DASAR-DASAR KODING
MODUL 1

Pendahuluan

Bismillah
BASIK adalah sebuah template JavaScript yang dirancang khusus untuk memudahkan pembuatan aplikasi berbasis kanvas. Template ini ditujukan sebagai pengantar dalam pembelajaran bahasa pemrograman JavaScript, terutama bagi pemula.
Desain BASIK disusun dengan prinsip-prinsip berikut:

✅ **Perintah yang Disederhanakan:** Semua perintah dalam BASIK memiliki bentuk yang sederhana dan struktur prosedural. Pendekatan ini menghindari konsep-konsep kompleks seperti dot-notation, class, closure, dan lainnya, agar lebih mudah dipelajari dan digunakan sebagai pengenalan terhadap pemrograman.

⚡ **Umpan Balik Langsung:** Setiap perintah dapat langsung menampilkan hasilnya. Hal ini memungkinkan siswa untuk fokus pada penyusunan logika dan pemecahan masalah tanpa hambatan teknis.

🧰 **Fungsi Bawaan Siap Pakai:** BASIK menyediakan berbagai perintah bawaan yang dapat langsung digunakan. Dengan ini, siswa dapat berkreasi dan mengembangkan kreativitas, imajinasi sambil belajar.

📂 **Struktur Proyek yang Siap Digunakan:** BASIK memiliki struktur proyek yang siap pakai, sehingga memudahkan siswa untuk memulai proyek, mengumpulkan tugas, atau membagikan hasil karya mereka kepada teman atau kepada yang lain.

🎨 **Contoh yang Variatif:** Disediakan berbagai contoh yang dapat dijadikan bahan pembelajaran dan inspirasi untuk mendorong kreativitas siswa.

🌐 **Dapat Digunakan Secara Luring:** BASIK dapat dijalankan tanpa koneksi internet. Hasilnya langsung dapat dilihat melalui browser tanpa perlu menginstal server atau infrastruktur tambahan.

📱 **Adaptif untuk Berbagai Perangkat:** Aplikasi yang dibuat dengan BASIK dapat dijalankan baik di perangkat desktop maupun di ponsel secara langsung.

Semoga buku ini dapat menjadi pengantar yang bermanfaat bagi siswa dalam mempelajari pemrograman dengan cara yang menyenangkan dan mudah diakses.

💻 Apa itu Pemrograman?

Pemrograman itu seperti memberi tahu komputer apa yang harus dilakukan. Kita menulis “perintah” dalam bentuk kode, dan komputer akan mengikuti perintah itu. Kodenya ditulis dengan cara khusus agar komputer bisa mengerti.

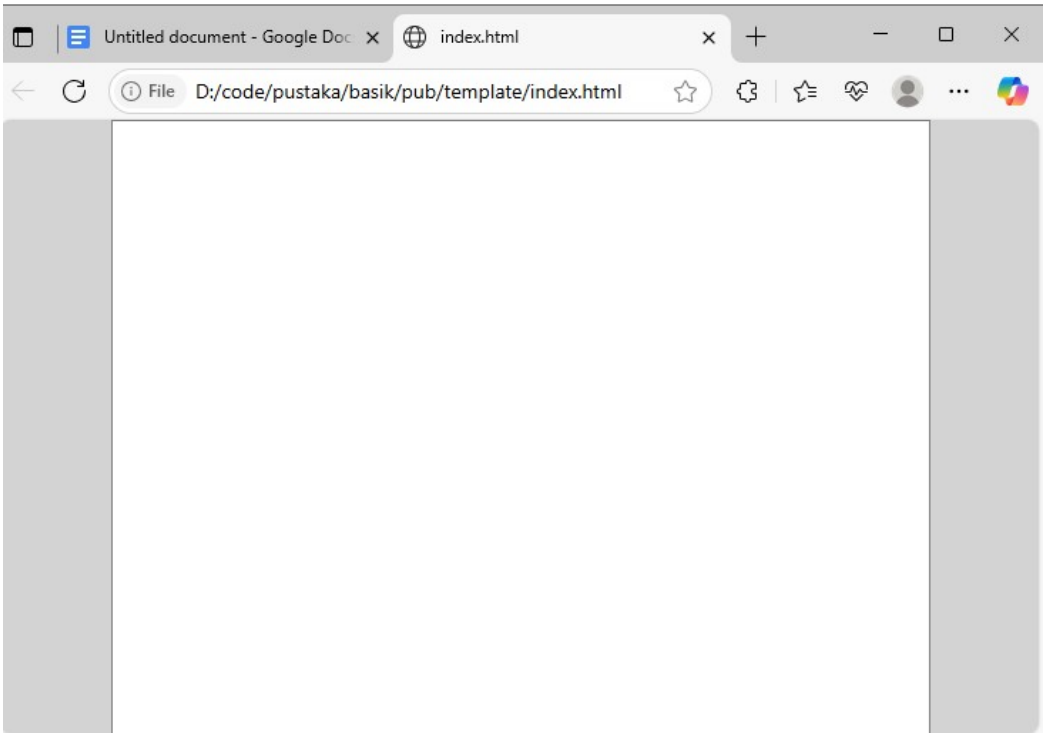
Di pelajaran ini, kita akan belajar bahasa pemrograman yang bernama JavaScript menggunakan BASIK. Dengan JavaScript, kita bisa membuat berbagai macam aplikasi—mulai dari yang sederhana hingga yang rumit!

