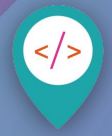


Software Park Thailand
</Code Camp>

JAVASCRIPT JS

Nuttachai Kulthammanit

Slide Design By Patteera Banlangthammas

หัวข้อ

Software Park Thailand
</Code Camp>

- ตัวดำเนินการแบบตรรกะ
- วงวน for และ while
- Switch Cases
- ฟังก์ชัน
- Function Expression
- Arrow Function

ตัวดำเนินการแบบตรรกะ

1. ตัวดำเนินการแบบตรรกะ

1.1. หรือ - OR (||)

- การใช้ตัวดำเนินการแบบ หรือ จะใช้เครื่องหมาย ||
- ถ้ามีเงื่อนไขใดเงื่อนไขหนึ่งเป็นจริงจะเป็นจริงทันที
- สามารถใส่ได้มากกว่า 1 เงื่อนไข

```
if (เงื่อนไข1 || เงื่อนไข2 || เงื่อนไข3)
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.1. หรือ - OR (||)

- ตัวอย่างการใช้ OR

```
alert(true || true);   // true
alert(false || true);  // true
alert(true || false);  // true
alert(false || false); // false
alert(false || false || true); //true
alert(true || false || false); //true
```

1. ตัวดำเนินการแบบตรรกะ

1.2. คุณสมบัติพิเศษ JavaScript กับ OR

- OR จะหาตัวที่เป็น truthy value ตัวแรก (OR finds the first truthy value)
- วิธีการหา truthy value ของ OR
  1. เริ่มเช็คค่าจาก ซ้าย ไป ขวา
  2. JavaScript จะแปลง Operand แต่ละตัว เป็น boolean ถ้าแปลงแล้วได้เป็น true ก็จะจบ OR ทันที และ คืนค่าที่เป็น Original ของ Operand นั้น
  3. ถ้า Operands ทุกตัวถูกแปลงเป็นเป็น boolean หมดแล้วแต่ยังไม่เจอ true เลย(เป็น false หมด) ก็จะคืนตัวสุดท้ายออกไป

*** พูดง่าย ๆ ก็คือ OR จะหาตัวแรกที่เป็น truthy และคืนค่านั้นออกมา และจะคืนค่าตัวสุดท้ายออกมาถ้าทุกตัวเป็น falsy**

1. ตัวดำเนินการแบบตรรกะ

1.2. คุณสมบัติพิเศษ JavaScript กับ OR

- วิธีการหา truthy value ของ OR - ตัวอย่างที่ 1

1. เริ่มเช็คค่าจาก ซ้าย ไป ขวา
2. JavaScript จะแปลง Operand แต่ละตัว เป็น boolean ถ้าแปลงแล้วได้เป็น true ก็จะจบ OR ทันที และ คืนค่าที่เป็น Original ของ Operand นั้น
3. ถ้า Operands ทุกตัวถูกแปลงเป็นเป็น boolean หมดแล้วแต่ยังไม่เจอ true เลย(เป็น false หมด) ก็จะคืนตัวสุดท้ายออกไป

```
alert(1 || 0);
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.2. คุณสมบัติพิเศษ JavaScript กับ OR

- วิธีการหา truthy value ของ OR - ตัวอย่างที่ 2

1. เริ่มเช็คค่าจาก ซ้าย ไป ขวา
2. JavaScript จะแปลง Operand แต่ละตัว เป็น boolean ถ้าแปลงแล้วได้เป็น true ก็จะจบ OR ทันที และ คืนค่าที่เป็น Original ของ Operand นั้น
3. ถ้า Operands ทุกตัวถูกแปลงเป็นเป็น boolean หมดแล้วแต่ยังไม่เจอ true เลย(เป็น false หมด) ก็จะคืนตัวสุดท้ายออกไป

```
alert(true || 'no matter what');
```

1. ตัวดำเนินการแบบตรรกะ

1.2. คุณสมบัติพิเศษ JavaScript กับ OR

- ตัวอย่าง

```
alert(null || 1); // 1 (1 เป็น truthy value ตัวแรก)
alert(null || 0 || 1); // 1 (truthy value ตัวแรก)
alert(undefined || null || 0); // 0 (ทั้งหมดเป็น
falsy, คืนค่าสุดท้ายออกมา)
```

1. ตัวดำเนินการแบบตรรกะ

1.2. คุณสมบัติพิเศษ JavaScript กับ OR - การประยุกต์

- ใช้เมื่อต้องการใช้ค่าใดค่าหนึ่งจากค่าที่มีทั้งหมด เช่น

```
let currentUser = null;
let defaultUser = 'John';

/*
ถ้ามี currentUser จะใช้ currentUser
ถ้าไม่มี currentUser จะใช้ defaultUser แทน
ถ้าไม่มี currentUser จะใช้เป็น String ที่มีค่า 'Unknown'
*/
let userName = currentUser || defaultUser || 'Unknown'
```

1. ตัวดำเนินการแบบตรรกะ

1.3. Short circuit กับ OR

- การดำเนินการแบบตรรกะ จะเกิดขึ้นจากซ้ายไปขวา
- ถ้ามี True เกิดขึ้น OR จะจบการทำงานทันที
- ตัวอย่าง

```
let x;

true || (x = 1);

alert(x); // undefined, เพราะว่า OR จบก่อนที่จะรันคำสั่ง x = 1
```

