แก้ไขครั้งล่าสุดเมื่อวันที่ 28 กุมภาพันธ์ 2568

Chapter 6
Broadcasting Systems

- Asymmetric communication (sender - receiver ปริมาณข้อมูลไปกลับไม่เท่ากัน)
- Unidirectional broadcasting คือ กรณีสุดโต่ง
- วิทยุและทีวีกำลังจะเป็นแบบ fully digital
  - Digital audio broadcasting (DAB) - ไม่น่าจะเกิดขึ้นในประเทศไทย ต้องเปลี่ยนอุปกรณ์รับ
  - Digital video broadcasting (DVB) - ทีวีในเมืองไทยเปลี่ยนเป็นระบบ digital หมดแล้ว
  - คาดการณ์ว่าวิทยุ AM/FM น่าจะตายภายในปีหน้า (2020) ใช้ Podcast แทน (ไม่จริง)

อัตราการฟังวิทยุของประชากรอายุ 6 ปีขึ้นไป จำแนกตามกลุ่มอายุ พ.ศ. 2532 2537 2546 และ 2551
RATE OF POPULATION 6 YEARS OF AGE AND OVER BY LISTENING TO RADIO, AGE GROUP:1989, 1994, 2003 AND 2008

กลุ่มอายุ	2532	2537	2546	2551	Age group (years)
รวม	56.7	43.9	42.8	31.1	Total
6 - 14 ปี	41.9	31.4	26.1	12.7	6 - 14
15 - 24 ปี	69.4	56.8	57.5	37.1	15 - 24
25 - 29 ปี	59.6	45.0	45.0	34.6	25 - 29
60 ปีขึ้นไป	42.6	31.3	30.7	28.3	60 and over

ที่มา: การสำรวจสื่อมวลชน พ.ศ. 2532 2537 2546 และ 2551 (วิทยุและโทรทัศน์) สำนักงานสถิติแห่งชาติ กระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร
Source: The Mass Media Survey:1989, 1994, 2003 and 2008, National Statistical Office, Ministry of Information and Communication Technology

อัตราการชมโทรทัศน์ของประชากรอายุ 6 ปีขึ้นไป จำแนกตามกลุ่มอายุ พ.ศ. 2532 2537 2546 และ 2551
RATE OF POPULATION 6 YEARS OF AGE AND OVER BY VIEWING TELEVISON, BY AGE GROUP:1989, 1994, 2003 AND 2008

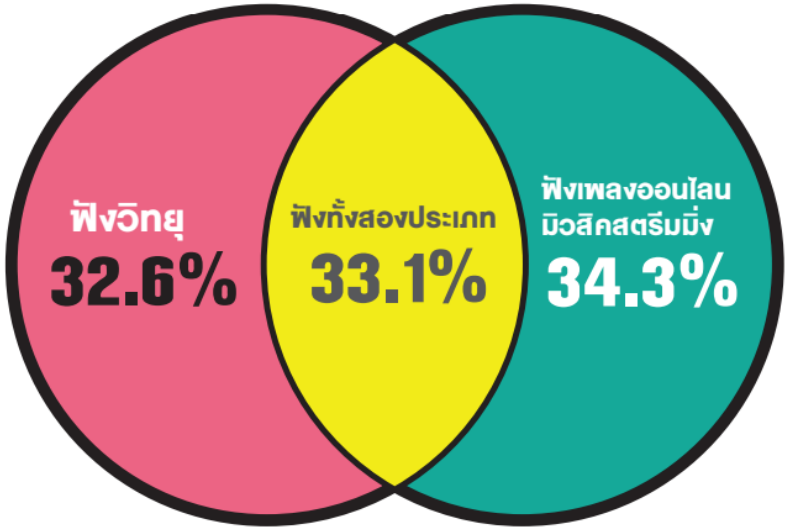
กลุ่มอายุ	2532	2537	2546	2551	Age group (years)
รวม	80.4	89.4	94.5	94.6	Total
6 - 14 ปี	85.3	92.8	97.0	97.0	6 - 14
15 - 24 ปี	83.1	91.0	96.1	96.0	15 - 24
25 - 29 ปี	79.0	89.7	95.6	95.7	25 - 29
60 ปีขึ้นไป	62.9	73.4	81.3	84.6	60 and over

ที่มา: การสำรวจสื่อมวลชน พ.ศ. 2532 2537 2546 และ 2551 (วิทยุและโทรทัศน์) สำนักงานสถิติแห่งชาติ กระทรวงเทคโนโลยีสารสนเทศและการสื่อสาร
Source: The Mass Media Survey:1989, 1994, 2003 and 2008, National Statistical Office, Ministry of Information and Communication Technology

สำรวจพฤติกรรมและแนวโน้ม การรับฟังสื่อทางเสียงของคนไทย ปี 2562

ผลสำรวจพฤติกรรมการรับฟังสื่อทางเสียง ปี 2562 พบว่า วิทยุ FM ได้รับความนิยมสูงสุดทั่วประเทศ เกือบร้อยละ 90 โดยที่ช่วงอายุมีผลต่อพฤติกรรมการรับฟังรายการวิทยุ กล่าวคือ กลุ่มอายุมากมีความสนใจรับฟังรายการข่าว ในขณะที่กลุ่มอายุน้อยสนใจรับฟังรายการบันเทิงมากที่สุด ทั้งนี้ ในส่วนของการรับฟังเพลงออนไลน์และมิวสิคสตรีมมิ่งพบว่า โทรศัพท์มือถือเป็นช่องทางการรับฟังที่สำคัญมากที่สุด

ประเภทสื่อทางเสียงที่รับฟัง



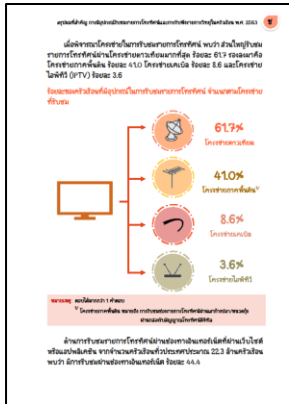
หมายเหตุ : คำนวนสัดส่วนจากจำนวนผู้รับฟังสื่อทางเสียง 5,564 คน

ที่มา : สำนักนโยบายและวิชาการกระจายเสียงและโทรทัศน์, สำนักงาน กสทช.
และ สถาบันอาณามริเวณศึกษา มหาวิทยาลัยธรรมศาสตร์

2. การรับฟังรายการวิทยุในครัวเรือน

จากจำนวนครัวเรือนทั่วประเทศประมาณ 22.3 ล้านครัวเรือน ในจำนวนนี้มีครัวเรือนที่รับฟังรายการวิทยุ 9.3 ล้านครัวเรือน หรือคิดเป็นร้อยละ 41.5 โดยในเขตเทศบาลมีครัวเรือนที่รับฟังรายการวิทยุ ร้อยละ 40.0 และนอกเขตเทศบาล ร้อยละ 42.9

สำหรับคลื่นวิทยุที่รับฟัง พบว่า ส่วนใหญ่รับฟังรายการวิทยุผ่านคลื่น FM มากที่สุด ร้อยละ 62.3 รองลงมาคือ รับฟังรายการวิทยุทั้งคลื่น FM และ AM ร้อยละ 33.4 และคลื่น AM ร้อยละ 3.2 ส่วนสถานที่ที่รับฟังรายการวิทยุ พบว่า ส่วนใหญ่รับฟังรายการวิทยุที่บ้าน/ที่พักอาศัย ร้อยละ 66.7 รองลงมาคือ รับฟังรายการวิทยุขณะเดินทาง/บนรถ ร้อยละ 23.5 และที่ทำงาน ร้อยละ 9.7



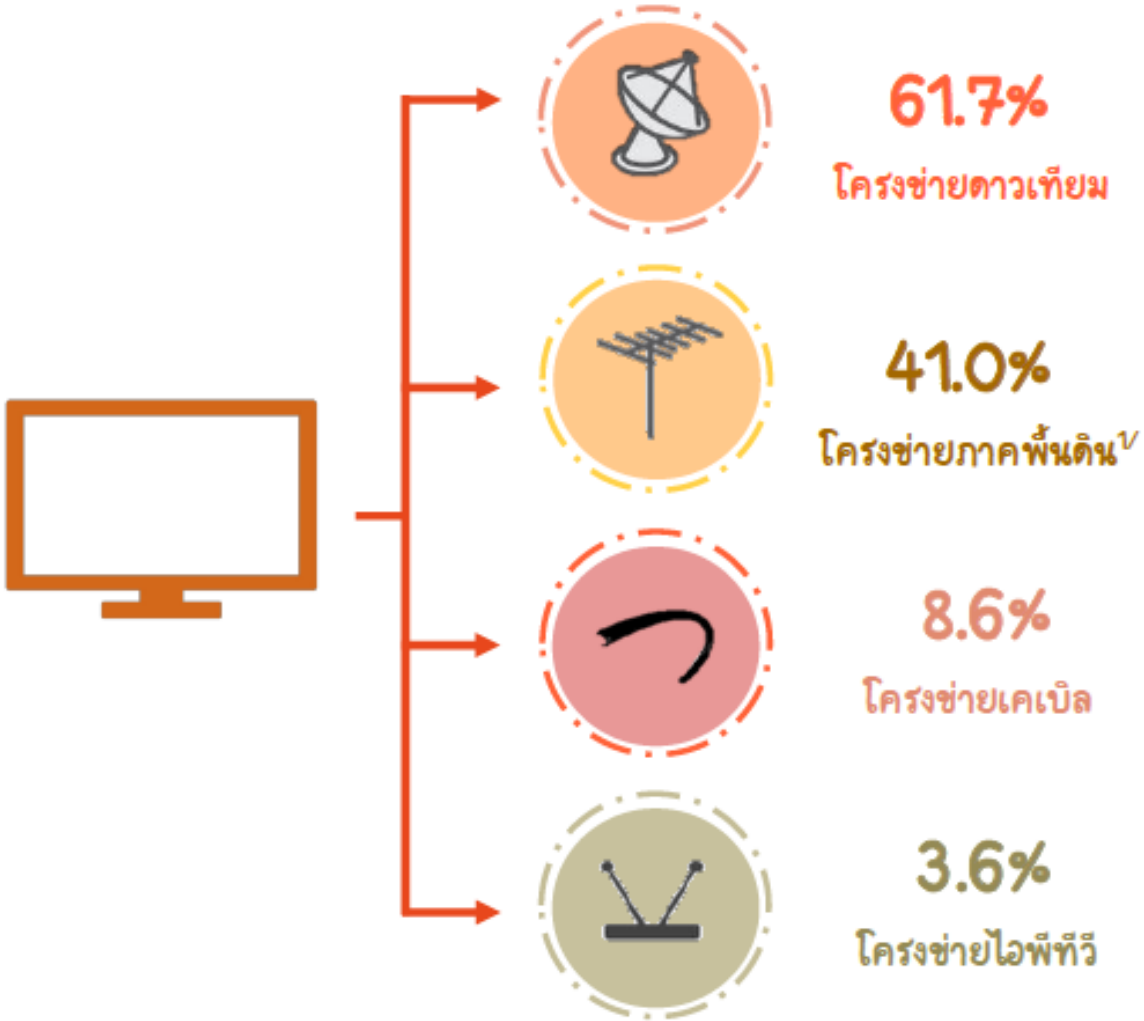
ร้อยละของครัวเรือนที่มีอุปกรณ์ในการรับชมรายการโทรทัศน์ จำแนกตามเขตการปกครอง



ปี	ทั่วราชอาณาจักร	ในเขตเทศบาล	นอกเขตเทศบาล
2560	96.0	96.2	95.8
2561 (ไตรมาส 1)	95.3	94.9	95.6
2561 (ไตรมาส 4)	95.3	94.7	95.8
2562	97.0	97.0	97.0
2563	94.2	93.0	95.3

หมายเหตุ: ตอบได้มากกว่า 1 คำตอบ
ปี 2560 – 2562 อุปกรณ์ในการรับชมรายการโทรทัศน์ คือ เครื่องรับโทรทัศน์ คอมพิวเตอร์ และโทรศัพท์มือถือแบบสมาร์ทโฟน
ปี 2563 อุปกรณ์ในการรับชมรายการโทรทัศน์ คือ เครื่องรับโทรทัศน์