Mari kita mulai latihan pertama!

Cari folder yang namanya “template”, lalu salin (copy) folder itu. Setelah disalin, beri nama folder barunya: "latihan".Kita akan menggunakan folder ini sebagai tempat kita berlatih.

Di dalam folder “latihan”, ada file yang namanya index.html. Klik dua kali file itu. Komputer akan otomatis membuka file-nya di browser (misalnya Chrome, Edge, atau Firefox), tergantung pengaturan di komputer kamu.

Kalau sudah terbuka, maka akan melihat tampilan awal dari halaman web yang akan kita ubah dan beri perintah.



Sekarang kamu akan melihat layar kosong dengan sebuah kotak di tengah. Kotak ini adalah tempat kita bisa menulis, menggambar, dan bermain dengan kode. Karena belum ada perintah, kotaknya masih kosong.

Langkah selanjutnya:
buka file bernama `index.js`

Disarankan untuk membuka file ini dengan aplikasi khusus untuk menulis kode, seperti **Notepad++**, **Visual Studio Code (VSCode)**, atau aplikasi sejenis. Bila tidak ada applikasi tersebut, maka sebaiknya di Install dulu.

Bila terpaksa, bisa juga gunakan notepad bawaan dari Windows, tampilannya mungkin kurang nyamn, tapi jangan membuka menggunakan microsoft office, atau adobe photoshop, karena nanti filenya tidak akan bisa dijalankan.

Di sini, saya akan menggunakan Notepad++.

Saat ini, file index.js masih kosong.
Kita akan menulis perintah pertama kita di sana.

Tulislah perintah seperti berikut:

```
tulis("Bismillah, Saya sedang belajar pemrograman");
```

Perhatikan baik-baik:
Saat menulis kode, kamu harus teliti dengan tanda-tanda seperti:

- Tanda kutip dua "
- Tanda kurung ( )
- Titik koma ; di akhir baris

Semua itu penting supaya kode bisa berjalan dengan benar.
Setelah selesai menulis, simpan file index.js.
Kembali lagi ke browser yang tadi sudah dibuka dan tekan tombol F5 atau klik tombol refresh.
Tampilan di layar akan berubah!
Kira-kira tampilannya akan seperti ini:

Kalau kamu belum melihat gambar seperti di atas...
Coba cek lagi cara kamu menulis perintahnya. Mungkin ada yang terlupa, seperti:

- Salah ketik huruf atau tanda
- Lupa menutup kurung atau tanda kutip
- Tidak pakai titik koma; di akhir

Ingat, komputer sangat teliti! Sedikit saja salah, hasilnya bisa berbeda atau tidak muncul sama sekali.

Perintah pertama yang kita tulis tadi adalah untuk “menulis” sesuatu ke layar. Sesuai dengan namanya: perintah untuk menulis.

