The Next Generation Web: Technologies and Services

Asoke K. Talukder(✉)

SRIT India Pvt. Ltd., Bangalore, India
asoke.talukder@renaissance-it.com

Abstract. Tim Berners-Lee, invented the World Wide Web (WWW) or Web in short in 1989. The Web became so popular that for many, it is synonymous with the Internet or simply the Net. There were many flavors of the original Web like Web 2.0, Web 3.0, etc. All these versions of Web are different use-cases of the original Web, which is fundamentally Client-Server in nature. A client Web browser (or user-agent) makes a request for a document (or a transaction), and the server services that request – kind of a synchronized request-response service (Pull service). The Next Generation Web (NGW) will have a fundamental paradigm shift – it will be two-ways (full-duplex) asynchronous Peer-to-Peer (P2P) communication between two Web browsers using HTML5, Secured WebSocket (wss://), Server Sent Events (SSE), and JavaScript. The Next Generation Web (NGW) will be for human to human, and human to machineries (robots) interaction. Major technologies in Next Generation Web include WebRTC, Web Speech API, and WebUSB. WebRTC is already standardized by W3C and IETF. Web Speech API is at the draft a state. WebUSB is still evolving. NGW is a technology and not a solution – NGW does not need any additional downloads or plugins or any intermediate server. WebRTC supports real-time ultra-low latency audiovisual media and non-media arbitrary data with recording facility. Web Speech API includes speech recognition, speech synthesis, and audio processing on the Web browser. WebUSB will allow USB devices connected to the Web for Collaborative Robotics (Cobotics). Added with AI and IoT, Next Generation Web will revolutionize the Web application and digital transformation ecosystem from computers to mobile phones starting from simple Web page viewing to complex Health care applications and cobotics that will touch everybody's life.

Keywords: Digital transformation · Next Generation Web · NGW · HTML5 · WebSocket · Server Sent Events · JavaScript · WebRTC · Web Speech API · WebUSB · Nodes.js · IoT · Cyber Physical Systems · AI · Knowledge Graph · Secured communication · HIPAA · Cobotics

1 Introduction

On 4th of October 1957, the Union of Soviet Socialist Republics (USSR) launched Sputnik I – the first artificial Earth satellite into the space. The United States Department of Defense responded to Sputnik's launch by creating ARPA (Advanced Research Projects

Agency) in 1958 [1]. ARPA research played a central role in launching the "Information Revolution", through ARPANET. ARPANET's initial demonstration in 1969 led to the Internet that we use today [2].

World Wide Web (WWW) or Web in short was invented by Tim Berners-Lee in 1989. He invented it while he was working at CERN, European Laboratory for Particle Physics in Geneva, Switzerland. Originally World Wide Web was conceived and developed to meet the demand for information-sharing between scientists and researchers in universities and research institutes around the world working in CERN projects (Fig. 1). Tim Berners-Lee had the idea of enabling researchers from remote sites in the world to organize and pool together information and research publications [3].

Fig. 1. (a) The World Wide Web (WWW) inventor Tim Berners-Lee. Picture Taken: 11 Jul 1994 (c) CERN. Picture source: https://home.cern/science/computing/birth-web/short-history-web. (b) The first page of Tim Berners-Lee's proposal for the World Wide Web, written in March 1989 (c) CERN. Picture source: https://home.cern/science/computing/birth-web/short-history-web

In the Web's first generation, Tim Berners-Lee invented the HTML (Hyper Text Markup Language), the Hyper Text Transfer Protocol (HTTP), and the Uniform Resource Locator (URL) standards with Unix-based servers and character based browser version of the World Wide Web. In the second generation, Marc Andreessen and Eric Bina developed NCSA Mosaic at the National Center for Supercomputing Applications, University of Illinois. Several million people suddenly noticed the 'Web' [4]. Web became so popular that today for many 'Internet' or 'Net' became synonymous with Web.

The Clinton Administration's Telecommunications Act of 1996 made Internet free and launched the age of modern Internet policy – thrusting market forces and technological innovations. The authors of this act envisioned about what to happen in the future and that government could facilitate it best by letting private investments happen [5]. This accelerated the innovations in the Web technology space.

The first Web browser named 'WorldWideWeb' was constructed by Tim Berners-Lee in 1990. It was character mode software to access the Web. It did not have much impact outside of the scientific community. The Web became commercial on October

13, 1994 when the Netscape Navigator Web browser was released by the Mosaic (later Netscape) Corporation founded by Marc Andreessen and Jim Clark. Netscape Navigator used GUI (Graphical User Interface) and instantly became popular [6]. It can be stated that Netscape Navigator took Web out of the research lab and released it to the public. The dot-com boom started with Netscape going public on 9th August 1995. In 1995 Microsoft released their Internet Explorer Web browser and the browser war started. Within few years Internet Explorer became the market leader in the Web browser marketplace.

In 1995 few other important events happened in the Web Search Engine space. Jerry Yang and David Filo introduced the Yahoo directory based search engine called "Yahoo!". Crawler based search engine AltaVista was created by researchers at Digital Equipment Corporation and launched in 1995. Paul Flaherty conceived the idea of a crawler based search engine at Digital. Louis Monier and Michael Burrows wrote the Web crawler and indexer respectively.

Google was officially launched in 1998 by Larry Page and Sergey Brin. Google developed a search algorithm at first known as "BackRub" in 1996, with the help of Scott Hassan and Alan Steremberg [7]. The search engine soon became popular and Google went public in 2004. Google became one of the main players in the Internet and Web space with their Google Search Engine, Gmail, Google Map, Chrome Web browser and various AI (Artificial Intelligence) innovations. Google also developed Android operating system for smartphones and other devices.

Web came a long way and evolved from a simple document rendering mechanism on a user display terminal to the public-facing interface of all applications running anywhere in the world. Majority of intuitive interactive applications today have a Web interface. Users use these applications using a Web browser. During the "Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2)" commonly known as COVID-19 pandemic of 2020 the Web came to the rescue of the world. Starting from grocery to high school all became virtual through the Web during the COVID-19 pandemic and lockdown.

The era of Next Generation Web (NGW) started in 2008. During 2008 to 2012 many technologies were invented to transform the Web ecosystems – the main ones that will play a pivotal role in the NGW are HTML5, WebSocket, Node.js, and WebRTC along with JavaScript invented in 1995. These technologies helped a Web browser to transform from a passive client-server type application to a real-time peer-to-peer (P2P) active device. We will see the impact of this transformation through Next Generation Web (NGW) in the post-COVID era starting in 2021 when AI (Artificial Intelligence) and IoT (Internet of Things) join hands with peer-to-peer Web and mobile phones that will touch every industry and everybody's life.