1. ตัวดำเนินการแบบตรรกะ

1.3. Short circuit กับ OR

- ตัวอย่างที่ 2

```
let x;

false || (x = 1);

alert(x); // 1, เพราะว่า OR จะรันคำสั่ง x = 1 ด้วย
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.3. Short circuit กับ OR

- ตัวอย่างใน React/Redux

```
const mapStateToProps = (state) => ({
    isAuth: state.auth.uid || 0
});
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.4. และ - AND (&&)

- การใช้ตัวดำเนินการแบบ และ จะใช้เครื่องหมาย &&
- ถ้ามีเงื่อนไขใดเงื่อนไขหนึ่งเป็นเท็จจะเป็นเท็จทันที
- สามารถใส่ได้มากกว่า 1 เงื่อนไขเช่นกัน

```
if (เงื่อนไข1 && เงื่อนไข2 && เงื่อนไข3)
```

1. ตัวดำเนินการแบบตรรกะ

1.4. และ - AND (&&)

- ตัวอย่างการใช้ AND

```
alert(true && true);   // true
alert(false && true);  // false
alert(true && false);  // false
alert(false && false); // false
alert(true && true && false); //false
alert(true && true && true); // true
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.5. คุณสมบัติพิเศษของ JavaScript กับ AND

- AND จะหาตัวที่เป็น falsy value ตัวแรก (AND “&&” finds the first falsy value)
- วิธีการหา falsy value ของ AND
  1. เริ่มเช็คค่าจาก ซ้าย ไป ขวา
  2. JavaScript จะแปลง Operand แต่ละตัว เป็น boolean ถ้าแปลงแล้วได้เป็น true ก็จะจบ AND ทันที และ คืนค่าที่เป็น Original ของ Operand นั้น
  3. ถ้า Operands ทุกตัวถูกแปลงเป็นเป็น boolean หมดแล้วแต่ยังไม่เจอ false เลย(เป็น true หมด) ก็จะคืนตัวสุดท้ายออกไป

*** พูดง่าย ๆ ก็คือ AND จะหาตัวแรกที่เป็น falsy และคืนค่านั้นออกมา และจะคืนค่าตัวสุดท้ายออกมาถ้าทุกตัวเป็น falsy**

1. ตัวดำเนินการแบบตรรกะ

1.5. คุณสมบัติพิเศษของ JavaScript กับ AND

- ตัวอย่าง

```
// ถ้า operand ตัวแรกเป็น truthy,
// AND จะคืนค่าตัวที่ 2 ออกมา:
alert(1 && 0); // 0
alert(1 && 5); // 5
```

1. ตัวดำเนินการแบบตรรกะ

1.5. คุณสมบัติพิเศษของ JavaScript กับ AND

- ตัวอย่าง

```
// ถ้า operand ตัวแรกเป็น falsy,
// AND จะคืนค่าตัวแรก. และ Operand ตัวที่ 2 จะถูกเมิน
alert(null && 5); // null
alert(0 && "no matter what"); // 0
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.6. คุณสมบัติพิเศษของ JavaScript กับ AND - การประยุกต์

- ใช้เมื่อต้องการให้คำสั่งข้างหลังทำงานเมื่อค่าของตัวแรกเป็น true
- ใช้บ่อยใน React

```
let x = 1;

(x > 0) && alert( 'Greater than zero!' );
```

1. ตัวดำเนินการแบบตรรกะ

1.6. คุณสมบัติพิเศษของ JavaScript กับ AND - การประยุกต์

- ตัวอย่าง ใน React

```
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
      <h1>Hello!</h1>
      {unreadMessages.length > 0 &&
        <h2>
          You have {unreadMessages.length} unread messages.
        </h2>
      }
    </div>
  );
}
```

1. ตัวดำเนินการแบบตรรกะ

1.7. ความสำคัญระหว่าง AND กับ OR

- ใน JavaScript AND จะมีความสำคัญกว่า OR (ทำ AND ก่อน OR)
- เช่น

```
a && b  ||  c && d  ||  e && f  || g
```

จะเปรียบเสมือน

```
(a && b) || (c && d) || (e && f) || g
```

1. ตัวดำเนินการแบบตรรกะ

1.8. นิเสธ - NOT (!)

- ใช้อยู่ทั้งหมด 2 กรณี
  1. เมื่อต้องการแปลงข้อมูลเป็นประเภท Boolean (Double NOT)
     a. result = Boolean(value)
     b. result = !!value

** ทั้ง a และ b ได้ผลลัพธ์เหมือนกัน

1. ตัวดำเนินการแบบตรรกะ

1.8. นิเสธ - NOT (!)

