DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Rev 1-19-22

Tech History

Part 3

Software & Networks (Internet)

by
Dr Jeff Drobman
Dr Jeff Software

Index

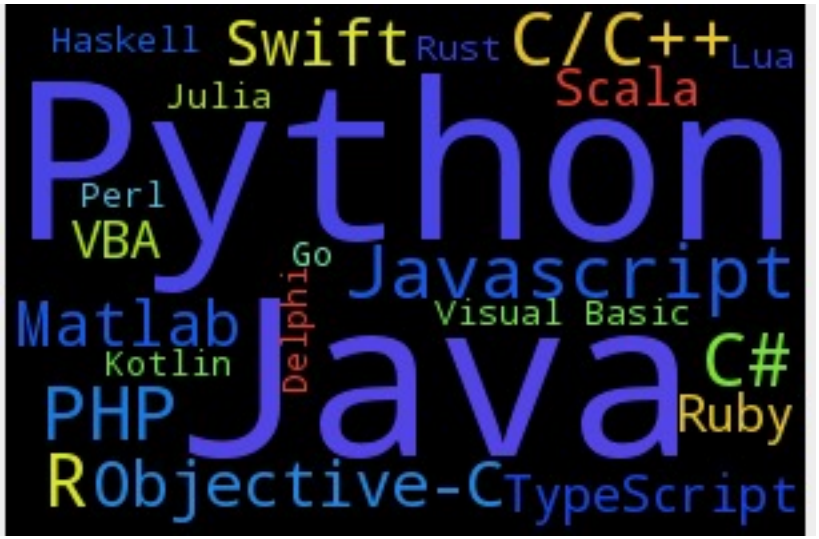
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Software

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Software

- Languages



Realms of Software

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

~70% of all software

❖Applications
- ❑ Desktop
- ❑ Mobile (Apps)
- ❑ Web

❖ APIs (Frameworks)

❖Web
- ❑ Markup
- ❑ Applications
- ❑ SQL databases

❖ Client-Server model
❖ Language “stacks” (e.g., LAMP)

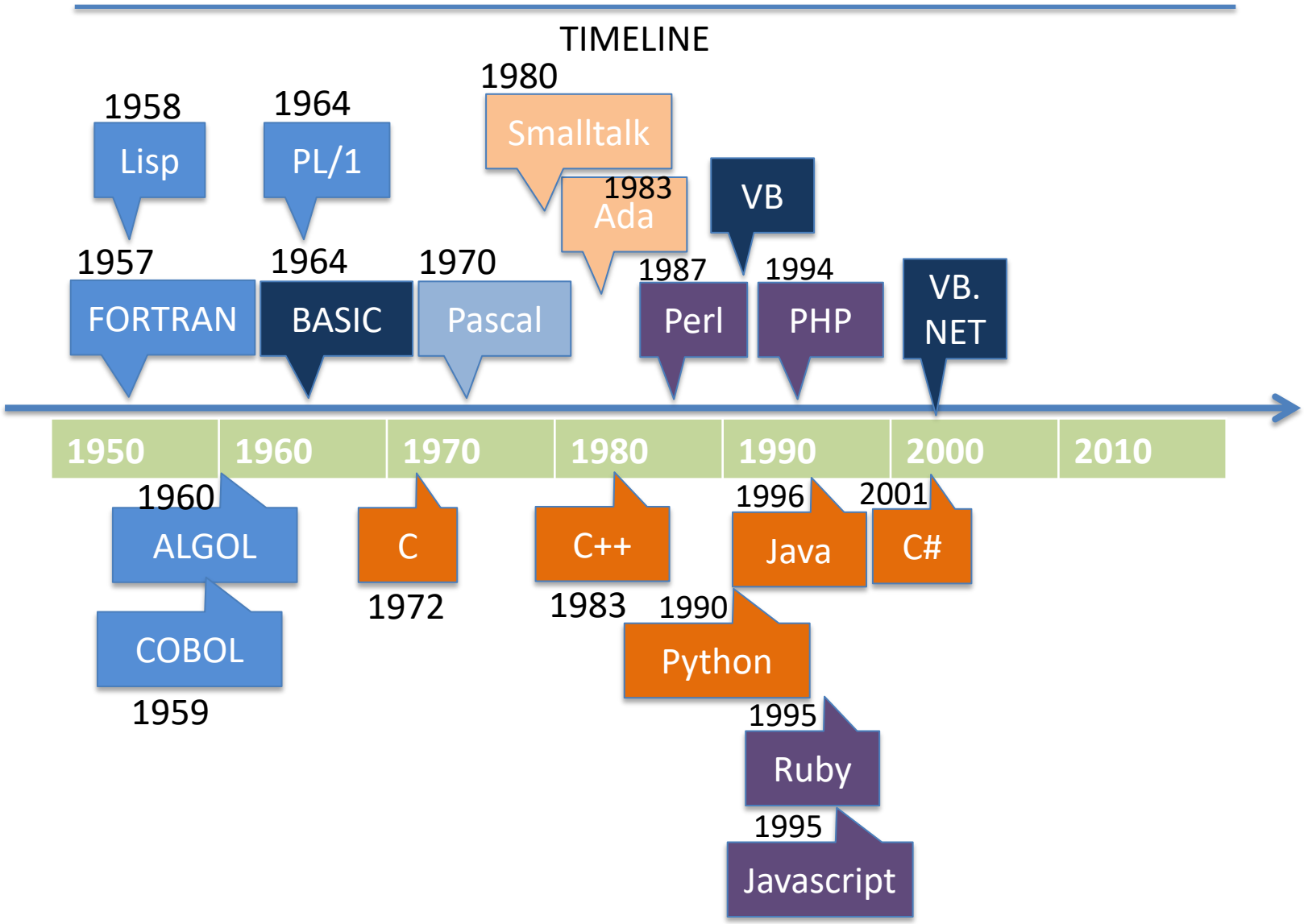
❖Embedded Control
- ❑ Small (8-bit)
- ❑ Medium (16-bit)
- ❑ Large (32/64-bit)

❖ From TV remotes to
❖ Autonomous cars and
❖ Robots

➢ Common required properties
- Performance
- Reliability (bug free)
- *Security*

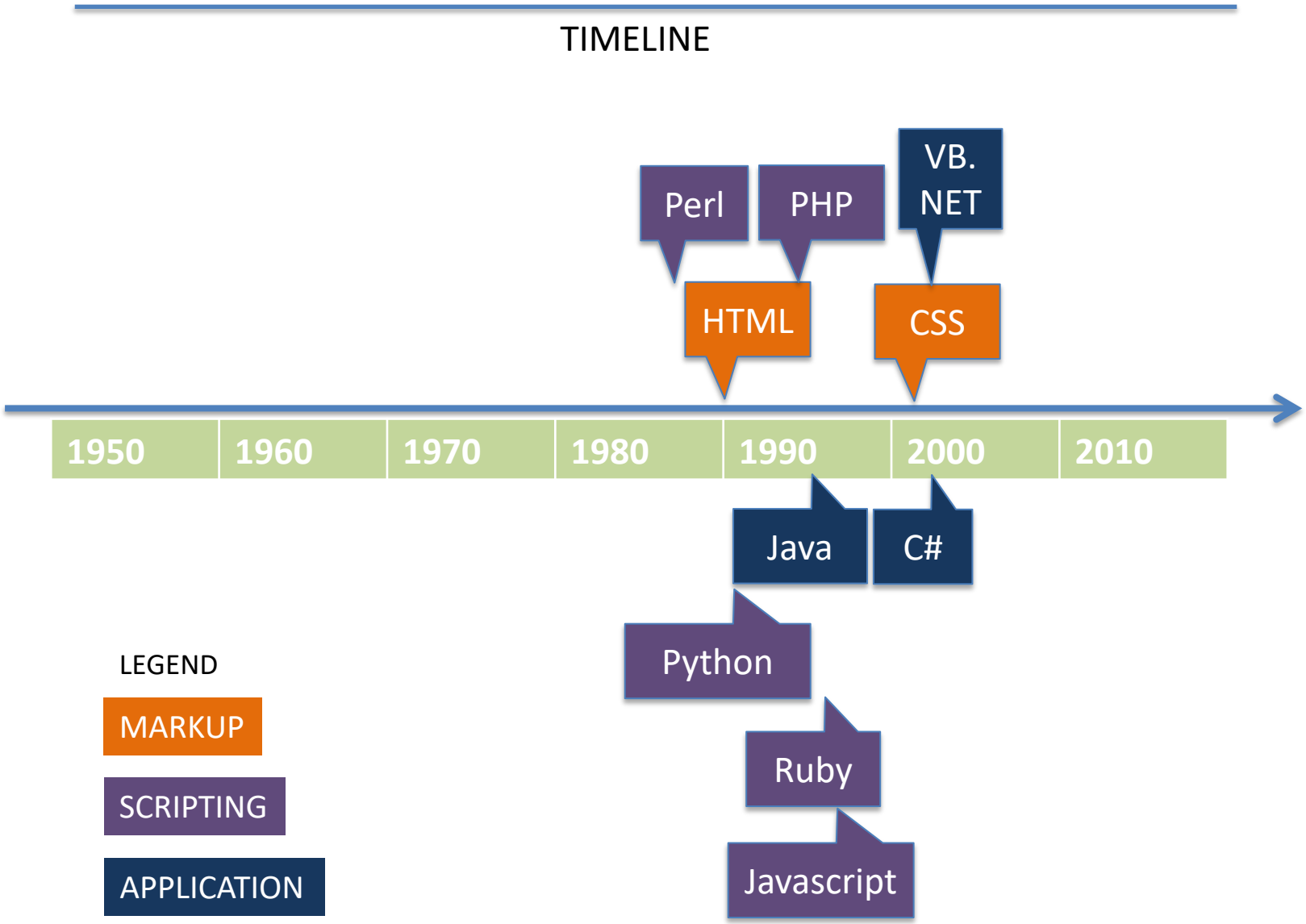
HL Languages

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Web Languages

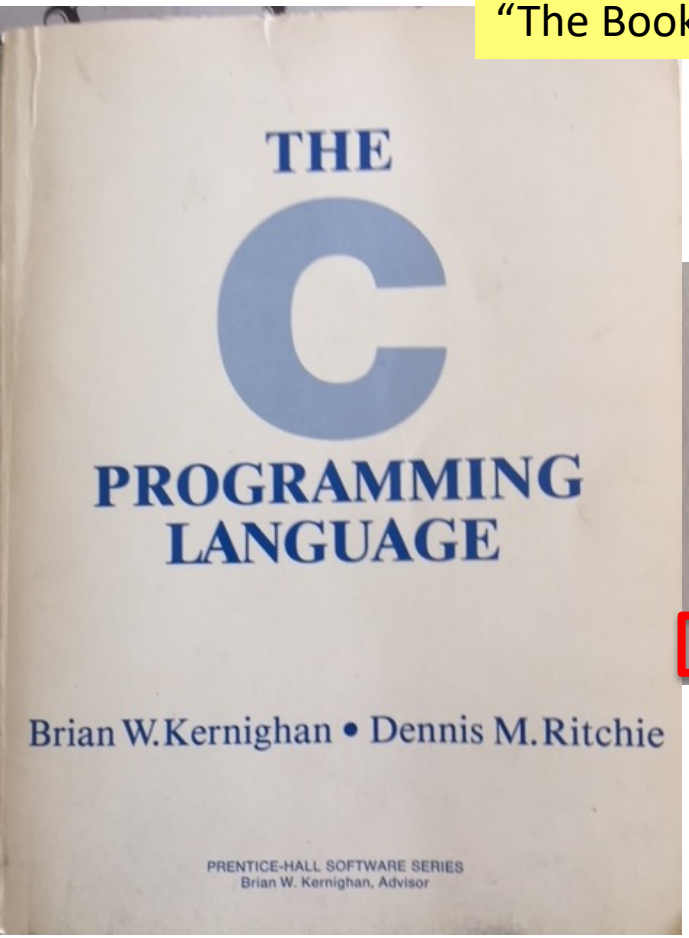
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



C History

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

"The Book"



© 1978

This book was set in Times Roman and Courier 12 by the authors, using a Graphic Systems phototypesetter driven by a PDP-11/70 running under the UNIX operating system.

- Kernighan & Ritchie

C Design

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators. C is not a "very high level" language, nor a "big" one, and is not specialized to any particular area of application. But its absence of restrictions and its generality make it more convenient and effective for many tasks than supposedly more powerful languages.

C was originally designed for and implemented on the UNIX† operating system on the DEC PDP-11, by Dennis Ritchie. The operating system, the C compiler, and essentially all UNIX applications programs (including all of the software used to prepare this book) are written in C. Production compilers also exist for several other machines, including the IBM System/370, the Honeywell 6000, and the Interdata 8/32. C is not tied to any particular hardware or system, however, and it is easy to write programs that will run without change on any machine that supports C.

C is a general-purpose programming language. It has been closely associated with the UNIX system since it was developed on that system, and since UNIX and its software are written in C. The language, however, is not tied to any one operating system or machine; and although it has been called a "system programming language" because it is useful for writing operating systems, it has been used equally well to write major numerical, text-processing, and data-base programs.

C is a relatively "low level" language. This characterization is not pejorative; it simply means that C deals with the same sort of objects that most computers do, namely characters, numbers, and addresses. These may be combined and moved about with the usual arithmetic and logical operators implemented by actual machines.

C Contents

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

- Pointers!

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

SDLC

Software Development Life Cycle

- *Software Engineering*

- 6 Stages (we use the 1st four)

- **Requirements**
- **Design**
- **Implementation**
  - Coding
- **Testing**
- Deployment
- Maintenance

Software Engr. at Google

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

ACM

What is the secret to software engineering at Google? Over the years, we've come to recognize three key principles that guide our practices and decisions: Time, Scale, and Tradeoffs. We recently published a book with O'Reilly on those principles, and we'll share the key ideas here.

Software engineering and programming are related but different problems. If programming is about producing code, software engineering is about maintaining that code for the duration of its usefulness. It is about the practices, policies, and decisions that support robust and reliable code. It is about building for growth and for the ability to manage change, sustainably.

At Google, we have learned many lessons related to the sustainability of software. Google arguably maintains one of the largest codebases ever. The expected lifespan of the codebase is at least another couple of decades. We've needed to figure out how to operate at a scale previously unattempted for a timespan longer than most others have considered. Learning from the difficulties that we have encountered along the way while wrangling with this unprecedented problem, we have developed practices around time, scaling, and evidence-based decision making. This is what has enabled us to operate as we do.

Software Engr. at Google

ACM

Presenter:

Titus Winters, *Senior Staff Software Engineer, Google*

Titus is a Senior Staff Software Engineer at Google, where he has worked since 2010. At Google, he is the library lead for Google's C++ codebase: 250 million lines of code that will be edited by 12K distinct engineers in a month. He served for several years as the chair of the subcommittee for the design of the C++ standard library. For the last 10 years, Titus and his teams have been organizing, maintaining, and evolving the foundational components of Google's C++ codebase using modern automation and tooling. Along the way, he has started several Google projects that are believed to be in the top 10 largest refactorings in human history. That unique scale and perspective has informed all of his thinking on the care and feeding of software systems. His most recent project is the book *Software Engineering at Google* (aka "The Flamingo Book"), published by O'Reilly in early 2020.

Moderator:

Hyrum Wright, *Senior Staff Software Engineer, Google*

Hyrum Wright is a Senior Staff Software Engineer at Google, where he leads the Code Health Team. His team is responsible for the maintainability of Google's source code, ensuring the scalable evolution of billions of lines of code. He has spent the last decade improving techniques for maintenance of large-scale software systems, and sharing those lessons inside and outside of Google. Hyrum is one of the editors of *Software Engineering at Google*, and occasionally teaches at Carnegie Mellon University. He coined the eponymous Hyrum's Law, but not its name.

Google Code Base

HOW BIG IS Google? We can answer that question in terms of revenue or stock price or customers or, well, metaphysical influence. But that's not all. Google is, among other things, a vast empire of computer software. We can answer in terms of code.

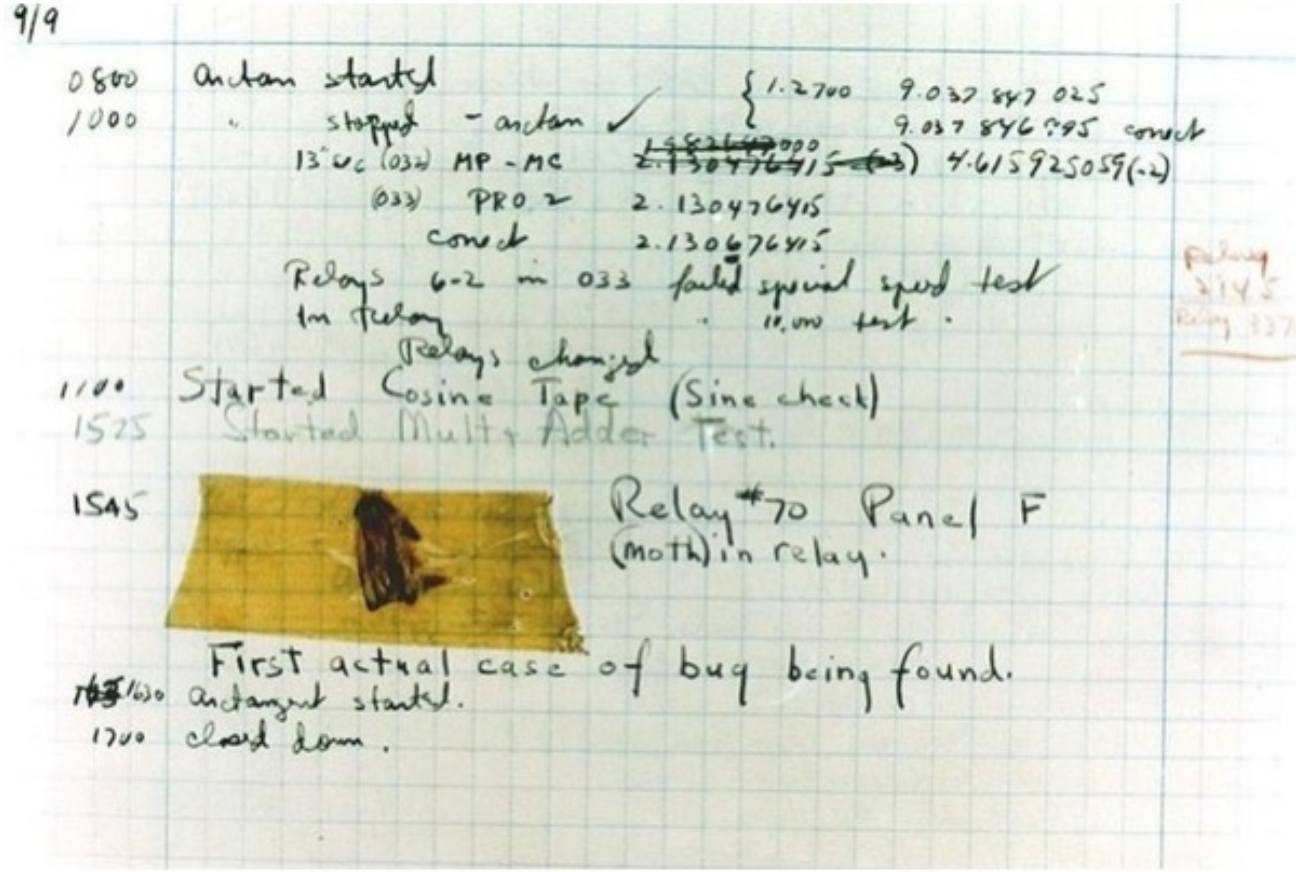
Google's Rachel Potvin came pretty close to an answer Monday at an engineering conference in Silicon Valley. She estimates that the software needed to run all of Google's Internet services—from Google Search to Gmail to Google Maps—spans some 2 **billion** lines of code. By comparison, Microsoft's Windows operating system—one of the most complex software tools ever built for a single computer, a project under development since the 1980s—is likely in the realm of 50 million lines.

So, building Google is roughly the equivalent of building the Windows operating system 40 times over.

Bugs/Debugging

Grace Hopper, inventor of the COBOL programming language, who worked in the Navy's engineering program at Harvard, found the bug. It was an actual insect.