2 Documents and Knowledge in the Web

Tim Berners-Lee thought of collaborative work where a researcher could actually link the text in the files themselves – kind of virtual embedding of one document within another. This is better than simply making a large number of research documents as files. This allowed scientists to arrange documents of related topics together simply through links. This would mean that while reading one research paper, a scientist could quickly display part of another paper that holds directly relevant text or diagrams or tables. Tim

Berners-Lee thought, multiple documents could be related or brought into the user's view by using some form of hypertext. Documentation of a scientific and mathematical nature would thus be represented as a 'web' of information held in electronic form on computers across the world. Tim Berners-Lee had worked on document production and text processing, and had developed his first hypertext system 'Enquire' in 1980 for his own personal use much before he invented World Wide Web. His prototype Web browser on the NeXT computer came out in 1990 [8].

Document presentation in computers was a challenge all along. There are different computers running various word-processing software; at the same time there are different printers from different vendors with proprietary formatting styles. How to design a document in word processing software that will print on a paper by all printers with exactly the same output? GML, SGML, runoff, nroff, and troff are the early versions of document formatting system. Word processing software will use any of these document formatting styles to send the document to the printer. UNIX group at Bell Labs were working on document presentation in UNIX. Roff is a descendant of the RUNOFF program developed by Jerry Saltzer. Douglas McIlroy and Robert Morris wrote runoff for Multics in BCPL based on Saltzer's program written in MAD assembler. Their program was "transliterated" by Ken Thompson into PDP-7 assembler language for his early UNIX operating system, circa in 1970 [9]. In 1973 Joseph Frank Ossanna, Jr authored the first version of document processing software troff for UNIX for Version 2 Unix at Bell Labs, in Assembly language and then ported to C. Another dominating document description language was PostScript developed for desktop publishing business at Adobe Systems by John Warnock, Charles Geschke, Doug Brotz, Ed Taft and Bill Paxton during 1982 to 1984. Another formatting software very popular in academic circle even today is LaTeX – it was written in the early 1980s by Leslie Lamport at SRI International.

While Unix group was building their document formatting software, IBM was working their own document formatting software generalized mark-up language focused on an alternative approach, whereby standard document structures such as headers, paragraphs, lists and so on were marked up by tags inserted into document text. In 1969, together with Ed Mosher and Ray Lorie, Charles F. Goldfarb, a lawyer by training invented Generalized Markup Language (GML) to solve the document representation problem for IBM. This work led to the Standard Generalized Mark-up Language (SGML), which became an international standard ISO 8879:1986 [10].

SGML enables a user to define a grammar for marked-up documents that defines the ways in which tags can be inserted into documents and printed on a printer in homogeneous fashion. Tim Berners-Lee chose SGML as a guide to define the HTML document format for the World Wide Web.

2.1 Semantic Web

The original WWW was "Web of documents", where a URL is used to fetch the document and present it to the user. In a business environment however WWW displays processed data or information from databases like your bank balance or ordering something from an e-commerce site. While presenting the data to the user, the browser uses some presentation styles such that the data is easy to read and ingested by the human user. The term "Semantic Web" conceived by Tim Berners-Lee refers to W3C's vision of the Web of linked data. The ultimate goal of the Web of data is to enable computers to do more useful artificial intelligent work and to develop systems that can support trusted interactions over the network. Tim Berners-Lee had an idea that WWW should not only fetch the data and display it blindly as a rendering agent, but WWW should read and understand the semantics (meaning) of the document or the data and present the meaning of the data for a layman who need not have the domain knowledge.

The goal of the Semantic Web is to make Internet data machine-understandable. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for transforming data into knowledge. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS [11]. For example if you enter '*solar system*' in Google, along with the search results on the left side of the screen, you will also notice some sematic output on the right side of the screen extracted from multiple Web sites about the solar system including pictures of stars and a recommendation. John Mark of New York Times named this concept of semantic Web as Web 3.0 through an article 'Entrepreneurs See a Web Guided by Common Sense' in 2006. It was called as Web 3.0 because in 1999 Darcy DiNucci coined the term Web 2.0 that refers to websites that emphasize user-generated content, ease of use, participatory culture and interoperability for end users.

3 HTML and HTTP

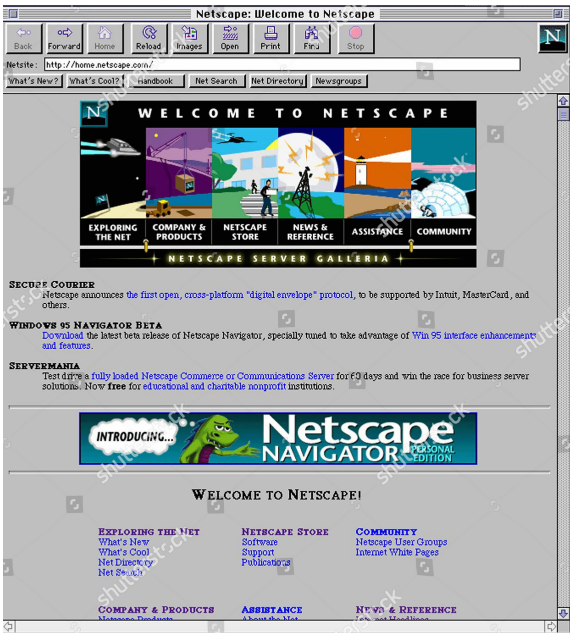
It can be stated that HTML (Hyper Text Markup Language) is formally an application of SGML. The HTML Document Type Definition (DTD) formally defines the set of HTML tags and the ways that they can be inserted into documents. First version of HTML was created by Tim Berners-Lee in late 1991 but was never officially released. The first official version of HTML was HTML 2.0 which was published in November 1995 as IETF standard through RFC 1866.

HTML is a document structuring protocol. But we needed a communication protocol over Internet to fetch the HTML document from the server for us to display on a computer terminal. HTTP (Hyper Text Transfer Protocol) is the protocol for same. The initial HTTP protocol for the Web was simple. The client sent a request: 'GET this filename' and the other end sends back the file and closed the connection. There was no content type to tell the browser what kind of file was being sent. There was no status code for the browser to know if there was an error during fetch. MIME (Multipurpose Internet Mail Extensions) was the multimedia extension to email. MIME was adapted by HTTP so that when a browser receives a file from the server, it has a status code and a content type. This content type tells the browser or a user program whether or not the file is text/HTML, video/MPEG, image/GIF etc., which gives the browser a chance to call up the correct viewers to display the content of the file.

