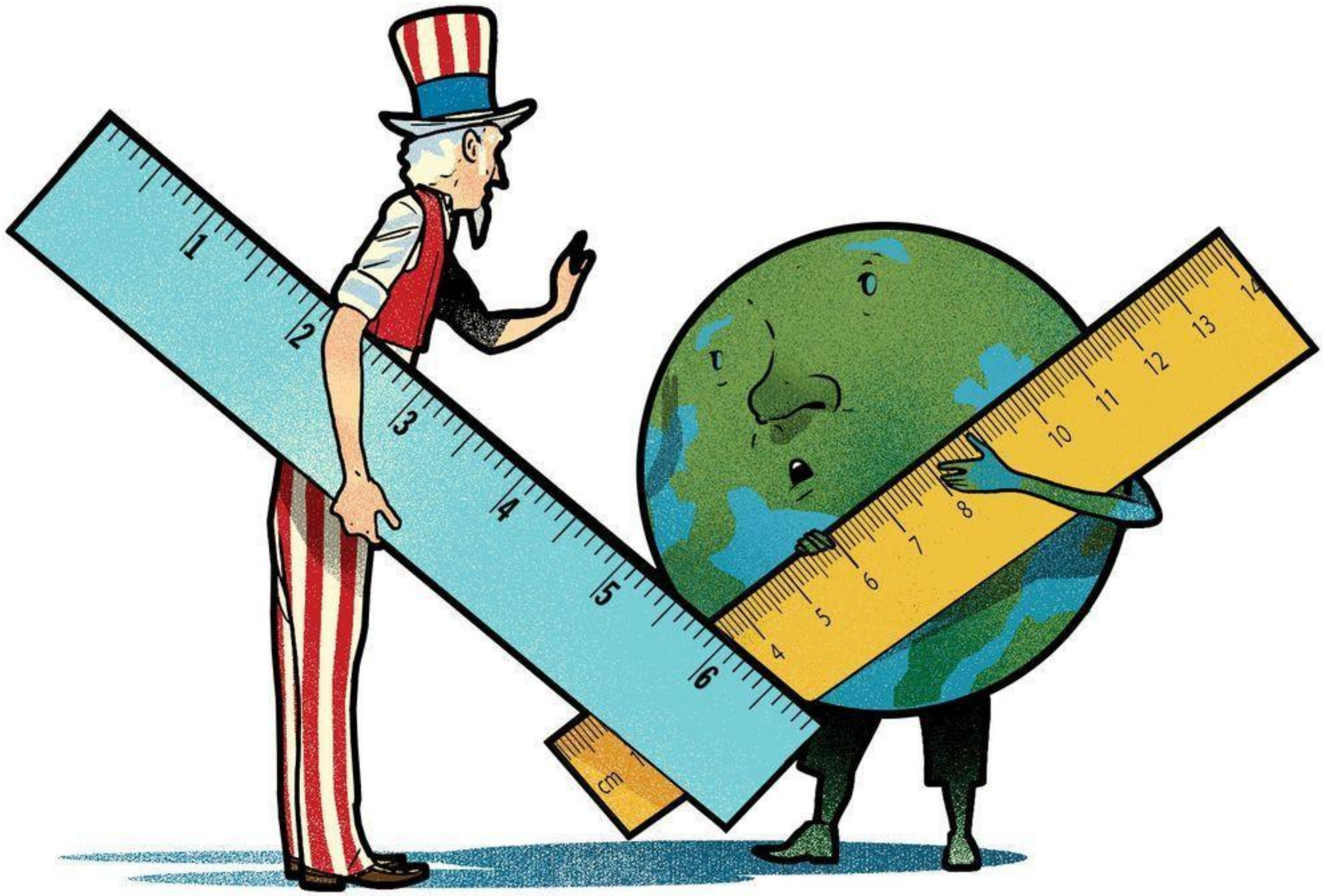
posted by revgeorge at 8:59 AM on September 12, 2004