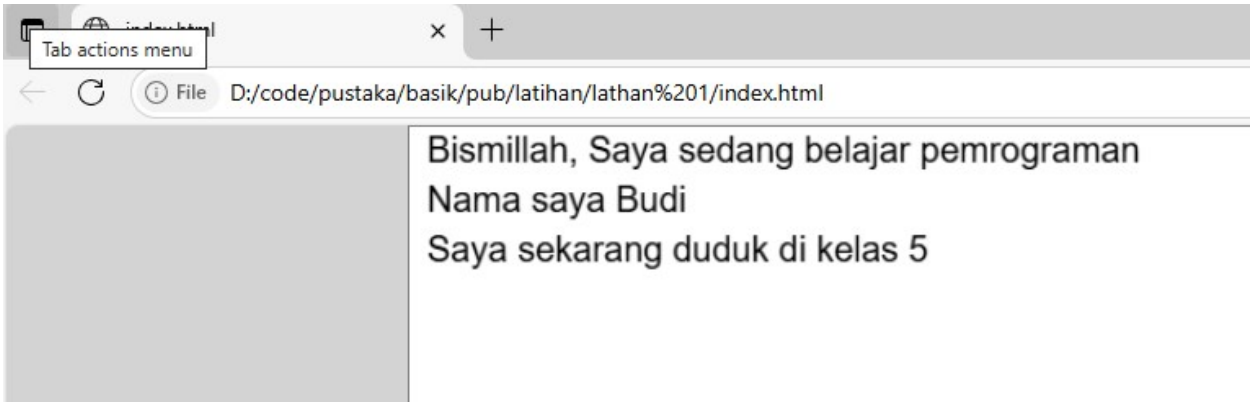
Sekarang, mari kita lanjut ke perintah berikutnya. Tulis di bawah perintah pertama, seperti ini:

```
tulis("Bismillah, Saya sedang belajar pemrograman");
tulis("Nama saya Umar");
tulis("Saya sekarang duduk di kelas 5");
```

Jangan lupa!
Setelah menulis perintah kedua:

- Simpan file index.js
- Refresh browser dengan menekan tombol F5 atau klik tombol refresh.

Kalau semua perintah ditulis dengan benar, maka tampilannya akan seperti ini:



Sampai di sini, kalau tampilan yang kamu lihat mirip seperti contoh di atas...

🏆 Selamat! Artinya kamu sudah berhasil 🏆:
- Memberikan perintah kepada komputer
- Komputer mengerti perintahmu
- Dan komputer berhasil menjalankannya!

Kamu baru saja “berbicara” dengan komputer lewat kode.

Kalau belum muncul seperti yang diharapkan...
- Jangan khawatir! Coba ulangi langkah-langkahnya:
- Periksa apakah ada huruf yang salah atau kurang
- Pastikan semua tanda seperti kurung dan kutip sudah benar
- Jangan lupa simpan file index.js sebelum refresh

Dan yang penting: jangan copy-paste! Menulis sendiri akan membuat kamu lebih paham dan terbiasa.

Berikutnya, mari kita kenali bentuk perintah dalam JavaScript!
Secara umum, perintah di JavaScript punya bentuk seperti ini:

```
perintah("parameter1", parameter2);
```

Penjelasan bentuk perintah:

Nama perintah biasanya satu atau dua kata, contohnya:
- `tulis()`
- `update()`
- `prompt()`

Kalau dua kata, penulisannya disambung dan huruf pertama kata kedua pakai huruf kapital, contohnya:

- `muatImage()`
- `buatKanvas()`

Tidak boleh ada spasi di nama perintah! Contoh yang salah: `buat kanvas()` ✕
Kurung buka dan tutup ( ) adalah tempat untuk menaruh parameter. Parameter itu seperti “info tambahan” yang kita berikan ke perintah.

Contoh:

```
tulis("Bismillah, Saya sedang belajar");
```

Di sini, `"Bismillah, Saya sedang belajar"` adalah parameter yang memberi tahu apa yang harus ditulis.

Perintah bisa punya 0, 1, atau lebih banyak parameter. Kalau lebih dari satu, pisahkan dengan tanda koma ,

Contoh:

- `buatKanvas(300, 200);`
- `tulisXY(100, 100, "Bismillah, saya sedang belajar coding");`
- `muatAnimasi("gambar.png", 32, 32);`

Di akhir perintah biasanya ada titik koma; Ini seperti tanda “selesai”. Tapi tidak apa-apa, kalau kamu lupa, JavaScript kadang masih bisa jalan. Tetap disarankan untuk dibiasakan ya!