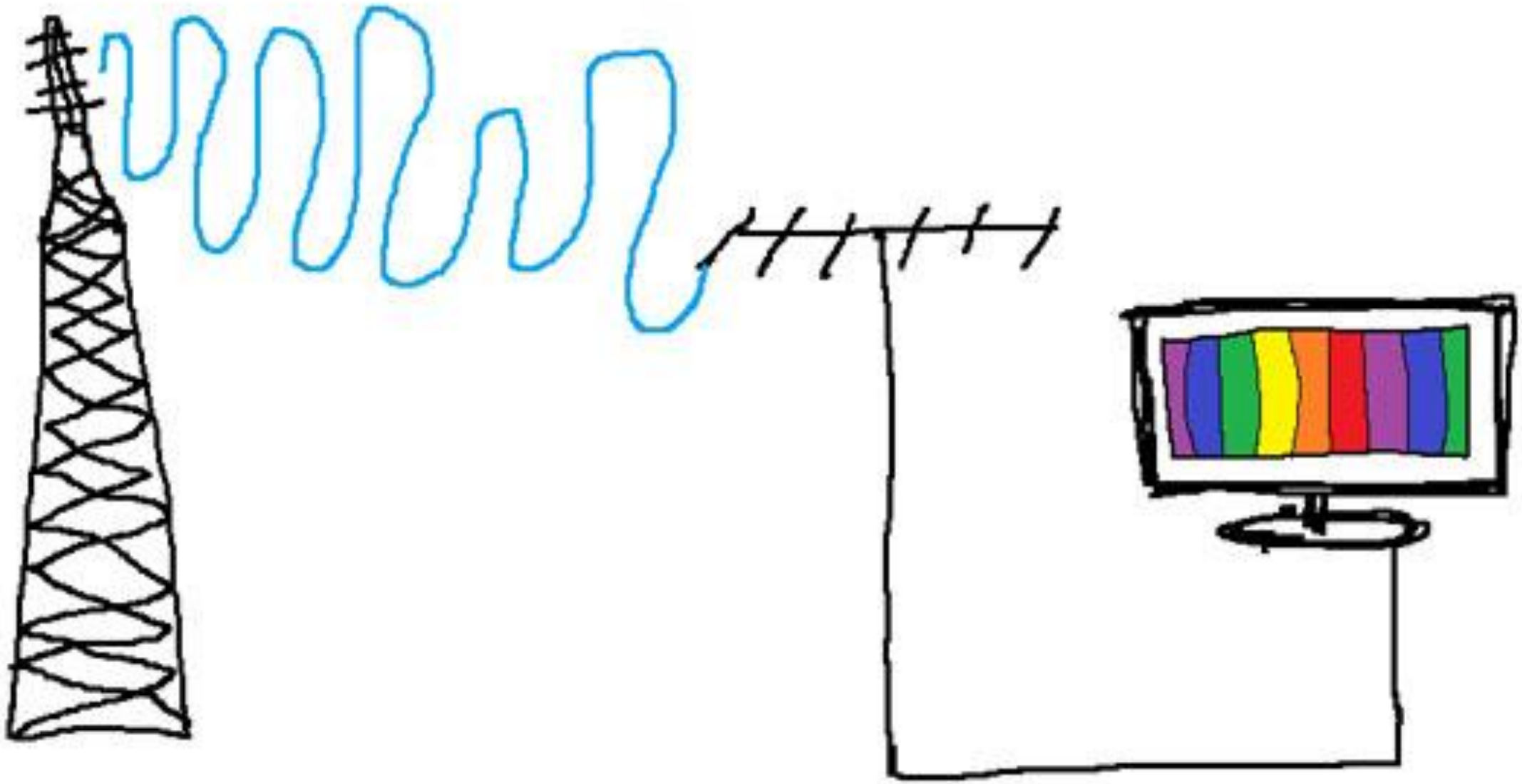
ร้อยละของครัวเรือนที่มีอุปกรณ์ในการรับชมรายการโทรทัศน์ จำแนกตามโครงข่ายที่รับชม

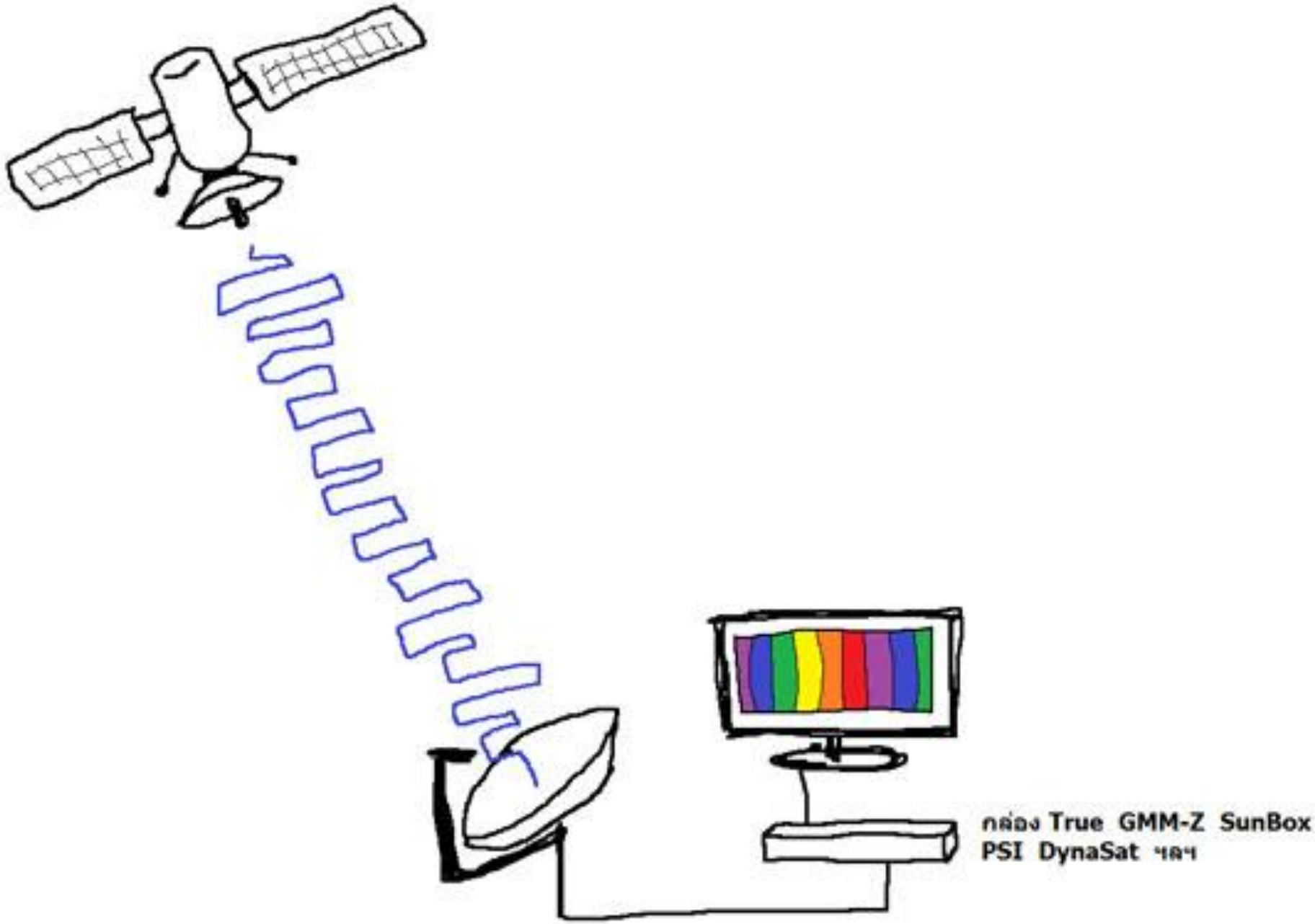


หมายเหตุ: ตอบได้มากกว่า 1 คำตอบ

1/ โครงข่ายภาคพื้นดิน หมายถึง การรับชมช่องรายการโทรทัศน์ผ่านเสาก้างปลา/หนวดกุ้ง ผ่านกล่องรับสัญญาณโทรทัศน์ดิจิทัล

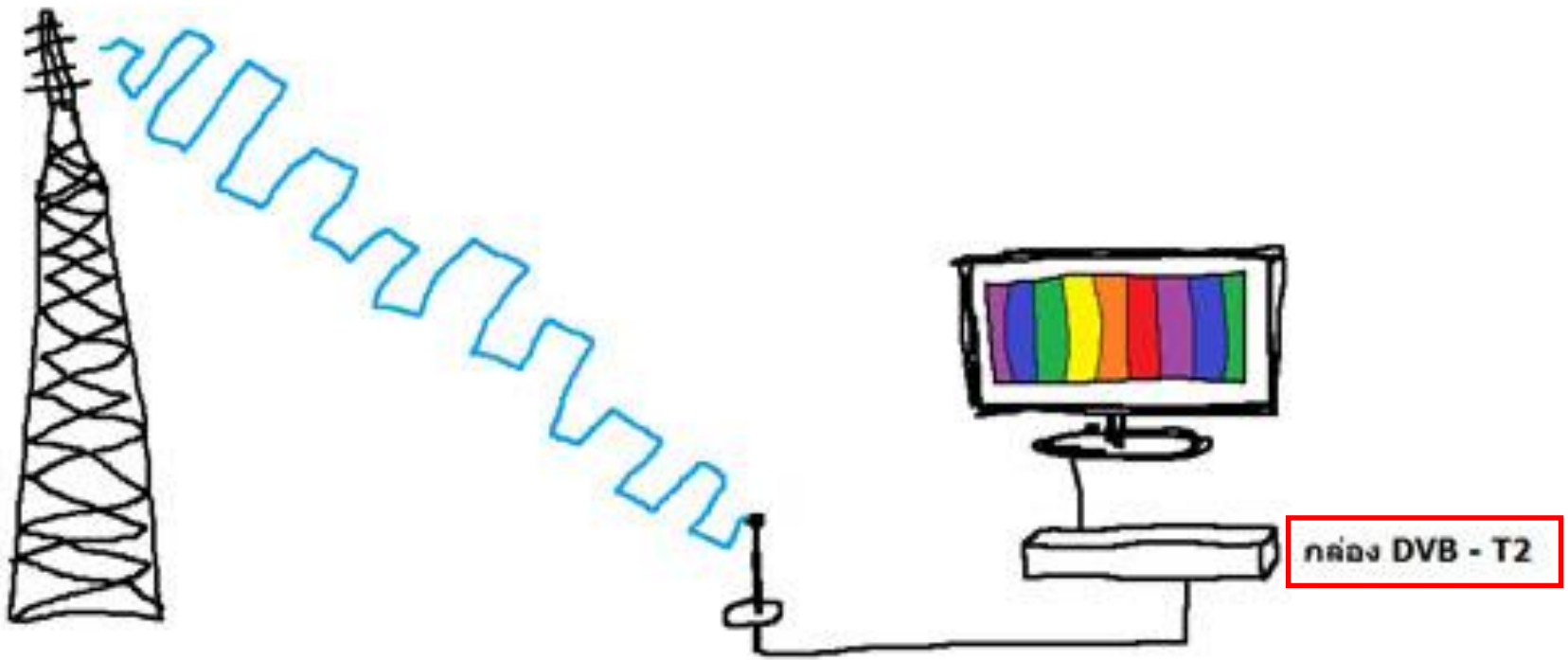
http://www.indyconsumers.org/main/index.php/news-articles/2015-02-17-04-53-24/media-telecom-162/106-570923002