**Prime numbers in Sumerian base system. Mystery revealed**

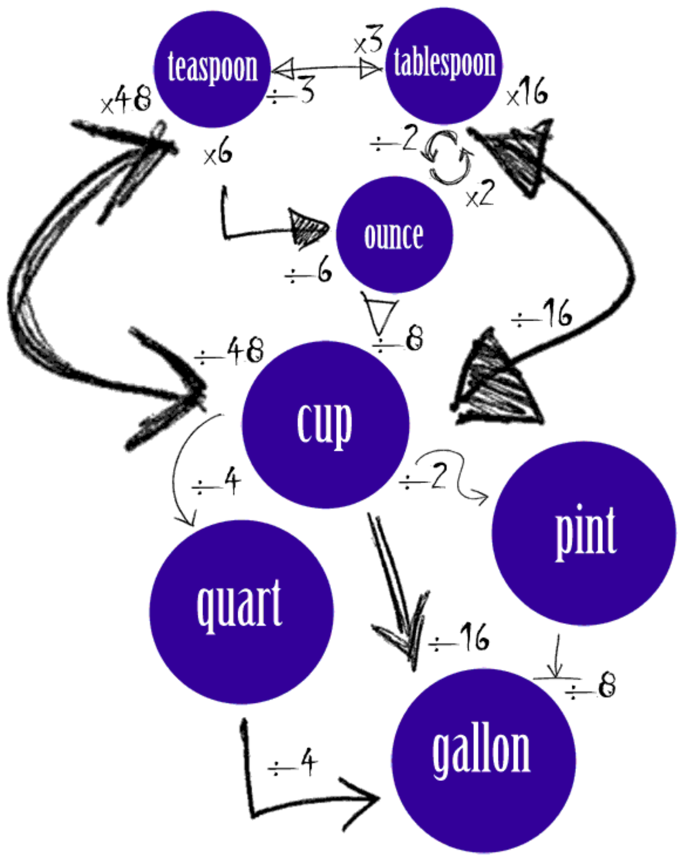
https://youtu.be/0aI_gJbKFds


All four bases have the same kind of bond to the sugar-phosphate backbone
There are no bonds between the bases in the longitudinal direction along the helix.
Free - phosphate end
Hydrogen bonds between base pairs.
5′
3′
Free-sugar end.
Sugar-phosphate backbone
Nitrogenous bases
T
A
C
G
A
T
G
C
Flattened segment of the double helix