- ตัวอย่าง Double NOT

```
          !!false === false
           !!true === true

             !!0 === false
!!parseInt("foo") === false // NaN is falsy
             !!1 === true
            !!-1 === true  // -1 is truthy

            !!"" === false // empty string is falsy
         !!"foo" === true  // non-empty string is truthy
       !!"false" === true  // ...even if it contains a falsy value

    !!window.foo === false // undefined is falsy
          !!null === false // null is falsy
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.8. นิเสธ - NOT (!)

- ใช้อยู่ทั้งหมด 2 กรณี

  2. เมื่อต้องการแปลงข้อมูลเป็นค่าตรงข้าม

  - result = !value

```
alert( !true );    // false
alert( !0 );       // true
```

1. ตัวดำเนินการแบบตรรกะ

1.9. แบบฝึกหัด

1. คำสั่งต่อไปนี้จะแสดงค่าเป็นอะไร

- `alert( null || 2 || undefined );`
- `alert( alert(1) || 2 || alert(3) );`
- `alert( 1 && null && 2 );`
- `alert( alert(1) && alert(2) );`
- `alert( null || 2 && 3 || 4 );`

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.9. แบบฝึกหัด

2. เขียนคำสั่ง if ที่เช็คอายุว่าอยู่ระหว่าง 18 - 60

3. เขียนคำสั่ง if ที่เช็คอายุว่าไม่อยู่ระหว่าง 18 - 60

1. ตัวดำเนินการแบบตรรกะ

1.9. แบบฝึกหัด

4. คำสั่ง alert ไหนที่จะถูกรันบ้าง

```
- if (-1 || 0) alert( 'first' );
- if (-1 && 0) alert( 'second' );
- if (null || -1 && 0) alert( 'third' );
```

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.9. แบบฝึกหัด

5. ให้เขียนระบบ login

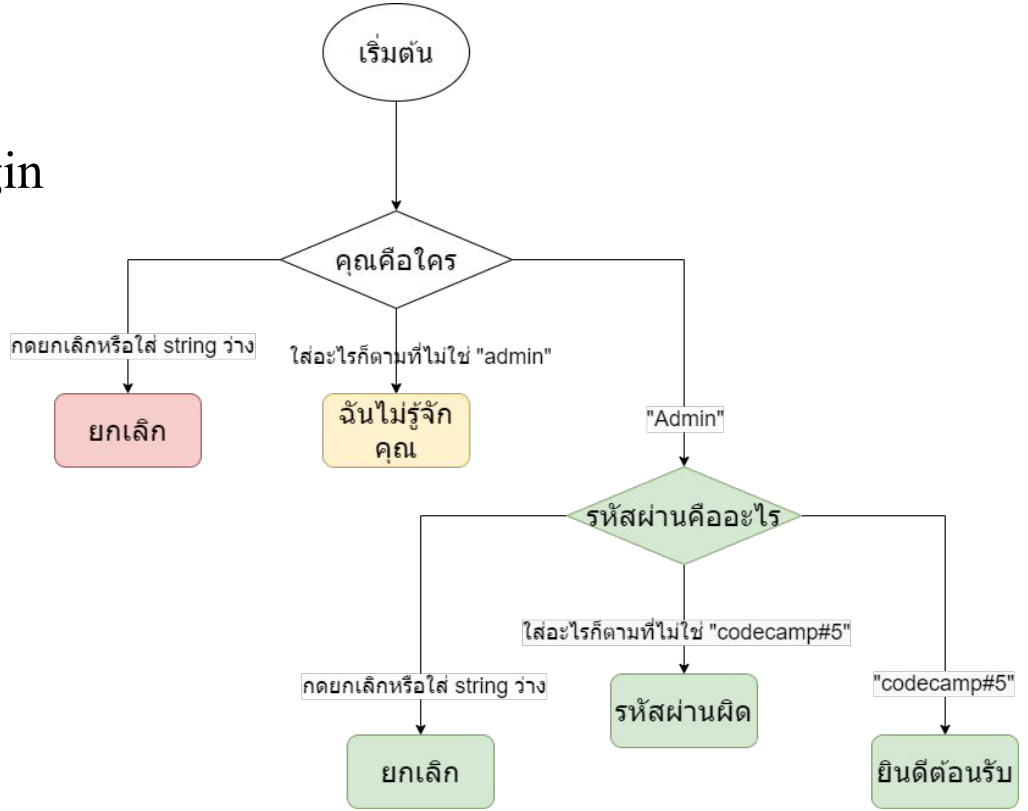
- ให้ใช้ prompt ในการถามใครเป็นคน login
- ถ้าผู้ใช้กรอกว่า “Admin” ให้ใช้ prompt ถาม password
    - วิธีเช็ค Password
    - ถ้า string นั้นเป็น “codecamp#5” ให้ alert “ยินดีต้อนรับ”
    - ถ้า string เป็นอย่างอื่นให้ alert เป็น “Wrong password”
    - ถ้าเป็น string ว่าง หรือ กด cancel ให้ alert ว่า “ยกเลิก”
- ถ้าผู้ใช้กรอกอย่างอื่นที่ไม่ใช่ “Admin” ให้ alert ว่า “ผมไม่รู้จักคุณ”
- ถ้าผู้ใช้กรอก input เป็น string ว่าง หรือกด Esc ให้ alert ว่า “ยกเลิก”

1. ตัวดำเนินการแบบตรรกะ

Software Park Thailand
</Code Camp>

1.9. แบบฝึกหัด

5. ให้เขียนระบบ login



หัวข้อ

Software Park Thailand
</Code Camp>

- ตัวดำเนินการแบบตรรกะ
- วงวน for และ while
- Switch Cases
- ฟังก์ชัน
- Function Expression
- Arrow Function

วงวน for และ while

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.1. วงวน while

- while จะรันคำสั่งใน loop body ไปเรื่อย ๆ จนกว่าเงื่อนไขใน while จะเป็น false

