Javascript for React (operators, Async+ REST API)

Preparation

Membuat 2 files, `index.html` dan `index.js`

index.html

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript For In Loop</h2>
<p>The for in statement loops through the properties of an object:</p>

<p id="demo"></p>

<script src="index.js"></script>

</body>
</html>
```

index.js

```
const demo = document.getElementById("demo");
```

Operators

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x === 5	true
		x === "5"	false
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true

Conditional branching

Statement `if(...)` mengevaluasi kondisi pada parentheses dan, jika hasilnya `true`, mengeksekusi kode blok

If-Else

Syntax

```
if (condition) {  //  block of code to be executed if the condition is true}
```

Contoh:

```
const getDay = () => {
  if (new Date().getDay() === 0) {
    return "Sunday";
  }
  if (newDate().getDay() === 1) {
    return "Monday";
  }
  if (newDate().getDay() === 2) {
    return "Tuesday";
  }
  if (newDate().getDay() === 3) {
    return "Wednesday";
  }
  if (newDate().getDay() === 4) {
    return "Thursday";
  }
  if (newDate().getDay() === 5) {
    return "Friday";
  }
  if (newDate().getDay() === 6) {
    return "Saturday";
  }
};
const today = getDay();
demo.innerHTML = `Today is ${today}`;
```

Ternary operator

Operator yang direpresentasikan oleh tanda tanya ?. Disebut ternary karena operator memiliki 3 operands.

```
const today = {
  day: 1,
  name: "Monday",
};

// Single Condition
new Date().getDay() === today.day
  ? (demo.innerHTML = `Today is ${today.name}`)
  : (demo.innerHTML = `Today is not ${today.name}`);

const anotherDay = {
  day: 5,
  name: "Friday",
};

// Multiple Condition
new Date().getDay() === today.day
  ? (demo.innerHTML = `Today is ${today.name}`)
  : new Date().getDay() === anotherDay.day
  ? (demo.innerHTML = `Today is ${today.anotherDay.day}`)
  : (demo.innerHTML = `Today is neither ${today.name} nor ${anotherDay.name}`);
```

Switch Case

```
const getDay = () => {
  switch (new Date().getDay()) {
    case 0:
      return "Sunday";
      break;
    case 1:
      return "Monday";
      break;
    case 2:
      return "Tuesday";
      break;
    case 3:
      return "Wednesday";
      break;
    case 4:
      return "Thursday";
      break;
    case 5:
      return "Friday";
      break;
    case 6:
      return "Saturday";
  }
};

const today = getDay();
demo.innerHTML = `Today is ${today}`;
```

Callbacks

> ### "I will call back later!"

Callback merupakan fungsi yang dilempar sebagai argumen ke fungsi yang lain. Fungsi Callback dapat berjalan setelah fungsi lain selesai.

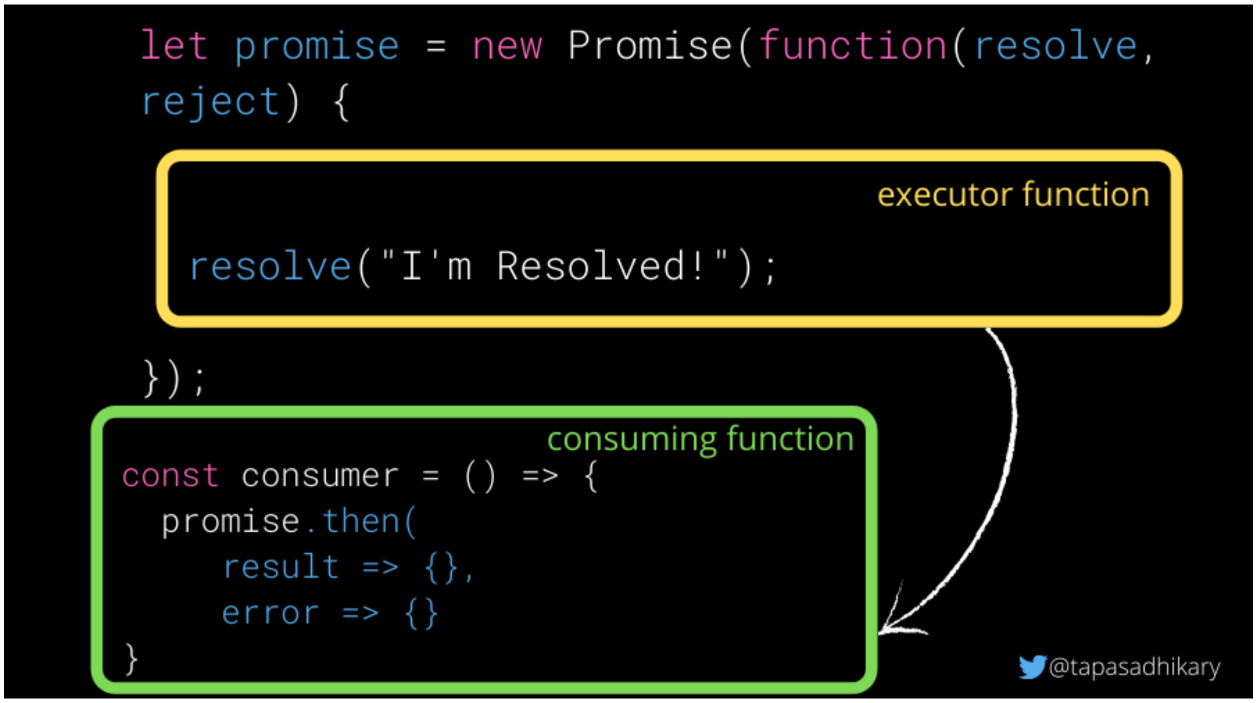
```
const myNumbers = [4, 1, -20, -7, 5, 9, -6];

// Call removeNeg with a Callback
const posNumbers = removeNeg(myNumbers, (x) => x >= 0);

// Display Result
document.getElementById("demo").innerHTML = posNumbers;

// Remove negative numbers
function removeNeg(numbers, callback) {
  const myArray = [];
  for (const x of numbers) {
    if (callback(x)) {
      myArray.push(x);
    }
  }
  return myArray;
}
```

Promises

> ### "I Promise a Result!"

Fungsi yang berjalan secara pararel dengan fungsi yang lain disebut dengan **asynchronous**. Dengan asynchronous programming, program JavaScript dapat memulai long-running task dan melanjutkan task yang lain secara pararel. Karena hal tersebut, metode JS tidak lagi menggunakan callback tetapi menggunakan Promises.

Dalam promises, ada 2 istilah:



```
let promise = new Promise(function(resolve,
reject) {

  resolve("I'm Resolved!");

});
const consumer = () => {
  promise.then(
     result => {},
     error => {}
}
```