Nama : Yanuar Nurul Hilal
NIM : 222112418
Kelas : 2KS4

Responsi Pemrograman Berbasis Web Pertemuan 6

1.Apakah yang dimaksud dengan hoisting? Jelaskan dengan contoh.
Jawab :

Hoisting merupakan keadaan di mana deklarasi variabel (dan fungsi) dianggap telah dipindahkan ke atas dari lokasi aslinya di dalam lingkup fungsi atau global scope, dan deklarasi ditempatkan di awal kode sebelum eksekusi sebenarnya. Hal ini berarti bahwa jika variabel atau fungsi dideklarasikan setelah digunakan, Javascript akan menganggapnya telah dideklarasikan di awal dan akan membuat variabel atau fungsi itu tersedia untuk bisa digunakan di seluruh scope.
Contoh 1 :

```
console.log(a);
var a = 7;
// output : undefined
```

Pada contoh tersebut meski variabel a dideklarasikan setelah console.log, variabel a akan tetap terbaca, karena hoisting, hoisting akan “memindahkan” deklarasi variabel a ke bagian atas. Outputnya akan undefined karena nilai a belum ditentukan pada saat sebelum baris kode console.log
Contoh 2 :

```
console.log(a);
// output : ReferenceError : y is not defined
```

Pada contoh tersebut output akan eror sebab variabel a belum dideklarasikan.
Kesimpulan :
Hoisting adalah perilaku di mana deklarasi variabel (dan fungsi) dianggap telah dipindahkan ke atas dari lokasi aslinya dan ditempatkan di awal kode sebelum eksekusi sebenarnya. Hal ini memungkinkan variabel atau fungsi untuk diakses bahkan jika dideklarasikan setelah digunakan, tetapi hanya deklarasi yang ditingkatkan, bukan nilai.

2.Mengapa disarankan menggunakan const pada saat mendeklarasikan objek dan array. Jelaskan dengan contoh.

Disarankan untuk menggunakan ‘const’ saat mendeklarasikan objek dan array karena memastikan bahwa nilai yang ditetapkan untuk variabel tersebut tidak dapat diubah setelah diberikan nilai awal. Hal ini mencegah perubahan tak disengaja atau tidak diinginkan pada nilai tersebut, yang dapat menyebabkan bug atau kesalahan dalam program.

Contoh 1 :

```
const arr =[1,2,3];
arr.push(5);
console.log(arr)
// output : 1,2,3,5
```

Pada contoh tersebut, walaupun ‘arr’ dideklarasikan menggunakan ‘const’, kita masih bisa menambahkan elemen baru ke dalam array tersebut menggunakan method ‘push’. Hal ini karena nilai referensi dari ‘arr’ tetap sama setelah method ‘push’ dijalankan, yang berarti nilai ‘arr’ sendiri tetap tidak berubah.

Contoh 2 :

```
const arr =[1,2,3];
arr =[3,2,1];
console.log(arr)
// output : Error: Assignment to constant variable
```

Pada contoh tersebut ketika dicoba untuk menetapkan nilai baru untuk variabel ‘arr’. Maka akan menghasilkan error karena nilai const tidak dapat diubah

Contoh 3 :

```
const person = {
 name : "Andi",
 Age : 20
};

person.age = 31;
console.log(person);
// output : {name : "Andi", age : 20}


person = {name: "Budi",age : 21 }
console.log(arr)
```

```
// output : Error: Assignment to constant variable
```

Pada contoh tersebut, ketika objek dideklarasikan menggunakan “const’. Properti pada objek tersebut masih bisa memodifikasi. Namun, jika dicoba untuk menetapkan objek baru ke variabel ‘const’ yang sama, maka akan menghasilkan ‘Error’.

3. Jelaskan implementasi array dan objek yang bisa diterapkan pada aplikasi sederhana yang tim Anda buat pada pertemuan sebelumnya!