The incident is recorded in Hopper's logbook alongside the offending moth, taped to the logbook page:



The "bug" and the page it's attached to reside at the Smithsonian's Museum of American History in Washington, ... (more)

History

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

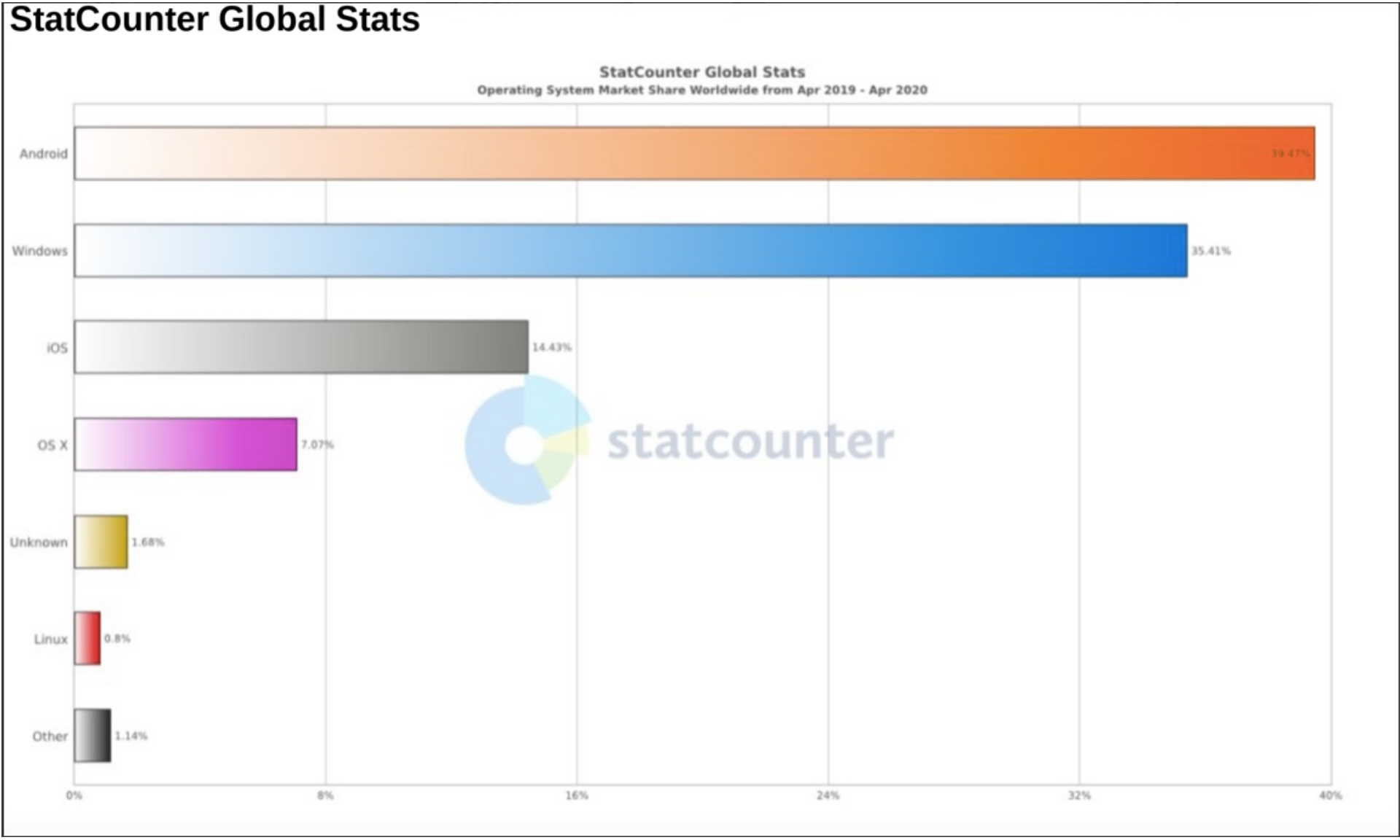
Software

OS

OS Stats

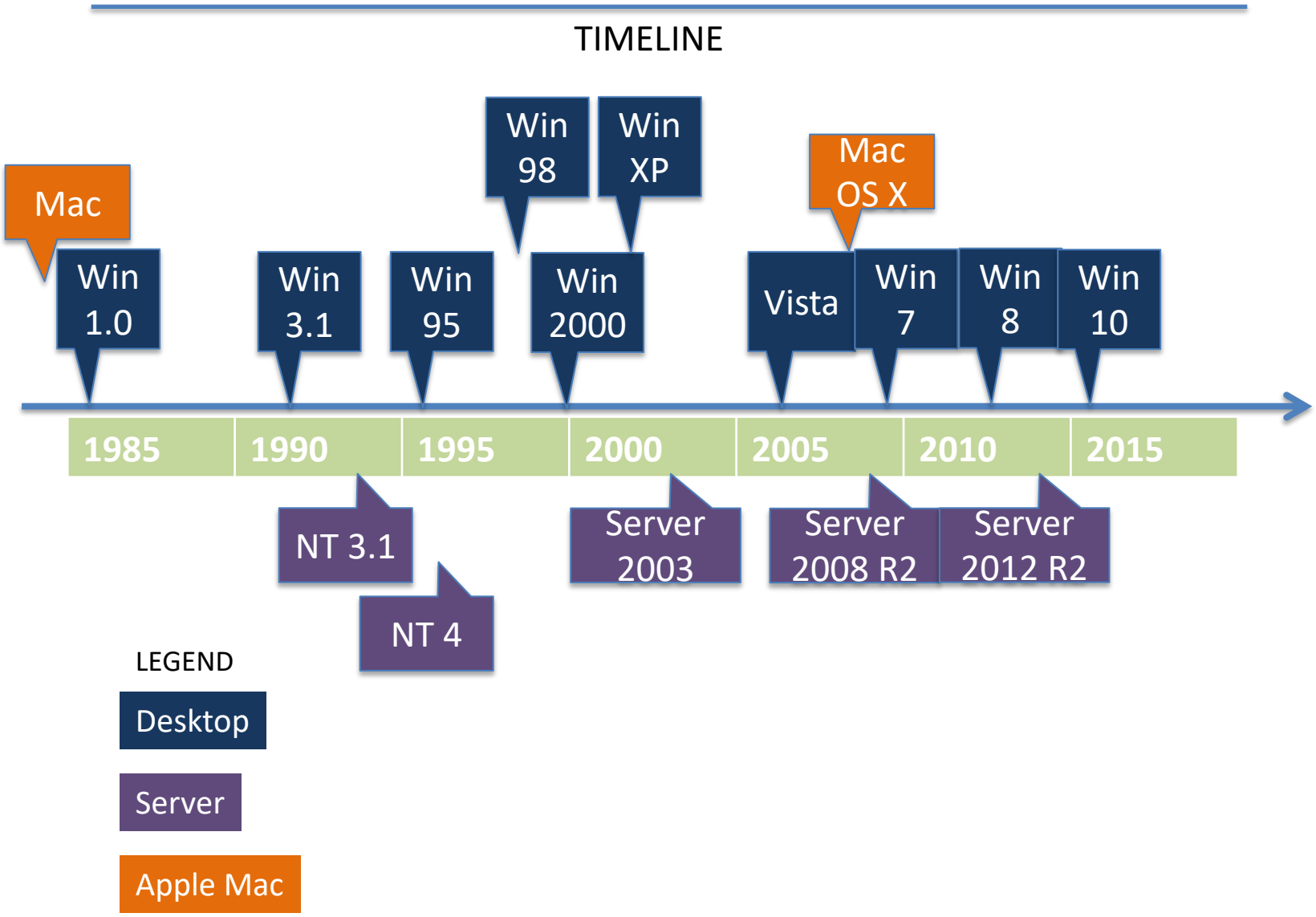
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

StatCounter Global Stats



MS Windows Timeline

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



OS X **10.8** Mountain Lion, **10.9** Mavericks, **10.1.10** Yosemite, **10.11** El Capitan, MacOS **10.12** Sierra, MacOS **10.13** High Sierra

MS Windows by Gates

In fact, between 1988 and 2000, there were actually two separate Windows groups...one working on NT, and the other working on the desktop versions (e.g., Windows 3.x, Windows 95, Windows 98, Windows ME), which were based on 16-bit code and not on the NT kernel. With the advent of XP in 2001, Windows development was in the hands of the NT group, with some continuing maintenance on the 16-bit products.

In the late 1980s, Bill Gates was not developing product code. In fact, the last product he contributed a significant amount of code to was the BASIC interpreter built into the Tandy Model 100, which was released back in 1983. Bill contributed to some products after 1983, but his primary role was managing and growing the company. That said, he remained very technical, drilling down into the details in design reviews, understanding how everything worked, making recommendations for changes to the design, etc.

By the way, last I checked, the Windows source code (including all kernel mode and user mode code, utilities, networking, etc.) had approximately 55 million lines of code.

Unix Timeline

TIMELINE

Ken Thompson (sitting) and Dennis Ritchie working together at a PDP-11

Thompson & Ritchie

Steve Jobs

Bell/GE MIT Multics

Bell Labs Unix

UCB BSD Unix

Next Step OS

Mac OS X

1965 1970 1975 1980 1985 1990 1995 2000 2005 2010 2015

Sun OS

Sun Solaris

Red Hat

Ubuntu

Fedora

CP/M

SCO Xenix

Gary Kildall

Linux

Linus Torvalds

MS DOS

Win 1.0

Microsoft

LEGEND

Linux

Sun/SCO

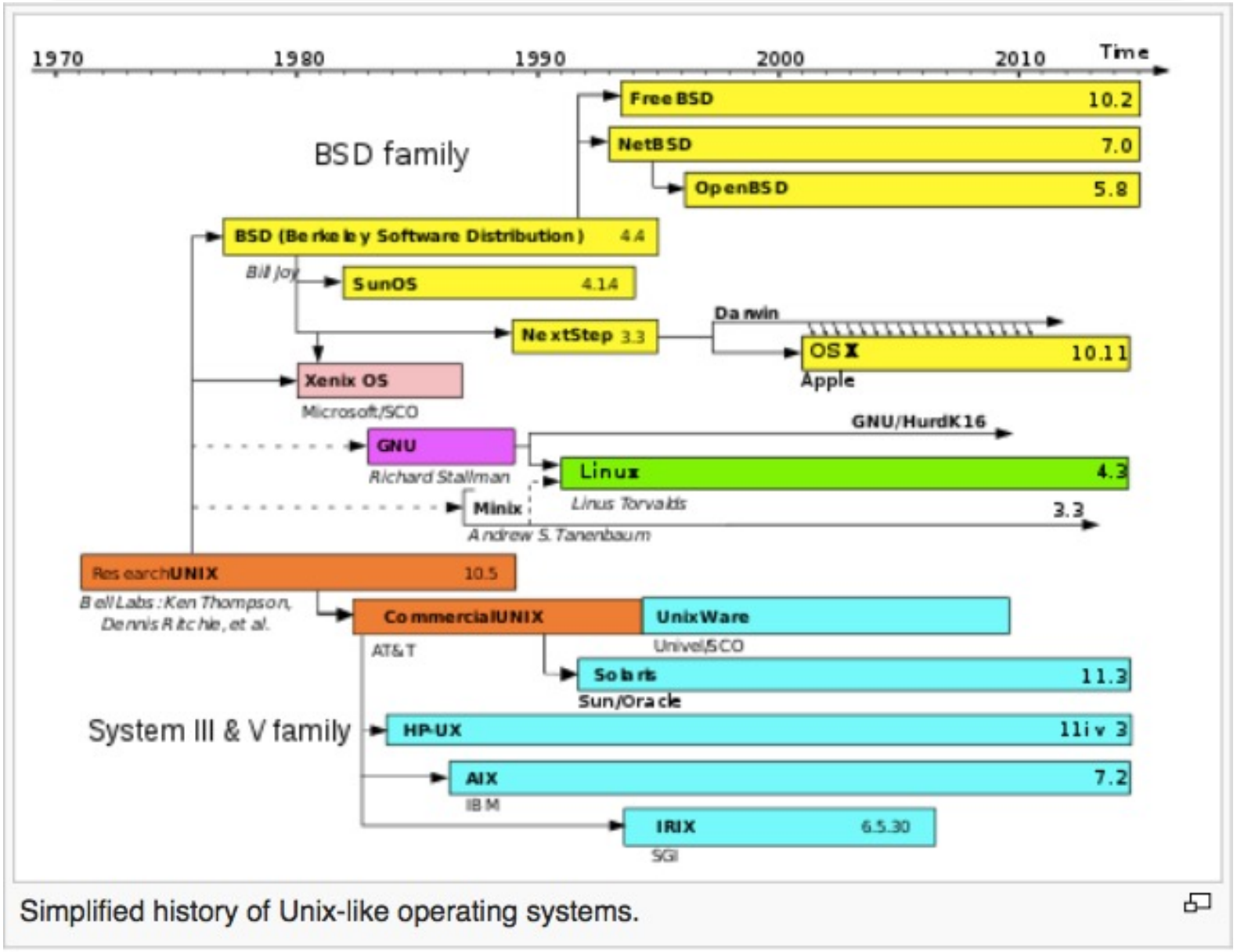
Bell Labs

Micro

Unix Genealogy

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Wikipedia

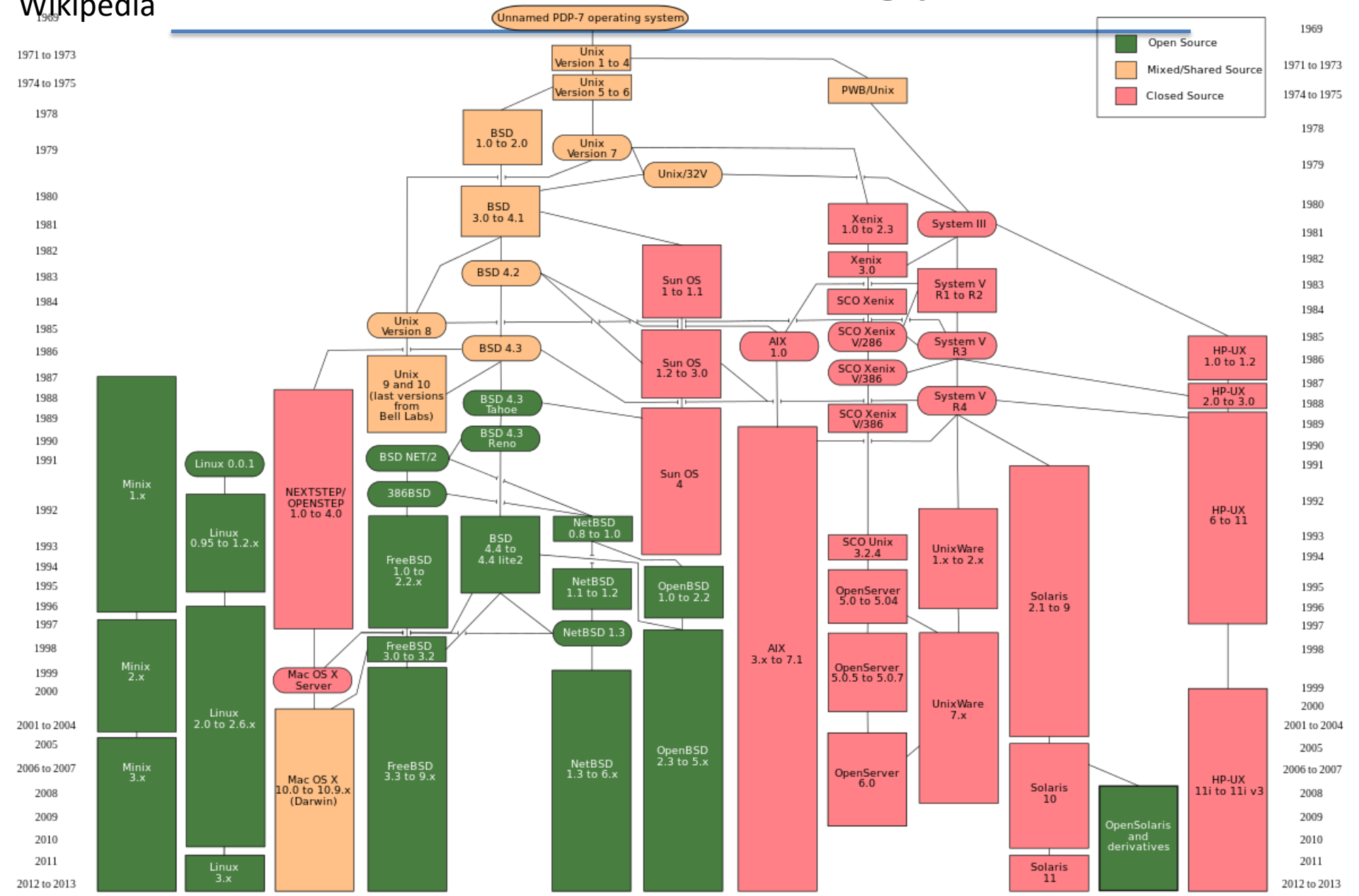


Simplified history of Unix-like operating systems.

Unix Genealogy

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Wikipedia



Unix Specs

Wikipedia

Unix (trademarked as **UNIX**) is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.[3]

Initially intended for use inside the Bell System, AT&T licensed Unix to outside parties from the late 1970s, leading to a variety of both academic and commercial variants of Unix from vendors such as the University of California, Berkeley (BSD), Microsoft (Xenix), IBM (AIX) and Sun Microsystems (Solaris). AT&T finally sold its rights in Unix to Novell in the early 1990s, which then sold its Unix business to the Santa Cruz Operation (SCO) in 1995,[4] but the UNIX trademark passed to the industry standards consortium The Open Group, which allows the use of the mark for certified operating systems compliant with the Single UNIX Specification (SUS). Among these is Apple's macOS,[5] which is the Unix version with the largest installed base as of 2014.

From the power user's or programmer's perspective, Unix systems are characterized by a modular design that is sometimes called the "Unix philosophy", meaning that the operating system provides a set of simple tools that each perform a limited, well-defined function,[6] with a unified filesystem as the main means of communication[3] and a shell scripting and command language to combine the tools to perform complex workflows. Aside from the modular design, Unix also distinguishes itself from its predecessors as the first portable operating system: almost the entire operating system is written in the C programming language[7] that allowed Unix to reach numerous platforms.

Many clones of Unix have arisen over the years, of which Linux is the most popular, having displaced SUS-certified Unix on many server platforms since its inception in the early 1990s.

Unix

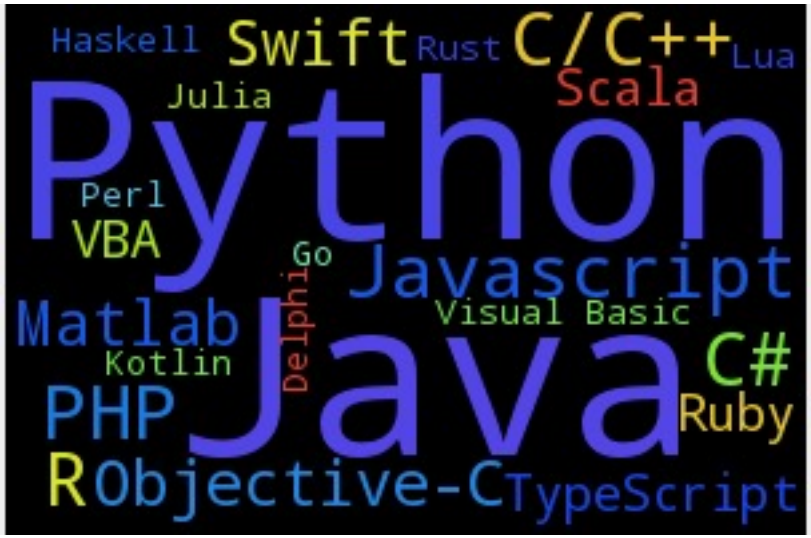
Evolution of Unix and Unix-like systems

Developer	Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, and Joe Ossanna at Bell Labs
Written in	C and assembly language
OS family	Unix
Working state	Current
Source model	Historically closed-source, while some Unix projects (including BSD family and Illumos) are open-source
Initial release	Development started in 1969; 47 years ago First manual published internally in November 1971[1] Announced outside Bell Labs in October 1973[2]
Available in	English
Kernel type	Monolithic
Default user interface	Command-line interface and Graphical (X Window System)

Software

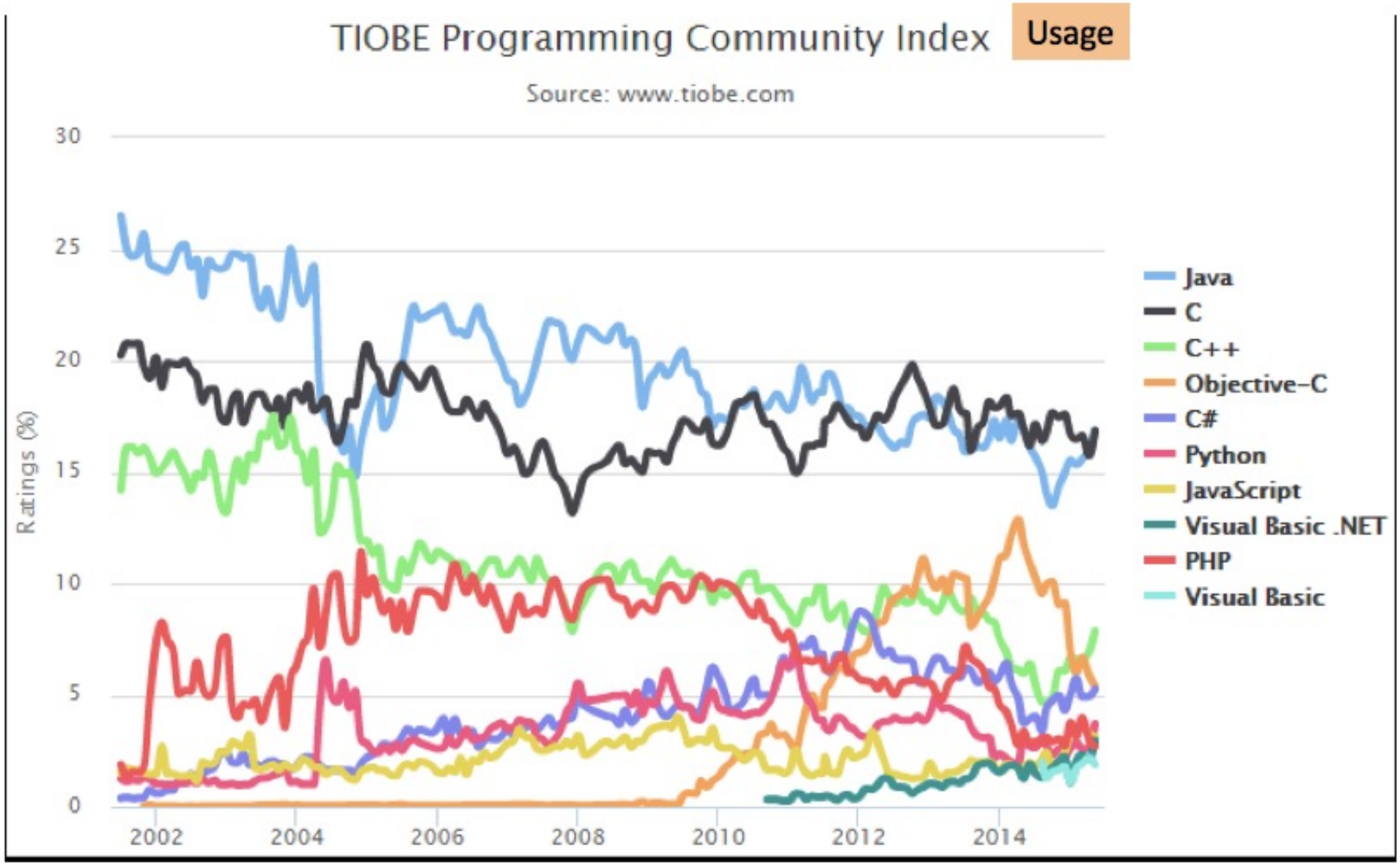
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Java
Vs. Other HLLs



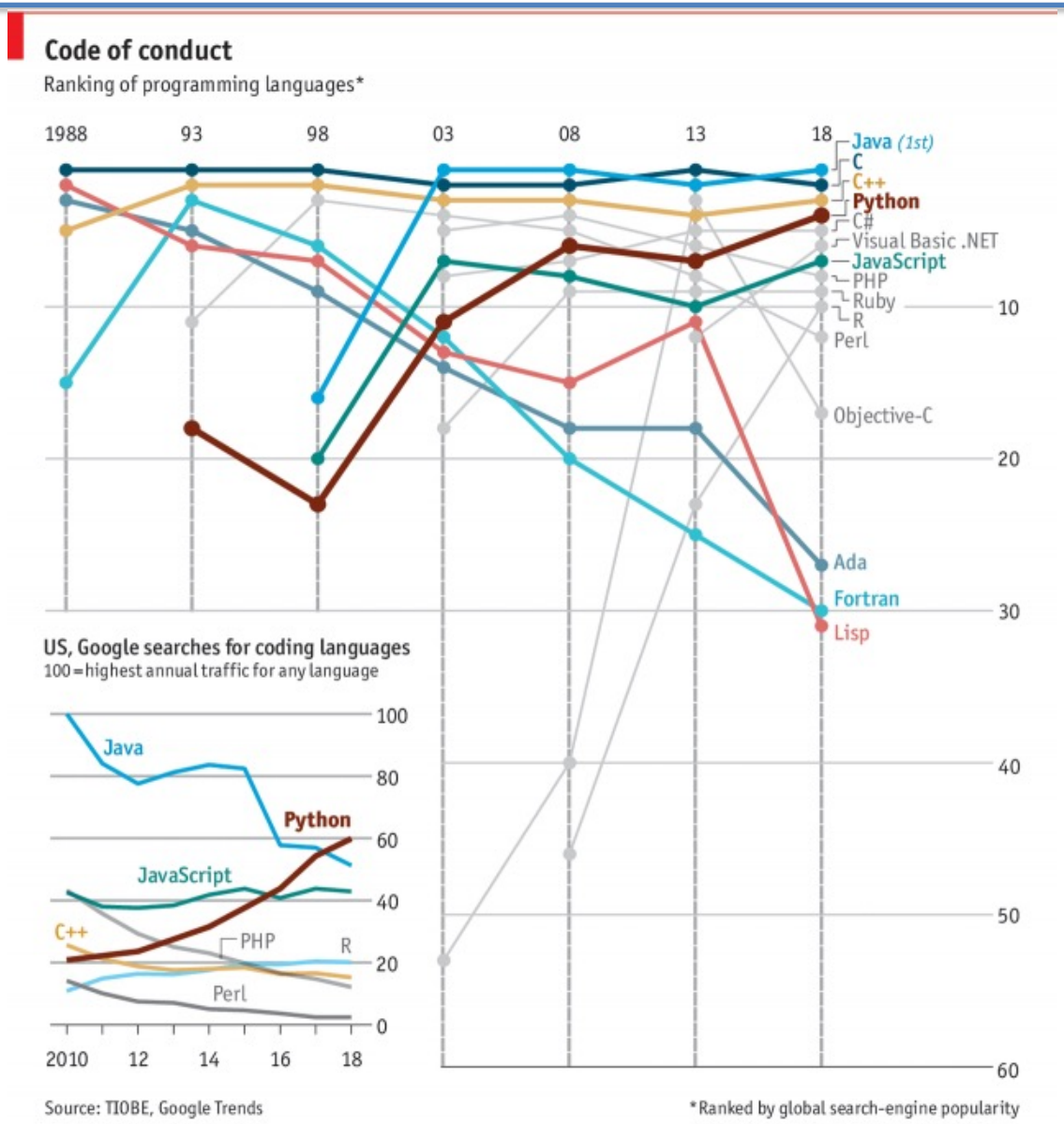
HLL Usage

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



HLL Usage by Search

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



GUI

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

1980
- ❖ Xerox PARC (started it all – Alan Kay)
  - Windows with mouse (point & click)

1983-4
- ❖ **Apple** *Lisa* & Macintosh ("Mac")
  - *Personal* Computer
  - Adopted Xerox GUI
  - Graphics Applications (MacDraw, MacPaint)
  - Fancy fonts

1990
- ❖ Microsoft
  - Windows: Apple copycat
  - .NET for web forms + ASP

2007
- ❖ Mobile
  - Apple iPhone + iPad (iOS)
  - Google Android
  - Microsoft Windows Phone

Social Media

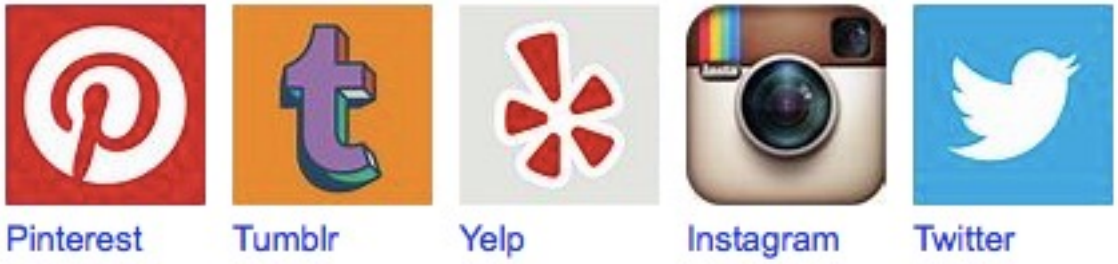
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

- MySpace (started it all)
  - Personal web pages
- **Facebook** (the big one)
  - Personal – Friends
  - Updates with *photos*
  - Profiles – extensive with "Likes"
- LinkedIn
  - Professional – Networks
  - Profiles with *Resumes*
- Twitter
  - Personal + Professional
  - Blogging
- Others
  - Pinterest
  - Instagram
  - Snapchat
  - Quora

Collecting & mining personal *likes* and *activities*

Social Networks

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Foursquare Labs, Inc.