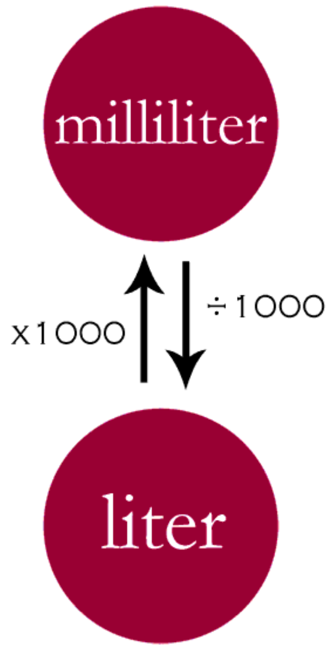


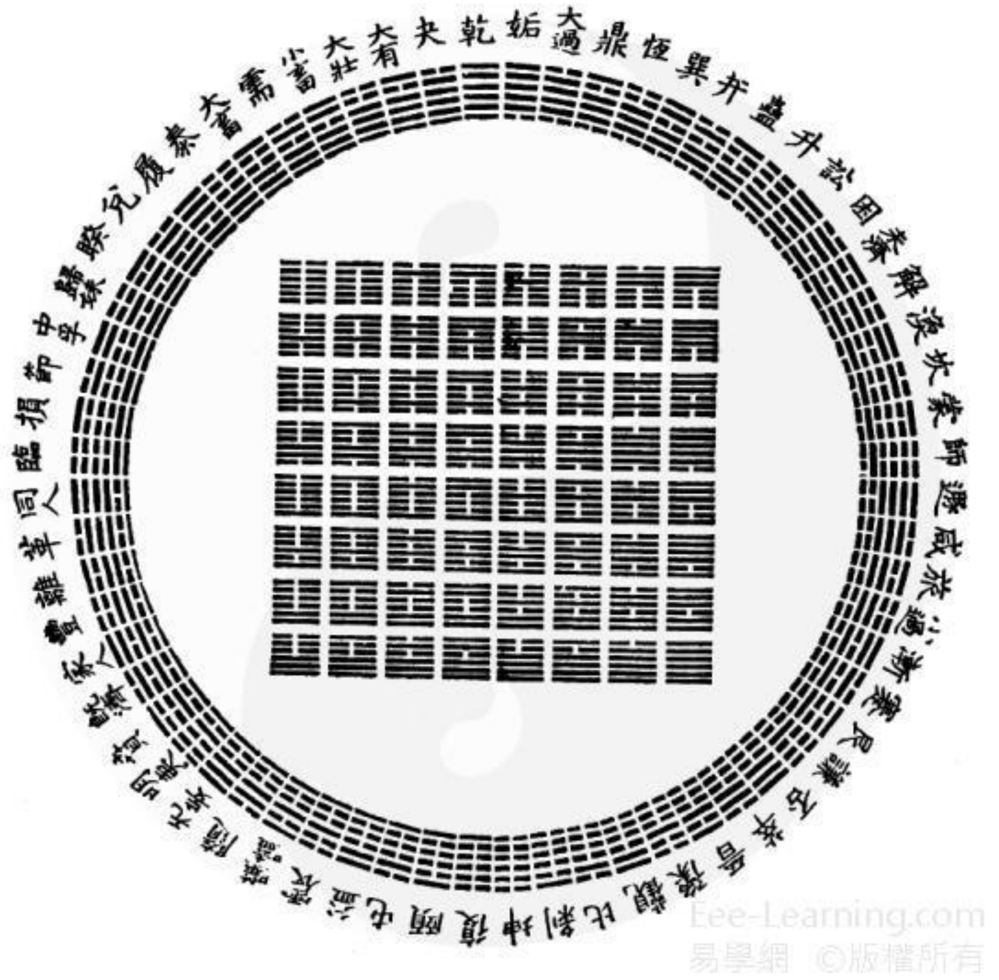

1 2 3 4 5 6
cm 1
4 5 6 7 8 10 11 12 13 14

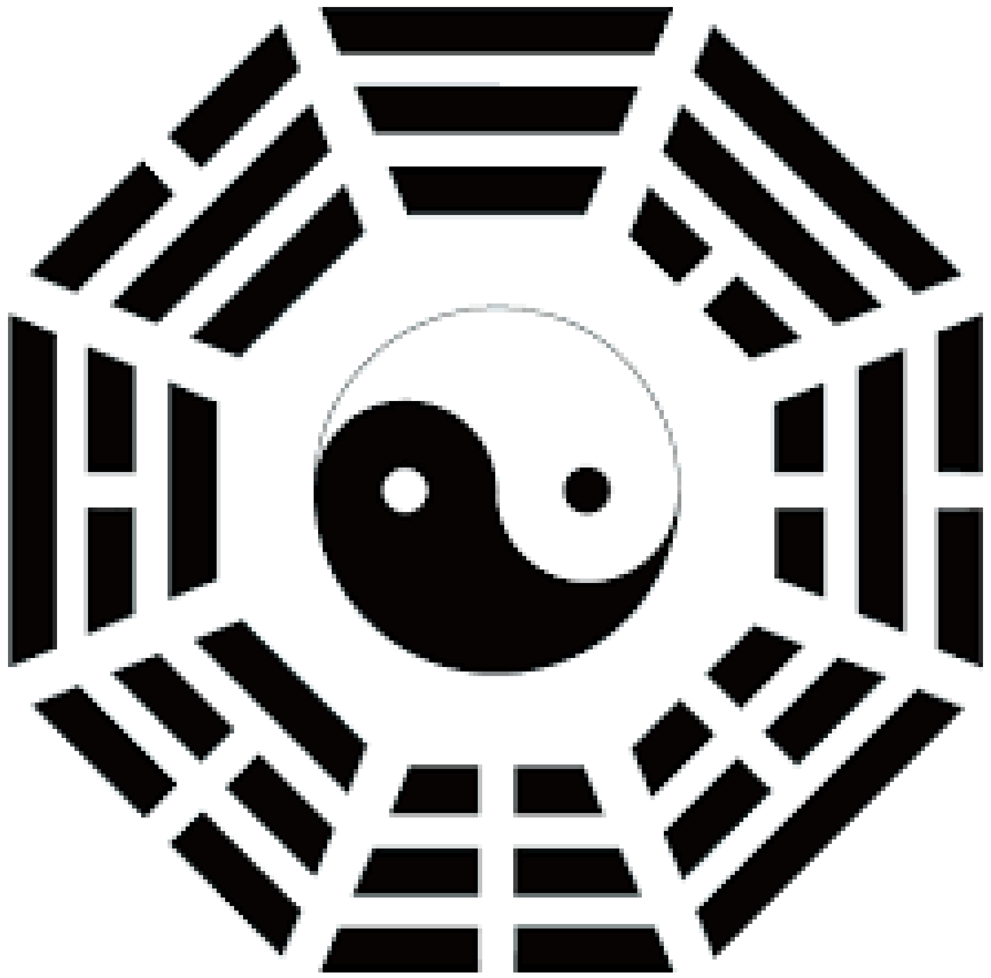
## US Customary Units



## Metric Units




乾
夬
大有
大壯
小畜
需
大畜
泰
履
兌
睽
歸妹
中孚
節
損
臨
同人
革
離
豐
家人
既濟
賁
明夷
無妄
隨
噬嗑
震
益
屯
頤
復
坤
剝
比
觀
豫
晉
萃
否
謙
艮
蹇
漸
小過
旅
咸
遯
師
蒙
坎
渙
解
未濟
困
訟
升
蠱
井
巽
恆
鼎
大過
姤



SW
W
NW
S
SE
E
NE
200
210
220
230
240
250
260
270
280
290
300
310
320
330
340
350
360
10
20
120
130
140
150
160
170
180
190

| Enero | Febrero | Marzo | Abril | Mayo | Junio | Julio | Agosto | Septiembre | Octubre | Noviembre | Diciembre |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 🌗 5 JUEVES - 19:37 - Cetus | 🌗 4 SÁBADO - 04:02 - Aries | 🌗 5 DOMINGO - 11:16 - Taurus | 🌗 3 LUNES - 18:27 - Gemini | 🌗 3 MIÉRCOLES - 02:30 - Cancer | 🌗 1 JUEVES - 12:25 - Leo | 🌗 1 SÁBADO - 00:33 - Virgo | 🌕 7 LUNES - 18:22 - Capricornius | 🌕 6 MIÉRCOLES - 06:46 - Aquarius | 🌕 5 JUEVES - 18:13 - Cetus | 🌕 4 SÁBADO - 05:16 - Aries | 🌕 3 DOMINGO - 16:07 - Taurus |