กล่อง True GMM-Z SunBox
PSI DynaSat ฯลฯ

เสาสั้น ติดบ้าน หรือ ติดรถ

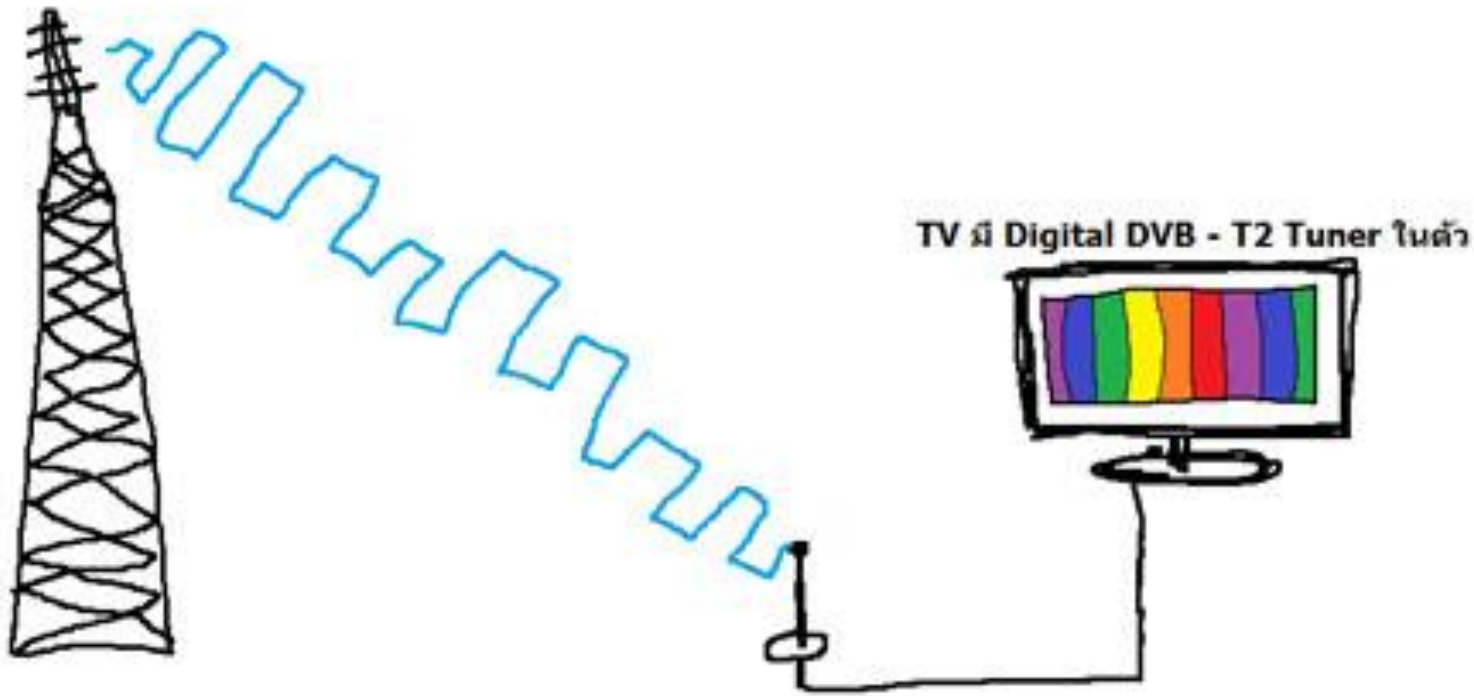
เสายาว ติดนอกบ้าน



T = Terrestrial
ใช้ S หรือ C ไม่ได้

เสาสั้น ติดบ้าน หรือ ติดรถ

เสายาว ติดนอกบ้าน



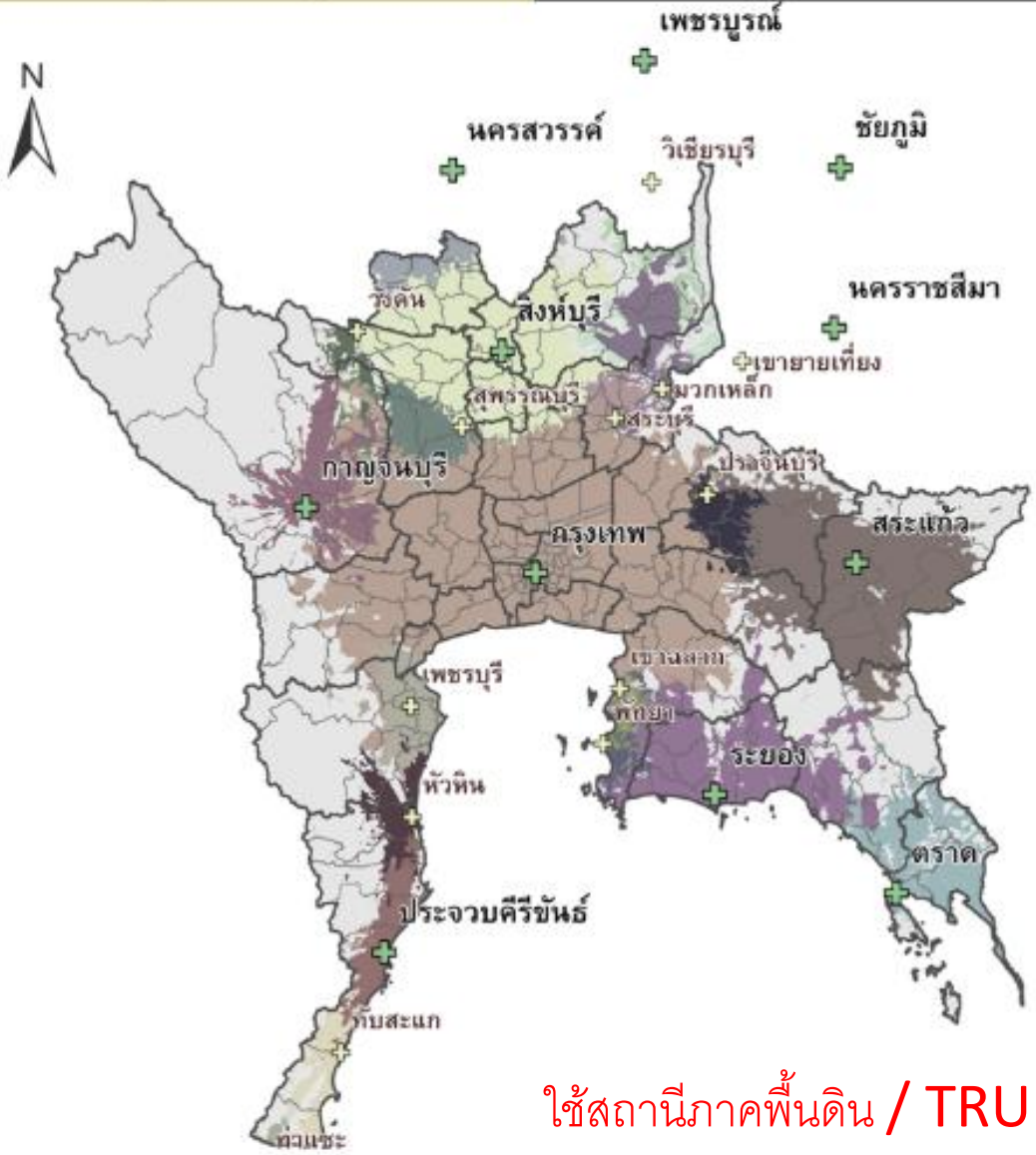
กสทช.

บริการสาธารณะ			หมายเลขช่อง	สถานะ
กองทัพบก	ททบ.5		1	ออกอากาศ
กรมประชาสัมพันธ์	NBT		2	ออกอากาศ
ส.ส.ท. (ไทยพีบีเอส)	Thai PBS		3	ออกอากาศ
[ส.ส.ท. (ไทยพีบีเอส) เด็ก เยาวชนและครอบครัว]	[Thai PBS]		4	
[การศึกษา ความรู้]	สาธารณะประเภท 1		5	
[ศาสนา ศิลปะ วัฒนธรรม]	สาธารณะประเภท 1		6	ไม่มีการทดลองออกอากาศ เนื่องจากยังไม่มีผู้ได้รับใบอนุญาตให้บริการ
[สุขภาพ กีฬา คุณภาพชีวิต]	สาธารณะประเภท 1		7	
[ความมั่นคงของรัฐ]	สาธารณะประเภท 2		8	
[ความปลอดภัยสาธารณะ]	สาธารณะประเภท 2		9	
สถานีวิทยุโทรทัศน์รัฐสภา	สาธารณะประเภท 3		10	ออกอากาศ

พื้นที่ครอบคลุมของสัญญาณดิจิตอลทีวีของประเทศไทยภายหลังการขยายโครงข่ายปีที่ 1-3

จังหวัดในภาคกลาง ตะวันออก ตะวันตก (26 จังหวัด)

กรุงเทพ สมุทรปราการ นนทบุรี ปทุมธานี พระนครศรีอยุธยา อ่างทอง ลพบุรี สิงห์บุรี ชัยนาท สระบุรี ชลบุรี ระยอง จันทบุรี ตราด ฉะเชิงเทรา จันทบุรี นครนายก สระแก้ว ราชบุรี กาญจนบุรี สุพรรณบุรี นครปฐม สมุทรสาคร สมุทรสงคราม เพชรบุรี ประจวบคีรีขันธ์



คำอธิบายสีของพื้นที่ครอบคลุม : ชื่อสถานี (กำหนดการขยายโครงข่าย)

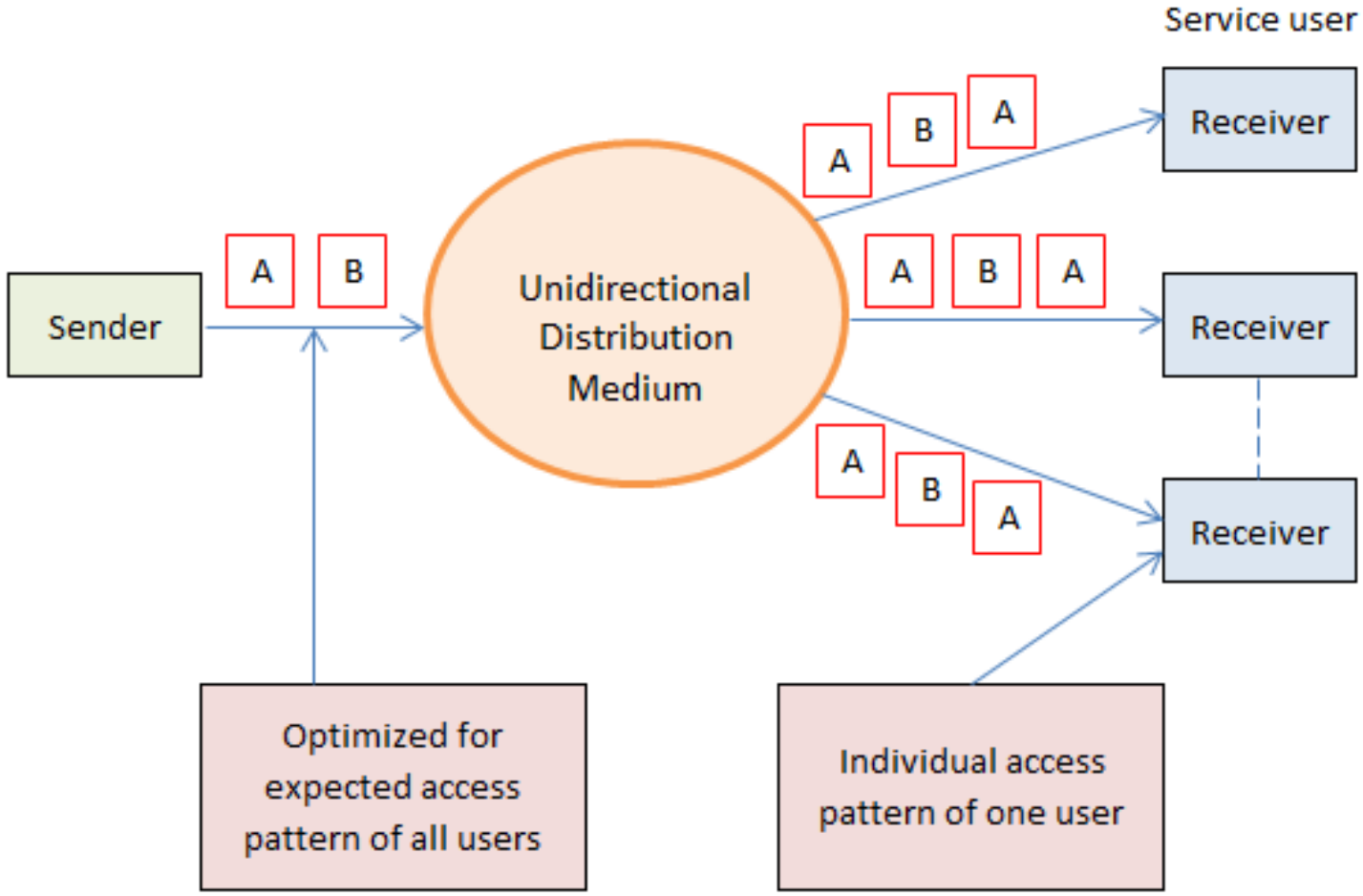
- กรุงเทพ (1 เม.ย. 2557)
- กาญจนบุรี (1 ก.พ. 2558)
- ชัยภูมิ (1 เม.ย. 2558)
- ชุมพร (1 ก.พ. 2558)
- ตราด (1 ก.พ. 2558)
- ทับสะแก (1 ธ.ค. 2558)
- ท่าแซะ (1 มิ.ย. 2559)
- นครราชสีมา (1 เม.ย. 2557)
- นครสวรรค์ (1 ส.ค. 2557)
- ประจวบคีรีขันธ์ (1 ธ.ค. 2557)
- ปราจีนบุรี (1 ธ.ค. 2558)
- พัทยา (1 ก.พ. 2559)
- มวกเหล็ก (1 ก.พ. 2559)
- ระยอง (1 พ.ค. 2557)
- วังดัน (15 มิ.ย. 2558)
- วิเชียรบุรี (1 มิ.ย. 2559)
- สระบุรี (1 ธ.ค. 2558)
- สระแก้ว (1 ส.ค. 2557)
- สิงห์บุรี (1 มิ.ย. 2557)
- สุพรรณบุรี (1 มิ.ย. 2559)
- หัวหิน (1 มิ.ย. 2557)
- เขาฉลาก (1 ก.พ. 2559)
- เขายายเที่ยง (15 มิ.ย. 2558)
- เพชรบุรี (1 มิ.ย. 2559)
- เพชรบูรณ์ (1 ธ.ค. 2557)

คำอธิบายสัญลักษณ์

สถานีหลัก | สถานีเสริม | ขอบเขตจังหวัด | ขอบเขตอำเภอ

ใช้สถานีภาคพื้นดิน / TRUE ใช้ดาวเทียมไทยคม / CTH ใช้ดาวเทียมเวียดนาม

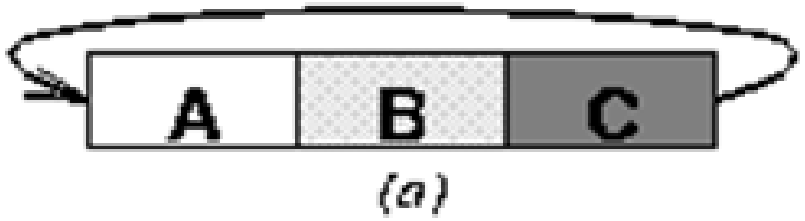
ผู้ส่ง broadcast ออกไปทุก packet แต่ผู้รับเลือกแค่บาง packet เท่านั้น เช่น เลือกรับชมแค่บางโปรแกรม
ผู้ส่งต้องตัดสินใจว่าจะเรียง packet ในแกนเวลายังไงให้ดีที่สุด (ปัญหาคล้าย ๆ CPU Scheduling)



ประเทศที่มีพื้นที่มาก แต่ประชากรน้อย เช่น นอร์เวย์
วิทยุดิจิตอลอาจจะเหมาะสมกว่า cellular system

วิทยุดิจิตอลส่งทั้ง voice และ data
มี voice ได้หลายช่อง และมี data ได้หลายช่อง

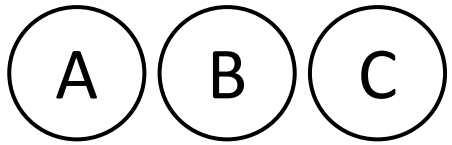
Cyclical repetition of data blocks (เรียกว่า broadcast disk)



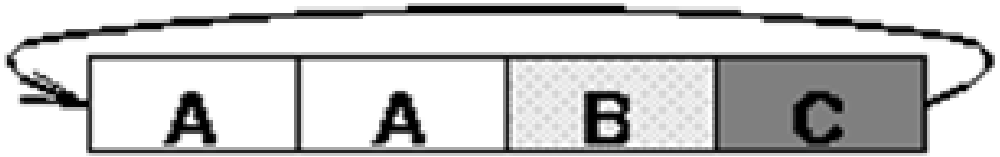
(a)

Flat Broadcast

แต่ละช่องได้ time slot เท่า ๆ กัน



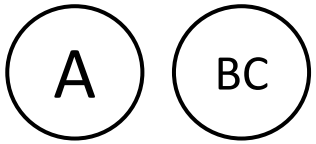
disk A หมุนเร็วกว่า 2 เท่า



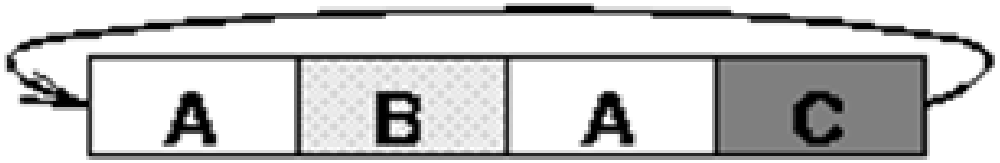
(b)