Foursquare is a local search and discovery service mobile app which provides a personalised local search experience for its users. By taking into account the places a user goes, the things they have told the app that they like, and th… +

Founded: 2009 · New York City, NY

Founders: Naveen Selvadurai · Dennis Crowley

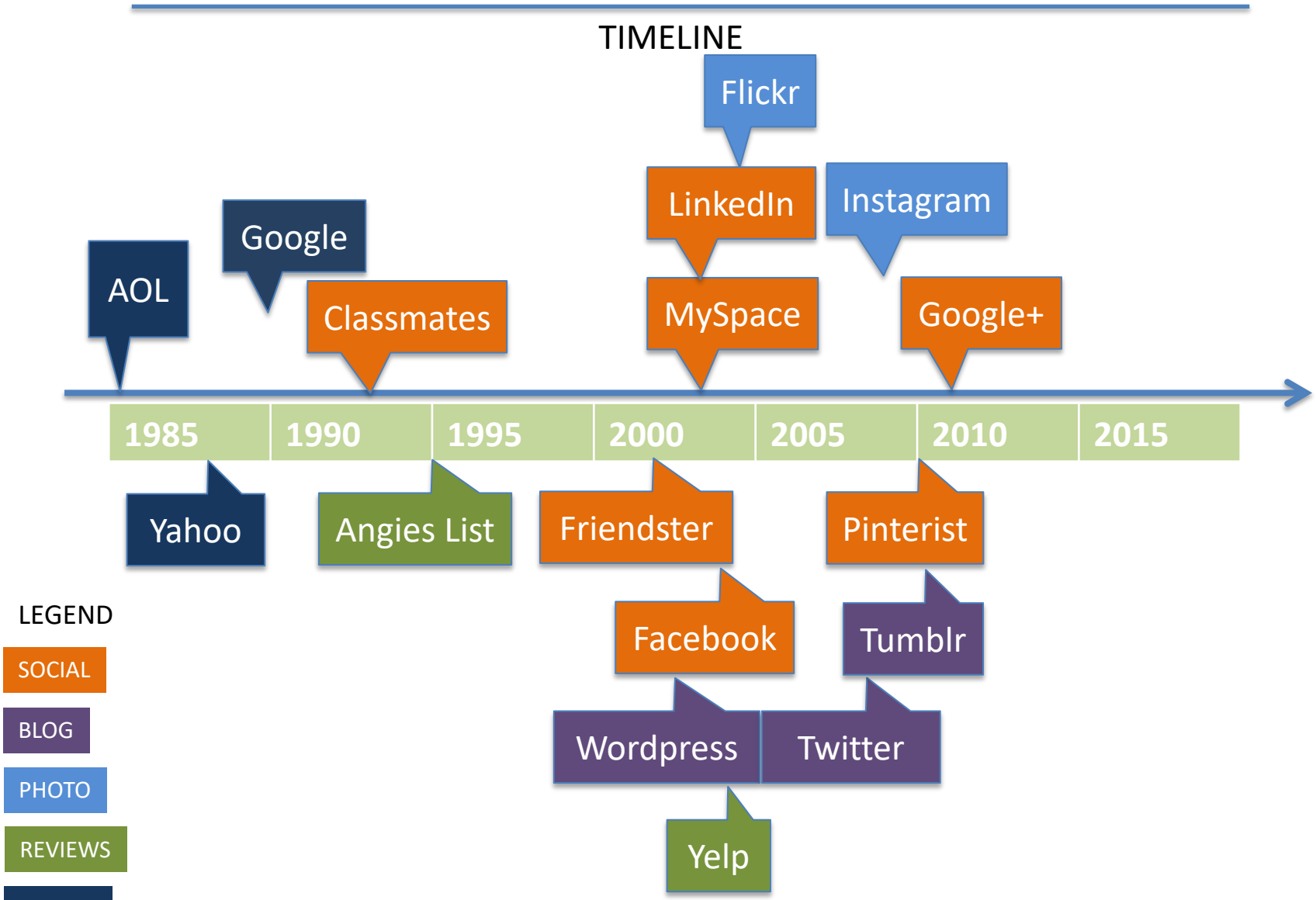
Headquarters: New York City, NY

Angie's List

Angie's List is a US-based, paid subscription supported website containing crowd-sourced reviews of local businesses. For the quarter ending on September 30, 2015, Angie's List reported total revenue of US$87,000,000 an… +

Social Networks

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Wireless Networks

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

- WiFi
- Cellular

WiFi

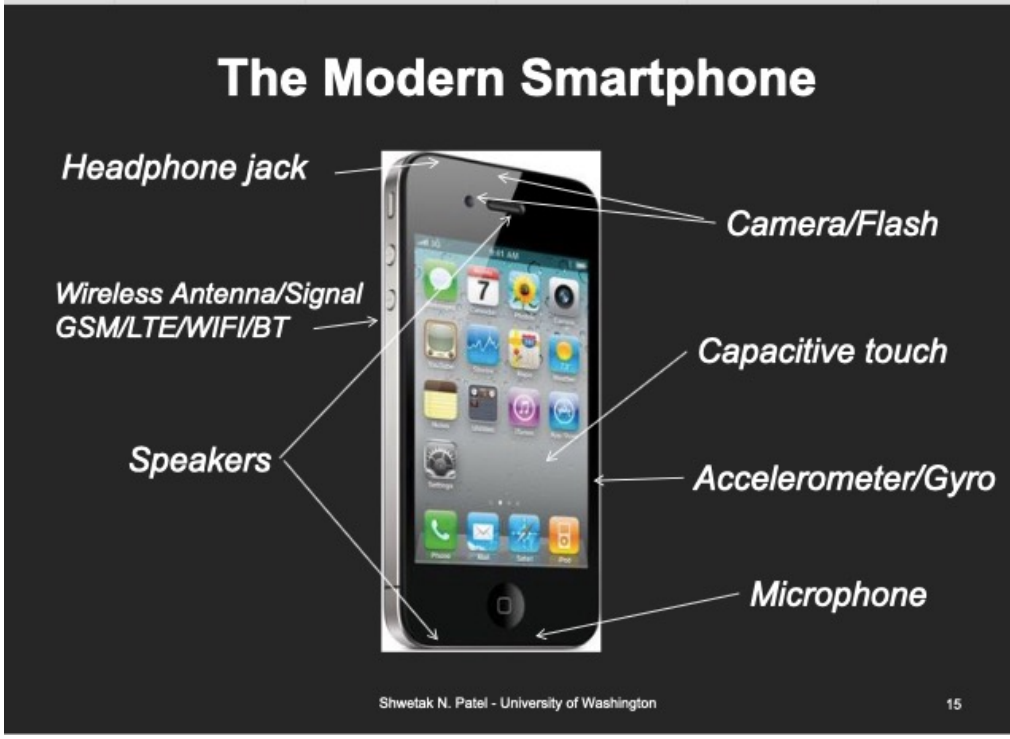
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Wi-Fi generations

Generation/IEEE Standard	Maximum Linkrate	Adopted	Frequency
Wi-Fi 6 (802.11ax)	600–9608 Mbit/s	2019	2.4/5 GHz 1–6 GHz ISM
Wi-Fi 5 (802.11ac)	433–6933 Mbit/s	2014	5 GHz
Wi-Fi 4 (802.11n)	72–600 Mbit/s	2009	2.4/5 GHz
Wi-Fi 3 (802.11g)	3–54 Mbit/s	2003	2.4 GHz
Wi-Fi 2 (802.11a)	1.5 to 54 Mbit/s	1999	5 GHz
Wi-Fi 1 (802.11b)	1 to 11 Mbit/s	1999	2.4 GHz
(Wi-Fi 1, Wi-Fi 2, Wi-Fi 3 are unbranded[41] but have unofficial assignments[42])			

Cell Phones

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Smart phones

Top Smart Phones

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

- Samsung Galaxy Note 7/8 ➢ ON FIRE!
  - Biometrics for authentication
    - Fingerprints
    - **Iris scan**

- Apple iPhone 8/X/11/12
  - Wireless headphones
  - Biometrics for authentication
    - Fingerprints
    - **Facial scan**
  - iOS 15
  - A14/15 ARM SoC

Apple intros totally wireless AirPods that use new W1 chip

Fastest CPU
Fastest GPU
Faster Neural Engine
ML Controller
ML Accelerators
Core ML 3
8.5 Billion Transistors

Cellular Network

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

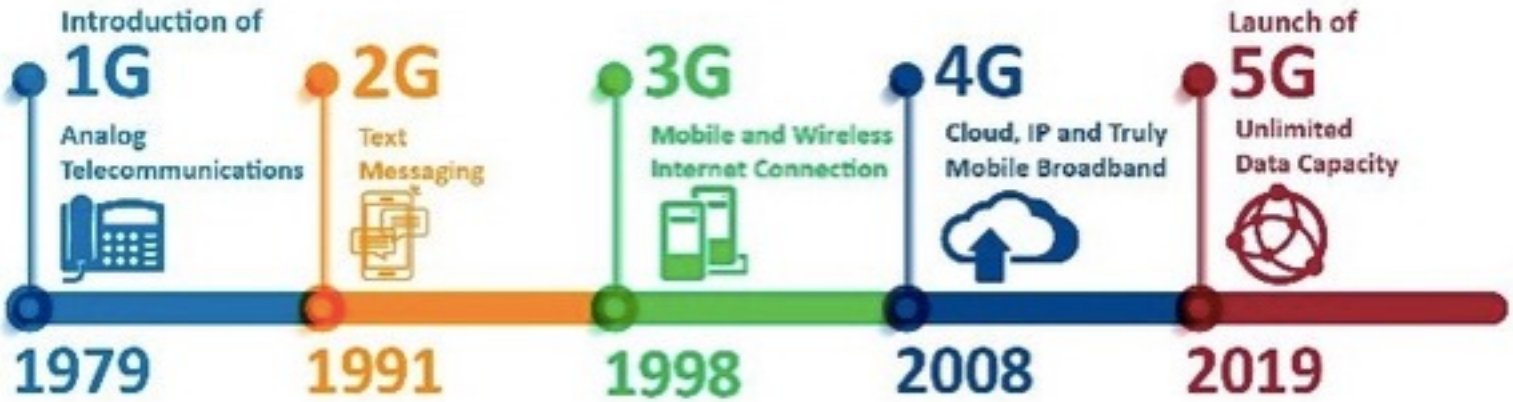
Feature	1G	2G	3G	4G	5G
Voice	A	A	D	D	D
Data rates	300K	1-2M	10M	100M	>1G
Technology	TDMA	TDMA CDMA GSM	CDMA	LTE	LTE+ QPSK QAM
Channels (V/D)	2	2	2	1	1
Year	1979	1991	1998	2008	2019

5G

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

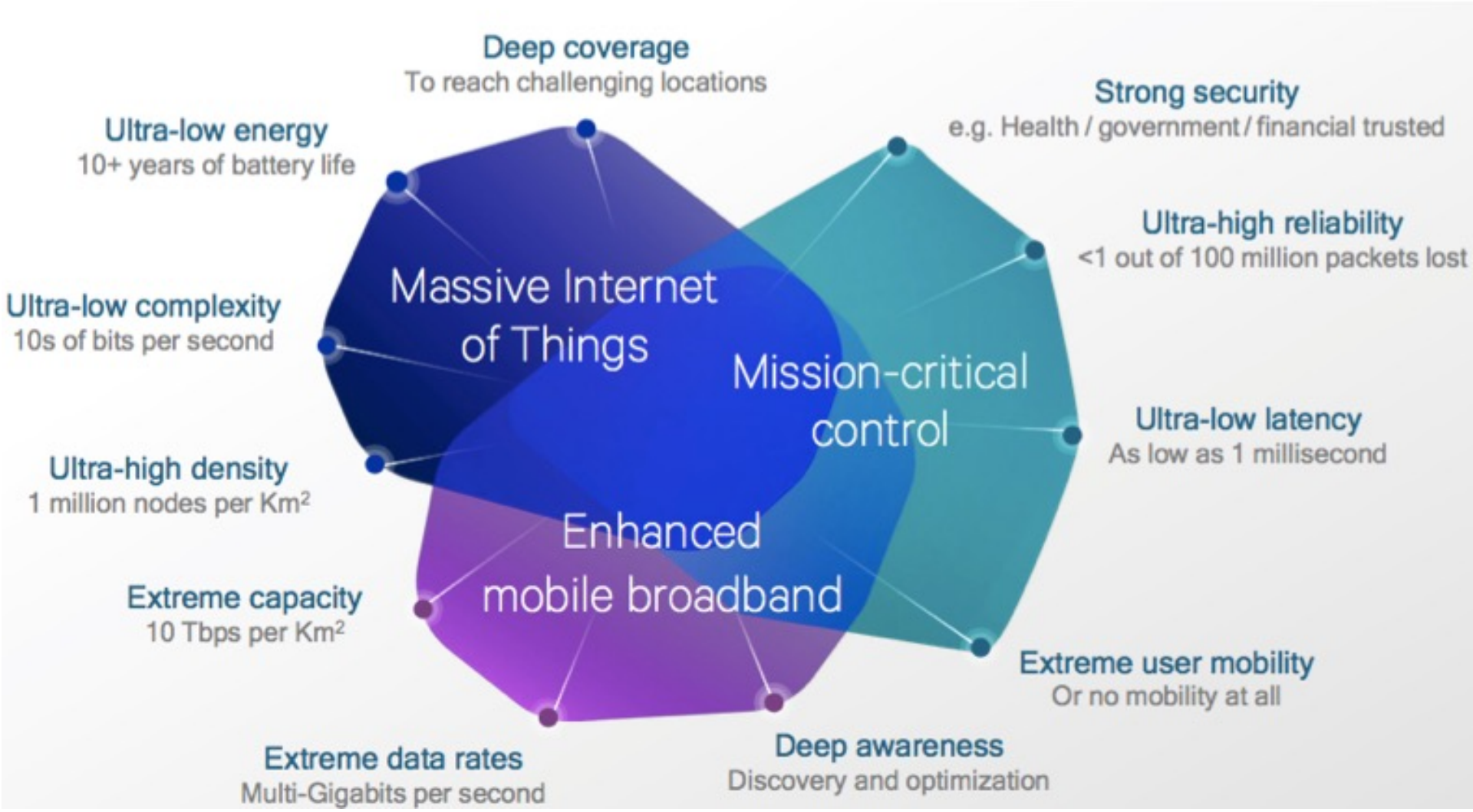
Mobile communications: from 1G to 4G

The Evolution of 5G



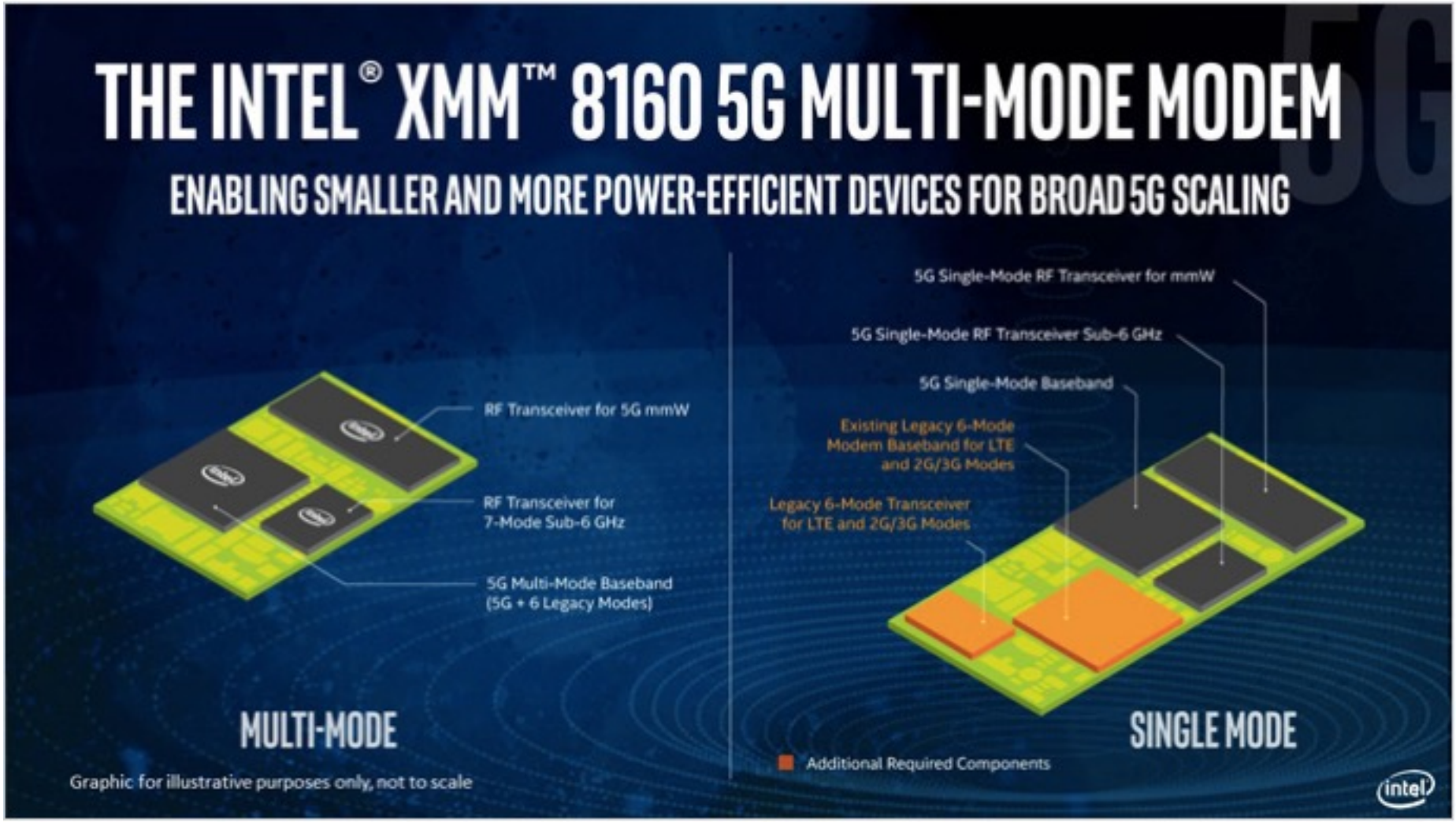
5G

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Intel 5G Board

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Features of the Intel XMM 8160. Image from Intel

5G Tech Specs

Kurt Behnke, former Telecom R&D and Manager Operations at Ericsson (1997-2017)

Answered 23h ago·Upvoted by
Sankaran CB, M.S. Telecommunications & Software Engineering, Illinois Institute of Technology Chicago - Illinois Tech (200...

It is neither revolutionary coding progress nor any novel coding. Just „normal progress"

- Modulation: add 256QAM (and potentially 1024QAM) to the LTE list of coding schemes (QPSK, 16QAM, 64QAM)
- Error correction schemes: Efficieny improvement.
- basic frame timing: the LTE frame can be squeezed in time dimension by factors 2,4,8,16 and 32. At the same time the OFDM spacing is stretched by the same factor (necessarily). This is the trick to cut down on radio latency, but it comes at the cost of higher bandwidth consumption.
- In the high frequency bands TDD is now the rule, no longer the exception. This gives another latency improvement, but is mainly used for MIMO support.
- Core network architecture is flattened and distributed. That is a major latency improvement.

5G Tech Specs

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Eslam Ibrahim

Answered 19h ago

As mentioned before , modulation techniques supported by 5G systems will be : QPSK , 16-QAM , 64-QAM , 256-QAM and also 1024-QAM (Mostly for back-haul links such as microwave/millimeter wave links).

For coding , i believe you mean Forward Error Correction (FEC) , there'll be 2 main types that will be used which are : Low-Density Parity Check (LDPC) , and Polar Codes by Arikan , in addition to the use of the well-known Turbo codes.

For reducing latency of communications , especially for Machine-to-Machine (M2M) communications , new frame structures and also new ARQ (Automatic Repeat Request) will be used to reduce the time of re-transmissions and thus reduce latency.