```
while (เงื่อนไข) {
  // คำสั่ง
  // เรียกว่า "loop
body"
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.1. วงวน while

- ตัวอย่าง

```
let i = 0;
while (i < 3) { // แสดง 0, 1 และ 2
  alert( i );
  i++;
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.2. วงวน for

- for จะรันคำสั่งใน loop body ไปเรื่อย ๆ เหมือนกับ while loop แต่จะมีการกำหนดค่าเริ่มต้น และค่าต่อไปได้

```
for (ค่าเริ่มต้น; เงื่อนไข; ค่าต่อไป) {
  // ... loop body ...
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.2. วงวน for

- ตัวอย่าง

```
let i = 0;
while (i < 3) {
  alert(i);
  i++;
}
```

```
for (let i = 0; i < 3; i++) {
  alert(i);
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.3. break การหยุดวงวน

- เมื่อโปรแกรมเจอ break โปรแกรมจะออกจาก loop ที่กำลังรันอยู่

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.3. break การหยุดวงวน

- ตัวอย่างที่ 1

```
for (let i = 1; i <= 9; i++) {
  console.log(i)
  if (i == 6) {
    break;
  }
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.3. break การหยุดวงวน

- ตัวอย่างที่ 2

```
for (let i = 1; i <= 9; i++) {
  if (i == 6) {
    break;
  }
  console.log(i)
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.3. break การหยุดวงวน

- ตัวอย่างที่ 3

```
let sum = 0;
while (true) {
  let value = +prompt("ใส่เลข", '');
  if (!value) break; // (*)
  sum += value;
}
alert('Sum: ' + sum);
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.4. continue

- เมื่อโปรแกรมเจอ continue โปรแกรมจะจบ loop รอบนั้น ที่กำลังรันอยู่ และข้ามไปรอบใหม่ทันที

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.4. continue

- ตัวอย่างที่

```
for (let i = 0; i < 10; i++) {

  // ถ้าเป็นเงื่อนไขเป็น true คำสั่งหลังจากบรรทัดนี้จะถูกข้าม
  if (i % 2 == 0) continue;

  alert(i); // 1, then 3, 5, 7, 9
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.5. แบบฝึกหัด

1. เลขที่ถูก alert เป็นลำดับสุดท้ายคือเลขอะไร

```
let i = 3;

while (i) {
  alert( i-- );
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.5. แบบฝึกหัด

2. code ทั้งสองอันนี้จะแสดง alert ออกมาเหมือนกันทั้งหมดหรือไม่

```
let i = 0;
while (++i < 5) alert( i );
```

```
let i = 0;
while (i++ < 5) alert( i );
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.5. แบบฝึกหัด

3. code ทั้งสองอันนี้จะแสดง alert ออกมาเหมือนกันทั้งหมดหรือไม่

```
for (let i = 0; i < 5; i++) alert( i );
```

```
for (let i = 0; i < 5; ++i) alert( i );
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.5. แบบฝึกหัด

4. ให้เขียน loop ทั้งแสดงเลข 2 ถึง 10 ออกมา

5. เปลี่ยน code for loop ด้านล่างนี้ให้เป็น while loop โดยที่ผลลัพธ์ยังเหมือนเดิม

```
for (let i = 0; i < 3; i++) {
  alert( `number ${i}!` );
}
```

2. วงวน for และ while

Software Park Thailand
</Code Camp>

2.5. แบบฝึกหัด

6. ให้เขียนเกมส์ทายตัวเลขสำหรับเล่นสองคน โดย

- ให้ผู้เล่นคนแรกพิมพ์เลขใส่ใน prompt ที่อยู่ระหว่าง 1 ถึง 100 โดยไม่ให้ผู้เล่นคนที่สองรู้ว่าตัวเลขเป็นอะไร
- และให้ผู้เล่นคนที่สองทายเลขโดยการพิมพ์เลขใส่ใน prompt จนกว่าจะถูก ถ้าไม่ถูก จะต้องบอกด้วยว่าเลขที่ผู้เล่นคนที่สองพิมพ์เข้ามา มากกว่าหรือ น้อยกว่าคำตอบนั้น

หัวข้อ

Software Park Thailand
</Code Camp>

- ตัวดำเนินการแบบตรรกะ
- วงวน for และ while
- Switch Cases
- ฟังก์ชัน
- Function Expression
- Arrow Function

Software Park Thailand
</Code Camp>

Switch Cases

3. Swtich Cases

3.1. Swtich cases

- ใช้เมื่อต้องการ match ค่าของตัวแปร กับคำสั่งที่ต้องการทำ
- Syntax ของการเขียน Swtich
- เป็น Type Matter ด้วย (เหมือนกับการเปรียบเทียบด้วย ===)

```
switch(x) {
  case 'value1':  // if (x === 'value1')
    // คำสั่ง
    [break]
  case 'value2':  // if (x === 'value2')
    // คำสั่ง
    [break]
  default:
    // คำสั่ง
    [break]
}
```