| 🌕 12 JUEVES - 12:01 - Gemini | 🌕 11 SÁBADO - 00:46 - Leo | 🌕 12 DOMINGO - 14:37 - Virgo | 🌕 11 MARTES - 05:37 - Virgo | 🌕 10 MIÉRCOLES - 21:34 - Libra | 🌕 9 VIERNES - 13:34 - Ophiuchus | 🌕 9 DOMINGO - 04:40 - Sagittarius | 🌓 15 MARTES - 01:30 - Taurus | 🌓 13 MIÉRCOLES - 06:39 - Taurus | 🌓 12 JUEVES - 12:41 - Gemini | 🌓 10 VIERNES - 20:54 - Leo | 🌓 10 DOMINGO - 08:09 - Leo |
| 🌓 19 JUEVES - 22:34 - Virgo | 🌓 18 SÁBADO - 19:54 - Libra | 🌓 20 LUNES - 16:19 - Sagittarius | 🌓 19 MIÉRCOLES - 10:16 - Sagittarius | 🌓 19 VIERNES - 00:51 - Aquarius | 🌓 17 SÁBADO - 11:49 - Pisces | 🌓 16 DOMINGO - 19:49 - Pisces | 🌑 21 LUNES - 18:25 - Leo | 🌑 20 MIÉRCOLES - 05:04 - Virgo | 🌑 19 JUEVES - 18:49 - Virgo | 🌑 18 SÁBADO - 11:52 - Libra | 🌑 18 LUNES - 07:04 - Sagittarius |
| 🌑 28 SÁBADO - 00:31 - Capricornius | 🌑 26 DOMINGO - 14:55 - Aquarius | 🌑 28 MARTES - 02:34 - Cetus | 🌑 26 MIÉRCOLES - 11:58 - Cetus | 🌑 25 JUEVES - 19:52 - Taurus | 🌑 24 SÁBADO - 02:55 - Orión | 🌑 23 DOMINGO - 10:07 - Cancer | 🌗 29 MARTES - 07:53 - Scorpius | 🌗 28 JUEVES - 02:33 - Sagittarius | 🌗 27 VIERNES - 22:02 - Capricornius | 🌗 26 DOMINGO - 16:43 - Aquarius | 🌗 26 MARTES - 09:02 - Pisces |
|  |  |  |  |  |  | 🌗 30 DOMINGO - 15:04 - Virgo |  |  |  |  |  |