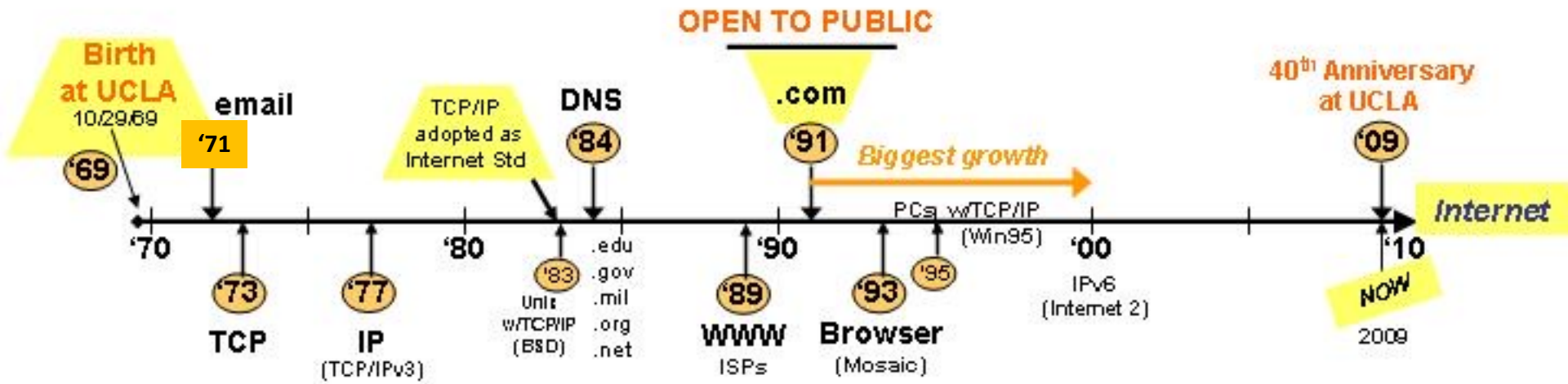
I recommend you read more about Ultra-Reliable Low Latency Communications or URLLC , one of the good papers is **here** .

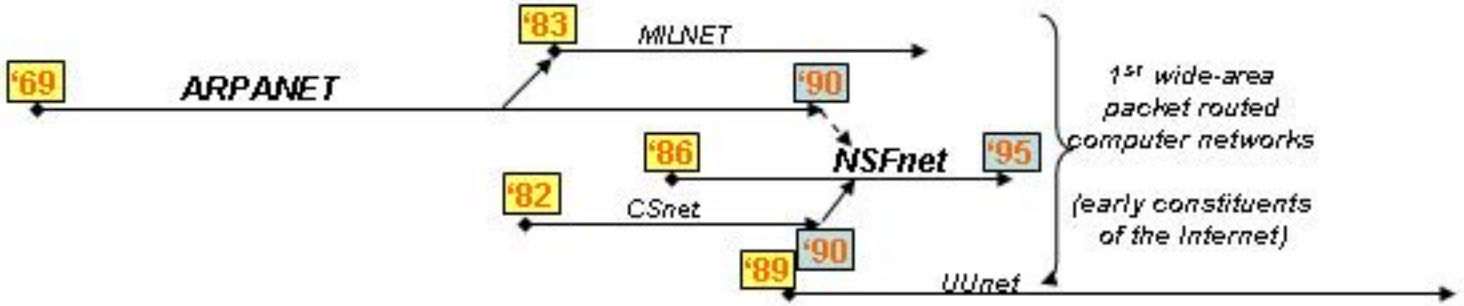
History

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Internet

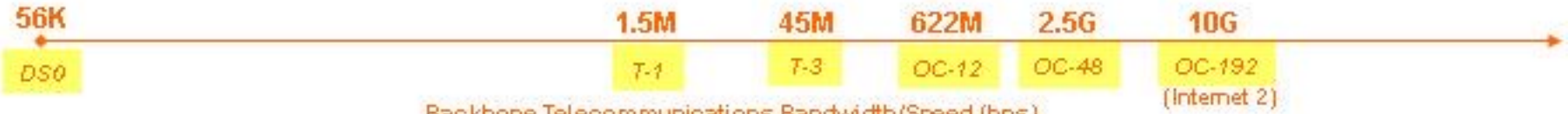
INTERNET TIMELINE





Number of Computers Connected (Users)

100M web sites



Backbone Telecommunications Bandwidth/Speed (bps)

WAN-Internet

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

- Internet born on Oct. 29, 1969 as ARPANET (in a UCLA lab)
- Initially 4 nodes (UCLA, SRI, UCSB, Utah)

- Architecture is "store & forward packet-switching" (routing)
- Topology is "mesh" (a regular network)
- Protocol-based (Invented by Dr. Robert Kahn & Dr. Vinton Cerf)
  - TCP (1973)
  - TCP/IP v3 (1977)
  - TCP/IP v6 (2000 → 2012)

Kahn & Cerf

- Became "The Internet" on Jan. 1, 1983 (all nets adopted TCP/IPv3)

- Domain names.TLD (.org, .edu, .gov, .net, .mil; then .com in 1990)
  - Uses DNS (Domain Name Servers) network (1984)
  - ICANN registers domains (URLs) & assigns IP addresses
    (located in Marina del Rey/Playa Vista)

- Addressing is 32-bit (IPv3, v4) & now 128-bit (IPv6) also

- Fiber optic cables over ***SONET/SDH***

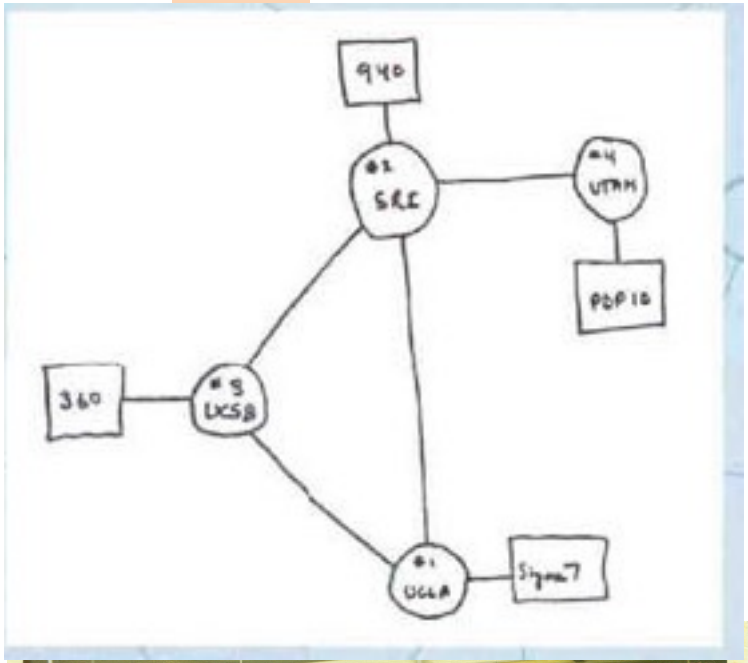
ARPANET

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

1969

1st Four Nodes

1. UCLA
2. SRI
3. UCSB
4. Utah



ARPANET

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

1969

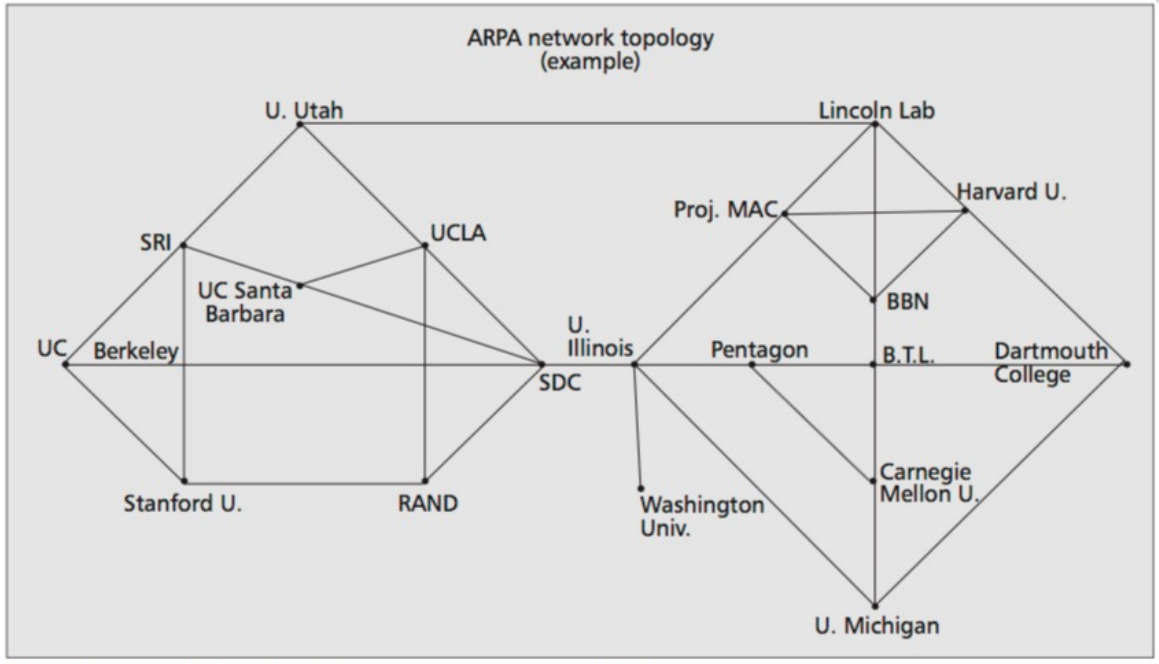


Figure 1. *19-node ARPANET as shown in the original RFQ.*

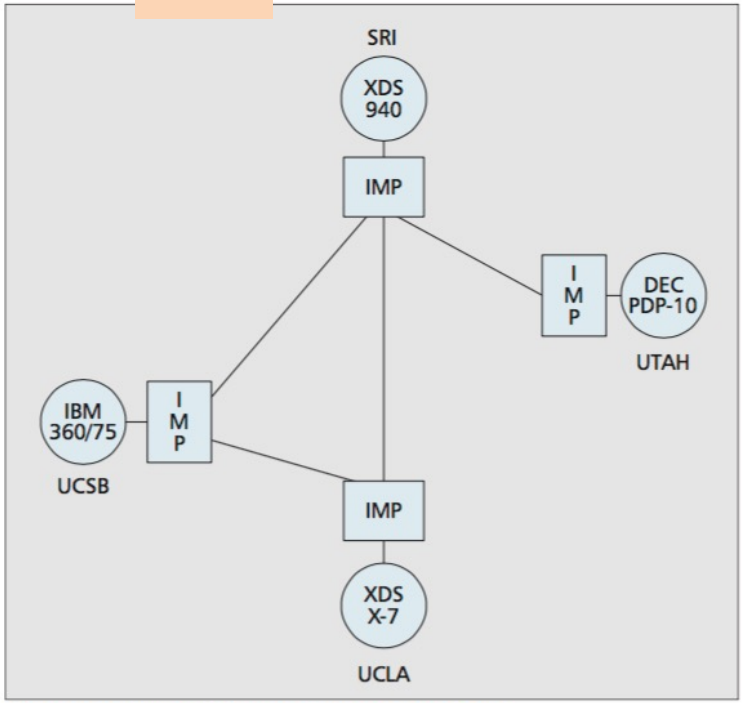
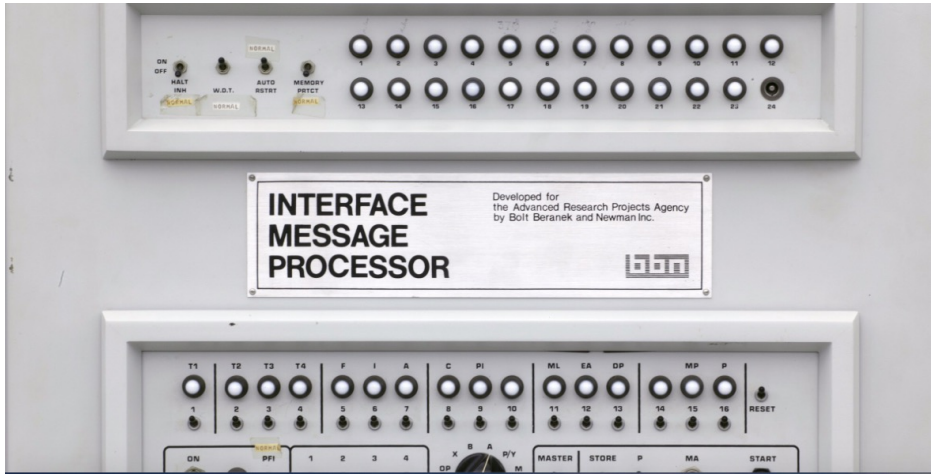
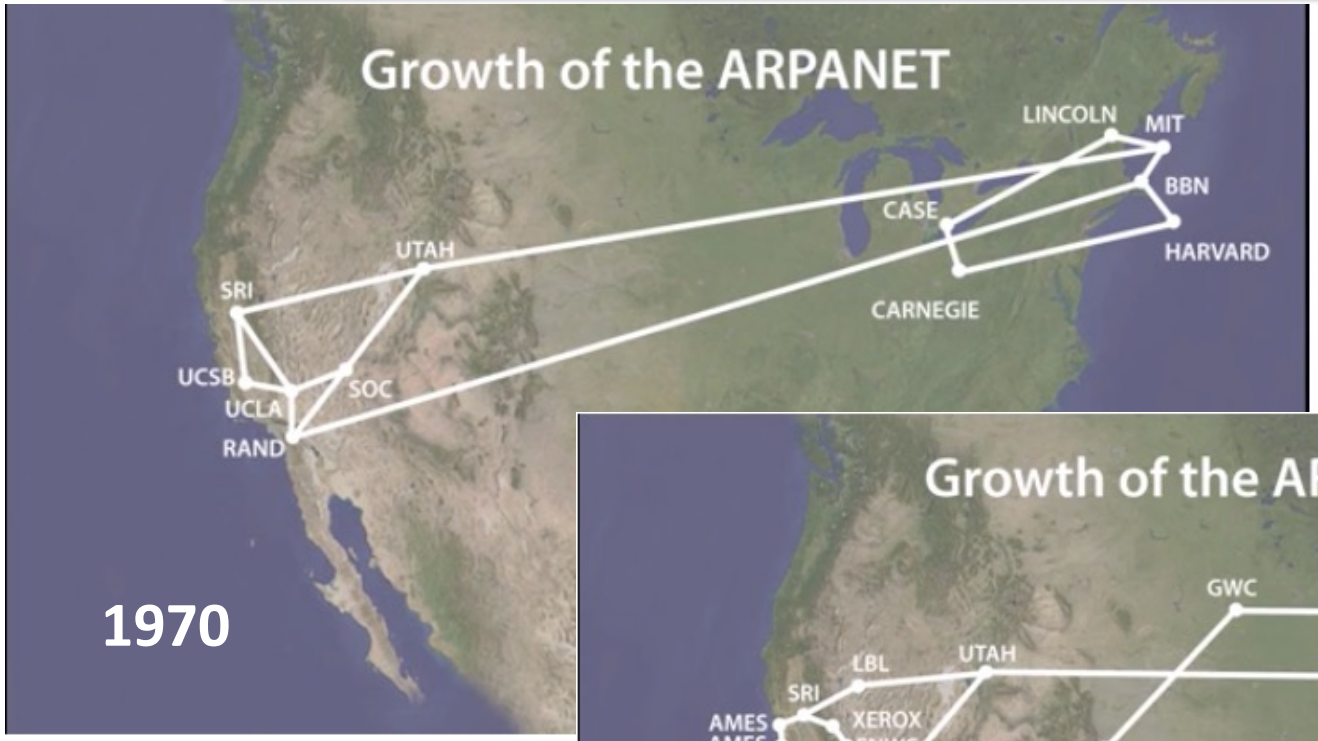


Figure 2. *The initial four-node ARPANET (1969).*

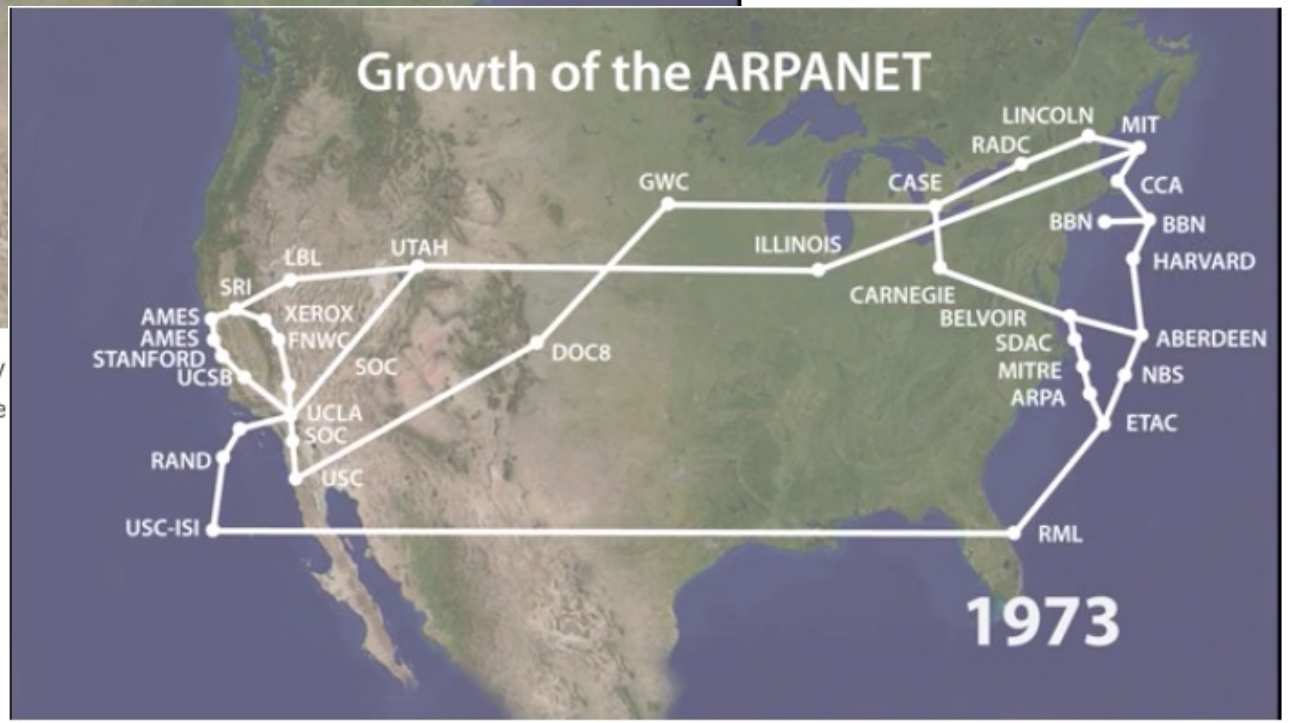


ARPANET

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



The Computer History Museum honored Larry
communications and for his role in the

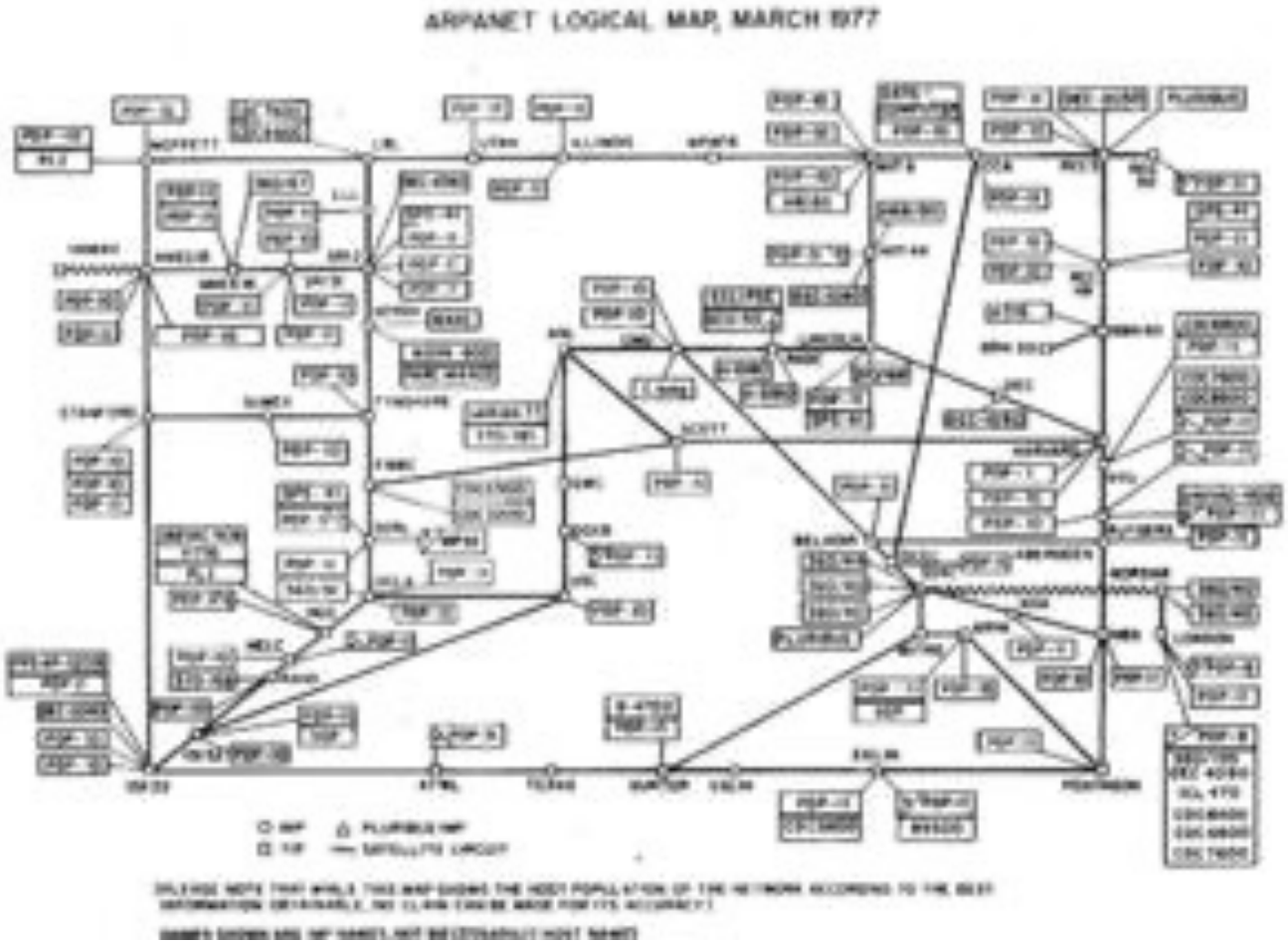


The Computer History Museum honored Larry Roberts in 2017 for his contributions to human and machine communications and for his role in the development of the ARPANET and the X.25 protocol.