3. Swtich Cases

Software Park Thailand
</Code Camp>

3.1. Swtich cases

- ตัวอย่างที่ 1

```
let a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Too small' );
    break;
  case 4:
    alert( 'Exactly!' );
    break;
  case 5:
    alert( 'Too large' );
    break;
  default:
    alert( "ไม่ตรงกับ case ใด ๆ เลย" );
}
```

3. Swtich Cases

Software Park Thailand
</Code Camp>

3.1. Swtich cases

- ตัวอย่างที่ 2
- ถ้าไม่ใส่ break หลังจาก case match แล้วตัวข้างล่างจะถูกรันต่อไปจนจบ

```
let a = 2 + 2;

switch (a) {
  case 3:
    alert( 'Too small' );
  case 4:
    alert( 'Exactly!' );
  case 5:
    alert( 'Too big' );
  default:
    alert( "ไม่ตรงกับ case ใด ๆ เลย );
}
```

3. Swtich Cases

3.1. Swtich cases

- การรวม case หลาย cases ไว้ด้วยกัน
- จากตัวอย่างเป็นการรวม case 3 กับ case 5 ไว้ด้วยกัน

```
let a = 2 + 2;

switch (a) {
  case 4:
    alert('Right!');
    break;

  case 3: // (*) grouped two cases
  case 5:
    alert('Wrong!');
    alert("Why don't you take a math class?");
    break;

  default:
    alert('The result is strange. Really.');
}
```

3. Swtich Cases

Software Park Thailand
</Code Camp>

3.2. Switch cases กับ if-else

```
let a = 2 + 2;
switch (a) {
  case 3:
    alert( 'Too small' );
    break;
  case 4:
    alert( 'Exactly!' );
    break;
  case 5:
    alert( 'Too large' );
    break;
  default:
    alert( "ไม่ตรงกับ case ใด ๆ เลย" );
}
```

```
let a = 2 + 2;

if (a === 3) {
  alert('Too small');
} else if (a === 4) {
  alert('Exactly!');
} else if (a === 5) {
  alert('Too big');
} else {
  alert("ไม่ตรงกับค่าใด ๆ เลย")
}
```

3. Swtich Cases

Software Park Thailand
</Code Camp>

3.3. แบบฝึดหัด

1. แปลง Code ดังกล่าวเป็น if-else statement

```
switch (browser) {
  case 'Edge':
    alert( "You've got the Edge!" );
    break;

  case 'Chrome':
  case 'Firefox':
  case 'Safari':
  case 'Opera':
    alert( 'Okay we support these browsers too' );
    break;

  default:
    alert( 'We hope that this page looks ok!' );
}
```

3. Swtich Cases

Software Park Thailand
</Code Camp>

3.3. แบบฝึดหัด

2. แปลง Code ดังกล่าวเป็น Switch cases

```
let a = +prompt('a?', '');

if (a == 0) {
  alert( 0 );
}
if (a == 1) {
  alert( 1 );
}

if (a == 2 || a == 3) {
  alert( '2,3' );
}
```

หัวข้อ

- ตัวดำเนินการแบบตรรกะ
- วงวน for และ while
- Switch Cases
- ฟังก์ชัน
- Function Expression
- Arrow Function

Software Park Thailand
</Code Camp>

ฟังก์ชัน

4. ฟังก์ชัน

4.1. ฟังก์ชัน

- เป็นการเขียน Code ที่สามารถนำโค๊ดนั้น ๆ มาใช้ซ้ำได้
- ไว้สำหรับเขียน Code ที่เราต้องใช้บ่อย ๆ
- มี build-in function (ฟังก์ชันที่เค้าเขียนมาให้) เช่น alert, prompt

```
function ชื่อฟังก์ชัน(parameters) {
  // body
}
```

4. ฟังก์ชัน

4.1. ฟังก์ชัน

- วิธีเรียกใช้ function
- เขียนชื่อและตามด้วยวงเล็บเปิดและปิด ใส่ parameters ด้วย (ถ้ามี)

```
function showMessage() {
  alert( 'Hello everyone!' );
}

showMessage();
showMessage();
```

4. ฟังก์ชัน

4.2. Local variables

- ตัวแปรที่ประกาศในฟังก์ชัน และใช้ได้ในเฉพาะฟังก์ชันนั้น เรียกว่า “Local variables”

```
function showMessage() {
  let message = "Hello, I'm Codecamp!"; // local variable
  alert( message );
}

showMessage(); // "Hello, I'm Codecamp!"
alert( message ); // <-- พัง! ตัวแปร local variable ใช้ได้ในฟังก์ชันเท่านั้น
```

4. ฟังก์ชัน

4.3. Global variables

- ตัวแปรที่ประกาศนอกฟังก์ชันต่าง ๆ และฟังก์ชันสามารถเรียกใช้ตัวแปรนั้นได้ เรียกว่า “Global variables”
- เราควรจะใช้ Global variables ให้น้อยที่สุด