# JavaScript Data Types

JAVASCRIPT CHAPTER 2

# Adding Links, Image Files

HTML CHAPTER 6-7

# Sending Text

LESSON 7

## Vocabulary

- **HTML** - HyperText Markup Language: the language of the web, code for web pages
- **HTTP** - HyperText Transfer Protocol: designed for communication on the web, for sending and receiving information
- **ASCII** – American Standard Code for Information Interchange – raw text that any computer can recognize
- **Abstraction** - a simplified representation of something more complex. Abstractions allow you to hide details to help you manage complexity, focus on relevant concepts, and reason about problems at a higher level.
- **Protocol** - A set of rules governing the exchange or transmission of data between devices.

## Objectives

- You will how coding language works to encode text. (Encryption and Decryption)
- The OSI 7 layers architecture. Layers upon layers of encoding all trace back to binary and work together to encode complex information.
- Parcel, Data Format, Layers.

## Objectives

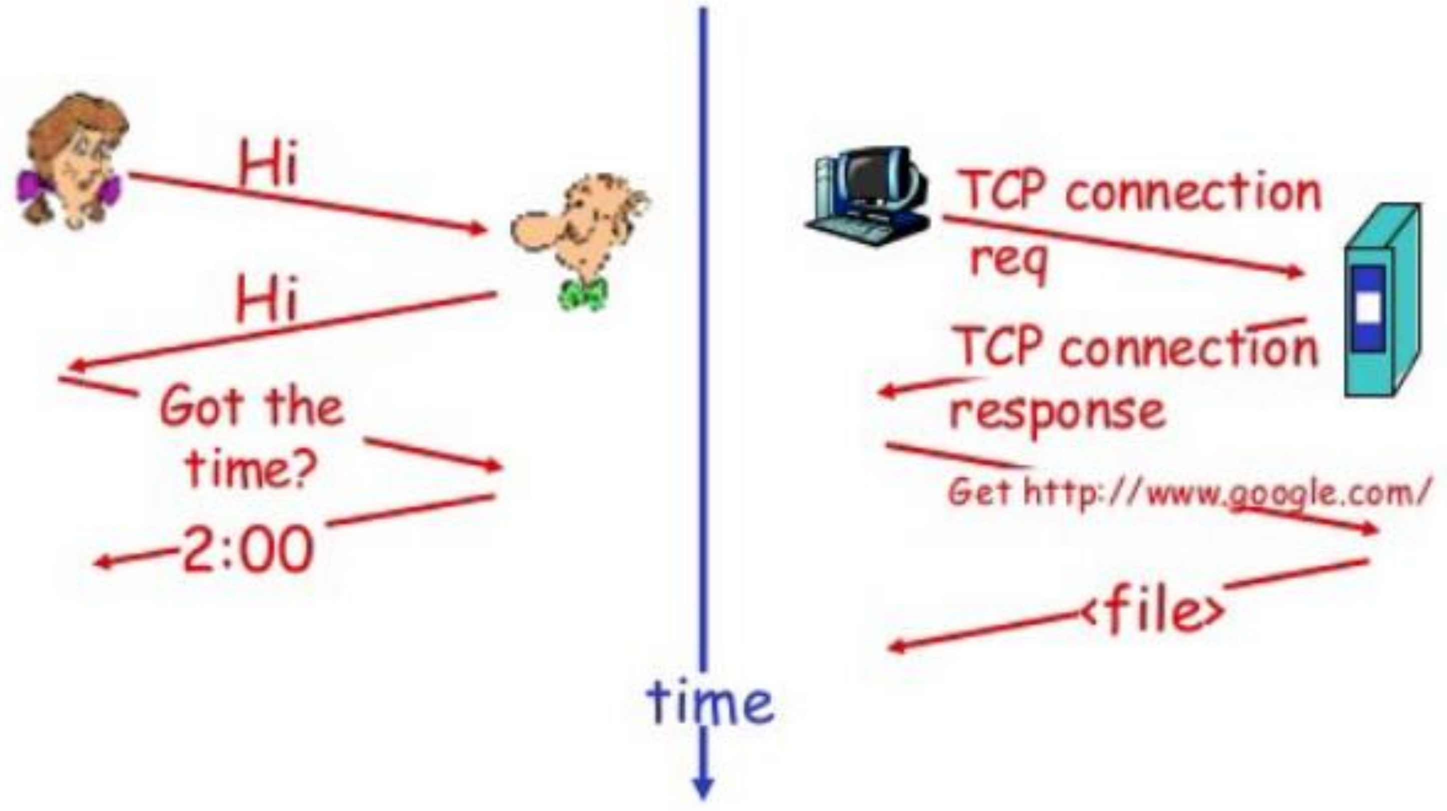
- You will also make a connection to the internet and protocols.
- Information travelling across the internet will often need to contain both the contents of the message itself, and information that helps to format, route or interpret this data.