Skewed Broadcast

บางช่องได้ time slot มากกว่า



disk ทุกตัวหมุนเร็วเท่ากัน



(c)

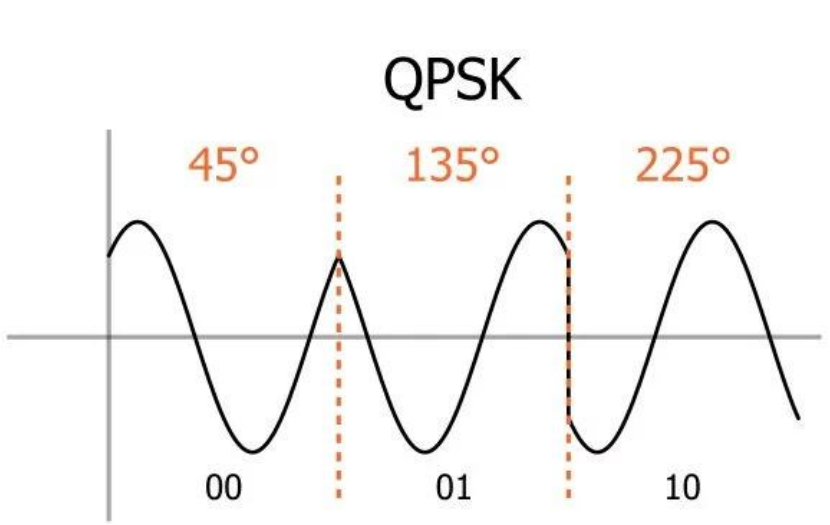
Multi-disk Broadcast

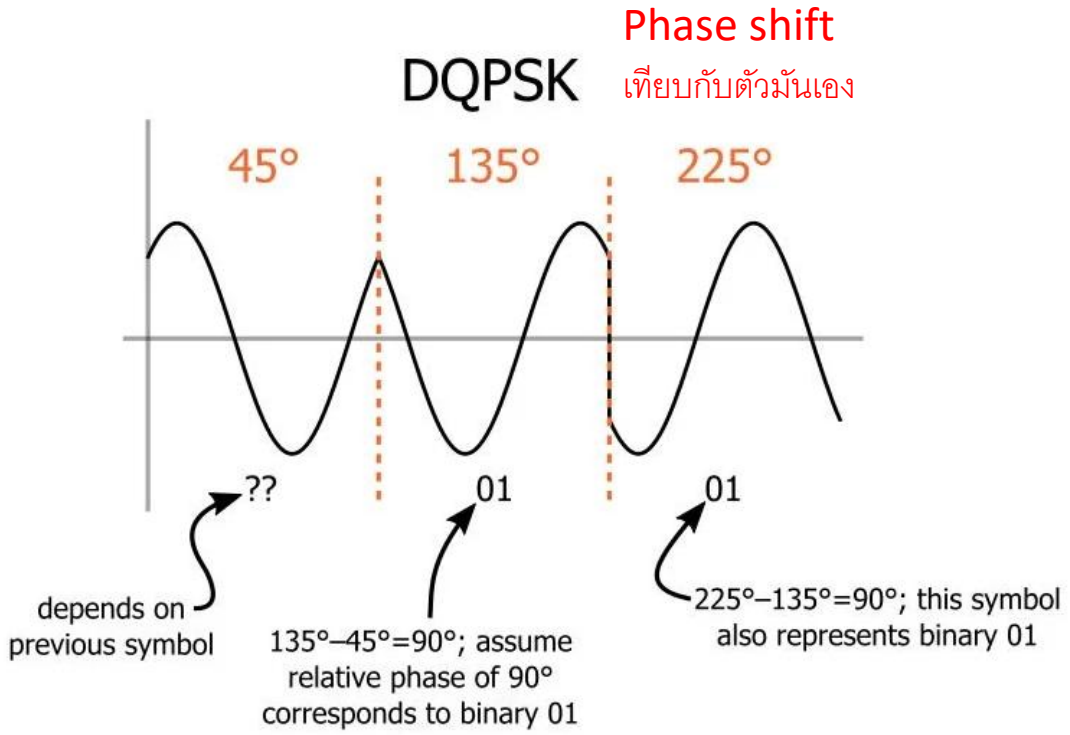
The performance characteristic of third (a multi-disk) program is identical to the case that A is stored on a single-page disk spinning twice as fast as the 2-page disk storing B and C.

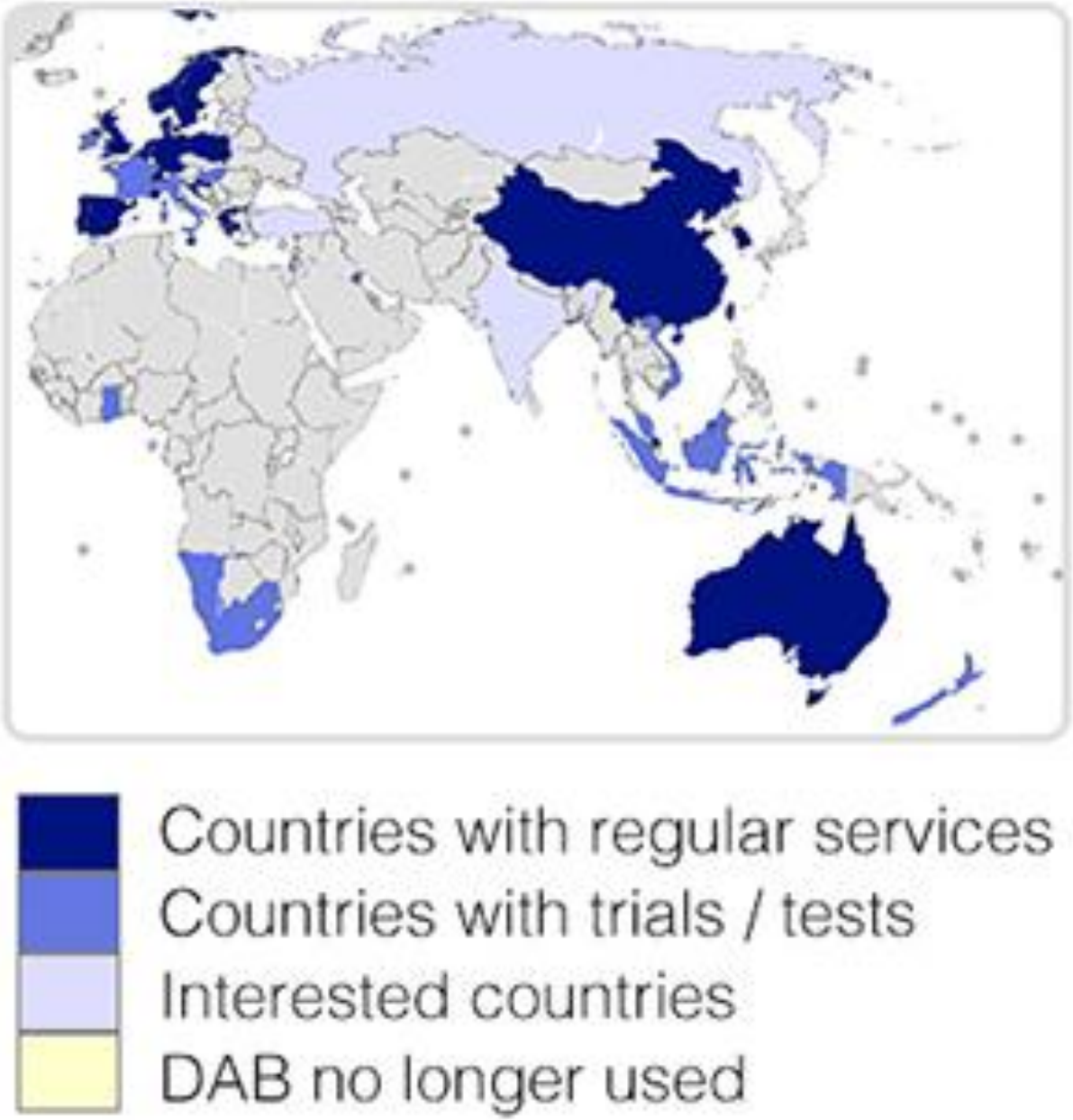
https://dl.acm.org/doi/pdf/10.1145/568271.223816

Digital Audio Broadcasting (DAB)

- ใช้ single frequency networks (SFN) แบ่งช่องโดยใช้ time division (TD)
- ใช้ความถี่ VHF และ UHF ส่งสัญญาณ ใช้พลังงานน้อยกว่าสถานีวิทยุ FM
- ใช้ modulation แบบ Differential Quadrature Phase Shift Keying (DQPSK)







https://www.jd.co.th/product/dab-008-dab-box-digital-radio-tuner-receiver-audio-broadcast-receiver-for-car-stereo-car-radio-android-5.1-and-above_4106066.html

Orthogonal Frequency Division Multiplexing (OFDM)

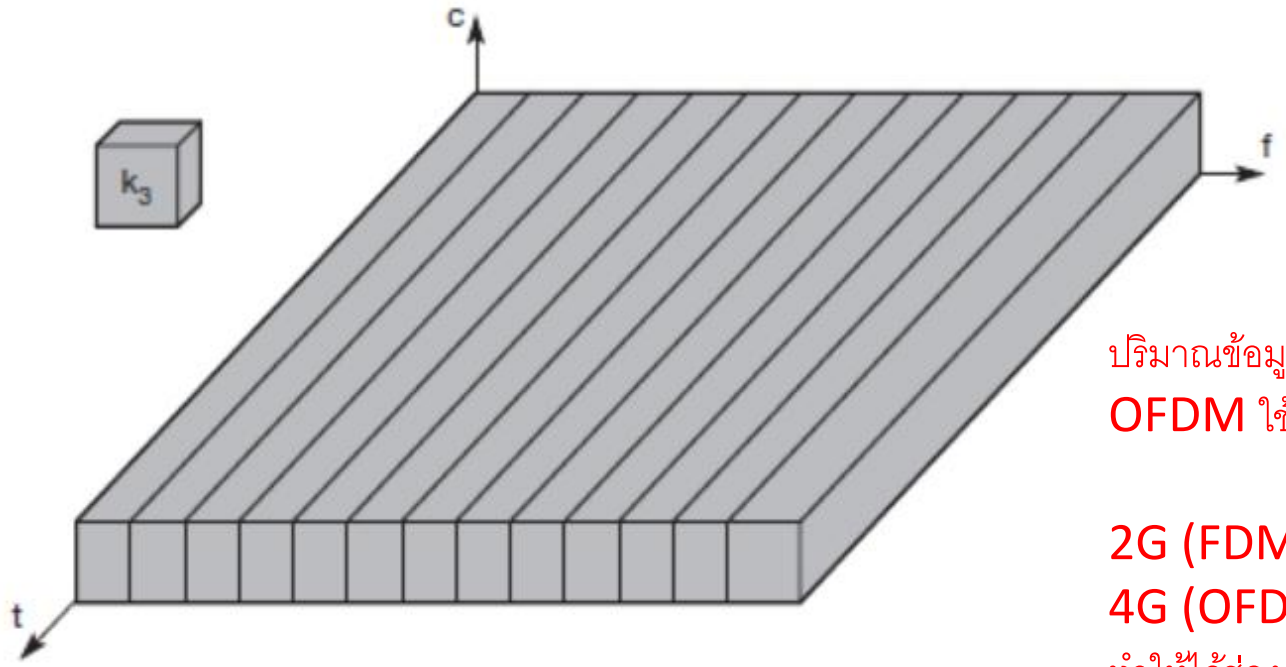


Figure 2.30
Parallel data transmission on several subcarriers with lower rate

ปริมาณข้อมูลมาก ใช้ subcarriers เดียวไม่พอ
OFDM ใช้กับ DAB, DVB, 4G

2G (FDM) ความถี่ห่างกันมากถึง 200 kHz
4G (OFDM) ความถี่อยู่ชิดกันมาก 15 kHz
ทำให้ได้ช่องสัญญาณจำนวนมาก

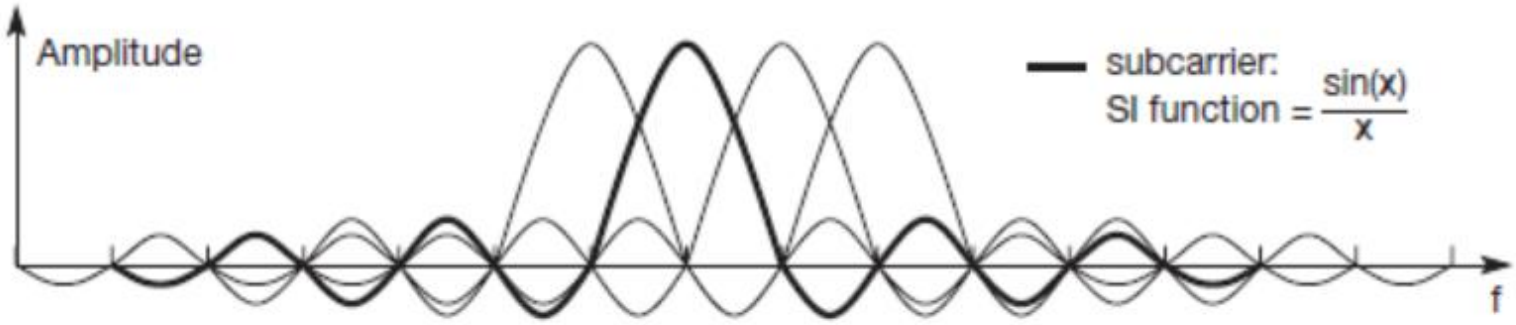


Figure 2.31
Superposition of orthogonal frequencies

Frequency Domain

รูปนี้จะอธิบายทีหลัง

EXAMPLE DAB ENSEMBLE

The following is an ensemble transmitted at 225.648 MHz in southern Germany. The ensemble contains six radio programs and two data channels.