The HTTP we use today is the product of collaboration between CERN and a group at the National Center for Supercomputer Applications (NCSA) at the University of Illinois at Champaign-Urbana [12]. Hypertext Transfer Protocol standard HTTP/1.0 was released through RFC 1945 in May 1996. In May 2000 HTTPS (Hyper Text Transport Protocol Secured) was standardized through RFC 2818 that offered HTTP over TLS (Transfer Layer Security).

There are vast numbers of Web servers and Web services. How does HTTP locate just the file you need from somewhere on the Internet? The answer is through the use of the URL – the Uniform Resource Locator. This is like the telephone number of a file within a computer in the Internet. The URL does it by combining the domain name with the location (path) of the file to be sent to your machine. Uniform Resource Locators (URL) was published in December 1994 through RFC 1738.

HTML also evolved. HTML 3.0 released in 1995 was a major version of HTML with many additional features. Some of the important features are, support of CSS (Cascaded Style Sheets), Tables, and support for Non-Visual Media to cater for the visually challenged individuals.



Secure Courier
Netscape announces the first open, cross-platform "digital envelope" protocol, to be supported by Intuit, MasterCard, and others.

Windows 95 Navigator Beta
Download the latest beta release of Netscape Navigator, specially tuned to take advantage of Win 95 interface enhancements and features.

Servermania
Test drive a fully loaded Netscape Commerce or Communications Server for 60 days and win the race for business server solutions. Now **free** for educational and charitable nonprofit institutions.

Welcome to Netscape!

Exploring the Net	Netscape Store	Community
What's New	Software	Netscape User Groups
What's Cool	Support	Internet White Pages
Net Directory	Publications	
Net Search		

Company & Products | Assistance | News & Reference

Fig. 2. World's first commercial graphic Web Browser the 'Netscape Navigator' homepage on August 9, 1995 the day Netscape went public and the Internet Boom started

Next major version of HTML was HTML 4.0. W3C recommendation of HTML 4.0 was published in December 1997 and revised in April 1998. HTML 4.0 offered many new features. Some of the notable features are Internationalization, to allow representation of the non-English languages through the support of Unicode. It added Frames, which allowed the ability to present multiple documents in one window. Another feature in HTML 4.0 was Advanced Tables. Another major enhancement of HTML 4.0 was scripting for the Web browser – to do some processing tasks to create dynamic Web pages and use HTML as a means to build networked applications.

4 Web Browsers

Tim Berners-Lee is credited with developing, both the first web server, and the first web browser. He developed the first browser called ‘WorldWideWeb’ in 1990. Later the WorldWideWeb program was renamed as Nexus [13]. The browser developed by Tim Berners-Lee was character mode and was not popular. Marc Andreessen and Eric Bina – two programmers working at NCSA (National Center for Supercomputing Applications) at the University of Illinois at Urbana–Champaign created a browser that used GUI (Graphical User Interface). The graphical browser was named as Mosaic. Mosaic is credited with sparking the Internet boom of the 1990s.

Mosaic was the first web browser aiming to bring multimedia content to non-technical general users, and therefore included images and text on the same page. Mosaic’s creator Marc Andreessen, established the company Netscape in 1994. Netscape Navigator (Fig. 2) web browser from Netscape Communications Corporation (originally Mosaic Communications Corporation) dominated the Web market at the beginning; but lost to Internet Explorer from Microsoft and other competitors in the so-called first browser war. However, Netscape contributed towards the growth of Web technologies in many ways through many innovations. A major contribution of Netscape was the JavaScript – the most widely used programming language for client-side scripting of web pages. Netscape also developed SSL which was used for securing online communications before its successor TLS took over.

4.1 WAP Browser

At the end of the 20th century while Internet boom happened, wireless mobile hand phones were also becoming popular. Originally two dominant players in this space were Nokia and Ericsson with their mobile smartphones much before iPhone and Android were launched.

For smart mobile phones to access Web content over HTTP, WAP (Wireless Application Protocol) was introduced. WAP can be considered a scaled downed version of HTML designed for small wireless smartphone screen. WAP is a technical standard specifically designed for accessing information over a mobile wireless network.

In India WAP was introduced in 2000 by CellNext Solutions a company in wireless media space. CellNext collaborated with IBM to WAP enable IBM AS400 applications. IBM business users using AS400 could access their business applications through smartphones using WAP browsers in 2001. CellNext also introduced the first wireless email in India in 2001 that used WAP browser and WAP protocols. The WAP email system for CellNext was developed by Integra Micro Systems.

5 The Next Generation Web (NGW)

Web fundamentally works in Client-Server mode. It is a human to machine (H2M) interface, where a human being uses a Web browser to access a computer or an application or a service in a remote host. A client browser (or user-agent) makes a request using a URL for a document, data, or a transaction, and the server services that request. It is a kind of a synchronized request-response service. This is also referred to as a Pull service, where the user pulls a document from the remote computer using the Web Browser.

The Next Generation Web (NGW) will have a fundamental paradigm shift – it will be two-ways asynchronous Peer-to-Peer (P2P) communication. It is a conversation between two persons or two Web browsers – kind of human to human (H2H) communication using HTML5, JavaScript, WebSocket, and Node.js. In NGW there is no concept of predefined requester or responder or a sever. Anybody can become a requester or a responder with changing roles – either party can send messages or data to the other party in unsolicited fashion in real-time. It is a Push service where end-points do not Pull anything; they Push the data instantly. NGW offers all these peer-to-peer technologies without any intermediate server, plugin, or downloads.

The NGW will also be human to machineries (H2M) interaction, which is called cobotics. Cobotics is collaborative robotics where human beings collaborate with one or more machineries or robots and always is in charge. In robotics robots are autonomous and work independently without the direction of human being; whereas, in cobotics human beings are in charge and uses robotics for some function. A good example of cobotics will be robotic surgery where a human surgeon is in charge and guides a robotic surgeon to do the surgery. We will discuss then in following sections.

Next Generation Web fundamentally includes HTML5, along with WebSocket Secured (wss://), Server Sent Events (SSE), WebRTC, Web Speech API, WebUSB, JavaScript, and Node.js. A good repository to look for examples code and tutorials in HTML5 is https://www.html5rocks.com/en/. You may also like to refer Web Fundamentals at Google developers at https://developers.google.com/web.

Fig. 3. Official logos of HTML5, Node.js, and WebRTC. These are some of the important technologies in the Next Generation Web ecosystem.