Komentar

Komentar itu seperti catatan kecil yang kita tulis di dalam kode. Komentar tidak akan dijalankan oleh komputer, jadi fungsinya hanya untuk memberi penjelasan atau pengingat.

Cara menulis komentar:
Gunakan tanda // di awal baris, lalu tulis komentarmu.

Contoh:
Tambahkan komentar seperti contoh di atas ke file index.js

```
//ini adalah komenter
//komputer tidak akan menjalankannya
//ini hanya untuk catatan saja
tulis("Bismillah, Saya sedang belajar pemrograman");
tulis("Nama saya Budi");
tulis("Saya sekarang duduk di kelas 5");
```

Simpan file dan jangan lupa Refresh browser-nya

📌Hasilnya? Tidak ada perubahan tampilan! Karena komentar memang tidak memengaruhi hasil program.

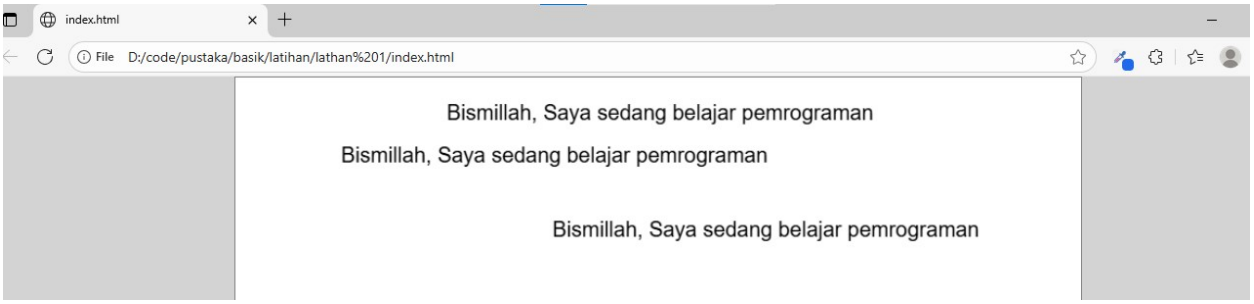
Posisi Teks

Kita bisa menentukan posisi teks dengan perintah posisiTeks().

Contoh:

```
posisiTeks(200, 40);
tulis("Bismillah, Saya sedang belajar pemrograman");
posisiTeks(100, 80);
tulis("Bismillah, Saya sedang belajar pemrograman");
posisiTeks(300, 150);
tulis("Bismillah, Saya sedang belajar pemrograman");
```

📌 Hasilnya: teks akan terlihat pada posisi yang berbeda-beda.



Perintah `posisiTeks()` memiliki dua parameter bertipe **number/angka**. Penulisan parameter bertipe number/angka tidak menggunakan tanda kutip.

Koordinat layar di kanvas akan di mulai dari pojok kiri atas dan bergerak ke kanan dan ke bawah.

`posisiTeks(200, 40)` artinya posisinya bergerak horizontal ke kanan 200 pixel, dan 40 pixel vetikal ke arah bawah.

Perintah-perintah dalam pemrograman akan dijalankan dari atas ke bawah secara berurutan. Perintah posisiTeks() akan mempengaruhi perintah setelahnya, dan tidak berpengaruh terhadap perintah sebelumnya.

🏆 Kesimpulan Modul Ini, kita sudah belajar:

- Cara menulis perintah di JavaScript
- Cara memberi parameter ke perintah
- Cara membuat komentar sebagai catatan di dalam kode
- Dan bagaimana komputer menjalankan perintah yang kita tulis

📄 Variable - Si Kotak Ajaib Penyimpan Data

☕ Apa itu Variable?

Bayangkan kamu punya banyak kotak kecil.
Setiap kotak punya label nama, dan kamu bisa memasukkan benda ke dalamnya.
Nah, dalam dunia pemrograman, variable adalah kotak itu.
Kita bisa menyimpan data di dalamnya, lalu memanggilnya kapan saja.