- SWR 1 BW 192 kbit/s, stereo
- SWR 2 192 kbit/s, stereo
- SWR 3 192 kbit/s, stereo
- Hit Radio Antenne 1 192 kbit/s, stereo
- DAS DING 160 kbit/s, stereo
- SWR traffic information 16 kbit/s, data
- SWR service information 16 kbit/s, data

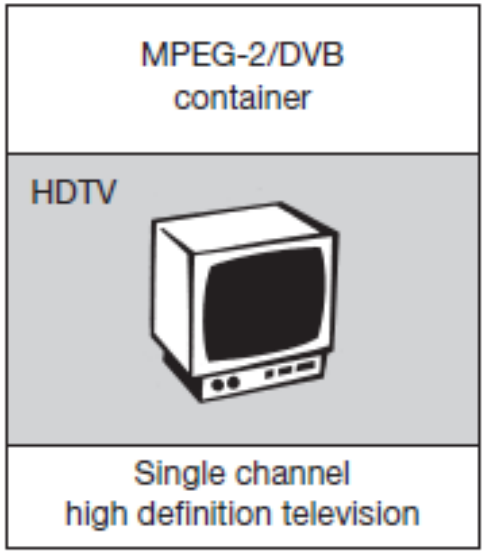
Voice

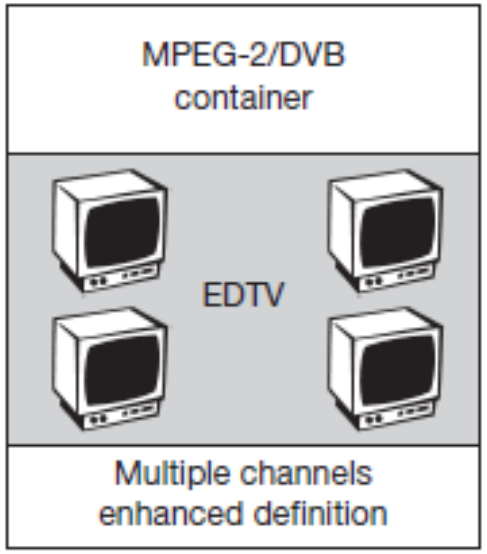
Data

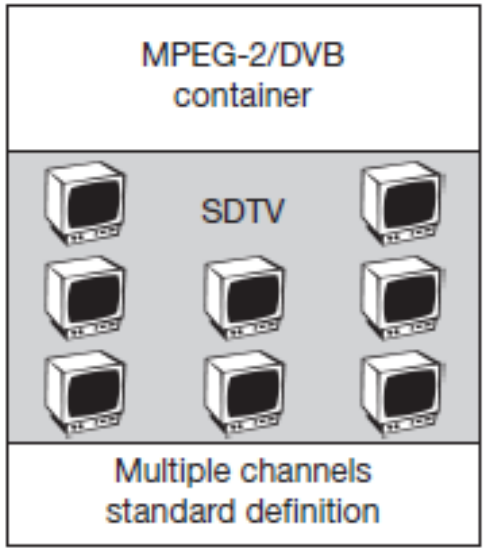
Within every frequency block of 1.5 MHz, DAB can transmit up to six stereo audio programmes with a data rate of 192 kbit/s each. Depending on the redundancy coding, a data service with rates up to 1.5 Mbit/s is available as an alternative. For the DAB transmission system, audio is just another type of data (besides different coding schemes).

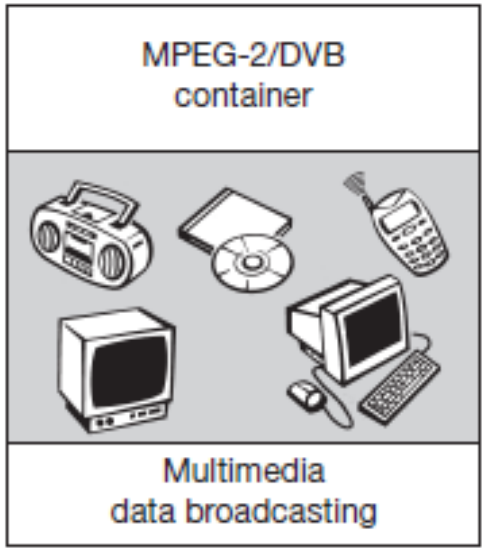
Digital Video Broadcasting (DVB)

- DVB มี 3 มาตรฐาน
  - DVB-S ใช้ satellite
  - DVB-C ใช้ cable technology
  - DVB-T ใช้ terrestrial transmission
- ทุกมาตรฐานใช้ MPEG-2 เป็นหลัก
- DVB-T2 ใช้ MPEG-4 ได้









Digital Radio https://www.youtube.com/watch?v=HewjoIVNvME

Digital TV https://www.youtube.com/watch?v=ylNuJaB1OuQ

The story of two groups - MPEG and VCEG						
Year	**MPEG**	**Part**	**Layer/Profile/Type**	**Usage**	**VCEG**	Variants
1984	Not formed	Practically not useful			H.120	
1988	Not formed	Videoconferencing			H.261	
1993	MPEG-1	VHS and Television Recording				
		Part 1	Systems			
		Part 2	Video	VCD	H.261	
		Part 3	Audio			
			Layer 1			
			Layer II			
			Layer III	**MP3**		
1999	**MPEG-2**	Broadcast, Distribution, DVD				
		Part 1	Systems			
			Program Stream			
			Transport Stream			
		Part 2	Video		H.262	HDV, XDCAM
		Part 3	Audio			
			Layer 1			
			Layer II			
			Layer III	**MP3**		
2004	**MPEG-4**	Broadcast, Internet, Blu-ray				
		Part 1	Systems			
		Part 2	Video		H.263	HDCAM SR
		Part 3	Audio			
		Part 10	Advanced Video Coding	**MPEG-4 AVC**	**H.264**	AVCHD, XAVC
		Part 14	MP4 Container	**MP4**		
2013	MPEG-H	Part 2	Video	**HEVC**	**H.265**	

International Organization for Standardization (ISO) vs. International Telecommunication Union (ITU)

https://www.winxdvd.com/answers/mp4-vs-h264.htm

MPEG2 vs. MPEG4

The Moving Pictures Experts Group, or MPEG, is the body responsible for the standards that we often use for video encoding. MPEG2 is the standard that was created to encode high quality videos, meant to be used for the, then emerging, DVD media. MPEG4 was developed much later, as an encoding method for devices with limited resources. Portable devices, like media players and mobile phones, use this format, as well as online stores who provide the hiring of video and audio files.

MPEG4 is the preferred format for devices, as it yields a file that is under 1G for most full length movies. This is a far cry from MPEG2, which can only produce files with five times the size. Storing MPEG2 files will not be a problem on DVDs, as the usual DVD capacity is over 4GB, but is a major issue with portable devices. MPEG4 also made it practical to buy and download videos online, as MPEG2 videos are quite large, and take a long time to download. The small file size of MPEG4 files directly translates to a lower bandwidth needed, when streaming recorded or real-time videos through the internet.

http://www.differencebetween.net/technology/difference-between-mpeg2-and-mpeg4/

MPEG-H

มีจุดเด่นที่ 3D Audio
https://youtu.be/oErdKT2oS6U

สาเหตุที่เลิกผลิตแผ่น CD/DVD

- มีอินเทอร์เน็ตความเร็วสูงและ flash drive เข้ามาแทนที่
- ความนิยมในการใช้บริการ streaming เพลงและวิดีโอเพิ่มสูงขึ้น

หยุดผลิตตั้งแต่วันที่ 15 พ.ค. 2561

Podcast

> **Podcast** มาจากคำว่า iPod + Broadcast = Podcast
> หรือ **Pod** ที่มาจาก Portable on Demand (ความต้องการแบบพกพา)
>
> สมัยก่อน iPod รุ่นแรกๆ ฟังวิทยุ ไม่ได้ จึงมี Podcast สำหรับดาวน์โหลดเก็บไว้ฟังย้อนหลัง
> ส่วน ในปัจจุบันผู้คนนิยมฟัง Podcast ในรูปแบบ สตรีมมิ่ง (Streaming) กันมากขึ้น

Podcast คือ ไฟล์เสียงออดิโอดิจิทัล ที่สร้างขึ้นมาเพื่อเผยแพร่บนอินเทอร์เน็ต มักจะเป็นเรื่องราวการพูดคุยเกี่ยวกับหัวข้อต่าง ๆ ที่ผู้ชมเพลิดเพลินไปกับเรื่องราวนั้น ๆ ได้โดยไม่รู้ตัว คล้ายกับการจัดรายการวิทยุ โดย Podcast จะมีผู้ดำเนินรายการออกมาเล่าเรื่องราวต่าง ๆ ให้ฟัง อาจจะเป็นตอนเดียว หรือ เป็นซีรีส์ พูดคุยแบบยาวๆ ผู้ชมสามารถสมัครสมาชิก และ ติดตาม เพื่อดาวน์โหลดตอนใหม่ ๆ มาเก็บไว้บนเครื่องคอมพิวเตอร์ แอปพลิเคชัน หรือเครื่องเล่นแบบพกพารูปแบบต่าง ๆ ได้นั่นเอง

https://tips.thaiware.com/1467.html#what-is-podcast

ทำไม Podcast ถึงได้รับความนิยม ?

รายการเสียง Podcast ได้รับความนิยมก็เพราะว่า "เนื้อหาและการเข้าถึงนั้นตอบโจทย์ผู้ฟัง" ด้วยหัวข้อที่หลากหลาย มีรายการหลายรูปแบบ ทำให้ผู้ฟังดำดิ่งไปกับเนื้อหาที่ตนเองชื่นชอบและอยากฟัง อีกทั้งการฟัง Podcast ไม่จำเป็นต้องฟังตรงเวลา มีอินเทอร์เน็ตดาวน์โหลดมาเก็บไว้ มีเวลาเมื่อไหร่ก็สามารถฟังได้ทุกเมื่อ

นอกจากนี้ รูปแบบรายการเสียง Podcast ยังมีเรื่องเล่าที่กระชับฟังแล้วได้ความรู้ เข้าใจทันที หรือแบบยาว ฟังเพลินๆ ถึง 3 ชั่วโมงก็มี นั่นเป็นสิ่งที่ทำให้ Podcast เหมาะแก่การฟังในช่วงเวลาต่างๆ เช่นฟัง Podcast ขณะนั่งในรถ, เดินทางไปทำงาน, ฟังเพลินๆ ขณะทำงาน, ออกกำลังกาย, หรือ ฟังขณะอยู่บ้าน ได้ทุกที่เป็นต้น