5.1 HTML5

Following HTML 4.0, next major release of HTML was HTML5. Because it a new generation of HTML It is not called HTML 5.0 – instead it is called HTML5. HTML5 has a logo of its own (Fig. 3). HTML 2.0 to HTML 4.0 were enhancing the browser capabilities as a passive device for human-facing interfaces. In HTML5 there were many enhancements which make HTML5 browser an active device working like a computer. In HTML 4.0 scripting was added for local client-side processing. HTML5 is known to be a Living Standard and is maintained by a consortium of the major browser vendors (Apple, Google, Mozilla, and Microsoft), the Web Hypertext Application Technology Working Group (WHATWG). Some of the revolutionary features added in HTML5 to make it work as an active device are Video, Audio, Picture, Canvas, in combination with WebSocket, and Server Sent Events.

5.2 WebSocket Secured (WSS://)

In 1971 TCP Socket was formalized through RFC147 titled "The Definition of a Socket". The first paragraph of RFC147 standard states that "A socket is defined to be the unique identification to or from which information is transmitted in the network. The socket is specified as a 32 bit integer with even sockets identifying receiving sockets and odd sockets identifying sending sockets." TCP sockets are used for full-duplex transmission of data between two end-points in the Internet.

TCP sockets are the communication primitives of Internet and it requires lot of effort to develop any communication systems using socket. The avatar of TCP socket in the Web is WebSocket, which was first referenced as TCPConnection in the HTML5 specification. In 2008 first version of the WebSocket protocol was standardized. In December of 2009, Google Chrome 4 was the first browser to support WebSocket. Two new internet schemes "ws://", and "wss://" were introduced for WebSocket and WebSocket Secured respectively. WebSocket Secured uses SSL (Secured Socket Layer). The goal of WebSocket is to provide a mechanism for browser-based applications that need two-way full-duplex communication with servers that need not open multiple HTTP connections and can work without pooling (see below).

5.3 Server Sent Events

Server-Sent Event (SSE) is a mechanism for Web servers to initiate data transmission towards clients. It is generally known as Push service initiated by a server. They are commonly used to send unsolicited message, updates, alerts, notifications, or continuous data streams to a client Web browser. It is designed to enhance native, cross-browser streaming through a JavaScript API called EventSource, through which a client requests a particular URL in order to receive an event stream. The idea behind this may be equivalent to subscribe-publish in message queues [14]. A web application "subscribes" to a stream of updates generated by a server and, whenever a new event occurs, a notification is sent to the client.

To understand the importance of Server-Sent Events, we need to understand the limitations of its AJAX (Asynchronous JavaScript And XML) predecessors, which include Polling or Long Term Polling for alerts, notification, or unsolicited message to the user. Polling or a Pull service is a traditional technique used by the vast majority of AJAX applications for alerts. The basic idea is that the client application repeatedly polls a server for data or pulls some data from the server. Fetching data revolves around a periodical request-response format. The client periodically makes a request to the server and waits for the server to respond with data. If there is data ready to be sent to the client, the client fetches the data; if no data is available, an empty response is returned. This type of applications consumes lot of Internet resources.

Long polling (Hanging GET/COMET) is a slight variation of polling. In long polling, if the server does not have data available, the server holds the request open until new data is made available. Hence, this technique is often referred to as a "Hanging GET". When information becomes available, the server responds, closes the connection, and the process is repeated.

Server-Sent Events in HTML5 on the other hand have been designed from the ground up to be efficient. When communicating using SSE, a server can push data to your application whenever it wants, without the need to make an initial request. In other words, updates can be streamed from server to client as they occur. SSE opens a single unidirectional channel between server and client. The main difference between Server-Sent Events and long-polling is that SSE is handled directly by the browser and the user simply has to listen for the message.

One of the reasons SSEs is not very popular because WebSockets provide a richer protocol to perform bi-directional full-duplex data exchange. Having a two-way channel is more attractive for things like games, messaging applications, and for cases where you need near real-time updates in both directions. However, in some scenarios data does not need to be sent from the client very often. A few examples would be friends' status updates, stock tickers, news feeds, or other automated data push mechanisms.

SSEs are sent over traditional HTTP – they do not require a special protocol or server implementation. WebSockets on the other hand, require full-duplex connections and new Web Socket servers to handle the protocol. In addition, Server-Sent Events have a variety of features that WebSockets lack by design such as automatic reconnection, event IDs, and the ability to send arbitrary events.

5.4 JavaScript and Node.js

JavaScript is the scripting language that runs on a browser. JavaScript was invented by Brendan Eich while working with Netscape and introduced in 1995. In 1995 Microsoft released Internet Explorer and the browser war started. On the JavaScript front, Microsoft reverse-engineered the Navigator interpreter to create its own scripting language called JScript [15] and dominated the browser market. This started to change in 2004, when the successor of Netscape, Mozilla cofounded by Brendan Eich, released the Firefox browser. Firefox was well-received by many, taking significant market share from Internet Explorer.

In 2009 another major development happened. Node.js written initially by Ryan Dahl was introduced. Node.js is the same JavaScript but the execution engine sits outside of the browser – in a server. This in other words mean that a browser can now execute the JavaScript code preloaded in its memory and also continue executing JavaScript code at the server side through Node.js. With HTML5, client side JavaScript, and server side Node.js made a browser to be an active device. A combination of JavaScript and Node.js makes two browsers to communicate peer-to-peer and perform different functions which were not possible earlier.

5.5 Real-Time Communication Over Web (WebRTC and RTCWEB)

In May 2010, Google bought GIPS (Global IP Solutions) developing many real-time communication software components. Google open-sourced the GIPS technology and engaged with relevant standards bodies like the World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF) to ensure industry consensus on real-time communication over Internet and the peer-to-peer Web.

WebRTC is a Web communication in real-time standardized by W3C [16]. The real-time communication is standardized by IETF (Internet Engineering Task Force) as RTCWEB. WebRTC being real-time browser to browser communication – it must be able to connect two browsers without a static IP. Even if two computers are within firewall with local IP address in two different domains in two different countries WebRTC should be able to connect these two browsers and communicate. How does WebRTC connect two computers with local IP address?

In standard Human to Computer pull service interface the web browser functions like a user-facing interface of the remote computer application. Let us take a simple example where you want to do some search using Google (www.google.com). If you do '`ping` www.google.com' you will see the IP address of www.google.com shown on your terminal, which is a static IP address that is cataloged in the DNS (Domain Name Server) and does not change from time to time. If you use the command '`ifconfig`' (`ipconfig` for Windows computer) you will see your own IP address. If you are inside a firewall, you will see a local IP address assigned to your computer by the DHCP (Dynamic Host Configuration Protocol) server something similar to 192.168.1.xxx. This dynamic IP address is unknown to the external world and changes from time to time. This in other words implies that a computer within a LAN has a local dynamic IP address which is not visible from outside of the LAN. NAT (Network Address Translation) does an address translation and connects you to Google.

