ข้อ 1)

|  |
| --- |
| onsole.log('1');  console.log('2');  console.log('3');  console.log('4');  เป็น Synchononus Process ที่จำเป็นตองรอให้ทำงานเสร็จก่อนถึงจะไปทา Process ถัดไปได้ |

|  |
| --- |
| console.log('1');  console.log('2');  setTimeout(() => { console.log('3'); }, 2000);  console.log('4');  เป็น Asynchononus Process ที่สามารถข้ามไปทำ Process ถัดไปได้ทันทีโดยไม่ต้องรอ |

--------------------------------------------------------------------------------------------------------------------------------------

ข้อ 2)

|  |
| --- |
| const fn1 = (value) => new Promise((resolve) => {  console.log('fn1', value);  resolve(value + 1);  });  const fn2 = (value) => new Promise((resolve) => {  console.log('fn2', value);  resolve({  a2: value + 1,  b2: value + 2  });  }); |

|  |
| --- |
| const fn3 = (value1, value2) => new Promise((resolve) => {  console.log('fn3', value1, value2);  resolve(value1 + value2);  });  const fn4 = (value) => new Promise((resolve) => {  console.log('fn4', value);  resolve();  }); |

|  |
| --- |
| const main = () => {  const a = 0;  fn1(a)  .then((a1) => fn2(a1))  .then(({ a2, b2 }) => fn3(a2, b2))  .then((a3) => fn4(a3))  .then(() => console.log('End'));  }; |