4. ฟังก์ชัน

4.3. Global variables

- ตัวอย่างที่ 1

```
let userName = 'Sonter';

function showMessage() {
  let message = 'Hello, ' + userName;
  alert(message);
}

showMessage(); // "Hello, Sonter"
```

4. ฟังก์ชัน

4.3. Global variables

- ตัวอย่างที่ 2

```
let userName = 'John';
function showMessage() {
  userName = "Bob"; // เปลี่ยนค่า Global variable

  let message = 'Hello, ' + userName;
  alert(message);
}

alert( userName ); // John (ก่อนที่จะเรียกฟังก์ชัน)
showMessage();
alert( userName ); // Bob, (ตัวแปรถูกเปลี่ยนหลังจากเรียกฟังก์ชัน)
```

4. ฟังก์ชัน

4.3. Global variables

- ตัวอย่างที่ 3 - การ shadows จะทำให้ตัวแปร Global ถูกเมิน

```
let userName = 'John';

function showMessage() {
  let userName = "Bob"; // ประกาศตัวแปร local variable

  let message = 'Hello, ' + userName; // Bob
  alert(message);
}
// function จะสร้าง userName ของมันเองขึ้นมา
showMessage();
alert( userName ); // John, ไม่เปลี่ยน, function ไม่ได้ใช้ตัวแปร Global variable
```

4. ฟังก์ชัน

4.4. Parameters

- ฟังก์ชันบางฟังก์ชันต้องรับค่าจากข้างนอกมาเพื่อทำงาน
- ซึ่งจะรับค่าผ่านตัวแปร
- ตัวแปรนั้นเรียกว่า Parameters หรืออีกชื่อคือ function arguments

```
function showMessage(from, text) { // arguments: from, text
  alert(from + ': ' + text);
}

showMessage('Ann', 'Hello!'); // Ann: Hello!
showMessage('Ann', "What's up?"); // Ann: What's up?
```

4. ฟังก์ชัน

4.4. Parameters

- function ทำสำเนาค่าแล้วมาใส่ไว้ใน parameters เพราะฉะนั้นการเปลี่ยนแปลงค่าในฟังก์ชันจะไม่ส่งผลไปถึงตัวแปรที่ใส่มา

```
function showMessage(from, text) {
  from = '*' + from + '*'; // ใส่ดอกจันไปในตัวแปร from
  alert( from + ': ' + text );
}

let name = "Ann";
showMessage(name, "Hello"); // *Ann*: Hello

// ค่าของ from ยังเหมือนเดิม
alert( name ); // Ann
```

4. ฟังก์ชัน

4.5. Default value (ES6)

- function ที่รับ parameters ถ้าไม่ใส่ค่าเข้าให้กับ function นั้น อาจจะทำให้ function นั้นทำงานผิดพลาดได้ ตัวอย่างเช่น `showMessage(from, text)` ที่รับ parameters 2 ตัว สามารถใส่ parameters แค่ตัวเดียวก็ได้

```
showMessage("Ann");
```

- ผลลัพธ์ที่ได้จะเป็น `"Ann: undefined"`

4. ฟังก์ชัน

4.5. Default value (ES6)

- function สามารถกำหนด default value ได้โดยใส่เครื่องหมาย = ไว้หลัง parameters แต่ละตัว

```
function showMessage(from, text = "no text given") {
  alert( from + ": " + text );
}

showMessage("Ann"); // “Ann: no text given”
```

4. ฟังก์ชัน

4.5. Default value (สมัยก่อน ES6)

- function สามารถกำหนด default value ได้โดยใส่เครื่องหมาย = ไว้หลัง parameters แต่ละตัว

```
function showMessage(from, text) {
  if (text === undefined) {
    text = 'no text given';
  }

  alert( from + ": " + text );
}
```

```
function showMessage(from, text) {
  // ถ้า text เป็น falsy ก็จะใช้ค่าข้างหลัง
  text = text || 'no text given';
  ...
}
```

4. ฟังก์ชัน

4.6. การคืนค่าของฟังก์ชัน

- function สามารถคืนค่าได้ออกได้
- เขียนตามหลังคำว่า return

```
function sum(a, b) {
  return a + b;
}

let result = sum(1, 2);
alert( result ); // 3
```

4. ฟังก์ชัน

4.6. การคืนค่าของฟังก์ชัน

- ตัวอย่างที่ 2

```
function checkAge(age) {
  if (age >= 18) {
    return true;
  } else {
    return confirm('Do you have permission from your parents?');
  }
}

let age = prompt('How old are you?', 18);

if ( checkAge(age) ) {
  alert( 'Access granted' );
} else {
  alert( 'Access denied' );
}
```

4. ฟังก์ชัน

4.6. การคืนค่าของฟังก์ชัน

- ตัวอย่างที่ 3

```
function showMovie(age) {
  if ( !checkAge(age) ) {
    return;
  }

  alert( "Showing you the movie" ); // (*)
  // ...
}
```

4. ฟังก์ชัน

4.6. การคืนค่าของฟังก์ชัน

- ถ้าฟังก์ชันนั้นมีการเขียน return; จะออกจากฟังก์ชันและไม่คืนค่าใด ๆ ออกจากฟังก์ชัน

```
function showMovie(age) {
  if ( !checkAge(age) ) {
    return;
  }

  alert( "Showing you the movie" ); // (*)
  // ...
}
```