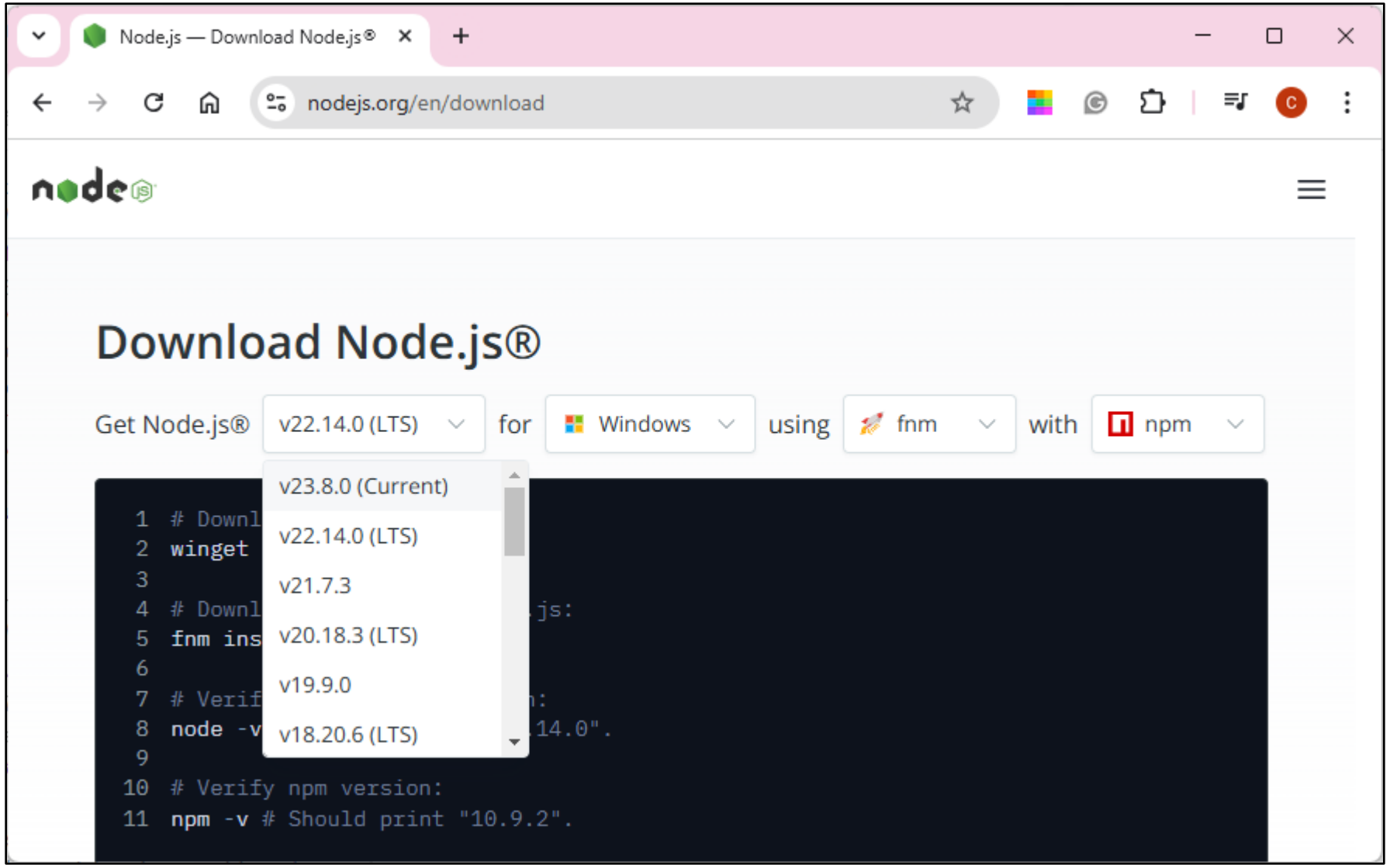
Video Streaming

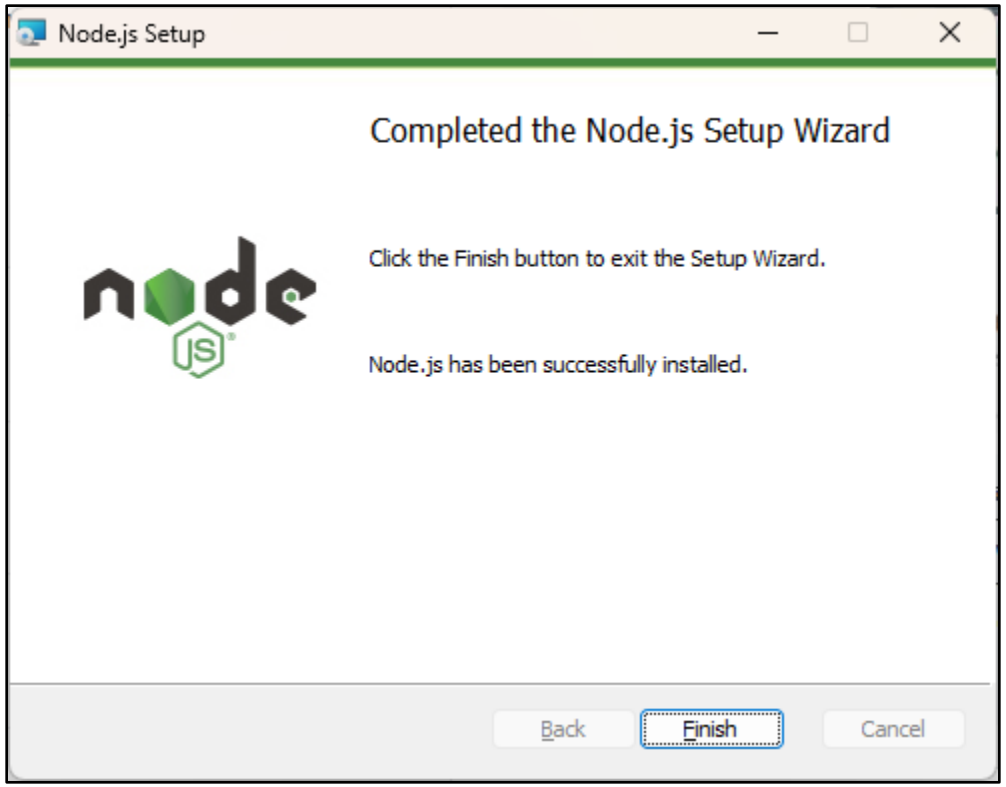
- อินเทอร์เน็ตความเร็วสูงเป็นสาเหตุที่ทำให้เกิด video streaming
- ดูได้ทุกที่ ทุกเวลา ไม่มีโฆษณา แต่มีค่าสมาชิก ผู้รับชมคือกลุ่มผู้มีรายได้ระดับหนึ่ง

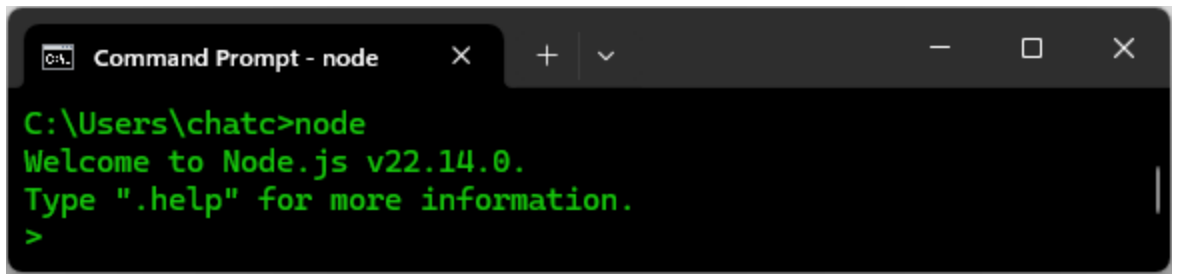
A language
that doesn't affect the way you think about programming
is not worth knowing.

Alan J. Perlis

Introduction to React




Node.js Setup
Completed the Node.js Setup Wizard
Click the Finish button to exit the Setup Wizard.
Node.js has been successfully installed.
Back
Finish
Cancel


Command Prompt - node
C:\Users\chatc>node
Welcome to Node.js v22.14.0.
Type ".help" for more information.
>

พัฒนา Application ด้วย

React และ React Native

สร้าง Web App และ Mobile App (iOS/Android)
โดยใช้ JavaScript เป็นหลักเพียงภาษาเดียว

สามารถทดสอบแอป
บนระบบ Windows
ได้ทั้ง iOS และ Android

- React สำหรับพัฒนา Web App ด้วย JavaScript และ JSX
- React Native สำหรับพัฒนา Mobile App ทั้ง iOS และ Android
- REST API สำหรับรับส่งข้อมูล ระหว่างฝั่ง Local และ Server

ดาวน์โหลดไฟล์ตัวอย่างได้ที่
http://www.developerthai.com

บัญชา ปะสีละเตสัง
ผู้เขียนหนังสือคอมพิวเตอร์ขายดี Best Seller
ด้าน Programming

อินน์

คำสั่งพื้นฐาน (น. 23)

npm install <package_name> …

npm install -g <package_name> … g หมายถึง global

npm uninstall <package_name> …

การสร้างและทดสอบ React Application

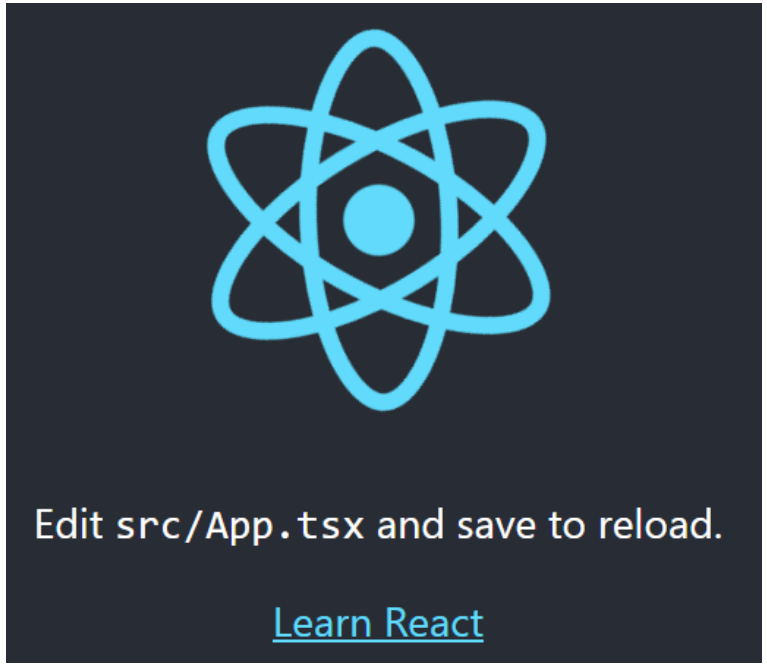
```
npm install -g create-react-app
create-react-app <app-name> --template typescript
cd test-app
code .
```

เอาไว้ก่อนยังไม่ต้องทำแบบ typescript
นิสิตที่สนใจไปศึกษาเพิ่มเองนะครับ

หรือเปิดจากเมนูของ VS Code โดยเลือกที่ File > Open Folder

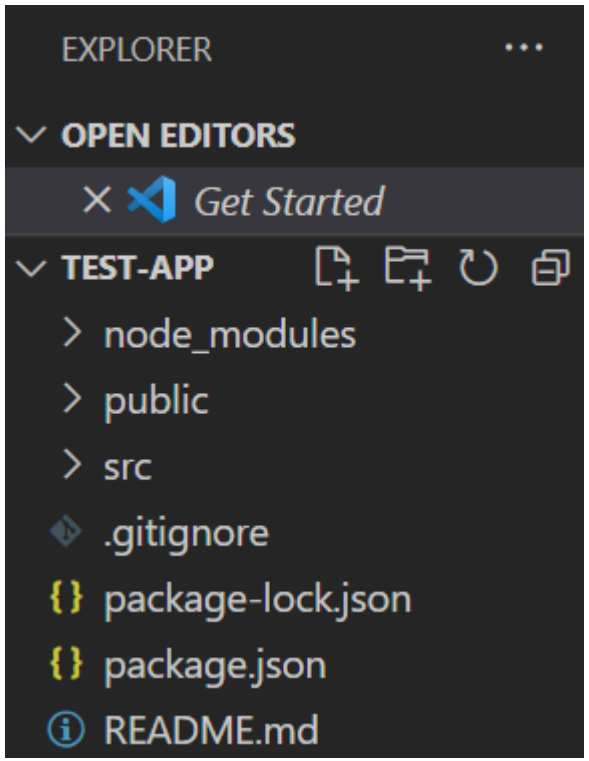
```
npm start          // to run the project
npm run build      // to create a production build
```

เหมือน dotnet build และ dotnet run (JavaScript ไม่ต้อง compile หรือ build)
เหมือน dotnet publish



http://localhost:3000/

ลักษณะโครงสร้างของ React Application (น. 30)



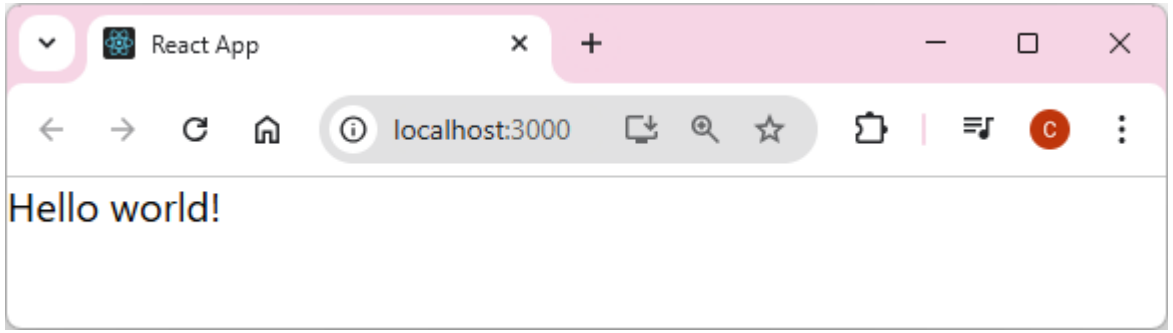
node_modules	โมดูลที่ใช้ในแอปพลิเคชันนี้ มีโมดูลเริ่มต้น และเราโหลดมาติดตั้งเพิ่มได้
public	ใช้เก็บไฟล์ static เช่น HTML, image
src	เก็บ source code (.js และ .css)
	App.js, App.css เป็น top-level component ของ React
	index.js, index.css เป็นหน้าแรกของแอปพลิเคชัน
package.json	การตั้งค่าต่าง ๆ ของแอปพลิเคชันนี้
package-lock.json	สำหรับโปรแกรม npm เราไม่ต้องไปแก้ไขเอง
.gitignore	รายชื่อไฟล์ที่ไม่ต้องเพิ่มลงใน Git
README.MD	MD = mark down คือ คำสั่งและข้อมูลสำหรับใช้บนแพลตฟอร์มอื่น เช่น GitHub

Note: ตั้งค่า VSC ให้ tab = 4 และไม่เดาค่า tab, Shift-Alt + F ใช้ beautify code

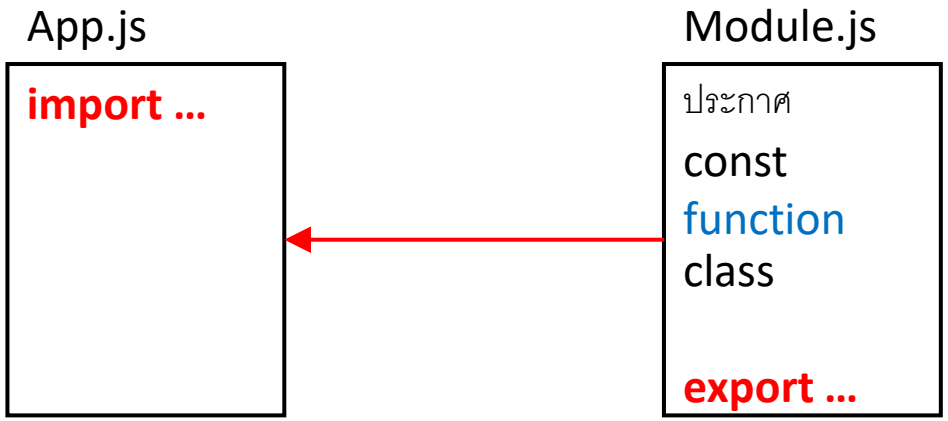
Hello world!

src > JS App.js > ...

```
1  function App() {
2    return <>Hello world!</>
3  }
4
5  export default App
```



การสร้างและส่งออกโมดูล / การนำเข้าโมดูล (น. 24 - 26)



คำสั่ง export ก็คล้าย ๆ public คือยอมให้ภายนอกโมดูล มองเห็น const, function, class ที่ประกาศไว้ใน module และนำไปใช้ได้

การส่งออก constant, function, class ใช้ export เขียนไว้ด้านล่างของ source code เช่น

```
export { PI, add, Random }
```

PI เป็น constant, add เป็นฟังก์ชัน, Random เป็น class

หรือจะใช้ inline export เขียนไว้ข้างหน้า const, function, class เช่น

```
export const PI = 3.14
export function add(num1, num2) { ... }
export class Random { ... }
```