## Protocol, Character Set, Data Frame

- One of the most powerful uses of the internet is sending text to people. Since the internet can only send bits around we need a way to encode text with bits...
- If it were up to you, how would you encode text in binary? Quickly, jot down an idea for encoding text.

## What's a protocol?

a human protocol and a computer network protocol:



## Protocols

### Protocols (Rules)

What to communicate, how to communicate and when to communicate.

### Key Elements of Protocols

1. Syntax: Structure of format of data
2. Semantics: Meaning of each section of bits.
3. Timing: When to send and how fast?

## HTML

- A lot of you have taken web design, and you used this language: HyperText Markup Language, or HTML
- This is the **language of the web**,  where the content and formatting of a web page are written.
Look at the background of a web page for the code.

## Protocol called: HTTP

- HyperText Transfer Protocol
- this is the foundation for communication on the web – it was designed to send and receive web page data over the internet.

## ASCII

- ASCII makes it all happen.
- ASCII – American Standard Code for Information Interchange
- This is the universally recognized raw text format that any computer can recognize

## ASCII

- ASCII encoding is the standard number-to-text encoding scheme used in computers and on the Internet.
- This protocol is 7 bits long and so can encode 128 different symbols, each corresponding to one of the binary numbers between 0 and 127.
- Originally the encoding was designed for older printers and so the codes associated with the numbers 0-31 were designated as "control codes" which instruct the printer to perform certain actions.

## ASCII

- ASCII Codes 32-126, however, contain many of the most commonly used symbols you use on the internet, including the lowercase and capital letters, the numbers 0-9, and most commonly-used punctuation marks.
- This printable character set is also used to represent all of the text formatting we use to add variety and emphasis to "plain-text" documents. The ability to represent formatting using only ASCII symbols is the result of cleverly designed protocols.

# ASCII

- ASCII codes were originally 7 bits long and so there are 128 possible values.
- 0-31 are “control characters” that are largely defunct and go unused; they were formerly used to control various aspects of machines and printers.
- 127 is the symbol for delete.

## ASCII

- Over time, 8 bits became a standard “chunk-size” for encoding information. ASCII made the transition to this 8-bit encoding by just adding an extra 0 to the front of the old 7-bit codes.

## Activity

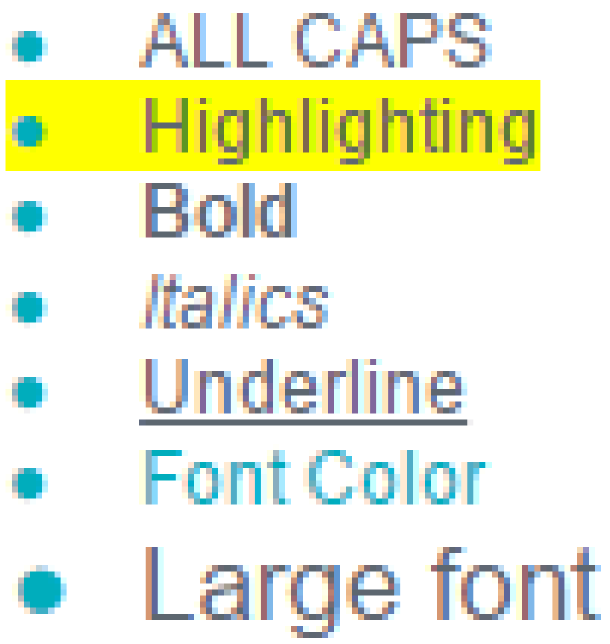
- Using the ASCII table, translate your first name from letters to numbers using the ASCII table.
- Write your name as: *Name!" (capital first letter, exclamation point at the end)
- Take a minute to do this
- Having a standardized protocol like ASCII to encode text enables us to send and receive textual information.
- This is very useful, but there are still instances when we will want even greater expressive power in our digital communications.