ARPANET

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

1977

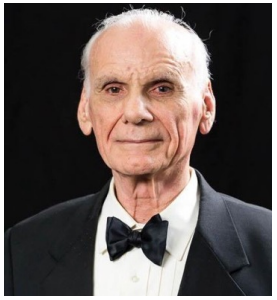


Internet

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

INTERNET
- ARPA
  - IPTO
    - ARPANET

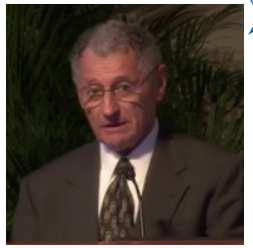
- 4 "Fathers"
  - **ARPA**



- **Dr. Lawrence Roberts**
  - ARPA Chief Scientist/Dir IPTO
  - Issued ARPANET RFP
- **Dr. Robert Kahn**
  - ARPA Director of IPTO
  - Co-invented TCP/IP
  - Primary architect of the IMP

Kleinrock: *IPTO (the Information Processing Techniques Office) was an office within ARPA. IPTO supported the computer and communications research projects of ARPA. It was that office that funded the ARPANET. It was formed in 1962 with JCR Lickliter as the first Director of IPTO, followed in 1964 by Ivan Sutherland, followed by Robert Taylor in 1966 and then Larry Roberts in 1969.*

  - **UCLA**



- **Dr. Leonard Kleinrock**
  - ARPANET Principal Investigator at UCLA
  - Ran the UCLA "Network Measurement" lab
  - Created mathematical theory of Packet Networks

- **Dr. Vinton Cerf**
  - UCLA PhD, worked with Kleinrock
  - Co-invented TCP/IP
  - Founded IAB, President ISOC
  - Guided 1st Internet email at MCI

Cerf: *Kahn was brought in as program manager after Larry's departure and promoted to deputy director of IPTO under Col David Russell. When Russell retired from the US Army in 1979, Bob became Director IPTO and stayed in that post until some time in 1985.*

Internet History-Long View

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

UCLA Prof. Leonard Kleinrock

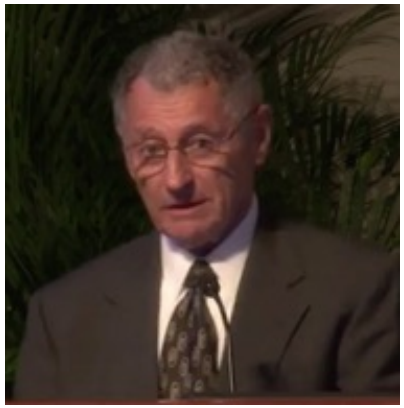
always interesting to listen to UCLA Prof. Len Kleinrock describe the historical background and significance of the Internet and its birth as ARPANET at UCLA 45 years ago. Len calls the Internet an "invisible nervous system" for humanity. He also addresses the "dark side" of security breaches as being underestimated by the Internet developers. here is his **10 min** speech to NAE. Vint Cerf calls this, "A good 10 minute capsule of 40 years of development for the NAE's 50th anniversary."

https://www.youtube.com/watch?v=H8GbcN7m984

Kleinrock's Internet

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

INTERNET



INVISIBLE

NERVOUS SYSTEM FOR HUMANITY

UBIQUITOUS

everywhere

NOMADIC

mobile

HISTORY OF COMMUNICATIONS
EDITED BY MISCHA SCHWARTZ

AN EARLY HISTORY OF THE INTERNET
LEONARD KLEINROCK

http://www.lk.cs.ucla.edu/data/files/Kleinrock/An%20Early%20History%20Of%20The%20Internet.pdf

Internet History-Day 1

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

UCLA Prof. Leonard Kleinrock

always interesting to listen to UCLA Prof. Len Kleinrock describe the historical background and significance of the Internet and its birth as ARPANET at UCLA 45 years ago. Len calls the Internet an "invisible nervous system" for humanity. He also addresses the "dark side" of security breaches as being underestimated by the Internet developers. here is his **7.5 min** speech on the birth of the ARPANET at UCLA in October 1969.

https://www.youtube.com/watch?v=vuiBTJZfeo8

Internet History-IMP

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

UCLA's Leonard Kleinrock displays Internet's first router

UCLA

UCLA Subscribe 33,520

123,126

Add to Share More 352 5

Uploaded on Jan 13, 2009
Internet pioneer and UCLA computer science professor Leonard Kleinrock displays the Internet's first router, or "switch" -- known as an Interface Message Processor -- and describes the process of connecting it with UCLA's host computer, leading to the first-ever Internet message sent on October 29, 1969.

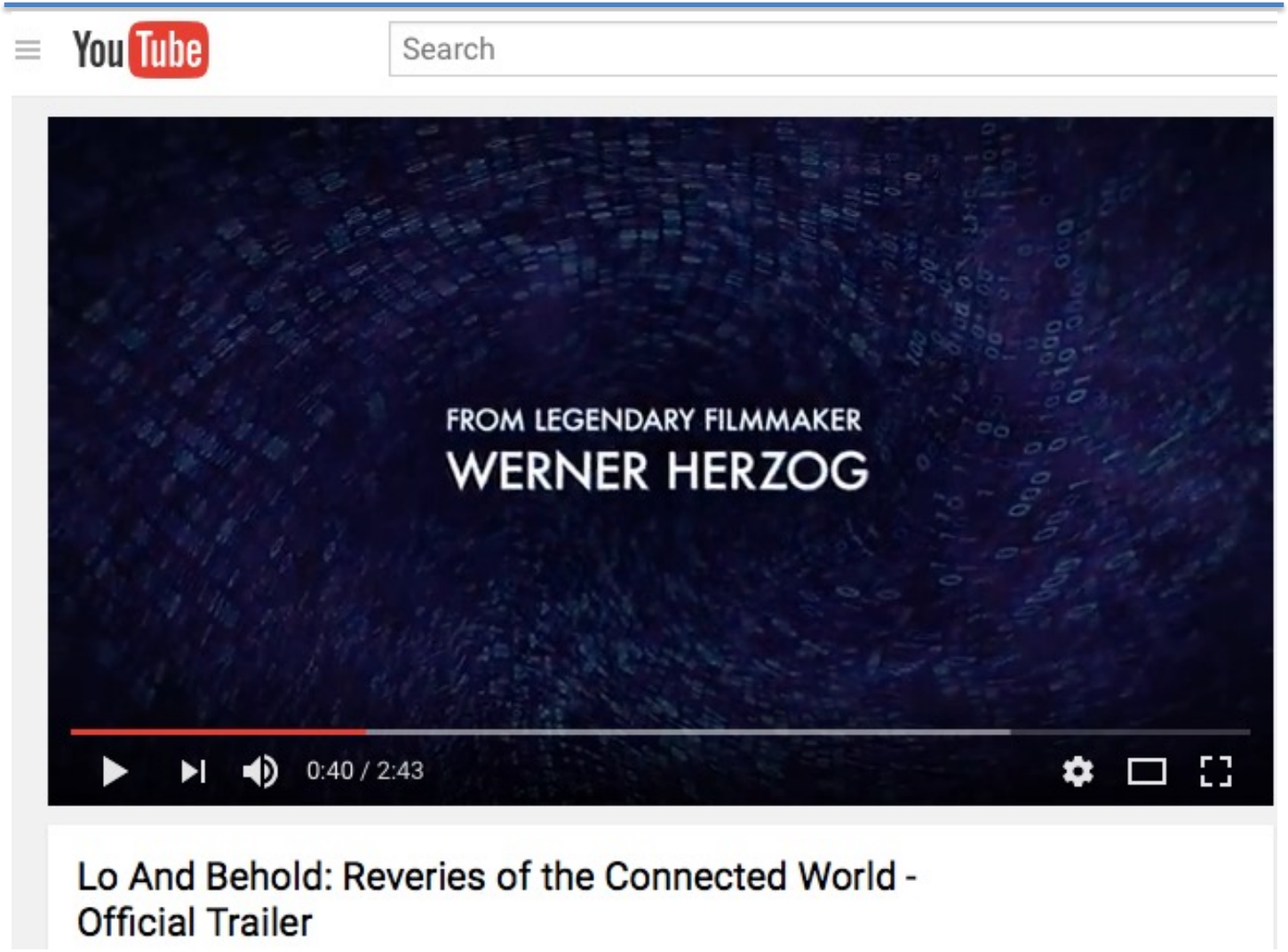
Always interesting to listen to UCLA Prof. Len Kleinrock. Here he shows us and describes the first "packet switch", built from a Honeywell computer by BBN, installed at UCLA first in Sept. 1969, in a lab in BH3420 (where it still resides) – called the "Interface Message Processor" or "IMP".
here is his **4 min** speech on the IMP.

https://www.youtube.com/watch?v=yU9oMOcRsuE

UCLA Prof. Leonard Kleinrock

Documentary Trailer

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



https://www.youtube.com/watch?v=Zc1tZ8JsZvg&feature=share

Internet Birthdays

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

2019

50th

On this day 50 years ago.

It's been 49 years since Leonard Kleinrock's team at UCLA sent the first-ever message across an innovative network from Boelter Hall on October 29th, 1969.

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

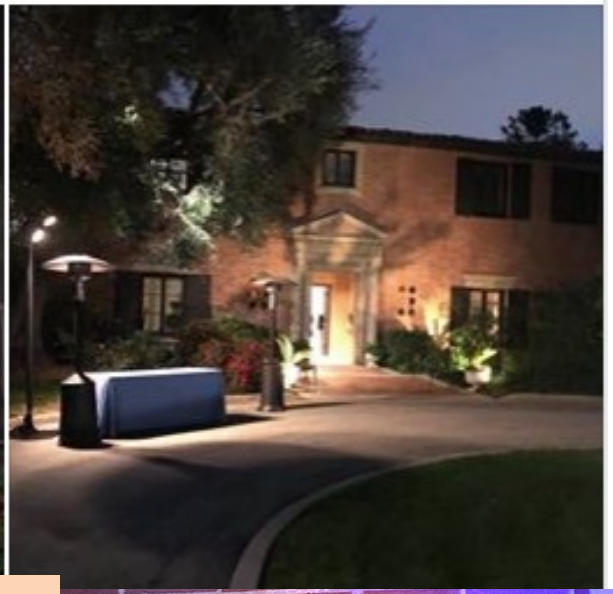
UCLA News

UCLA Samueli School of Engineering

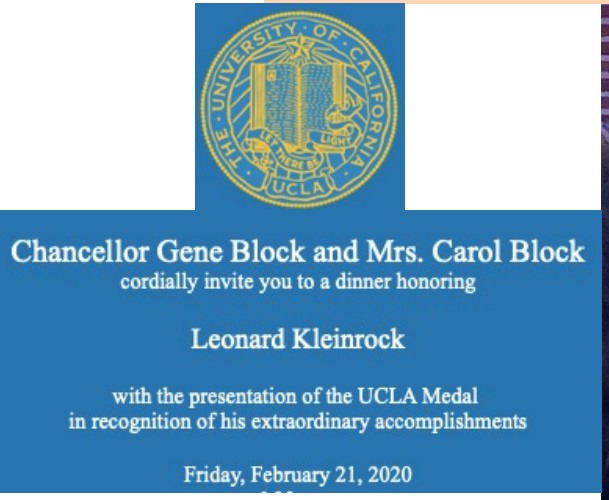
”See you in 50 years!”

Internet 50 – UCLA Medal

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Chancellor Block



Prof. Leonard Kleinrock

Internet Hall of Fame

Computer Science Professors Varghese and Zhang Inducted into Internet Hall of Fame

On Dec. 14, UCLA computer science professors George Varghese and Lixia Zhang were inducted into the Internet Hall of Fame's 2021 Class. The honor "recognizes individuals worldwide who have played an extraordinary role in the conceptualization, building, and development of the global Internet and recognizes those who have made crucial, behind-the-scenes contributions."

Internet Hall of Fame

21 Individuals from 11 Countries Form 2021 Inductee Class

December 14, 2021

Twenty-one pioneering individuals who fundamentally changed the world through their work building and developing the global Internet have been inducted into the Internet Hall of Fame. These engineers, physicists, mathematicians, academics, and others from 11 nations made outstanding contributions to the Internet's global growth, inventing the technologies that launched it, expanding its reach in their own regions and worldwide, and making it more secure, reliable, and accessible for millions.

The Internet they helped create brought the new cohort together in a virtual induction ceremony 14 December 2021. They logged on from points around the world to share the honor with their colleagues, about whom Internet Society President, Andrew Sullivan, noted: "Their contributions made it possible for us to look forward to our future, inextricably tied to the open, globally-connected, secure, and trustworthy Internet, and its ability to connect us reliably and consistently."

Internet Hall of Fame

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Live Panel Interview 2020

Making the Most of Online Learning — an Audience with Internet Pioneers

Amid the pandemic, UCLA computer science professor George Varghese found a creative way to enhance the undergraduate computer science networks course he taught in the fall of 2020 with five 40-minute live Zoom interview sessions with scientists who brought the internet to life, including UCLA Distinguished Professor Leonard Kleinrock, who directed the transmission of the first internet message from 3420 Boelter Hall to the Stanford Research Institute.

Internet

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

- Internet orgs
  - IAB (1979)
  - IETF (1986)
  - IESG
  - ISOC (1992)
  - ICANN (1998)
  - IANA

Cerf: *Regarding **IAB**, I created the **Internet Configuration Control Board** around 1979 and put on that board many of the leads for the Internet research project. Upon my departure from ARPA in 1982, I made David Clark the chairman and Chief Internet Architect and Jon Postel the Deputy Chief. When I was replaced at ARPA by Barry Leiner, Barry re-named and restructured ICCB to be the **Internet Activities Board** and left Clark and Postel in place. In 1992, the IAB was renamed the **Internet Architecture Board** as it was relocated to the **Internet Society** which was launched that year and where I served as president until 1995.*

Internet Applications

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

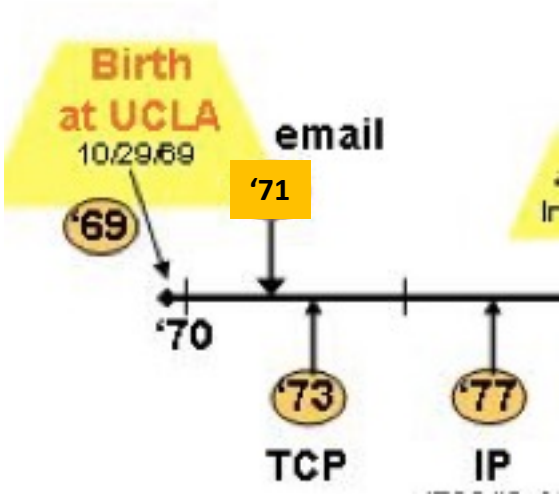
- Applications
  - **Email** (1971)
  - Newsgroups (1974)
  - BBS (1980) → *Blogs*
  - **WWW** (1989/91)
  - VoIP (1998) → *Skype*
  - Streaming (music, video)
  - IoT

Email Invention

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

INTERNET

Ray Tomlinson



BI Julie Bort, Business Insider
Mar. 6, 2016, 11:33 AM 4,112 1

FACEBOOK LINKEDIN TWITTER

Internet Hall of Famer Ray Tomlinson has died.

Tomlinson was the man who basically invented email as we know it today, including making the choice to use the "@" sign in an email address. He was 75.

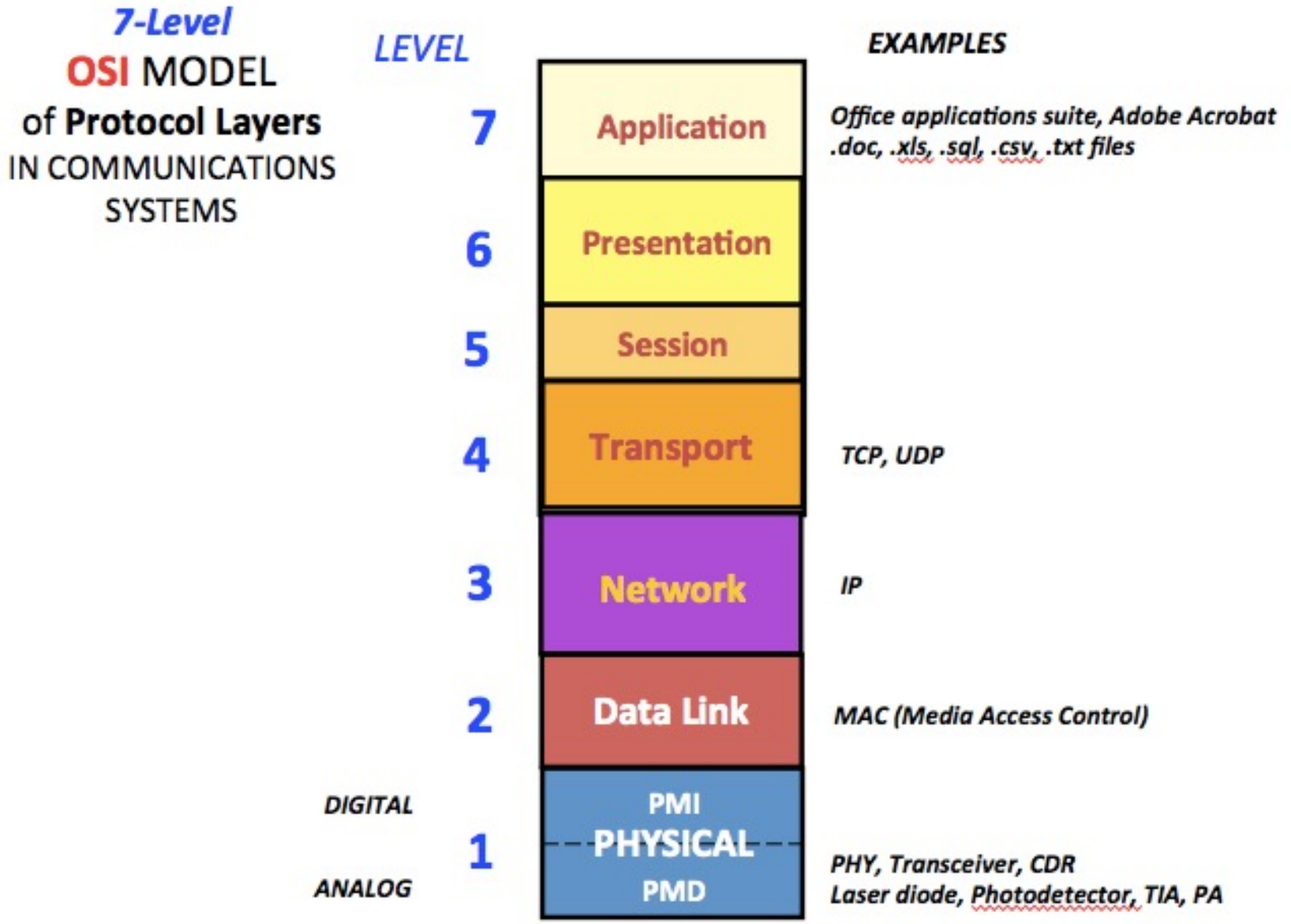
email inventor Ray Tomlinson

Tomlinson invented email, a system where a user on one network could send a message to someone on another network, in 1971.

He proceeded to win many awards over his lifetime for email. But he couldn't say what the first email ever sent actually said.