การนำเข้า ใช้ import เช่น

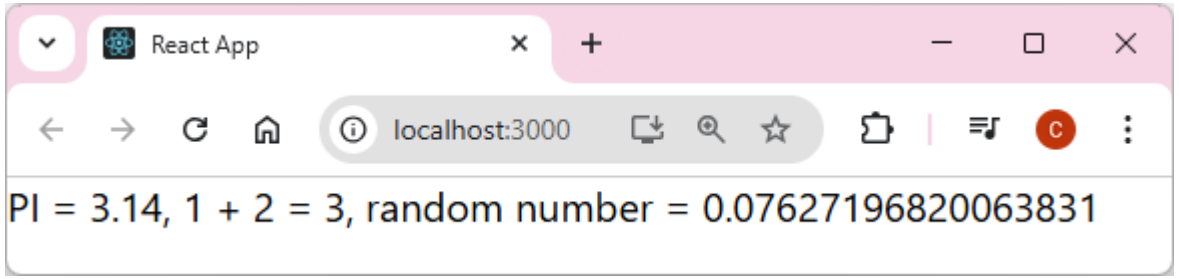
```
import { PI, add, Random } from './math'
```

เขียนโค้ด PI, add, Random ไว้ในไฟล์ math.js

Example

src > JS App.js > ...

```
1  import { PI, add, Random } from './math'
2
3  function App() {
4      return <>PI = {PI}, 1 + 2 = {add(1, 2)}, random number = {Random.next()}</>
5  }
6
7  export default App
```



src > JS math.js > ...

```
1  const PI = 3.14
2
3  function add(num1, num2) {
4      return num1 + num2
5  }
6
7  class Random {
8      static next() {
9          return Math.random()
10     }
11 }
12
13 export { PI, add, Random }
```

การสร้างและส่งออกโมดูล / การนำเข้าโมดูล (น. 24 - 26)

การนำเข้า constant, function, class ด้วย import เช่น

```
import { PI, add, Random } from './math'
import { add as plus } from './math'
import * as  math from './math'
```

ต้องมีไฟล์ math.js
เปลี่ยนชื่อ add เป็น plus
เปลี่ยนชื่อเป็น math.PI, math.add, math.Random สังเกตว่าไม่มีวงเล็บปีกกา

หรือจะใช้การส่งออกแบบ default เช่น

```
export default function add { … }
```

math.js แบบที่ 1

```
function add(num1, num2) { … }
export default add
```

math.js แบบที่ 2

```
import { add } from './math’
import add from './math'
import { default as plus } from './math'
```

ถ้าไม่ใช้ default การ import ใน App.js ต้องมีวงเล็บปีกกา ไม่ขึ้นกับจำนวนชิ้นที่ import
สังเกตว่าไม่ต้องมีวงเล็บปีกกา เพราะเติม default ใน math.js แล้ว
default คือ add เปลี่ยนชื่อเป็น plus

ไม่จำเป็นต้องใช้ default กับโมดูลที่มีสมาชิก const, function, class เพียงอันเดียว
แต่ในโมดูลหนึ่งจะมี default ได้แค่อันเดียวเท่านั้น (1 โมดูล = 1 ไฟล์ .js)

```
import add, { PI, Random } from './math'
```

add เป็น default, แต่ PI กับ Random ไม่ใช่ จึงต้องอยู่ในวงเล็บปีกกา

หลักการของ React JSX (JavaScript Syntax eXtension)

JavaScript ปกติ

```
const text = '<div>Hello world!</div>'
```

เขียนแบบ JSX

```
const text = <div>Hello world!</div>
```

ไม่ต้องมี single quote

หรือ

```
const text = (
        <div>
            Hello world!
        </div>
    )
```

ถ้ามีหลายบรรทัด ก็ใส่ไว้ในวงเล็บ

โปรแกรมที่แปลง JSX code เป็น JavaScript คือ **Transpiler**

เพื่อให้เขียนโค้ดแบบ JSX ได้ จะต้อง import React from 'react'

กำหนดค่าให้ตัวแปรเป็น HTML ได้ แต่ต้องอยู่ในแท็กเปิด/ปิด เช่น <table> … </table> หรือแท็กว่างก็ได้ <> … </> ในกรณีที่ไม่ต้องการ HTML tag

หลักการของ React JSX (ต่อ)

```
let a = 10
let b = 20
let x = <div>{a} + {b} = {a+b}</div>
```
ในวงเล็บปีกกาคือ นิพจน์ (expression) ดังนั้น a, b คือ ตัวแปร

```
let a = ['red', 'green', 'blue']
let b = <div>{a}</div>
```
b = ' redgreenblue '

```
let a = '<b>React</b>'
let b = <span>{a}</span>
let c = <span dangerouslySetInnerHTML={{__html: a}}/>
```
เพื่อให้ HTML tag ถูกประมวลผล ในที่นี้คือ …

```
let success = true
const msg = <div>{ success ? 'yes' : 'no' }</div>
```
Conditional (ternary) operator

```
let a = <div className="container">…</div>
let b = <label htmlFor="login">Login</label>
```
HTML attribute ใช้ class แต่ JSX attribute ใช้ className (จะได้ไม่ชนกับ class)

HTML attribute ใช้ for แต่ JSX attribute ใช้ htmlFor (จะได้ไม่ชนกับ for)

attribute ที่เกิดจาก 2 คำมารวมกันเขียนแบบ camel case เช่น maxLength tabIndex onClick onMouseDown (HTML เขียนยังไงก็ได้)

comment ใช้ /* … */ ใช้ <!-- … --> เหมือน HTML ไม่ได้

Differences between var, let, and const

var	let	const
The scope of a *var* variable is functional scope.	The scope of a *let* variable is block scope.	The scope of a *const* variable is block scope.
It can be updated and re-declared into the scope.	It can be updated but cannot be re-declared into the scope.	It cannot be updated or re-declared into the scope.
It can be declared without initialization.	It can be declared without initialization.	It cannot be declared without initialization.
It can be accessed without initialization as its default value is “undefined”.	It cannot be accessed without initialization, as it returns an error.	It cannot be accessed without initialization, as it cannot be declared without initialization.

https://www.geeksforgeeks.org/difference-between-var-let-and-const-keywords-in-javascript/

PascalCase .NET (C#)
camelCase JavaScript

`snake_case`

Pros: Concise when it consists of a few words.
Cons: Redundant as hell when it gets longer.
`push_something_to_first_queue, pop_what, get_whatever…`

`PascalCase`

Pros: Seems neat.
`GetItem, SetItem, Convert, ...`
Cons: Barely used. (why?)

`camelCase`

Pros: Widely used in the programmer community.
Cons: Looks ugly when a few methods are n-worded.
`push, reserve, beginBuilding, ...`

`skewer-case`

Pros: Easy to type.
`easier-than-capitals, easier-than-underscore, …`
Cons: Any sane language freaks out when you try it.

`SCREAMING_SNAKE_CASE`

Pros: Can demonstrate your anger with text.
Cons: Makes your eyes deaf.
`LOOK_AT_THIS, LOOK_AT_THAT, LOOK_HERE_YOU_MORON, …`

`nocase`

Pros: Looks professional.
Cons: Misleading af.
`supersexyhippothalamus, bool penisbig, …`

`fUcKtHeCaSe`

Pros: Can live outside of the law.
Cons: Can be out of a job.

ReactDOM (DOM = Document Object Model)

แบบที่ 1

```
import ReactDOM from 'react-dom'
ReactDOM.render(…)
```

แบบที่ 2

```
import { render } from 'react-dom'
render(…)
```

การใช้เมธอด render

```
render(element, container [, callback])
```

Note ปรับเนื้อหา เขียนอธิบายเพิ่ม

ReactDOM (DOM = Document Object Model)

The ReactDOM in React is responsible for **rendering the elements or Components in the actual DOM** of the web page. It is a package in React that **provides DOM-specific methods** that can be used at the top level of a web app to enable an efficient way of managing DOM elements of the web page. ReactDOM provides the developers with an API containing the various methods to manipulate DOM.

https://www.geeksforgeeks.org/reactjs-reactdom

public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

src/App.js เริ่มโค้ดที่นี่

```
function App() {
  return <div>... </div>
}

export default App
```

src/index.js

```
const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(
  <React.StrictMode>
    <App/>
  </React.StrictMode>
)
```

ถ้าพบว่า useEffect ซ้ำสองครั้ง หรือ call API ซ้ำสองครั้ง ให้เอา strict mode ออก

StrictMode is a tool for highlighting potential problems in an application.

https://reactjs.org/docs/strict-mode.html

Style (CSS)

แบบที่ 1: Inline

```
const divStyle = {
    color: 'red',
    backgroundColor: 'powerBlue',
    fontSize: 'larger'
}

return <div style={divStyle}>Hello world</div>
```

หรือจะใส่อาร์เรย์ของสไตล์แทน divStyle ก็ได้ เช่น [style1, style2] เป็นต้น

```
<div style={{color: 'red', backgroundColor: 'powderblue' }}>
  Hello world
</div>
```

แบบที่ 2: External

สร้างไฟล์ style.css เก็บไว้โฟลเดอร์ src แล้ว import เข้ามาใช้งาน เช่น

```
import './style.css'
```

Array.map()

```
array.map(function(item, [index]) {
   return …
})
```

หรือเขียนแบบ arrow function

```
array.map((item, [index]) => {
   return …
})
```

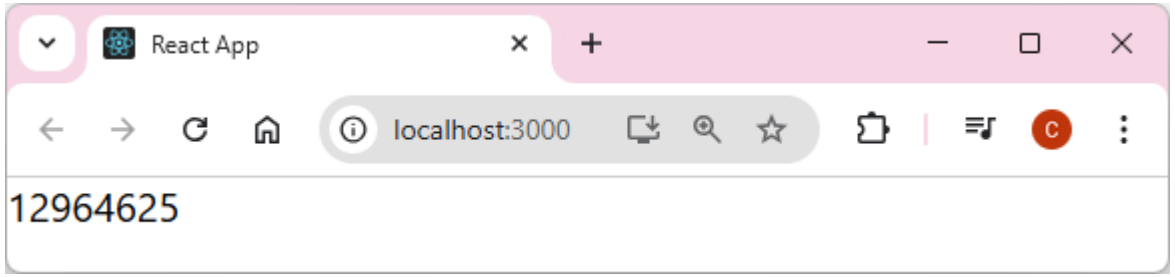
หรือเขียนแบบตั้งชื่อฟังก์ชัน

```
function xxx(item, [index]) {
   return
}
array.map(xxx)
```

array.map(…) จะทำฟังก์ชัน xxx กับสมาชิกทุกตัวในอาร์เรย์แล้วคืนค่าเป็น array
แต่ {array} ใน React จะเอาสมาชิกทุกตัวแปลงเป็น string แล้วเอามาต่อกัน

App.js

```
src > App.js > ...
1   function App() {
2
3     const numbers = [1,2,3,4,5]
4
5     return <>{numbers.map((x, i) => {
6       return x ** i
7     })}</>
8   }
9
10  export default App
```



ไฟล์รูปภาพ

เก็บไฟล์รูปภาพไว้ในโฟลเดอร์ public และนำมาใช้ดังนี้

```
<img src="./logo.svg" alt=""/>
```

วิธีแรกใช้กับการติดตั้งใน subdirectory ไม่ได้ ให้เก็บไฟล์รูปภาพไว้ในโฟลเดอร์ src และนำมาใช้ดังนี้

```
import logo from './logo.svg'

function App() {
   return <img src={logo} width="50%" alt=""/>
}
```

โปรเจ็ค React ปกติจะตั้งค่าสำหรับติดตั้งบน domain name หรือ sub-domain name

https://xxx.chula.ac.th

ตัวอย่าง subdirectory

https://chula.ac.th/xxx

Class

Arrow function

```
() => {
        …
        …
        return …
}

(a, b) => {
        …
        …
        return a + b
}
```

Functional Component

```
import React from 'react'

function Header() {
   ...
   return ( ... )
}

export default Header
```

นิยมตั้งชื่อฟังก์ชันแบบ PascalCase

ใน App.js เอา component มาประกอบกัน เช่น

```
import { Header } from './Header.js'
import { Content } from ...
import { Footer } from ...

return [<Header/>, <Content/>, <Footer/>]
```

หรือเขียนแบบนี้ก็ได้

```
return {
  <>
    <Header/>
    <Content/>
    <Footer/>
  </>
}
```

การนำ component ไปใส่ใน component อื่น ๆ ก็อ้างถึง <Header/> <Content/> <Footer/> ได้เลย

Props

```
import React from 'react'

function Header(props) {
   ...
   return <div>{props.text}</div>
}

export default Header
```

เมื่อจะใช้ก็ import เข้ามาดังนี้

```
import Header from './Header.js'
```
ไม่ต้องมี .js ก็ได้

เมื่อใช้ Header ก็ใส่ props ไปด้วย เช่น <Header text="Hello word"/> เป็นต้น
ถ้ามี props มากกว่า 1 ตัว ก็เขียนต่อกันไป

ในฟังก์ชัน Header ตัวแปร props เป็น read-only จะไป assign ค่าใหม่ เช่น ให้ props.text = 'Hi' ไม่ได้

Class Component

src > JS Header.js > ...

```jsx
import React from 'react'

function Header(props) {
    return <center>{props.code} {props.course}</center>
}

export default Header
```

src > JS Content.js > ...

```jsx
import React from 'react'

function Content(props) {
    return <ul>
        {
            props.fruits.map((x) => {
                return <li>{x}</li>
            })
        }
        </ul>
}

export default Content
```

src > JS Footer.js > ...

```jsx
import React from 'react'

function Footer(props) {
    return <center>{props.id} {props.name}</center>
}

export default Footer
```