Untuk implementasi array dan objek pada pembuatan web sederhana ticketing adalah membuat objek pemesan (orang yang memesan tiket) dimana objek tersebut memiliki atribut noId, nama, noTelepon, kuantitas, kategori, dan pembayaran. Lalu dari objek tersebut dibuat array dengan nama pesanan. Pada web sederhana kelompok kami juga bisa membuat objek invoice dimana atribut dari invoice adalah noId, kategori, dan harga
Implementasi codenya pada halaman 1 sebagai berikut:

```
const form = document.getElementById('form');
const noId = document.getElementById('noId');
const name = document.getElementById('name');
const noTelepon = document.getElementById('noTelepon');
const kuantitas = document.getElementById('kuantitas');
const kategori = document.getElementById('kategori');
const pembayaran = document.getElementById('pembayaran');

form.addEventListener('submit', function(e){e.preventDefault();

 const noIdValue = noId.value;
 const nameValue = name.value;
 const noTeleponValue = noTelepon.value;
 const kuantitasValue = kuantitas.value;
 const kategoriValue = kategori.value;
 const pembayaranValue = pembayaran.value;


// Membuat objek data pemesan

Const pemesan = {
noId: noIdValue, name: nameValue,
noTelepon: noTeleponValue, kuantitas: kuantitasValue, kategori:
kategoriValue, pembayaran: pembayaranValue
};
```

```
// Simpan objek data pemesan dalam array pesanan
Let pesanan =[]
pesanan.push(pemesan);

// Simpan array pesanan dalam Local Storage
localStorage.setItem('pesanan', JSON.stringify(pesanan));

// Ambil nilai harga yang dipilih
let hargaValue = ''; if(kategoriValue === 'VIP'){
hargaValue = '500000';
} else if(kategoriValue === 'Silver'){
hargaValue = '250000';
} else if(kategoriValue === 'Bronze'){
hargaValue = '100000';
}

// Membuat objek data invoice
Const invoice ={
noId: noIdValue,
kategori: kategoriValue, harga: hargaValue
};

// Simpan objek data invoice dalam Local Storage
localStorage.setItem('invoice', JSON.stringify(invoice));

// Redirect ke halaman invoice
window.location.href = "invoice.html";
});
```

Pada code diatas, kita membuat objek pemesan yang berisi data pemesan yang diambil dari nilai input pada form. Selanjutnya, objek pemesan tersebut disimpan dalam array pesanan yang kemudian disimpan dalam Local Storage menggunakan method JSON.stringify().

Selain itu, kita juga membuat objek invoice yang berisi data untuk invoice, yaitu noId, kategori, dan harga. Objek invoice tersebut juga disimpan dalam Local Storage menggunakan method JSON.stringify(). Lalu pada halaman kedua perlu mengambil data dari Local Storage yang telah disimpan pada page sebelumnya dan menampilkannya pada halaman invoice. Implementasi code pada halaman kedua sebagai berikut.

```
const invoiceData = localStorage.getItem('invoice');
const pesananData = localStorage.getItem('pesanan');

// Mengubah string kembali menjadi objek

const invoice = JSON.parse(invoiceData); const pesanan =
JSON.parse(pesananData);

// Menampilkan data pada invoice
document.getElementById('no-Id').textContent = invoice.noId;
document.getElementById('kategori').textContent = invoice.kategori;
document.getElementById('harga').textContent = invoice.harga;
document.getElementById('kuantitas').textContent =
pesanan[0].kuantitas; document.getElementById('name').textContent =
pesanan[0].name; document.getElementById('no-Telepon').textContent =
pesanan[0].noTelepon; document.getElementById('pembayaran').textContent =
pesanan[0].pembayaran;

// Menghitung total harga dan pajak
const totalHarga = invoice.harga * pesanan[0].kuantitas;
const tax = 0.1 * totalHarga;

// Menampilkan total harga dan pajak

document.getElementById('totalHarga').textContent = totalHarga;
document.getElementById('tax').textContent = tax;

// Menghitung jumlah pembayaran
const jumlahPembayaran = totalHarga + tax;

// Menampilkan jumlah pembayaran

document.getElementById('jumlahPembayaran').textContent = jumlahPembayaran;
```

Pada code diatas, kita mengambil data invoice dan pesanan dari Local Storage menggunakan localStorage.getItem() dan kemudian mengubahnya kembali menjadi objek menggunakan JSON.parse().

Selanjutnya, kita menampilkan data dari invoice dan pesanan pada invoice dengan menetapkan nilai teks pada elemen HTML yang sesuai menggunakan `document.getElementById().textContent .` Terakhir, kita menghitung total, harga, pajak, dan jumlah pembayaran dan menampilkannya pada invoice.