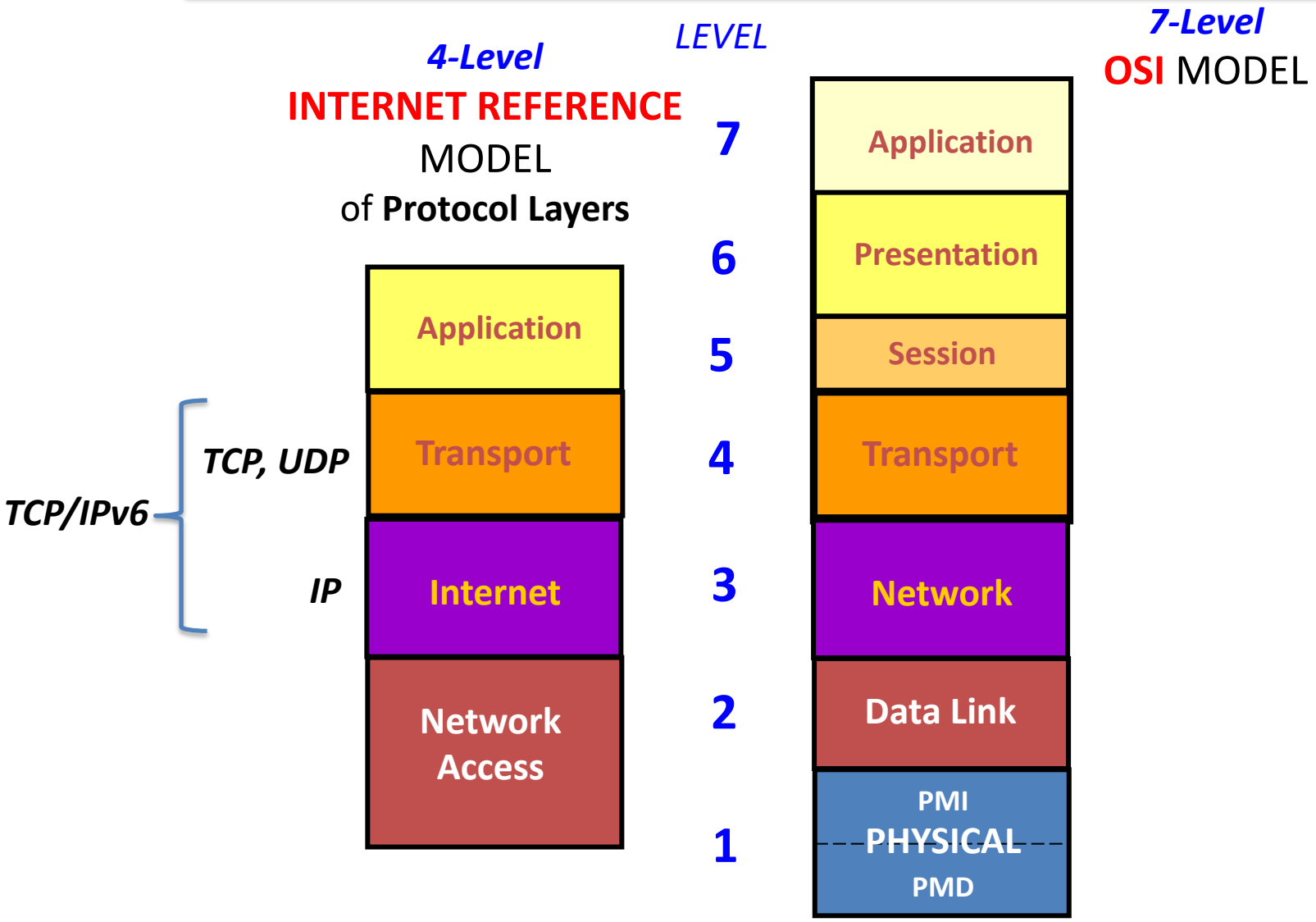
OSI Protocol Layers

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



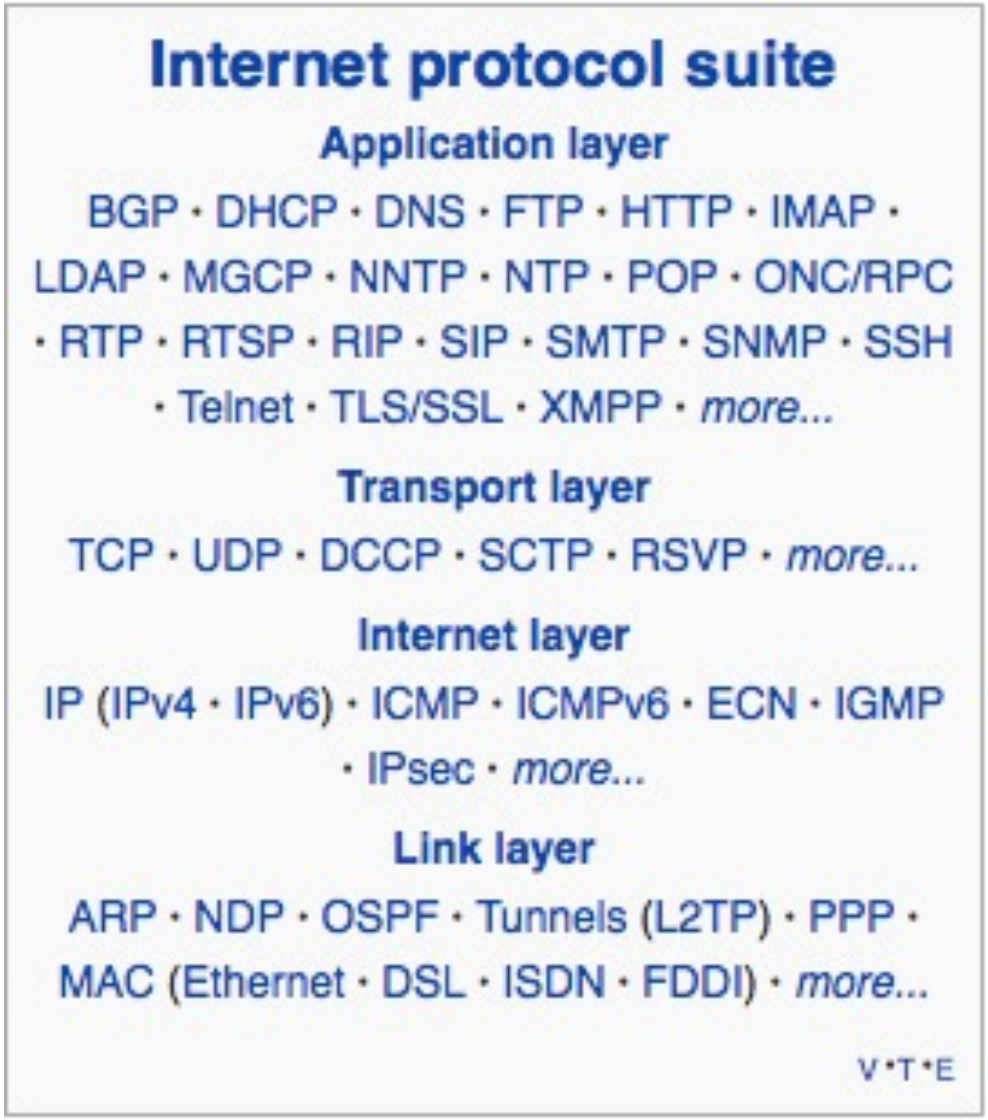
Internet Protocol Layers

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



Internet Protocols

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



WWW

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

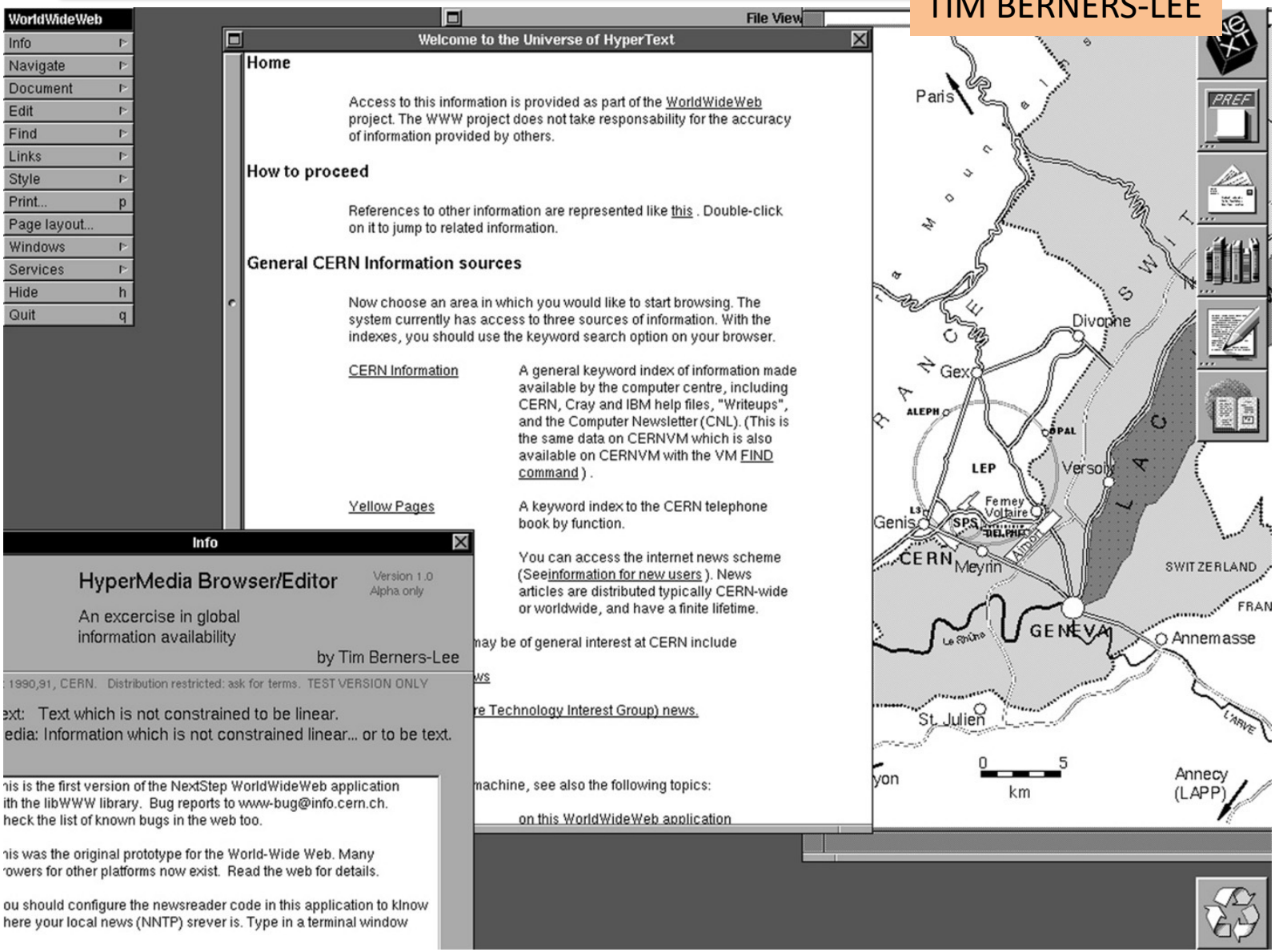
- Invented by Sir Tim Berners-Lee in 1989 (at CERN)
- **HTTP** protocol (Hyper-Text *Transfer* Protocol)
- **HTML** language (Hyper-Text *Markup* Language)
- Browsers – as rendering applications
  - *Mosaic* – invented by Marc Andreeson at U of WI (1992)
  - *Netscape Navigator* – upgrade of Mosaic by Andreeson
  - *Internet Explorer* by Microsoft
  - *Safari* by Apple
  - *Chrome* by Google
  - *Firefox* by Mozilla
- Languages (applications)
  - Java (*functional* on client VM)
  - PHP (*scripting* on server)
  - Javascript (*scripting* on client)
  - MySQL (*database* -– LAMP on server)
- Generations
  - Web 1.0 – static
  - Web 2.0 – dynamic: runs applications (incl. databases)
  - Web 3.0 – semantic: predictive/anticipatory

Tim Berners-Lee

First Website

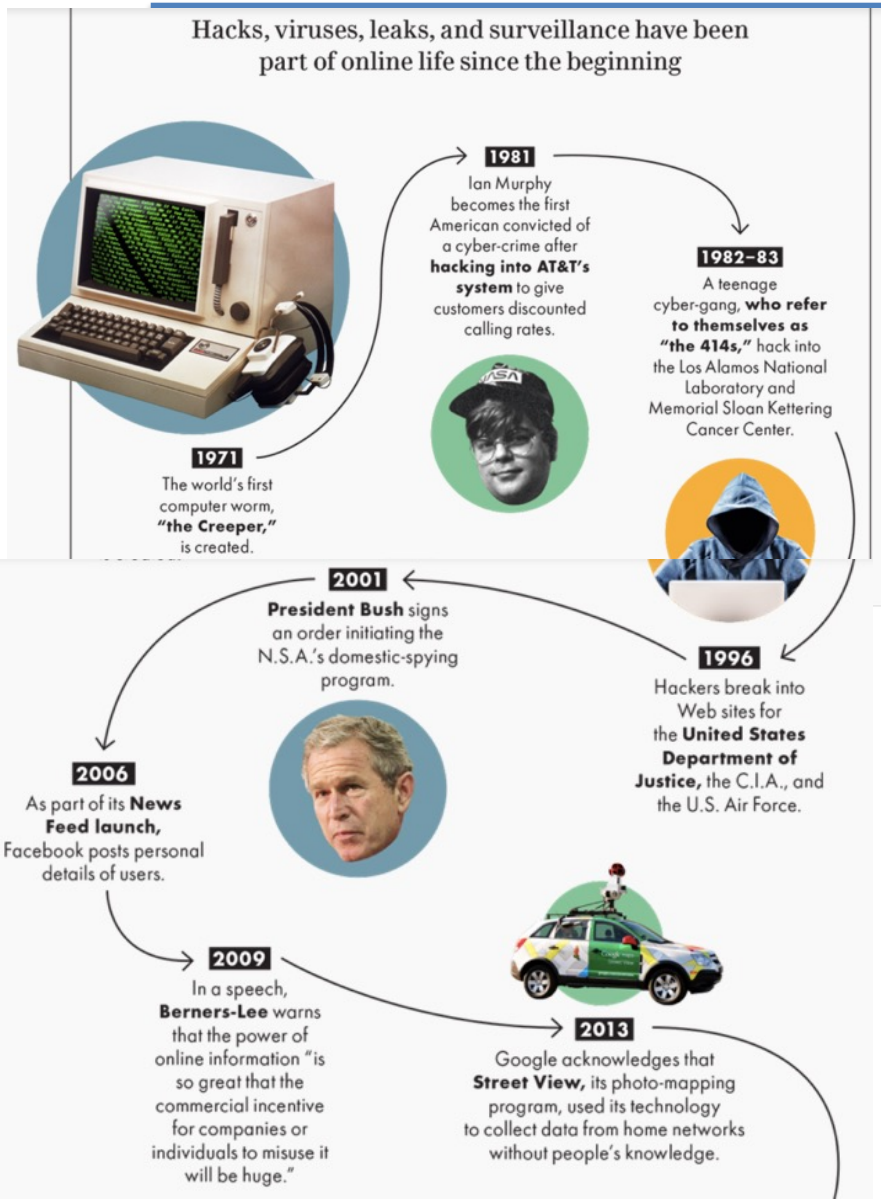
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

TIM BERNERS-LEE



Internet Malware/Hacking

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman



IoT – Wearables

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

History

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Museum

Early ICs

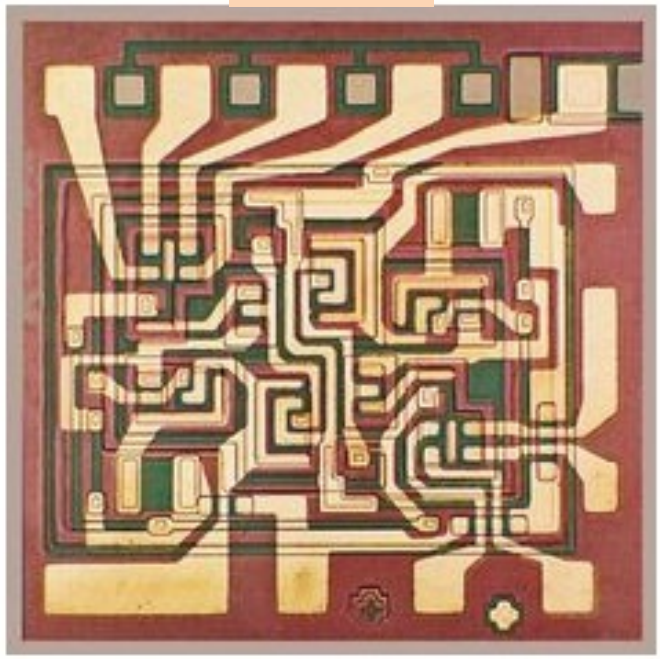
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Fairchild

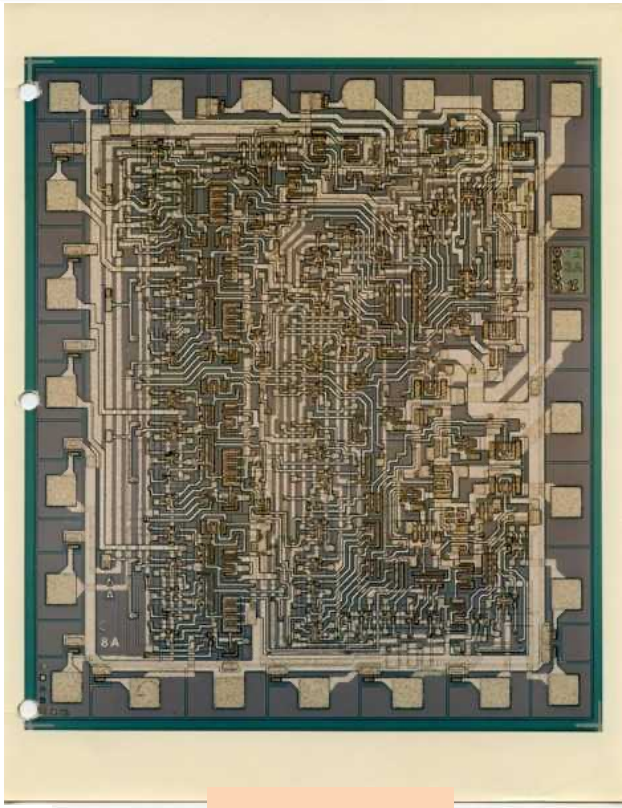
FIRST IC

Fairchild's first IC, the "F" element flip-flop compared to a dime coin. Photo: Getty Images

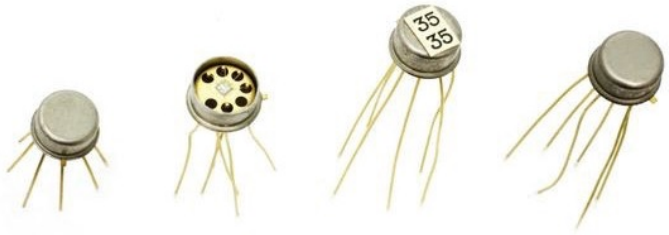
Fairchild "F"
Single flip-flop
(1961)



Fairchild Pheonix TTL gate die (1964)
Courtesy: Fairchild Semiconductor International, Inc.



F9340 MSI



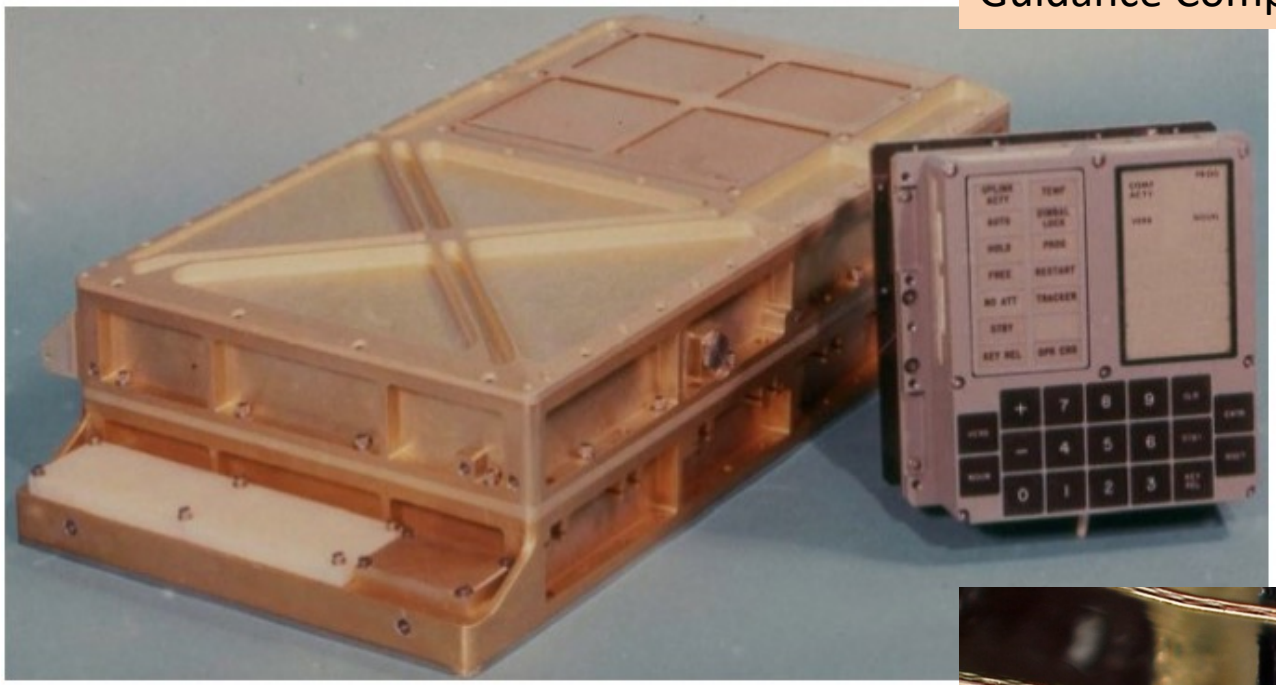
Fairchild "G" – Gate

Fairchild "G" Element (3-input NOR gate) IC in metal can package. Photo: © Mark Richards/Computer History Museum

Apollo Computer/Chips

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

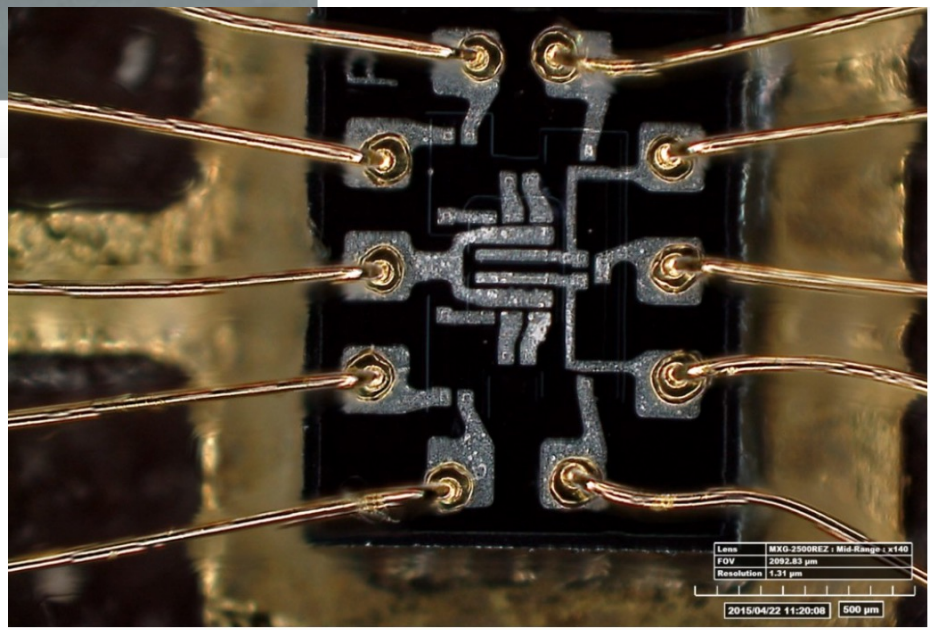
Guidance Computer (AGC)



AGC case and keyboard. Photo: NASA

Ca. 1965

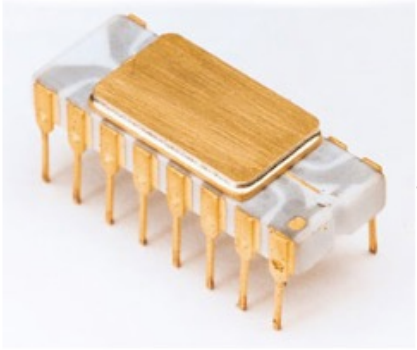
AGC chip



MPU/MCU Museum

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

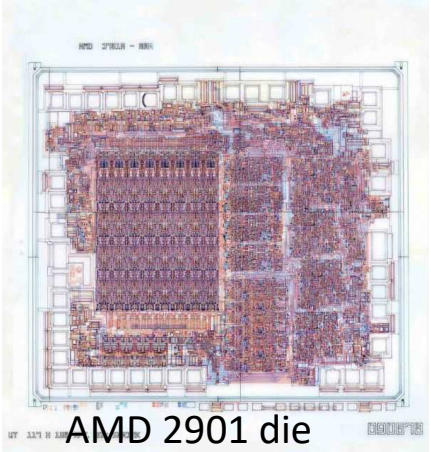
[Nostalgia Dept.]
Intel 4004 is announced, November 15, 1971



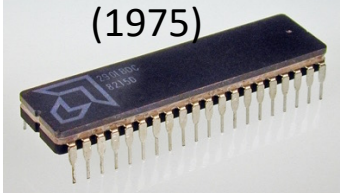
The venerable 16-pin side-brazed DIP.
(Click image to view full size)

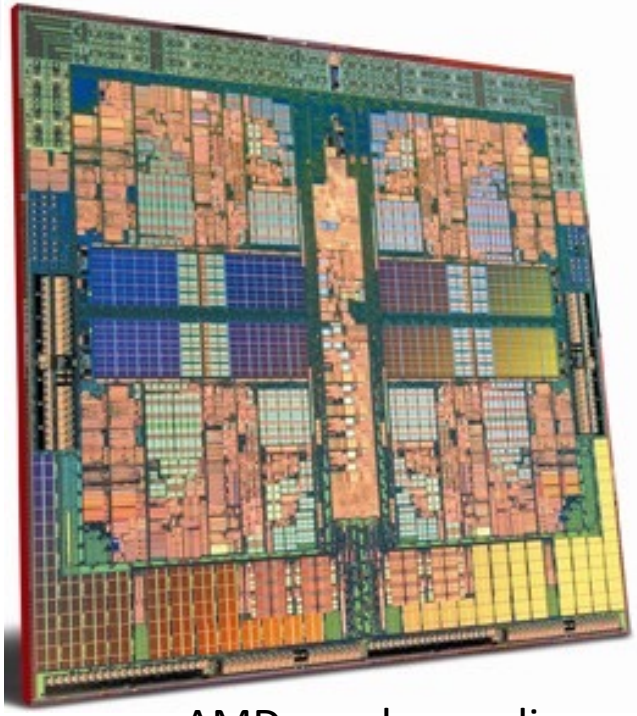
The building-block 4004 CPU held 2300 transistors. The microprocessor, the size of a little fingernail, delivered the same computing power as the first electronic computer built in 1946, which, in contrast, filled a room.