The telephony network is peer-to-peer communication network – anybody can initiate a call as a caller and talk to a callee or the called party. In telephony network all telephones have a unique address that we call a 'telephone number'. This telephone number is unique in the world like the static IP of Google. For example the telephone number of SRIT in Bangalore, India is +91-80-4195-1999. Anybody anywhere in the world can use the telephone network to call this number to talk to someone at SRIT. For example, if you are in USA with your mobile number +1-408-504-2551 calling SRIT number using your mobile phone, the telephony signaling network will use the SS7 (Signaling System 7) signaling network to locate the callee (SRIT in this case) in Bangalore, India. Once the callee is located by the signaling network the call is established between you and the person at SRIT using the telephone network. The telephone network is separate from the signaling network. After the call is established, the signaling network disappears from the scene and does not play any other role.

Similar to the telephony network WebRTC uses a signaling function to locate the computers within a firewall. This is explained in Fig. 4. WebRTC consults with a signaling server to resolve this challenge of locating the callee computer (browser) within the firewall (with local IP address). Signaling is out of the scope of WebRTC. WebRTC system can use any signaling protocol. Signaling can use a STUN (Session Traversal Utilities for NAT) server, TURN (Traversal Using Relays around NAT) server, or even a SIP (Session Initiation Protocol) server to resolve the challenges related to NAT and local IP address.

It can be argued that if WebRTC uses a server why do we say that WebRTC communication is peer-to-peer without an intermediate server, plugin, or downloads? In reality WebRTC does not use any intermediate server, plugins, or downloads to communicate or exchange data. Once the connection between two browsers is established, similar to the telephony network the signaling server disappears from the scene – it does not have any role to play. After the addressing issue is resolved by the signaling server, WebRTC connects two Web browsers and they communicate directly peer-to-peer without any intermediate server, plugin, or downloads.

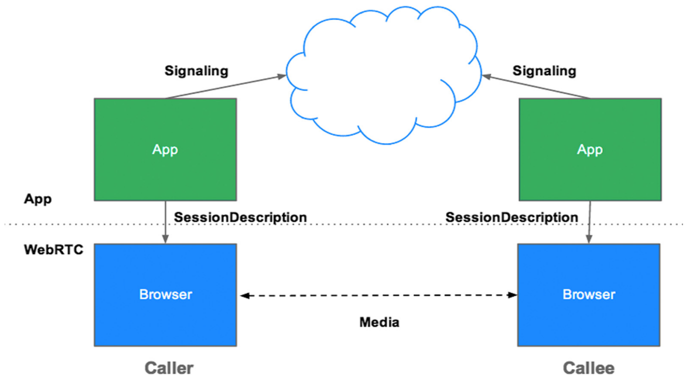


Fig. 4. WebRTC Architecture (Picture credit: Sam Dutton. (2013). WebRTC in the real world: STUN, TURN and signaling. Source of this architecture diagram is Sam Dutton. “WebRTC in the real world: STUN, TURN and signaling”. Published: November 4th, 2013 (https://www.html5rocks.com/en/tutorials/webrtc/infrastructure)

Moreover WebRTC uses HTTPS ensuring that the communication is end-to-end encrypted. WebRTC exchanges media directly using real-time audiovisual medium or arbitrary non-media data. In Fig. 4 upper part shows the caller connecting the callee through a signaling server; once the connection is established – WebRTC works as shown in the lower part of the figure where media (audio, video, and data) are transacted directly point-to-point without the signaling or any intermediate server.

WebRTC achieves all these complex functions through three simple APIs accessible through JavaScript in combination with WebSocket. These three APIs are:

1. `getUserMedia`: This API gives an access to turn real-time media streams like the microphone and the webcam in the device into basic JavaScript objects that can be easily manipulated.
2. `RTCPeerConnection`: This API is the core of peer-to-peer function in WebRTC to establish a peer-to-peer connection.
3. `RTCDataChannel`: This API is responsible for the exchange of real-time non-media data. Examples could be peer-to-peer file sharing, real-time chat, and other forms of IoT data.

WebRTC technology will transform IoT (Internet of Things) and PCS (Physical Cyber Systems) use-cases starting from healthcare to building automation. The real-time P2P feature of reliable secured data communication of WebRTC over the Web will ensures that any IoT device can be directly connected into the Web.

Below is a W3C code example taken from EXAMPLE 9 in Sect. 10.1 ‘Simple Peer-to-peer Example’ from https://www.w3.org/TR/webrtc/#simple-peer-to-peer-example of the WebRTC 1.0 standard [16]. The code assumes the existence of some signaling mechanism that is SignalingChannel. The STUN/TURN server can be used to get variables like their public IP address or to set up NAT traversal. They also have to send data for the signaling channel to each other using the same out-of-band mechanism. You can look at the live open source WebRTC application at https://appr.tc/.

```
const signaling = new SignalingChannel(); // handles JSON.stringify/parse
const constraints = {audio: true, video: true};
const configuration = {iceServers: [{urls:
'stun:stun.example.org'}]};
const pc = new RTCPeerConnection(configuration);

// send any ice candidates to the other peer
pc.onicecandidate = ({candidate}) => signal-
ing.send({candidate});

// let the "negotiationneeded" event trigger offer generation
pc.onnegotiationneeded = async () => {
  try {
    await pc.setLocalDescription();
    // send the offer to the other peer
    signaling.send({description: pc.localDescription});
  } catch (err) {
    console.error(err);
  }
};

pc.ontrack = ({track, streams}) => {
  // once media for a remote track arrives, show it in the re-
mote video element
  track.onunmute = () => {
    // don't set srcObject again if it is already set.
    if (remoteView.srcObject) return;
    remoteView.srcObject = streams[0];
  };
};

// call start() to initiate
function start() {
  addCameraMic();
}
```

```
// add camera and microphone to connection
async function addCameraMic() {
  try {
    // get a local stream, show it in a self-view and add it to
be sent
    const stream = await naviga-
tor.mediaDevices.getUserMedia(constraints);
    for (const track of stream.getTracks()) {
      pc.addTrack(track, stream);
    }
    selfView.srcObject = stream;
  } catch (err) {
    console.error(err);
  }
}

signaling.onmessage = async ({data: {description, candidate}})
=> {
  try {
    if (description) {
      await pc.setRemoteDescription(description);
      // if we got an offer, we need to reply with an answer
      if (description.type == 'offer') {
        if (!selfView.srcObject) {
          // blocks negotiation on permission (not recommended
in production code)
          await addCameraMic();
        }
        await pc.setLocalDescription();
        signaling.send({description: pc.localDescription});
      }
    } else if (candidate) {
      await pc.addIceCandidate(candidate);
    }
  } catch (err) {
    console.error(err);
  }
};
```

WebRTC is a technology and not a service or a solution; so, you can do almost anything using WebRTC – imagination is the limit. WebRTC offers many features and facilities. You can share screens (display device) by using `navigator.mediaDevices.getDisplayMedia`. You can do media processing like mixing of images, change color, add background etc. You can stream the canvas content or stream audio or video from external devices like a camera or microphone. You can even record and playback the media on a local disk and many more functions.