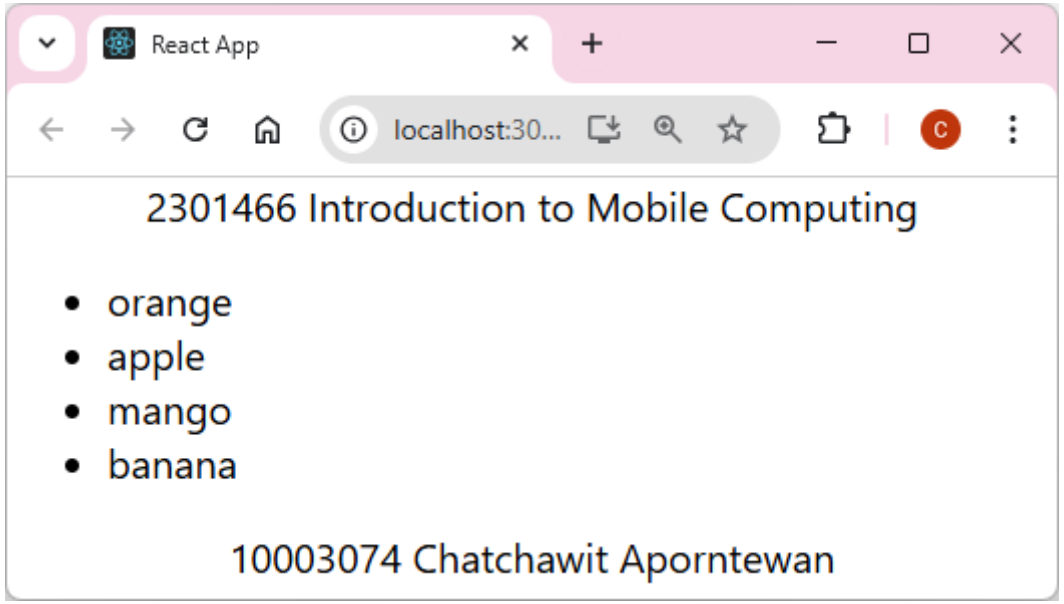
src > JS App.js > ...

```jsx
import Header from './Header'
import Content from './Content'
import Footer from './Footer'

function App() {

  return <>
            <Header code="2301466" course="Introduction to Mobile Computing"/>
            <Content fruits={['orange', 'apple', 'mango', 'banana']}/>
            <Footer id='10003074' name='Chatchawit Aporntewan'/>
        </>
}

export default App
```



ลองส่ง style, function ผ่าน props

JavaScript Events

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

https://www.w3schools.com/jsref/dom_obj_event.asp

```
function doSomething() {
   alert('You clicked button')
}

return <button onClick={() => doSomething()}>Ok</button>
```

JSX เขียนแบบ camelCase

HTML เขียนยังไงก็ได้

```
function doSomething(mesg) {
   alert(mesg)
}

return <button onClick={() => doSomething('hi')}>Ok</button>
```

```
function doSomething(x, y) {
   alert(x + y)
}

return <button onClick={() => doSomething(1, 2)}>Ok</button>
```

Refs

State

```
import React, {useState} from 'react'

function MyComponent() {
    let [fontSize, setFontSize] = useState(16)
}
```

fontSize เป็นตัวแปรแบบ state มีค่าเริ่มต้นคือ 16
setFontSize เป็นฟังกันสำหรับแก้ไขค่าตัวแปร fontSize เนื่องจากไม่สามารถแก้ไขค่าได้โดยตรง
เช่น setFontSize(18) ได้ แต่ fontSize = 18 ไม่ได้

หรือเขียนอีกแบบ

```
import React from 'react'

function MyComponent() {
    let [...] = React.useState()
}
```

สร้าง textbox 3 ตัว สร้างปุ่มที่เมื่อคลิกแล้วจะนำค่าใน textbox 2 ตัวแรก มาบวกกัน แล้วแสดงผลบวกใน textbox ตัวที่ 3

```
function App() {

    let [a, setA] = useState('')
    let [b, setB] = useState('')
    let [c, setC] = useState('')

    return (
        <div>
            <input type="number" placeholder='enter the first number' value={a} onChange={(event)=>setA(event.target.value)}/><br/>
            <input type="number" placeholder='enter the second number' value={b} onChange={(event)=>setB(event.target.value)}/><br/>
            <input type="number" placeholder='here the sum' value={c}/><br/>
            <button onClick={() => setC(Number(a)+Number(b))}>Ok</button>
        </div>
    )
}
```

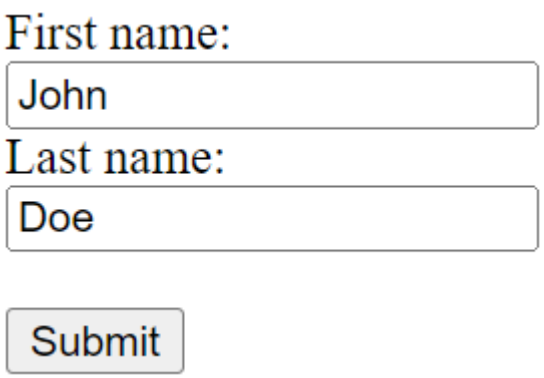
เมื่อค่าของตัวแปร state มีการเปลี่ยนแปลง UI จะเปลี่ยนตามทันที!

ตัวแปร state อาจจะเป็นยอด likes ยอด views เป็นขนาดตาราง m×n เป็น boolean ที่บอกว่าแสดงหรือซ่อน

HTML Form

เมื่อคลิกปุ่ม submit จะ POST ไปที่นี่ เขียนโค้ดแบบนี้คือให้เบราเซอร์ POST ให้

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```



https://www.w3schools.com/html/html_forms.asp

HTML Input

- `<input type="button">`
- `<input type="checkbox">`
- `<input type="color">`
- `<input type="date">`
- `<input type="datetime-local">`
- `<input type="email">`
- `<input type="file">`
- `<input type="hidden">`
- `<input type="image">`
- `<input type="month">`
- `<input type="number">`
- `<input type="password">`
- `<input type="radio">`
- `<input type="range">`
- `<input type="reset">`
- `<input type="search">`
- `<input type="submit">`
- `<input type="tel">`
- `<input type="text">` (default value)
- `<input type="time">`
- `<input type="url">`
- `<input type="week">`

https://www.w3schools.com/tags/tag_input.asp

Context

Route (ในเว็บไซต์)

npm install react-router-dom

ใน index.js

```
import { BrowserRouter } from 'react-router-dom'
… … …
<BrowserRouter>
   <App/>
</BrowserRouter>
```

ใน app.js

```
import { Routes, Route } from 'react-router-dom'
function App() {
 return (
  <Routes>
     <Route path="/" element={<Login/>} />
     <Route path="/login" element={<Login/>} />
     <Route path="/main" element={<Main/>} />
      <Route path="/credit" element={<Credit/>} />
  </Routes>
 )
}
```

การนำไปใช้ใน component

```
import { useNavigate } from 'react-router-dom'
…
let navigate = useNavigate()
…
<button onClick={() => navigate('/main')}>เข้าสู่ระบบ</button>
```

การบ้าน

สร้างหน้าล็อกอิน มี userid / password ปุ่ม sign in
เมื่อล็อกอินเข้าไปมีอย่างน้อย 2 หน้าคือ main และ credit
app คือ การเพิ่ม/แก้ไข/ลด กิจกรรม credit คือ คณะผู้จัดทำ
สามารถ navigate ไปยังหน้าต่าง ๆ ได้ และ sign out ได้

Effect

```
function MyComponent() {

   useEffect(function() {

      // สิ่งที่จะทำเมื่อเกิด effect

   }, [a, b, c, ...])
```

a, b, c, ... คือ ตัวแปรที่เมื่อเปลี่ยนค่าแล้วจะเกิด effect
ถ้าจะให้ทำ effect เฉพาะครั้งแรกที่โหลด component ให้ใช้ [ ] (empty array)

```
}
```

REST API

```
fetch(url, options)
    .then(…)
    .catch(…)
    .final(…)
```

ใช้ Axios ดีกว่า

npm install axios

import React, { useEffect } from 'react'
import axios from 'axios'

ตั้ง baseURL ได้ เช่น axios.defaults.baseURL = 'http://localhost:3000'
ระบุไว้ใน App() ครั้งเดียวก็พอ เวลา call API ก็ระบุแค่ /controller

```
function App() {

    useEffect(() => {
        axios.get('https://learningportal.ocsc.go.th/learningspaceapi/localdatetime').then((response) => {
            alert('status = ' + response.status + ' datetime = ' + response.data.datetime)
        }).catch((error) => {
            finally
        }).then(() => {

        })
    }, [])

    return <div>A X I O S</div>
}
```

ดัก status code ทั้งหมดที่ไม่ใช่ 2xx รวมถึง network error เช่น CORS, connection timeout

response.status ได้ 200
response.statusText ได้ 'OK'

https://axios-http.com/

แก้ Cross-Origin Resource Sharing (CORS)

```
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseCors(Options => Options.AllowAnyOrigin().AllowAnyMethod().AllowAnyHeader());

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

ถ้าเราเป็นเจ้าของ API ก็แก้โค้ดของตัวเองได้ เพื่อทดสอบบน web browser บน postman ไม่เกิด error นี้

Deploy a project in subdirectory

โดยปกติ npm run build แล้วก็อปปี้ไฟล์ทั้งหมดในโฟลเดอร์ build ไปติดตั้งบนเว็บไซต์ เช่น domain.com ก็เรียบร้อย
แต่ถ้าจะติดตั้งบน subfolder เช่น domain.com/subfolder จะต้องทำดังนี้

- ใน package.json เติม "homepage": "/project-name", เช่น ไว้ต่อจากบรรทัด "name": ...
- {process.env.PUBLIC_URL} และ %PUBLIC_URL% จะมีค่าเท่ากับ "/project-name"
- ใน index.html เติม <base href="%PUBLIC_URL%/"> ไว้ใน <head>...</head> เช่น ไว้ก่อนบรรทัด <title>...</title>
- ใน index.js เติม basename ดังนี้ <BrowserRouter basename={process.env.PUBLIC_URL}>
- ใน app.js กำหนด routes โดยไม่ต้องใส่ /project-name หรือ process.env.PUBLIC_URL เพราะกำหนด basename ไว้แล้ว เช่น

```
<Routes>
    <Route path="/" element={<SignIn/>} />
    <Route path="/main" element={<Main/>}/>
</Routes>
```

- ใช้รูปจากโฟลเดอร์ src เช่น import xxx from './image.png' และ ได้ตามปกติ
- ใช้รูปจากโฟลเดอร์ public เช่น ต้องใช้ ./ เท่านั้น ไม่งั้นบน production จะ error
- ใน development URL จะไม่มี project-name เช่น localhost:3000 หรือ localhost:3000/main
- ใน production URL จะมี project-name เช่น domain.com/project-name หรือ domain.com/project-name/main
- ใน navigate ไม่ต้องมี project-name เช่น <button onClick={() => navigate('/main')}>Go</button>
- ใน sidebar ใช้ <Link to="/main">Main</Link> โดย import { Link } from 'react-router-dom'
- รัน front-end บน local (เพื่อทดสอบ UI ใหม่) และใช้ back-end ของ production (API และข้อมูลจริง) ได้

https://skryvets.com/blog/2018/09/20/an-elegant-solution-of-deploying-react-app-into-a-subdirectory/

4. **Adjust the router**. Since every react-router uses history package, we need to import it and make some adjustments. First of all, install history package if you don't have it:

```
npm install --save history
```

In your application, add the following lines of code:

```
import { createBrowserHistory } from 'history';

...
                เอา history ไปส่งต่อให้ router
export const history = createBrowserHistory({
    basename: process.env.PUBLIC_URL
});

...
```

ข้อจำกัด

ปัจจุบันทำไม่ได้แล้ว เพราะพารามิเตอร์ **basename** ถูกยกเลิกไป

This will prepend necessary path and make router work in each environment: local as well as production.

Step 1

Install **react router dom**.

```
npm install --save react-router-dom
```

Step 2

Import the history package from react router dom.

```
import { useHistory } from "react-router-dom"
```

Step 3

Assign the history function to a variable (not necessary but recommended)

```
const history = useHistory()
```

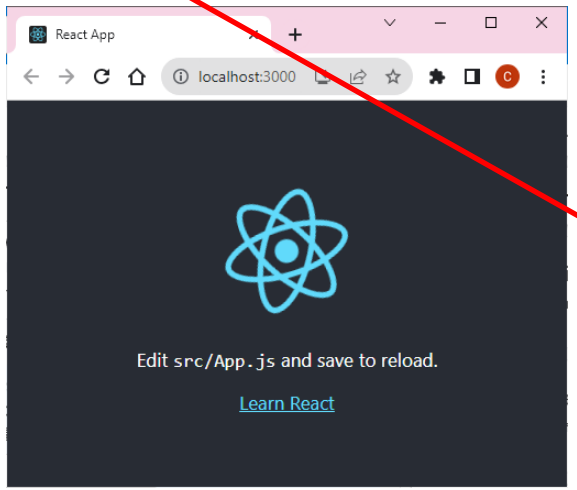
Step 4

Use the `push()` function to redirect the user *after a successful login, for example.*

```
history.push("/dashboard")
```

เลิกใช้ **history.push** แล้ว
ปัจจุบันใช้ **navigate** แทน

Create a project for deploying in a subdirectory



package.json

```
package.json > {} dependencies > @testing-library/user-event
1  {
2    "name": "ocsc-classroom",
3    "homepage": "/ocsccclassroom",
4    "version": "0.1.0",
5    "private": true,
```

index.html

```
public > index.html > html
27      <base href="%PUBLIC_URL%/">
28      <title>React App</title>
29    </head>
```

```
src > JS index.js > ...
7   const root = ReactDOM.createRoot(document.getElementById('root'));
8   root.render(
9     <BrowserRouter basename={process.env.PUBLIC_URL}>
10      <App />
11    </BrowserRouter>
12  );
```