4. ฟังก์ชัน

Software Park Thailand
</Code Camp>

4.6. การคืนค่าของฟังก์ชัน

- ไม่มีการเขียน return
- หรือมีการเขียน return;
- ฟังก์ชันนั้นจะคืนค่าเป็น undefined

```
function doNothing() { /* empty */ }

alert( doNothing() === undefined ); // true
```

```
function doNothing() {
  return;
}

alert( doNothing() === undefined ); // true
```

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

1. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
**
```

n = 2

```
***
```

n = 3

```
****
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

2. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
**
**
```

n = 2

```
***
***
***
```

n = 3

```
****
****
****
****
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

3. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
12
12
```

n = 2

```
123
123
123
```

n = 3

```
1234
1234
1234
1234
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

4. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
11
22
```

n = 2

```
111
222
333
```

n = 3

```
1111
2222
3333
4444
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

5. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
22
11
```

n = 2

```
333
222
111
```

n = 3

```
4444
3333
2222
1111
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

6. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
12
34
```

n = 2

```
123
456
789
```

n = 3

```
1234
5678
9101112
13141516
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

7. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
43
21
```

n = 2

```
987
654
321
```

n = 3

```
16151413
1211109
8765
4321
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

8. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
0
2
```

n = 2

```
0
2
4
```

n = 3

```
0
2
4
6
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

9. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
2
4
```

n = 2

```
2
4
6
```

n = 3

```
2
4
6
8
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดพิเศษ

10. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
12
24
```

n = 2

```
123
246
369
```

n = 3

```
1234
2468
36912
481216
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

11. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
_*
*_
```

n = 2

```
_**
*_*
**_
```

n = 3

```
_***
*_**
**_*
***_
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

12. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
*_
_*
```

n = 2

```
**_
*_*
_**
```

n = 3

```
***_
**_*
*_**
_***
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

13. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
*-
**
```

n = 2

```
*--
**-
***
```

n = 3

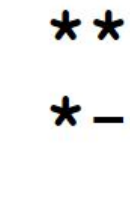
```
*---
**--
***-
****
```

n = 4

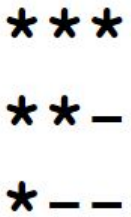
4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

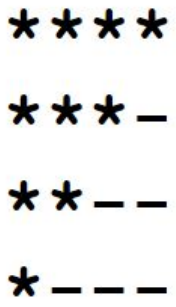
14. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้



n = 2



n = 3



n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

15. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
*-
**
*-
```
n = 2

```
*--
**-
***
**-
*--
```
n = 3

```
*---
**--
***-
****
***-
**--
*---
```
n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

17. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
-*
**
```

n = 2

```
--*
-**
***
```

n = 3

```
---*
--**
-***
****
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

18. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
**
-*
```

n = 2

```
***
-**
--*
```

n = 3

```
****
-***
--**
---*
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

19. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
-*
**
-*
```
n = 2

```
--*
-**
***
-**
--*
```
n = 3

```
---*
--**
-***
****
-***
--**
---*
```
n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

20. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
-1
23
-4
```
n = 2

```
--1
-23
456
-78
--9
```
n = 3

```
---1
--23
-456
78910
-111213
--1415
---16
```
n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

21. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
-*-
***
```

n = 2

```
--*--
-***-
*****
```

n = 3

```
---*---
--***--
-*****-
*******
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

22. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
***
_*_
```

n = 2

```
*****
_***_
__*__
```

n = 3

```
*******
_*****_
__***__
___*___
```

n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

23. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
 -*-
***
 -*-
```
n = 2

```
  -*-
 -***-
*****
 -***-
  -*-
```
n = 3

```
   -*-
  -***-
 -*****-
*******
 -*****-
  -***-
   -*-
```
n = 4

4. ฟังก์ชัน

แบบฝึกหัดขั้นสูง

24. จงเขียน method draw(int n) ให้ print ออกมาในกรณีที่ n มีค่าต่างๆ ได้ผลลัพธ์ดังนี้

```
-1-
234
-5-
```
n = 2

```
--1--
-234-
56789
-101112-
--13--
```
n = 3

```
---1---
--234--
-56789-
10111213141516
-1718192021-
--222324--
---25---
```
n = 4

หัวข้อ

Software Park Thailand
</Code Camp>

- ตัวดำเนินการแบบตรรกะ
- วงวน for และ while
- Switch Cases
- ฟังก์ชัน
- Function Expression
- Arrow Function

Function Expression

5. Function Expression

5.1. Function Expression คืออะไร

- การประกาศฟังก์ชันปกติ (Function declaration)

```
function sayHi() {
  alert( "Hello" );
}
```

- Function สามารถเก็บค่าใส่ตัวแปรได้เช่นเดียวกับ ข้อมูลประเภทอื่น ๆ (number, boolean, etc.) การทำแบบนี้เรียกว่า “Function expression”

```
let sayHi = function() {
  alert( "Hello" );
};
```