If you are interested in developing applications and services using WebRTC, I shall recommend that you please refer to sample code of WebRTC available at https://webrtc.github.io/samples.

5.6 Web Speech API (WSA)

Computer Telephony Integration (CTI) plays a significant role in many online telephony applications like Emergency services 911 in the US, Telephone banking, Customer support, Call center, Telephone triage and many more. These have been implemented using the IVR (Interactive Voice Response) systems [17]. IVR systems have been replaced by Visual IVR where the user interacts with a visual system in the Web and then connects to the IVR. This saved time and error – before the customer is connected to the service representative, all information about the customer is in front of the service center representative. In the Next Generation Web (NGW) Web Speech API removes these extra infrastructures. Web Speech API in the Web browser interfaces with a human and connects to the computer system directly. If you have SIP server, you can connect Web Speech API and the telephone network.

The Web Speech API (WSA) aims to enable web developers to provide, in a web browser, speech-input (speech-to-text) and speech output (text-to-speech) features that are typically not available when using standard speech-recognition software. The WSA itself is agnostic of the underlying speech recognition and synthesis implementation and can support both server-based and client-based embedded recognition and synthesis. The WSA specification is in draft state by W3C and available at https://wicg.github.io/speech-api/.

5.7 WebUSB

Every computer has a few USB (Universal Serial Bus) ports. This interface is used for many purposes starting from pen-drive to external hard disks or Webcam or mouse. The USB is the de-facto standard for wired peripherals. Most USB devices implement one of roughly a dozen standard device classes which specify a way for the device to advertise the features it supports and commands and data formats for using those features. Standard device classes include keyboard, mice, audio, video and storage devices. Operating systems support such devices using device-drivers provided by the device vendor. Many of these native codes have historically prevented these devices from being used by the Web. And that is one of the reasons the WebUSB API has been created: it provides a way to expose USB device services to the Web.

The WebUSB API provides a way to safely expose USB device services to the Web. It provides an API familiar to developers who have used existing native USB libraries and exposes the device interfaces defined by existing specifications. With this API you will have the ability to build cross-platform JavaScript SDKs for any device. The latest version of the W3C draft specification is available at https://wicg.github.io/webusb/.

6 Ancillary Technologies

There are other ancillary technologies that started emerging while Web was at its infancy. Java developed by James Gosling at Sun Microsystems was released in 1995 as a core component of Sun Microsystems' Java platform. Java like C++ is an object oriented procedural language. It was the language of choice for Web development. Java dominated the Web programing technology domain for a long time.

R language was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. The project was conceived in 1992, with an initial version released in 1995. R is an implementation of the S programming language created by John Chambers in 1976, while working at Bell Labs. R is a functional programing language and requires less coding and testing. Almost all statistical and mathematical algorithms and some AI libraries are ported in R. Soon R became the language of choice for data mining and statistical analysis.

Python was created by Guido van Rossum and released in 1991 at CWI (Centrum Wiskunde & Informatica) in the Netherlands. Python is a combination of object oriented with many features of functional programing. TensorFlow was developed by the Google Brain team for internal Google use. TensorFlow was made open source and released under the Apache License 2.0 on November 9, 2015. In no time TensorFlow was adopted by Python community to make it the de-facto standard and vehicle of AI. Python started gaining popularity when it started becoming the language for Artificial Intelligence (AI). All algorithms related to AI are available in Python.

While we talk about artificial intelligence we need to talk about human sensory organs like eye, ear, smell, taste, touch etc. Assuming there is a sensor to sense any of these signals, Python has a library for processing it. OpenCV for example – one of the leading computer vision libraries originally written in C++ has a port on Java and Python. For speech recognition or NLP (Natural Language Processing) you have many libraries in Python. For other AI functions like OCR (Optical Character Recognition), ANN (Artificial Neural Network) you have a Python package. Almost any algorithm you need for AI will be available either in Python or R.

Scratch, Python, HTML5, Java, C/C++ are the preferred languages for IoT (Internet of Thongs), CPS (Cyber Physical System), and IoE (Internet of Everything) driven by Raspberry Pi controller. Today's smartphones have many sensors. For example iPhone, Android operating systems on smartphones have Camera, Speaker, Motion Sensors, Position Sensors, and Environment Sensors. You can connect external sensors and wearables in these smartphones or your computers through the USB (Universal Serial Bus) port. Now you have sensors and the AI libraries with Next Generation Web. You just need to combine them and create innovative solutions.

As the price of smartphones are falling, and advanced wireless cellular networks like 4G and 5G with higher data bandwidths are becoming reality, the footprint of smartphones is increasing. Unlike a computer, smartphone is always powered on. Many people find it comfortable using the virtual keyboard or a soft-keyboard on a mobile smartphone compared to a keyboard in a computer. A computer will remain the platform of choice for development users and business users; but smartphones are likely to be the platform for general masses. Almost all web applications have a version for the smartphone or the mobile space. These applications in the mobile space are no longer called applications; they are called Apps. Google Android and iPhone iOS are the dominant players in the mobile space. They even have App stores where these mobile apps are generally hosted. There are many platforms and languages available in this space. However, Google Flutter is a very promising platform. Flutter is an open-source cross-platform graphical software development environment created by Google. Flutter is used to develop applications for

Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase written in Dart. Like Java – you write the code in Dart once and execute it in all environments. Dart even can compile to either native code or JavaScript.

In the Next Generation Web, CPS or IoE systems will be connected directly to a Web browser and send the sensor data real-time through WebRTC or Web Speech API, or WebUSB for server side processing. Eventually there will not be any Raspberry Pi controller at the sensor end. The sensor will collect the data and send it for real-time server side processing. NGW being real-time peer-to-peer with unlimited power, AI systems running at the server end can do many smart actions that were unthinkable earlier.

7 Next Generation Web Services

Next Generation Web is real-time peer-to-peer without any intermediate server or plugin, or downloads. This makes WebRTC end-to-end secure channel of communication without any third party software or systems in between. Any service needing high level of security can consider NGW as the carrier of technology and solutions. We already discussed ancillary technologies like CPS and AI. Here we will present two use-cases of NGW.