From: EDN Nov 20, 2015
Intel 4004 is announced, November 15, 1971

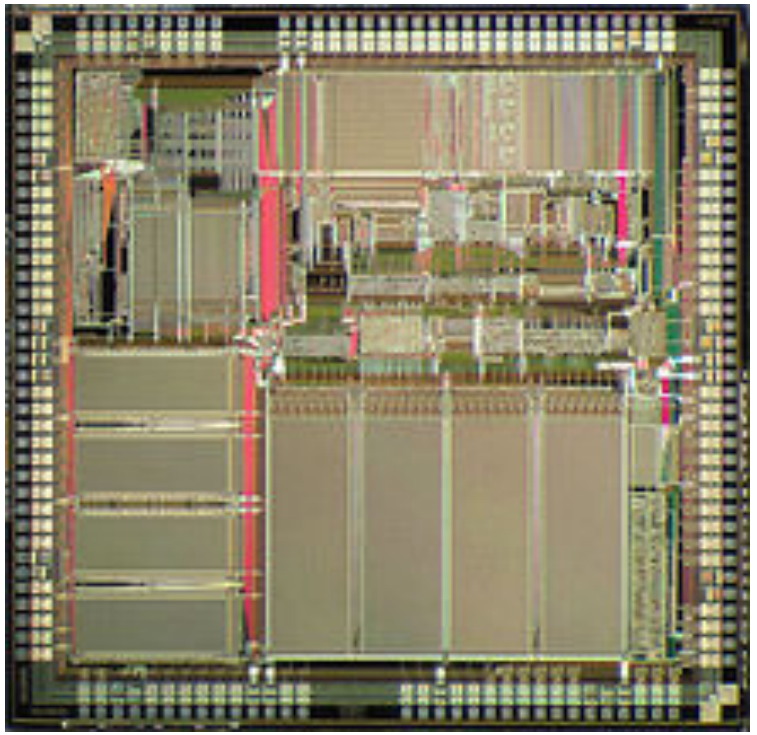


AMD 2901 die (1975)





AMD quad-core die



ARM610 die

Memory Museum

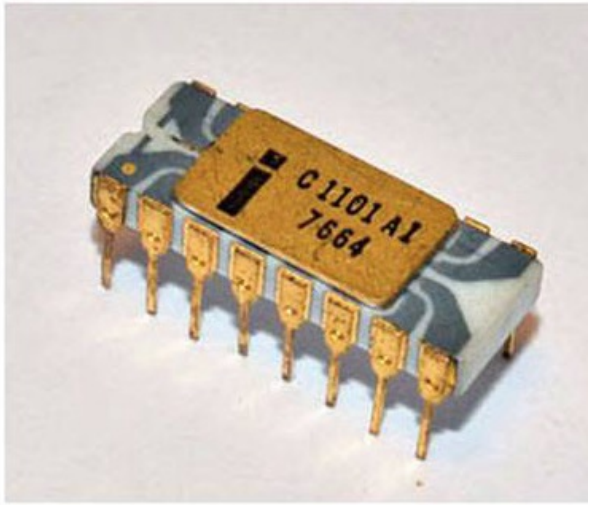
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

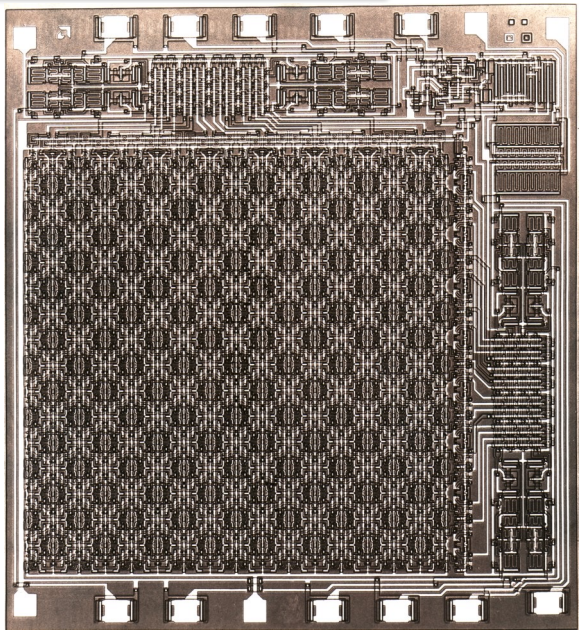
July 1969.

Intel 1101: The first MOS memory chip.

A 256-bit SRAM (Static Random Access Memory).

The 1101 was developed in parallel with the 3101, but lost the race to be Intel's first product. It was produced with Silicon gate MOS (Metal Oxide Semiconductor) technology, which gave Intel the edge it needed to produce high density memories.





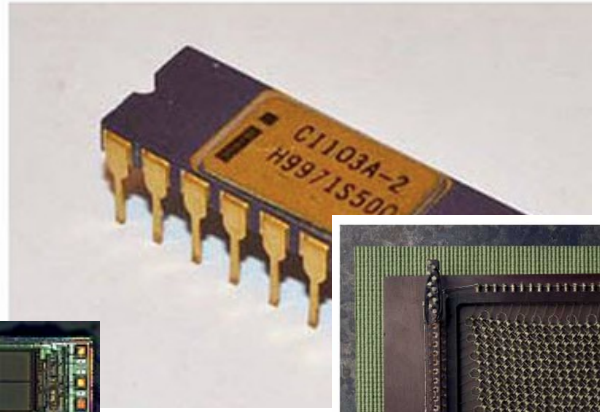
Am1101A

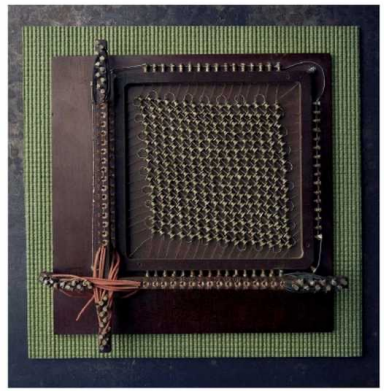
October 1970.

Intel 1103: The first DRAM memory chip.

A 1024-bit DRAM (Dynamic Random Access Memory).

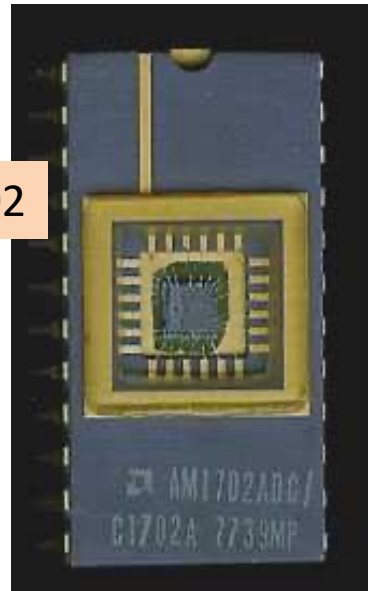
This is the chip that kicked magnetic Core Memory out of the game, making Intel a world leader in memories for a decade.

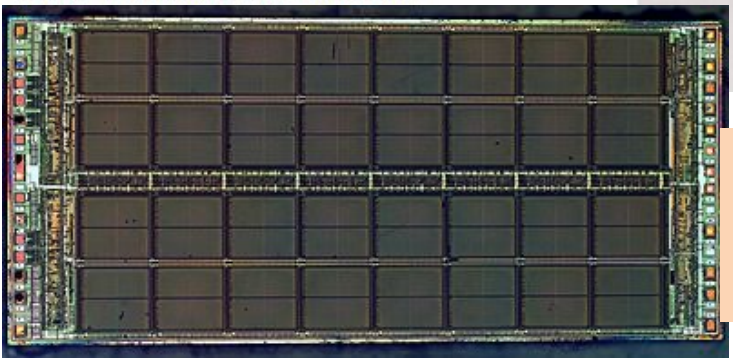




256-bit magnetic core memory (c. 1952) Slow data retrieval and storage speeds limited the utility of early computers. RCA researcher Jan Rajchman's solution was a memory array consisting of a wire matrix with doughnut-shaped magnetic cores at each intersection. By applying a current to a given set of horizontal and vertical wires, you could select a specific core and quickly change the direction of its magnetic field.

Am1702



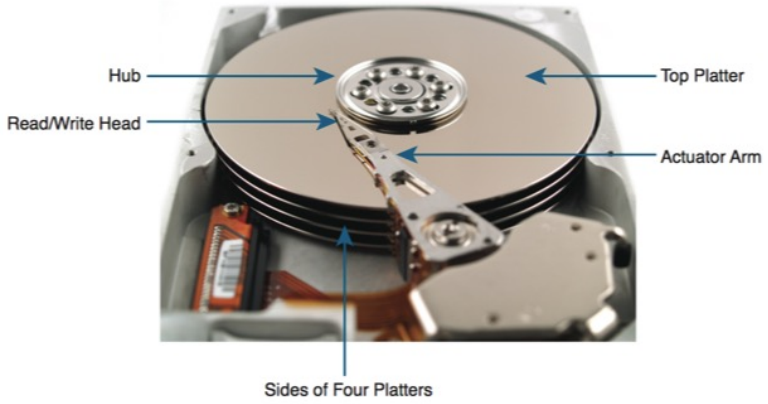


MT4C
1Mb
DRAM

Disk Storage

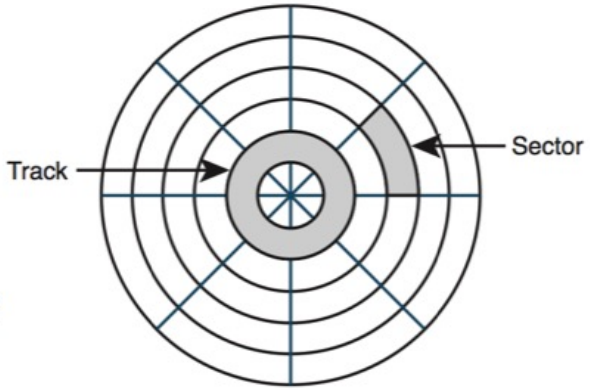
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Hard Disks vs. Floppy Disks (cont.)



Writing Data to Sectors, Tracks

A platter has many locations that can hold magnetic charges. Physically, these locations exist in concentric circles, with each circle called a ***track****. A* ***sector*** *refers to a subset of a track, as shown in the figure.*

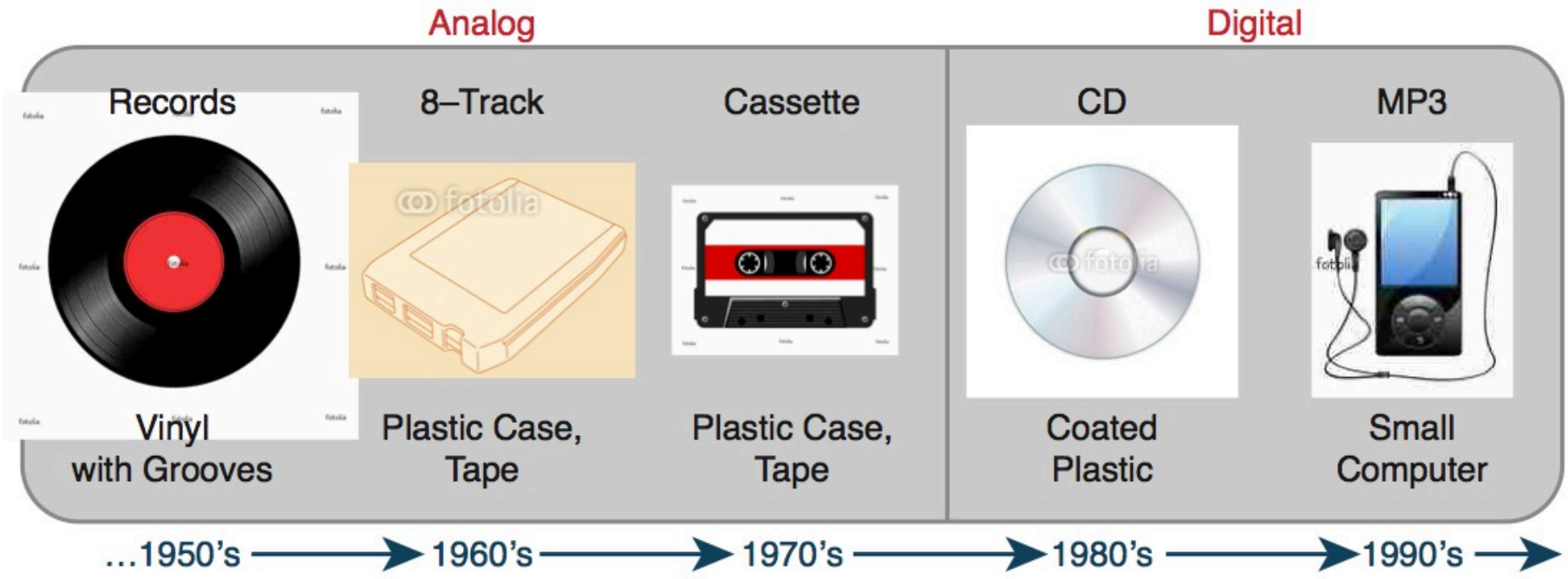


Tracks and Sectors in a Single Disk Drive Platter *Figure 1-15*

Music Storage

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

General Timeline of Commercial Music/Audio Products



Data

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Numbers
&
Codes

Ordinals

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Technical ordinals

10^(-24)	yacto
10^(-21)	zepto
10^(-18)	atto
10^(-15)	femto
10^(-12)	pico
10^(-9)	nano
10^(-6)	micro
10^(-3)	milli
10^(-2)	centi
10^(-1)	deci
10^(+1)	deka
10^(+2)	hecto
10^(+3)/2^(10)	kilo
10^(+6)/2^(20)	mega
10^(+9)/2^(30)	giga
10^(+12)/2^(40)	tera
10^(+15)/2^(50)	peta
10^(+18)/2^(60)	exa
10^(+21)/2^(70)	zetta
10^(+24)/2^(80)	yotta
10^(29)/2^(100)	**geo**

Gazillions

10^(+6) million
10^(+9) billion
10^(+12) trillion
10^(+15) quadrillion
10^(+18) quintillion
10^(+21) sexillion
10^(+24) septillion
10^(+27) octillion
10^(+30) nonillion
10^(+33) decillion
10^(+36) undecillion
10^(+39) duodecillion
10^(+42) tredecillion
10^(+45) quattuordecillion
10^(+48) quindecillion
10^(+51) sexdecillion
10^(+54) septendecillion
10^(+57) octodecillion
10^(+60) novemdecillion
10^(+63) vigintillion
10^(+100) googol
10^(+303) centillion
10^(10^(+100))
googolplex

Ordinal	Power of 2	Power of 10	Actual
1K	2^{10}	10^{3}	1024
1M	2^{20}	10^{6}	1,048,576
1G	2^{30}	10^{9}	1.074×10^{9}
1T	2^{40}	10^{12}	1.0995×10^{12}

Name	2^n	M/G	Actual
byte	2^{8}	--	256
short	2^{16}	64K	65,536
integer	2^{32}	4B	4.3×10^{9}
long	2^{64}	16 Q	1.84×10^{19}
IPv6	2^{128}	340 uD	3.4×10^{38}

Number Codes

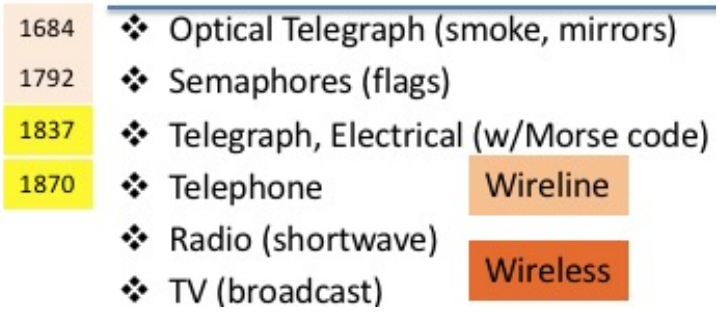
DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

❖Invented/Artificial

- ❑ Signaling
  - ➢ Smoke signals
  - ➢ Drums
  - ➢ Semaphores
- ❑ Communications
  - ➢ Morse code
  - ➢ Hollerith code (punch cards)
  - ➢ Paper tape codes
  - ➢ Encryption/cypher codes
  - ➢ **ASCII code** (also **EBCDIC**)
  - ➢ **Unicode** (16-bit)
  - ➢ UTF-8

❖Natural

- ❑ DNA – Genetic code
  - ➢ Base-4 {A,C,G,T)
- ❑ Fibonacci sequence
  - ➢ Shell growth
  - ➢ Leaf growth



Telegraph: Morse Code

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Base 2 = {dot, dash}

Each letter is a 1 to 4-bit character

1st Digital Code

1836-1844
by Samuel F.B. Morse et al.

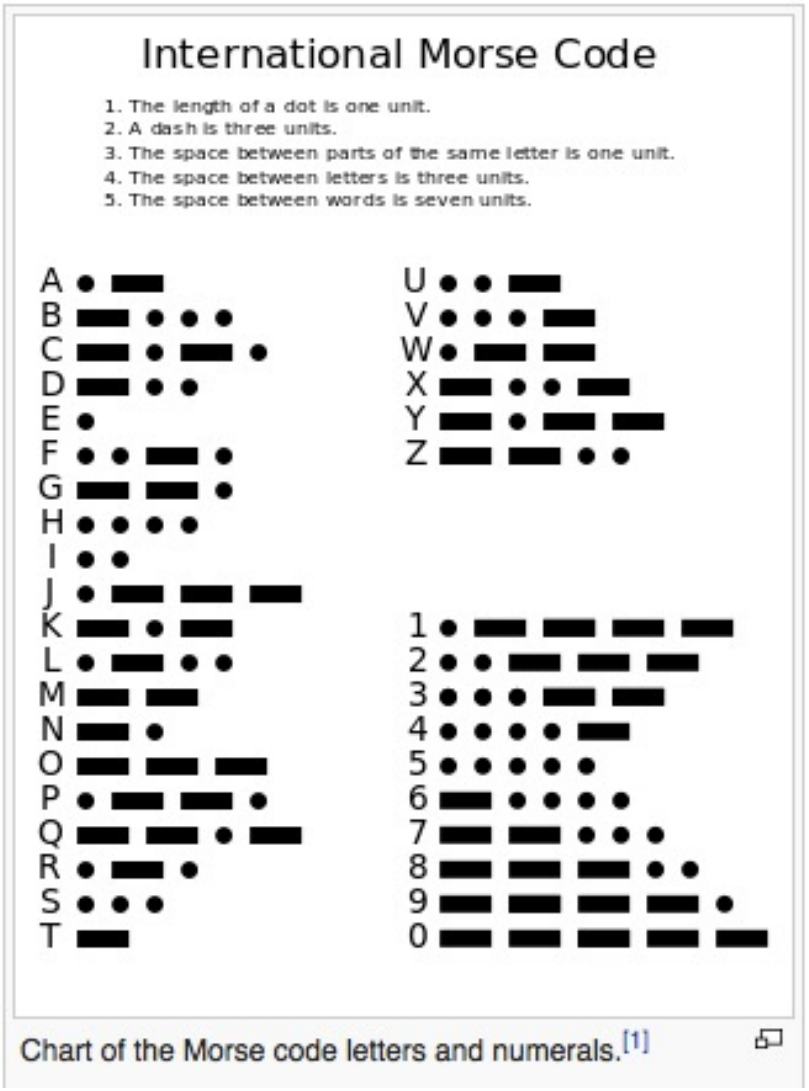


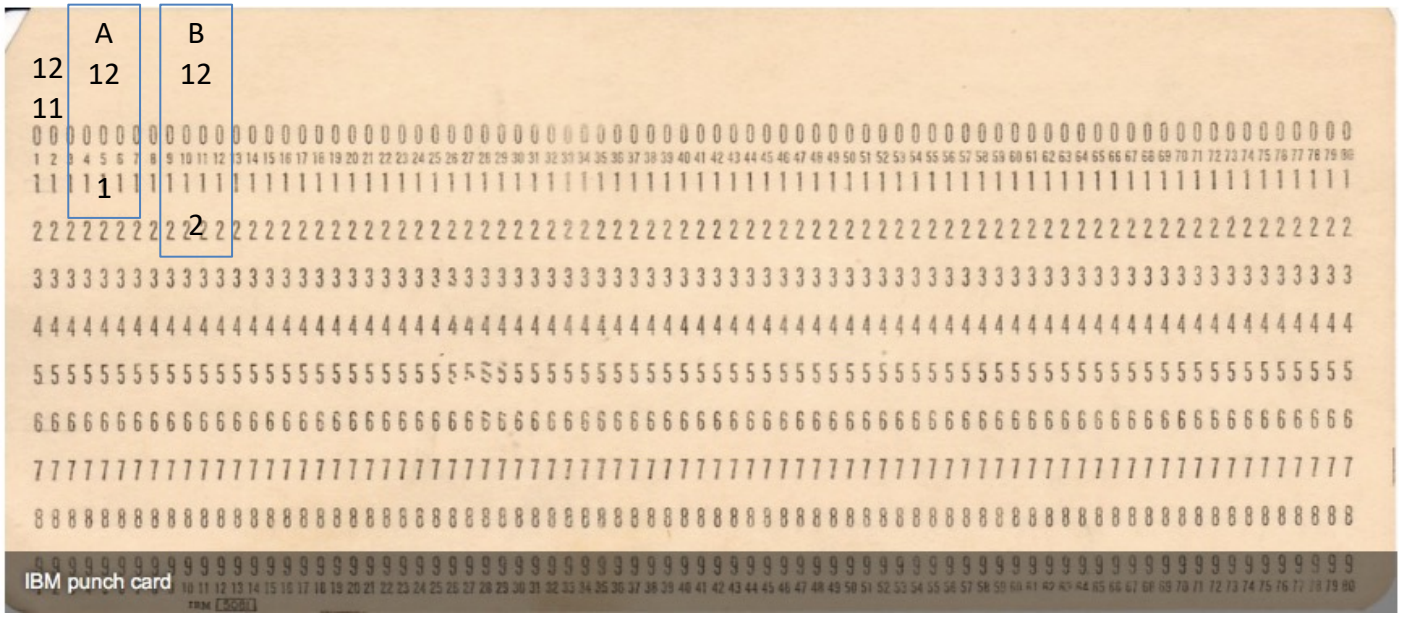
Chart of the Morse code letters and numerals.[1]

A typical "straight key". This U.S. model, known as the J-38, was manufactured in huge quantities during World War II, and remains in widespread use today. In a straight key, the signal is "on" when the knob is pressed, and "off" when it is released. Length and timing of the dots and dashes are entirely controlled by the telegraphist.

Punchcards

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

MODERN VERSION



Invented by Herman Hollerith for 1890 census

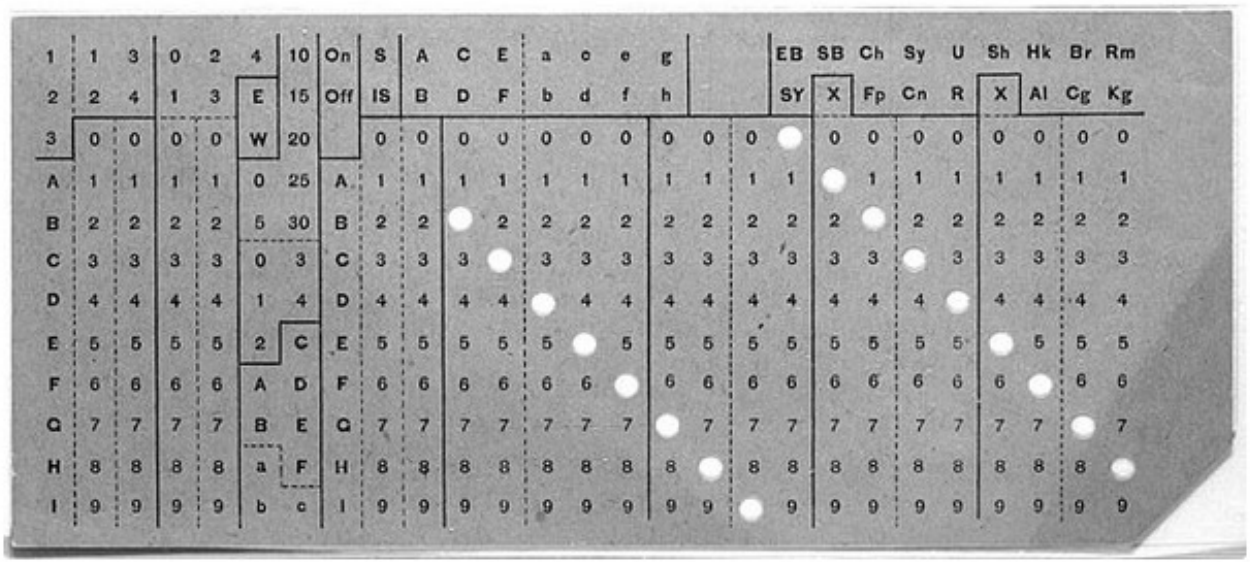
Punchcards

Invented by Herman Hollerith for 1890 census

"finished months ahead of schedule and far under budget." The editors of *The Electrical Engineer* were a bit more expressive when describing the Hollerith census operation in their Nov 11, 1891 edition:

> ***"This apparatus works unerringly as the mills of the gods, but beats them hollow as to speed."***

Hollerith's company and patents were acquired along with three other international business machine firms in 1923 to form the aptly named **International Business Machines**, or as it's known today, **IBM**.