5. Function Expression

Software Park Thailand
</Code Camp>

5.2. การเรียกฟังก์ชัน

- การเรียกฟังก์ชันจะต้องมี () ทุกครั้ง
- ถ้าไม่มีการเขียน () หลังชื่อฟังก์ชัน Function ก็จะไม่ถูกรัน

```
let showMessage = function () {
  alert('Hello World!')
}

/* จะไม่มีการเรียก showMessage แต่จะแสดงรายละเอียดของฟังก์ชัน */
alert(showMessage);

/* จะมีการเรียก showMessage() และ showMessage จะส่ง undefined ออกมาให้ alert ไปรันต่อ */
alert(showMessage());
```

5. Function Expression

5.2. การเรียกฟังก์ชัน

- ตัวอย่าง

```
function sayHi() {
  alert("Hello");
}

let func = sayHi;    // copy function ของ sayHi ไปให้ func

func(); // Hello
sayHi(); // Hello
```

5. Function Expression

Software Park Thailand
</Code Camp>

5.2. Callback Function คืออะไร (***สำคัญ***)

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}
function showOk() {
  alert( "You agreed." );
}
function showCancel() {
  alert( "You canceled the execution." );
}
// usage: functions showOk, showCancel are passed as arguments
to ask
ask("Do you agree?", showOk, showCancel);
```

5. Function Expression

5.2. Callback Function คืออะไร

- ทำไมชื่อว่า callback ?
- เพราะว่าฟังก์ชัน yes และ no อาจจะถูกเรียกถ้าจำเป็น (Be called back)

ทั้ง yes และ no เรียกว่า callback function หรือเรียกสั้นว่า callback

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}
```

5. Function Expression

Software Park Thailand
</Code Camp>

5.3. Function declaration VS Function Expression

- Function declaration จะประกาศก่อนหรือหลังเรียกใช้ก็ได้

```
sayHi("John"); // Hello, John

function sayHi(name) {
  alert( `Hello, ${name}` );
}
```

5. Function Expression

Software Park Thailand
</Code Camp>

5.3. Function declaration VS Function Expression

- Function expression จะประกาศก่อน ถึงจะเรียกใช้ก็ได้

```
sayHi("John"); // error!

let sayHi = function(name) {
  alert( `Hello, ${name}` );
};
```

หัวข้อ

- ตัวดำเนินการแบบตรรกะ
- วงวน for และ while
- Switch Cases
- ฟังก์ชัน
- Function Expression
- Arrow Function

Software Park Thailand
</Code Camp>

Arrow Function

6. Arrow Function

Software Park Thailand
</Code Camp>

6.1. Arrow function คืออะไร (ES6)

- เป็นการเขียนฟังก์ชันอีกรูปแบบนึงเช่นเดียวกับ Function expressions
- เพิ่มเข้ามาใน ES6

```
let func = (arg1, arg2, ...argN) => expression
```

6. Arrow Function

Software Park Thailand
</Code Camp>

6.1. Arrow function คืออะไร

- ถ้าเปรียบเทียบกับการประกาศฟังก์ชันปกติ
- Arrow function ก็คือการเขียนฟังก์ชันแบบย่อ

```
let func = (arg1, arg2, ...argN) => expression
```

```
let func = function(arg1, arg2, ...argN) {
  return expression;
};
```

6. Arrow Function

Software Park Thailand
</Code Camp>

6.1. Arrow function คืออะไร - ตัวอย่างที่ 1

```
let sum = (a, b) => a + b;

/* arrow function อันนี้เหมือนกับการเขียนแบบนี้

let sum = function(a, b) {
  return a + b;
};
*/

alert( sum(1, 2) ); // 3
```

6. Arrow Function

Software Park Thailand
</Code Camp>

6.1. Arrow function คืออะไร - ตัวอย่างที่ 2

- จากตัวอย่างที่ 1 มี arguments 2 ตัว คือ a และ b
- ถ้ามี arguments แค่ตัวเดียวเราสามารถละวงเล็บได้อีกด้วย

```
let double = n => n * 2;
/*
arrow function นี้เหมือนกับการประกาศฟังก์ชันด้านล่าง
let double = function (n) {
  return n * 2
}
*/
alert(double(3)); // 6
```

6. Arrow Function

6.1. Arrow function คืออะไร - ตัวอย่างที่ 3

- ถ้าไม่มี arguments เลยให้ใส่วงเล็บว่างเปล่าไว้

```
let sayHi = () => alert("Hello!");

sayHi();
```

6. Arrow Function

Software Park Thailand
</Code Camp>

6.1. Arrow function คืออะไร

- ถ้า body ใน arrow function มีมากกว่า 1 บรรทัด เราต้องใส่ {} ครอบ และต้องมีการใส่ return keyword ไว้

```
let sum = (a, b) => {  // ต้องใช้ปีกกาถ้า code มีหลายบรรทัด
  let result = a + b;
  return result; // ถ้าใส่ปีกกาต้องบอกด้วยว่าจะ return อันไหน
};

alert( sum(1, 2) ); // 3
```

6. Arrow Function

Software Park Thailand
</Code Camp>

6.2. แบบฝึกหัด

- แปลง function ข้างล่างให้อยู่ในรูป arrow function

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}

ask(
  "Do you agree?",
  function() { alert("You agreed."); },
  function() { alert("You canceled the execution."); }
);
```