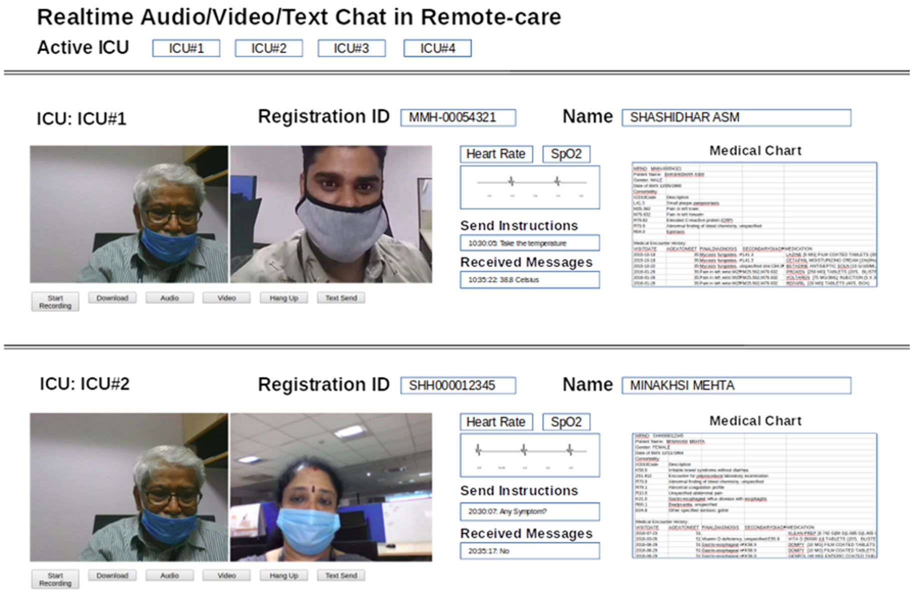
7.1 Use-Case I (HIPAA Compliant Remote Care)

In this use-case we discuss a HIPAA (Health Insurance Portability and Accountability Act) [18] compliant Contactless remote care. Healthcare organizations in the US require HIPAA compliance for patient privacy. A case in Oklahoma [19] about a doctor whose patient died has gained a lot of attention – the focus is on the fact that this doctor prescribed some restricted drugs and the entire doctor-patient interaction happened online, via Skype.

When somebody claims their system is "HIPAA Compliant", that usually means they have implemented their healthcare system to match the regulations of HIPAA as best as possible. The key technical points about HIPAA are that you must secure access to your system with strong authentication methods, and best practices to prevent hacking and data breaches. All patient data must be protected and conform to "Protected Health Information," or PHI. This means that information that can be used to identify individual patients should be encrypted when stored in your database, and encrypted when in-transit. The use of HTTPS URL's with SSL encryption is a must.

Figure 5 shows a Contactless ICU Care using WebRTC which is HIPAA compliant. In this picture there is a doctor and two patients (pictures are published with consent of individuals). In this figure a doctor is consulting with two patients from two different hospitals. The screen is split into two segments – each segment using separate peer-to-peer connection to these hospitals.

The dashboard in Fig. 5 shows Medical Chart with synthetic EHR (Electronic Health Records) data that shows the summary of all visits of the patient on a temporal scale with date, diagnosis, and prescribed medications. The Medical chart in Fig. 5 also includes at the top all comorbid diseases of the patient in machine understandable ICD10 codes.


Realtime Audio/Video/Text Chat in Remote-care

Active ICU: ICU#1, ICU#2, ICU#3, ICU#4

ICU: ICU#1 — Registration ID: MMH-00054321 — Name: SHASHIDHAR ASM

Send Instructions: 1030:05 Take the temperature

Received Messages: 1035:22 38.8 Celsius

ICU: ICU#2 — Registration ID: SHH000012345 — Name: MINAKHSI MEHTA

Send Instructions: 2030:07 Any Symptom?

Received Messages: 2035:17 No

Fig. 5. HIPAA compliant Contactless ICU Care. Here there are two patients and a doctor (picture are published with consent) with synthetic EHR data.

The audiovisual media channels between the ICU and the patient data is end-to-end encrypted through HTTPS and PKI (Public Key Infrastructure). Moreover, HTTPS on Chrome prevents CORS (Cross-Origin Resource Sharing). Because WebRTC is peer-to-peer end-to-end secured without any plugin, download, or intermediate server, there is no possibility of a man-in-the-middle or a hacker listening to the channel. Moreover, because HTTPS does not support CORS, the patient data in the Medical Chart is secured while on transit, though the doctor is consulting remote.

This is a follow-up session for the doctor to talk to the patients just to ensure that patients are fine. In this setup the doctor has one microphone which is used to talk to multiple patients. Unlike a conference call where one party can talk to all other parties or one can listen to everybody else, in a health care teleconsultation, each channel must be private. One patient should not be able to listen to the conversation the doctor is having with other patients. The voice channels between doctor and the ICU are point-to-point and independent of each other. Conversation with each patient is secured. Therefore, whenever the doctor selects one ICU, other voice channels to other ICUs are automatically muted. When the doctor selects a particular ICU the video channel of other ICUs are not muted – but the frame-rate is reduced.

Here along with the private audiovisual channels there is a full-duplex data channels for data chats between the doctor and nurse in-charge of the ICU. These data channels "Send Instructions" and "Received Messages" fields are also encrypted end-to-end. The heart rate is displayed on the display area that can show other real-time biomedical signals like ECG, SpO2 etc. These devices are connected through the USB (Universal Serial Bus). These streaming data through USB is directly transferred to the remote doctor encrypted end-to-end. The entire session of audiovisual interaction between the doctor and patient is recorded using the recording facility of WebRTC. This means, all interactions between the doctor and the patient are recorded at the doctor's end.

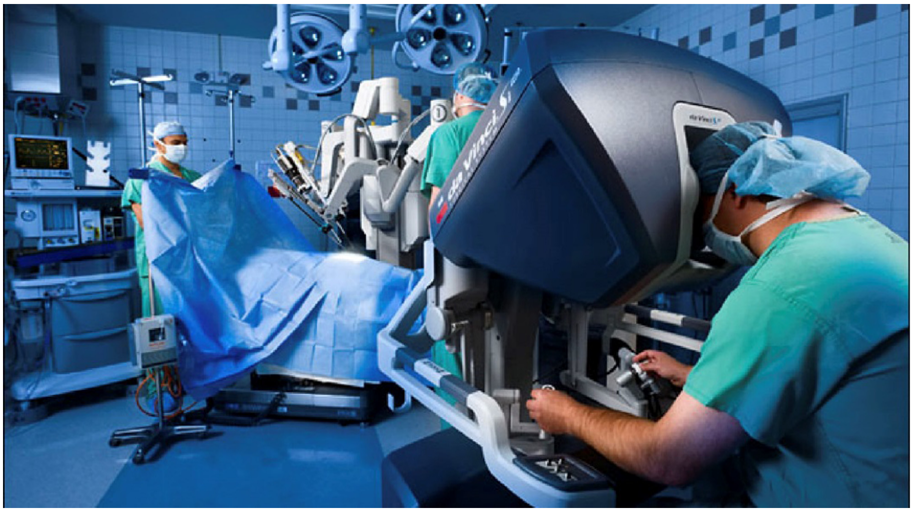


Fig. 6. Cobots in precision surgery where a human surgeon is controlling and supervising a robotic surgeon. Picture source: Robotic surgery has arrived [20].