Above: Hollerith punch card used in 1890 Census

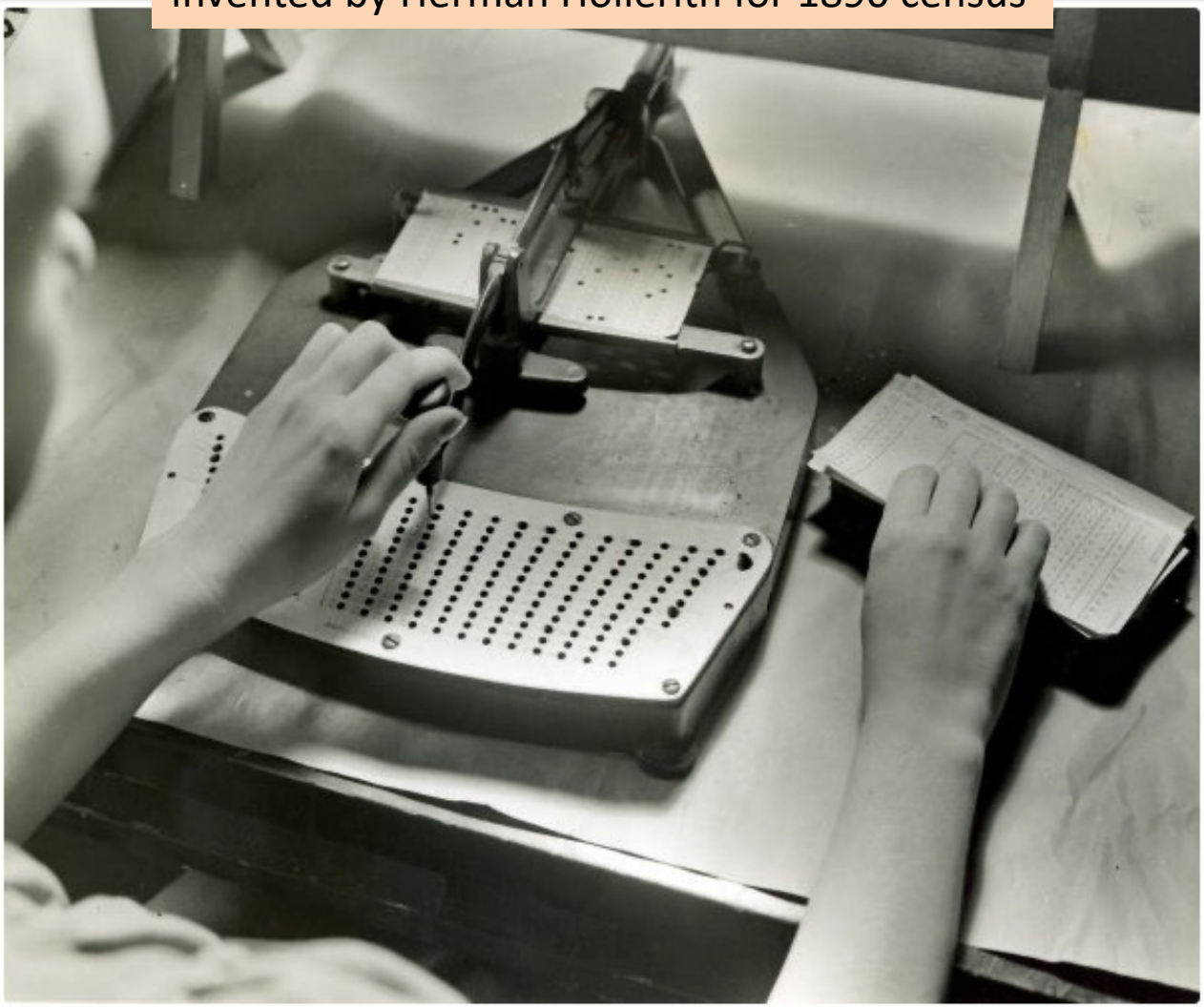
Punchcard Reader

Invented by Herman Hollerith for 1890 census

Using his newly invented punch cards (below) and Hollerith tabulation machines (above), Herman Hollerith's results earned him the contract to process and tabulate 1890 census data. Modified versions of his technology would continue to be used at the Census Bureau until it was replaced by the room-sized generations

Punchcard Reader

Invented by Herman Hollerith for 1890 census



Above: A clerk creates punch cards for the Hollerith Machine using a pantograph. Each
clerk was able to process 500 cards per day.

ASCII Codes

Table 1-3 ASCII Conversion Chart for Letters

Hex	Character	Hex	Character
41	A	61	a
42	B	62	b
43	C	63	c
44	D	64	d
45	E	65	e
46	F	66	f
47	G	67	g
48	H	68	h
49	I	69	i
4a	J	6a	j
4b	K	6b	k
4c	L	6c	l
4d	M	6d	m
4e	N	6e	n
4f	O	6f	o
50	P	70	p

ASCII Codes

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

IBM EBCDIC

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

EBCDIC

Extended **Binary Coded Decimal**

7-bit code used by large comp...
The collating sequence for the two codes is shown as ...

ASCII Code		
Character	*Decimal*	*Hex*
space	32	20
!	33	21
"	34	22
#	35	23
$	36	24
%	37	25
&	38	26
single quote	39	27
(	40	28
)	41	29
*	42	2A
+	43	2B
comma	44	2C
–	45	2D
.	46	2E
/	47	2F
0	48	30
⋮	⋮	⋮
9	57	39
:	58	3A
;	59	3B
<	60	3C
=	61	3D
>	62	3E
?	63	3F
@	64	40
A	65	41
⋮	⋮	⋮
Z	90	5A
a	97	61
⋮	⋮	⋮
z	122	7A

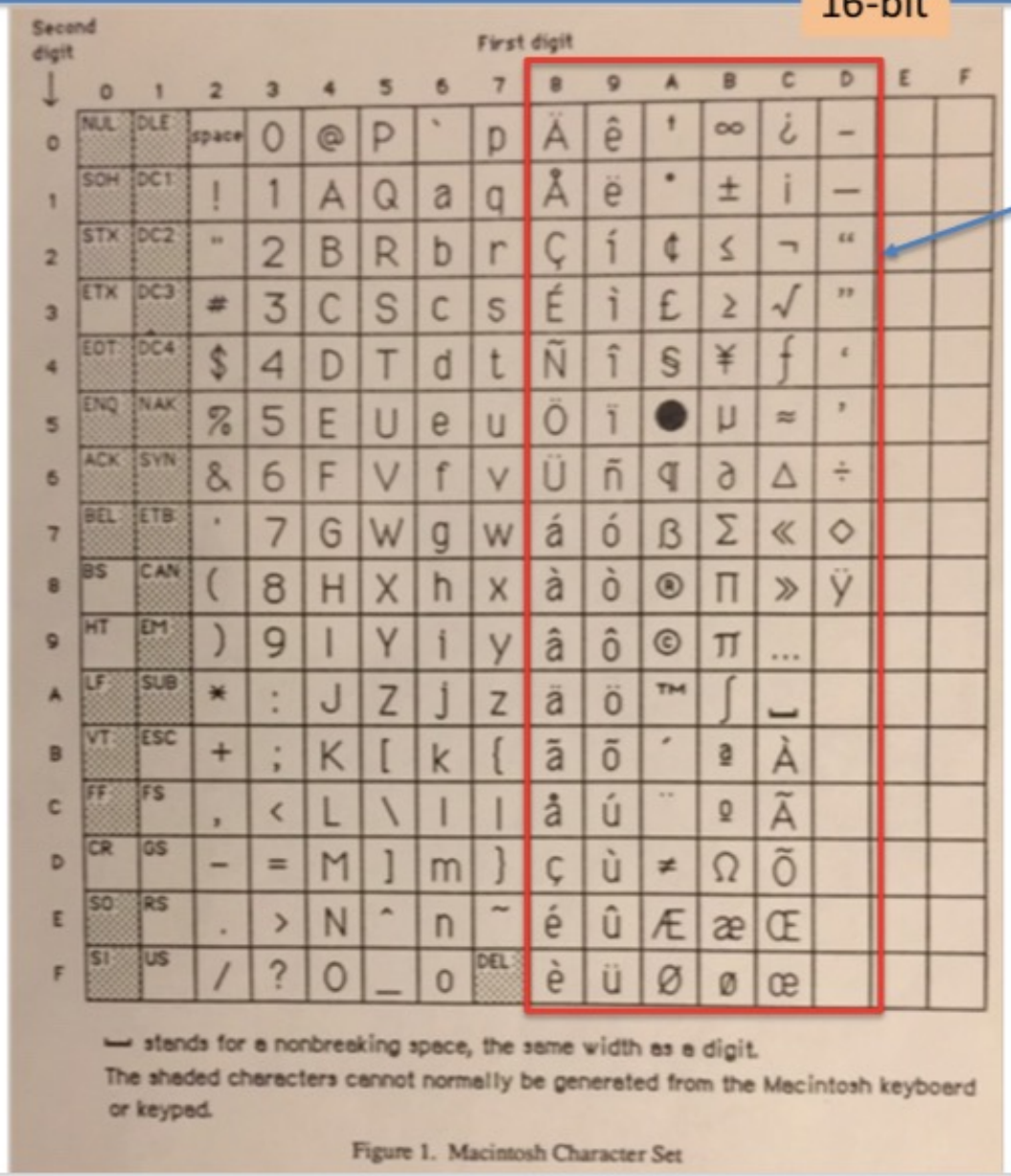
EBCDIC Code		
Character	*Decimal*	*Hex*
space	64	40
.	75	4B
<	76	4C
(	77	4D
+	78	4E
!	79	4F
&	80	50
$	91	5B
*	92	5C
)	93	5D
;	94	5E
minus –	96	60
/	97	61
comma	107	6B
%	108	6C
>	110	6E
?	111	6F
:	122	7A
#	123	7B
@	124	7C
single quote	125	7D
=	126	7E
"	127	7F
a	129	81
b	130	82
⋮	⋮	⋮
z	169	A9
A	193	C1
⋮	⋮	⋮
Z	233	E9
0	240	F0
9	249	F9

CSUN
CALIFORNIA STATE UNIVERSITY NORTHRIDGE
COMP110L

DR JEFF SOFTWARE
INDIE APP DEVELOPER

Old Mac Char Codes

16-bit



␣ stands for a nonbreaking space, the same width as a digit.
The shaded characters cannot normally be generated from the Macintosh keyboard or keypad.

Figure 1. Macintosh Character Set

unique special chars

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Unicode – 16-Bit

UTF-16

- 7 LSB are same codes as for ASCII
- 9 MSB add 2^{16}=65,536 - 128 new codes
- Japanese character sets (漢 字?),
  - *Kanji* uses same 5000 characters as base Chinese
  - *Hiragana/Katakana* uses (ひらがな or 平仮名?) a (カタカナ or 片仮名?).
- Other foreign languages
  - alphabets
  - special characters

(vis-à-vis, *oomlaut*)

Many modern applications can render a substantial subset of the many scripts in Unicode, as demonstrated by this screenshot from the OpenOffice.org application.

Initial repertoire covers these scripts: Arabic, Armenian, Bengali, Bopomofo, Cyrillic, Devanagari, Georgian, Greek and Coptic, Gujarati, Gurmukhi, Hangul, Hebrew, Hiragana, Kannada, Katakana, Lao, Latin, Malayalam, Oriya, Tamil, Telugu, Thai, and Tibetan.[19]

Unicode Transformation Format and Universal Coded Character Set [edit]

Unicode defines two mapping methods: the *Unicode Transformation Format* (UTF) encodings, and the *Universal Coded Character Set* (UCS) encodings. The Unicode codespace is divided into seventeen *planes*, numbered 0 to 16:

v·t·e	Unicode planes and used code point ranges					[hide]
Basic	Supplementary					
Plane 0	Plane 1	Plane 2	Planes 3–13	Plane 14	Planes 15–16	
0000–FFFF	10000–1FFFF	20000–2FFFF	30000–DFFFF	E0000–EFFFF	F0000–10FFFF	
Basic Multilingual Plane	Supplementary Multilingual Plane	Supplementary Ideographic Plane	*unassigned*	Supplementary Special-purpose Plane	Supplementary Private Use Area planes	
BMP	SMP	SIP	—	SSP	SPUA-A/B	

Unicode

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

Upper 128 chars of 8-bit Plane 0

C1 Controls and Latin-1 Supplement[1]

Official Unicode Consortium code chart (PDF)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+008x	XXX	XXX	BPH	NBH	IND	NEL	SSA	ESA	HTS	HTJ	VTS	PLD	PLU	RI	SS2	SS3
U+009x	DCS	PU1	PU2	STS	CCH	MW	SPA	EPA	SOS	XXX	SCI	CSI	ST	OSC	PM	APC
U+00Ax	NB SP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	SHY	®	¯
U+00Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
U+00Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
U+00Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
U+00Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
U+00Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Notes

1.^ As of Unicode version 12.0

MS Windows (1252)

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

1	! 33	1 49	A 65	Q 81	a 97	q 113	NOT USED 129	‘ 145	¡ 161	± 177	Á 193	Ñ 209	á 225	ñ 241
2	" 34	2 50	B 66	R 82	b 98	r 114	, 130	’ 146	¢ 162	² 178	Â 194	Ò 210	â 226	ò 242
3	# 35	3 51	C 67	S 83	c 99	s 115	*f* 131	“ 147	£ 163	³ 179	Ã 195	Ó 211	ã 227	ó 243
4	$ 36	4 52	D 68	T 84	d 100	t 116	„ 132	” 148	¤ 164	´ 180	Ä 196	Ô 212	ä 228	ô 244
5	% 37	5 53	E 69	U 85	e 101	u 117	… 133	• 149	¥ 165	µ 181	Å 197	Õ 213	å 229	õ 245
6	& 38	6 54	F 70	V 86	f 102	v 118	† 134	- 150	¦ 166	¶ 182	Æ 198	Ö 214	æ 230	ö 246
7	' 39	7 55	G 71	W 87	g 103	w 119	‡ 135	– 151	§ 167	· 183	Ç 199	× 215	ç 231	÷ 247
8	(40	8 56	H 72	X 88	h 104	x 120	ˆ 136	˜ 152	¨ 168	¸ 184	È 200	Ø 216	è 232	ø 248
9	) 41	9 57	I 73	Y 89	i 105	y 121	‰ 137	™ 153	© 169	¹ 185	É 201	Ù 217	é 233	ù 249
A	* 42	: 58	J 74	Z 90	j 106	z 122	Š 138	š 154	ª 170	º 186	Ê 202	Ú 218	ê 234	ú 250
B	+ 43	; 59	K 75	[91	k 107	{ 123	‹ 139	› 155	« 171	» 187	Ë 203	Û 219	ë 235	û 251
C	, 44	< 60	L 76	\ 92	l 108	| 124	Œ 140	œ 156	¬ 172	¼ 188	Ì 204	Ü 220	ì 236	ü 252
D	- 45	= 61	M 77	] 93	m 109	} 125	NOT USED 141	NOT USED 157	SHY 173	½ 189	Í 205	Ý 221	í 237	ý 253
E	. 46	> 62	N 78	^ 94	n 110	~ 126	NOT USED 142	NOT USED 158	® 174	¾ 190	Î 206	Þ 222	î 238	þ 254
F	/ 47	? 63	O 79	_ 95	o 111	127	NOT USED 143	Ÿ 159	¯ 175	¿ 191	Ï 207	ß 223	ï 239	ÿ 255

UTF-8

Variable Length version of Unicode

Number of bytes	Bits for code point	First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
1	7	U+0000	U+007F	0xxxxxxx			
2	11	U+0080	U+07FF	110xxxxx	10xxxxxx		
3	16	U+0800	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
4	21	U+10000	U+10FFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

UTF-8

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

UTF-8

From Wikipedia, the free encyclopedia

UTF-8 is a variable width character encoding capable of encoding all 1,112,064[1] valid code points in Unicode using one to four 8-bit bytes.[2] The encoding is defined by the Unicode Standard, and was originally designed by Ken Thompson and Rob Pike.[3][4] The name is derived from *Unicode* (or *Universal Coded Character Set*) *Transformation Format – 8-bit*.[5]

It was designed for backward compatibility with ASCII. Code points with lower numerical values, which tend to occur more frequently, are encoded using fewer bytes. The first 128 characters of Unicode, which correspond one-to-one with ASCII, are encoded using a single octet with the same binary value as ASCII, so that valid ASCII text is valid UTF-8-encoded Unicode as well. Since ASCII bytes do not occur when encoding non-ASCII code points into UTF-8, UTF-8 is safe to use within most programming and document languages that interpret certain ASCII characters in a special way, such as "/" (slash) in filenames, "\" (backslash) in escape sequences, and "%" in printf.

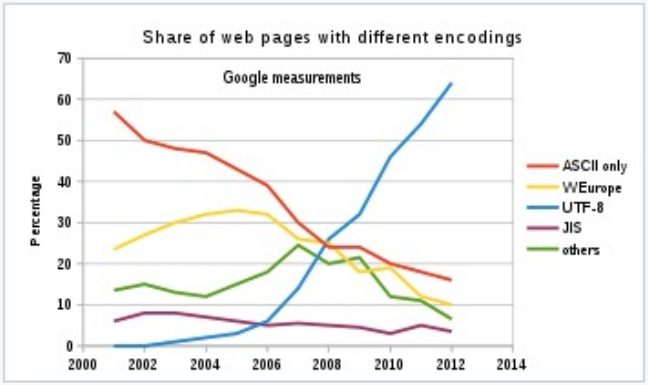
Since 2009, UTF-8 has been the dominant encoding (of any kind, not just of Unicode encodings) for the World Wide Web (and declared mandatory "for all things" by WHATWG[7]) and as of June 2019 accounts for 93.6% of all web pages (some of which are simply ASCII, as it is a subset of UTF-8) and 95% of the top 1,000 highest ranked[8] web pages. The next-most popular multi-byte encodings, Shift JIS and GB 2312, have 0.4% and 0.3% respectively.[9][10][6] The Internet Mail Consortium (IMC) recommended that all e-mail programs be able to display and create mail using UTF-8,[11] and the W3C recommends UTF-8 as the *default encoding* in XML and HTML.[12]

Contents [hide]

UTF-8

Language(s)	International
Standard	Unicode Standard
Classification	Unicode Transformation Format, extended ASCII, variable-width encoding
Extends	US-ASCII
Transforms / Encodes	ISO 10646 (Unicode)
Preceded by	UTF-1

V • T • E



Usage of the main encodings on the web from 2001 to 2012 as recorded by Google,[6] with UTF-8 overtaking all others in 2008 and over 60% of the web in 2012. Note that the ASCII-only figure includes all web pages that only contains ASCII characters, regardless of the declared header.

UTF-8 (low)

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

UTF-8

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
0_	NUL 0000	SOH 0001	STX 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
1_	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
2_	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
3_	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
4_	@ 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
5_	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
6_	` 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
7_	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

UTF-8 (high)

DR JEFF SOFTWARE
INDIE APP DEVELOPER
Jeff Drobman

8_	• +00	• +01	• +02	• +03	• +04	• +05	• +06	• +07	• +08	• +09	• +0A	• +0B	• +0C	• +0D	• +0E	• +0F
9_	• +10	• +11	• +12	• +13	• +14	• +15	• +16	• +17	• +18	• +19	• +1A	• +1B	• +1C	• +1D	• +1E	• +1F
A_	• +20	• +21	• +22	• +23	• +24	• +25	• +26	• +27	• +28	• +29	• +2A	• +2B	• +2C	• +2D	• +2E	• +2F
B_	• +30	• +31	• +32	• +33	• +34	• +35	• +36	• +37	• +38	• +39	• +3A	• +3B	• +3C	• +3D	• +3E	• +3F
2 C_	2 0000	2 0040	Latin 0080	Latin 00C0	Latin 0100	Latin 0140	Latin 0180	Latin 01C0	Latin 0200	IPA 0240	IPA 0280	IPA 02C0	Accents 0300	Accents 0340	Greek 0380	Greek 03C0
2 D_	Cyril 0400	Cyril 0440	Cyril 0480	Cyril 04C0	Cyril 0500	Armeni 0540	Hebrew 0580	Hebrew 05C0	Arabic 0600	Arabic 0640	Arabic 0680	Arabic 06C0	Syriac 0700	Arabic 0740	Thaana 0780	N'Ko 07C0
3 E_	Indic 0800	Misc. 1000	Symbol 2000	Kana… 3000	CJK 4000	CJK 5000	CJK 6000	CJK 7000	CJK 8000	CJK 9000	Asian A000	Hangul B000	Hangul C000	Hangul D000	PUA E000	Forms F000
4 F_	SMP… 10000	□ 40000	□ 80000	SSP… C0000	SPU… 100000	4 140000	4 180000	4 1C0000	5 200000	5 1000000	5 2000000	5 3000000	6 4000000	6 40000000		

Orange cells with a large dot are continuation bytes. The hexadecimal number shown after a "+" plus sign is the value of the six bits they add.

White cells are the leading bytes for a sequence of multiple bytes, the length shown at the left edge of the row. The text shows the Unicode blocks encoded by sequences starting with this byte, and the hexadecimal code point shown in the cell is the lowest character value encoded using that leading byte.

Red cells must never appear in a valid UTF-8 sequence. The first two red cells (C0 and C1) could be used only for a two-byte encoding of a 7-bit ASCII character which should be encoded in one byte; as described below such "overlong" sequences are disallowed. The red cells in the F row (F5 to FD) indicate leading bytes of 4-byte or longer sequences that cannot be valid because they would encode code points larger than the U+10FFFF limit of Unicode (a limit derived from the maximum code point encodable in UTF-16), and FE and FF were never defined for any purpose in UTF-8.

Pink cells are the leading bytes for a sequence of multiple bytes, of which some, but not all, possible continuation sequences are valid. E0 and F0 could start overlong encodings, in this case the lowest non-overlong-encoded code point is shown. F4 can start code points greater than U+10FFFF which are invalid. ED can start the encoding of a code point in the range U+D800–U+DFFF; these are invalid since they are reserved for UTF-16 surrogate halves.