## Activity Guide:

### Sending formatted text

- What if you wanted to send formatted text that included things like the ability to underline, bold, or italicize words....specify a different font size, or color?

## Things like this:



- ALL CAPS
- Highlighting
- **Bold**
- *Italics*
- Underline
- Font Color
- Large font

## Activity

- Today your challenge is to:
  - Invent a protocol for sending formatted text
  - Use the Internet Simulator to test out your protocol.

- You will also notice that the Internet Simulator has been updated so that you can now type ASCII text characters to send.

## Activity

- Test your protocols by having one member of the team make a secret formatted message to send to the other members of the team using the Internet Simulator.
- The receiver of the message can write it down. Compare with the original message – how close was it?!

## Activity

- If your protocols were tight, then your messages should match.

# Internet Simulator

UNIT 1 EXTRA LESSON 4

## Internet Simulator Activity

**Go Code.org!**

## Text-based Code

- What you likely did in the activity was invent a text-based code.
- Whether you are formatting languages like HTML, or Markdown or programming languages like Java, C++, or Python, all of these languages have one thing in common: they use ASCII text to encode other text or information

## Discussion

- Congrats!  You  just invented a coding language. At this point in the course a code and a protocol are very similar. Even though it's probably something no one else will use, the process you just went through gives a taste of inventing any kind of formal language or protocol that ultimately needs to be interpreted and processed by a computer.

## Discussion

- Look at Websites and HTML again – you can see the code behind the website in Chrome if you choose:
  - the three vertical dots
  - More Tools
  - Developer tools

- and you will see what goes into making a web page

## Lesson Take Away:

- sequences of binary states can be used to represent numbers
- numbers can be assigned to letters of the alphabet to encode text with plain text you can make a code you can use to apply other meanings (or formats) to text
- you can invent a "formatting language" (like HTML) to represent different ways you want text messages to appear.

## Practice

Fill out your rubric

Do your assessment and finish out the lesson in Code Studio

And, test

## Assessment Question/Answer:

- The word “Apple” translated into its ASCII number equivalent is:

  097 112 112 108 101
  097 108 108 111 119
  065 112 112 108 101
  065 110 110 105 101
  065 108 108 111 119

## Key Concepts and Pedagogy

CHAPTER COMMENTARY

## Protocols are **Creative** (and Arbitrary)

- In this chapter, students often design their own communication protocols to address a targeted problem.
- Students may be surprised to find how creative and seemingly arbitrary this process is.
- This highlights an important understanding that even widely used protocols on the internet were developed by someone to creatively solve a problem. Furthermore, any particular protocol succeeds primarily because the whole computing community agrees to use it.

## Activity Before Concept, Concept Before Vocabulary

- Rather than present the topics covered in this chapter in lecture form, students are presented inquiry-based activities in which they must discover or invent concepts themselves.
- Once the class has gone through this shared activity, the teacher will lead discussions that help synthesize the concepts they were exploring.
- Only then is technical vocabulary introduced. This approach is often shortened to "ABC CBV."

## An Equitable Introduction

- Many aspects of Unit 1 were designed specifically to make it an **equitable** introduction to computer science.
- Unlike other CS topics, (like programming, where some students may already have experience), the internet is likely to be both relevant and mysterious to all students.
- These activities encourage collaboration, inventiveness, and exploration. These features combine to build a positive and supportive classroom environment where it is safe to make mistakes or ask questions.
- The "ABC CBV" pedagogy ensures all students have a shared experience to refer to before more technical vocabulary is introduced.