7.2 Use-Case II (WebUSB in Cobotics)

In this use-case we discuss cobotics. Cobotics is collaborative robotics where humans are in charge and control remote machineries or robots. An autonomous robot performs tasks with a high degree of independence. WebRTC technology will transform IoT (Internet of Things) and PCS (Physical Cyber Systems) use-cases starting from building automation to cobotics. The real-time P2P feature of reliable data communication of WebRTC over the Web will ensures that any IoT device can be directly connected into the Web. These IoT devices will no longer need to be a standalone instrument driven by Raspberry Pi. WebRTC allows ultra-low latency connection for driving a robot. WebRTC offers end-to-end security; therefore, WebRTC can also be used as a VPN (Virtual Private Network).

Autonomous robots are expensive and have various constrains. An autonomous robot needs sufficient battery power, autonomous electromechanical flexibilities. In addition, robots must have sophisticated AI (Artificial Intelligence) for decision making along with synchronization of multiple sensors with AI functions. However, cobots are less expensive and easier to implement where cobots may function like robot coworkers or digital workers. An example could Flippy who flips and grills burgers in a California restaurant [21]. Figure 6 shows an example of cobotics where robotic surgeon is working as a coworker in an operating room along with nurses and human surgeons controlling the robots.

You can have multiple sensors and IoT devices with limited AI capabilities connected to the Web through WebRTC. An application in the Web managed by a person can control all these robots and implement a highly sophisticated cobot system. Some of the decision making can be shared between the human master controlling the robots and some decision making are delegated to some AI applications at the server end.

8 Conclusion

The United States Department of Defense responded to USSR's artificial satellite Sputnik's launch by creating ARPA in 1958. ARPA research played a central role in launching the ARPANET that laid the foundation in Internet that we use today.

In the Web's first generation, Tim Berners-Lee invented HTML, HTTP, and the URL standards. In the second generation, Netscape Communication created the world's first commercial graphic Web browser. On 9th August 1995 the Internet (Dot-Com) boom started with Netscape going public. The Clinton Administration's Telecommunications Act of 1996 made Internet free and helped private investors and innovation to drive the Internet.

While Web was an infant, Python was released in 1991. TensorFlow was developed by the Google Brain team for internal Google use. TensorFlow was made open source and released under the Apache License 2.0 on November 9, 2015. In no time TensorFlow was adopted by Python community to make it the de-facto standard and vehicle of AI.

Innovations between 2008 and 2018 transformed Web and Internet. Due to the active participation of many universities, standard bodies, governments, and business enterprises towards the cause of Open source, Next Generation Web (NGW) emerged. Next Generation Web is not simply a technology – it can be defined as an ecosystem of Web technologies. Along with Web technologies there were many other ancillary innovations in the Open domain software space that accelerated this journey. Some of the worth noting technologies are HTML5, JavaScript, HTTPS, WebSocket, Node.js, WebUSB, Web Speech API, WebRTC, Python3, Flutter, Dart, TensorFlow, ANN (Artificial Neural network), GAN (Generative Adversarial Network), Knowledge Graph, GNN (Graph Neural Network), and many other AI libraries.

During the COVID-19 pandemic and the lockdown of 2020, almost everything went online over the Web starting from online classes to online ordering of groceries. The technology roadmap of post-COVID-19 world will be the intersection of Artificial Intelligence, Cyber Physical Systems, Mobile Smartphone Apps, and the Next Generation Web. This will transform all industry verticals and touch everybody's life.

References

1. Van Sluyters, R.C.: Introduction to the Internet and World Wide Web. https://academic.oup.com/ilarjournal/article/38/4/162/656416
2. ARPANET. https://www.darpa.mil/about-us/timeline/arpanet
3. A short history of the Web. https://home.cern/science/computing/birth-web/short-history-web
4. Metcalfe, B.: Roads and Crossroads of Internet History Chapter 4: Birth of the Web. InfoWorld, vol. 17, no. 34, 21 Aug 1995

5. Thanks To Bill Clinton, We Don't Regulate The Internet Like A Public Utility. https://www.forbes.com/sites/realspin/2014/03/17/thanks-to-bill-clinton-we-dont-regulate-the-internet-like-a-public-utility/#564eee7c62e5
6. Netscape Navigator. https://en.wikipedia.org/wiki/Netscape_Navigator
7. History of Google. https://en.wikipedia.org/wiki/History_of_Google
8. A history of HTML. https://www.w3.org/People/Raggett/book4/ch02.html
9. roff. https://en.wikipedia.org/wiki/Roff_(software)
10. The Roots of SGML -- A Personal Recollection. http://www.sgmlsource.com/history/roots.htm
11. Semantic Web. https://www.w3.org/standards/semanticweb/
12. Introduction to the World Wide Web. https://www.w3.org/People/Raggett/book4/ch01.html
13. History of the web browser. https://en.wikipedia.org/wiki/History_of_the_web_browser
14. Message queue. https://en.wikipedia.org/wiki/Message_queue
15. JavaScript. https://en.wikipedia.org/wiki/JavaScript
16. WebRTC 1.0: Real-time Communication Between Browsers. W3C Editor's Draft 24 September 2020. https://w3c.github.io/webrtc-pc/
17. Talukder, A.K., Ahmed, H., Yavagal, R.: Mobile Computing Technology, Applications, and Service Creation. McGrawHill, New York (2010)
18. Summary of the HIPAA Security Rule. https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/index.html
19. Oklahoma Doctor Disciplined For Using Skype To Treat Patients?. https://telehealth.org/blog/oklahoma-doctor-disciplined-for-using-skype-to-treat-patients/
20. Robotic surgery has arrived. https://www.ucanaberdeen.com/robotic-surgery-has-arrived/
21. White Castle is testing a burger-grilling robot named Flippy. https://edition.cnn.com/2020/07/16/business/white-castle-flippy-robot/index.html