Contoh:

```
let namaSiswa = "Budi";
tulis(namaSiswa);
```

📌hasilnya adalah tulisan "Budi" di layar

Penjelasan:

- `let` adalah perintah untuk membuat kotak (variable).
- `namaSiswa` adalah nama kotaknya.
- `"Budi"` adalah isi kotaknya.
- `tulis(namaSiswa)` akan menampilkan isi kotak, yaitu `"Budi"`.

Perhatikan disini kita tidak menulis `namaSiswa tanpa tanda` kutip karena `namaSiswa` adalah nama variable, jika kita menulis dengan tanda kutip maka hasilnya nanti jadi `"namaSiswa"` dan bukan `"Budi"`.

Aturan Penamaan Variable

- Tidak boleh pakai spasi.
- Gunakan huruf kecil di awal, dan huruf besar untuk kata kedua (contoh: namaSiswa).
- Nama variable harus jelas dan mudah dimengerti.

Operator "=" dan "+"

Javascript memiliki beberapa operator antara lain = dan +:

- Tanda = digunakan untuk mengisi kotak.
- Tanda + digunakan untuk menggabungkan teks atau menjumlahkan angka.
  Bila digunakan dengan teks dan angka, maka hasilnya akan digabungkan. Namun bila digunakan dengan angka dan angka maka hasilnya akan dijumlahkan.

Contoh:

```
let namaSiswa = "Budi";
tulis("Nama saya: " + namaSiswa);
```

📌Hasilnya adalah tulisan Nama saya: Budi

Variable Bisa Diisi Belakangan

```
let namaSiswa;
namaSiswa = "Budi";
tulis("Nama saya: " + namaSiswa);
```

📌Hasilnya adalah tulisan Nama saya: Budi. Sama dengan di atas karena kita hanya memindahkan perintah untuk mengisi variable namaSiswa ke bawah, dan isinya tetap sama

Variable juga bisa diisi dengan Angka

```
let roti = 9;
let coklat = 5;
let total =  roti + coklat
tulis("total bekal yang dibawa: " + total)
```

Angka ditulis tanpa tanda kutip.

📌Hasilnya adalah tulisan angka 14. Hasil dari roti + coklat. 9 + 5 = 14.

🗃 Tipe Data di JavaScript

- **String**: Teks, seperti "Budi"
- **Number**: Angka, seperti 90
- **Object**: Kita akan bahas nanti

🎩 Tabel Contoh Penggunaan Variable

Perintah	Penjelasan
`let nama;`	Membuat kotak (variable) bernama nama, belum diisi
`let nama;` `nama = "Umar";`	Membuat kotak (variable) nama, lalu diisi dengan teks "Umar"
`let nama = "Umar";`	Membuat dan mengisi variable nama sekaligus

`let roti = 9;`
`let coklat = 5;`
`let total =  roti + coklat`
	Menghitung total makanan dan menyimpannya di kotak total

`let donat = 10;` `donat = donat + 10;`	Membuat variable donat dan mengisinya dengan angka 10 **Menambahkan** isi dari kotak donat dengan 10 dan **memasukkan** ke dalam kotak donat. Sekarang isinya jadi 20. 📌 *Ini adalah salah satu pola yang sering digunakan dalam pemrograman*

✂️ Operator Singkatan

Kadang kita bisa menulis perhitungan dengan cara yang lebih singkat.

Contoh:

```
let angka = 5;
angka = angka + 10;
```

Bisa disingkat menjadi:

```
let angka = 5;
angka += 10;
```

Tanda += artinya: tambahkan angka 10 ke variabel `angka`.

Tambah 1 dengan Cara Singkat

```
let angka = 5;
angka = angka + 1;
```

Bisa disingkat menjadi:

```
let angka = 5;
angka++;
```

Tanda ++ artinya: tambahkan 1 ke variabel `angka`.

🎯 Kesimpulan

- Gunakan `+`, `-`, `*`, dan `/` untuk menghitung angka.
- Gunakan + juga untuk menggabungkan teks.
- Gunakan += atau ++ untuk menulis lebih singkat.

🌀 Belajar Perulangan dengan Perintah `for`

Kadang-kadang, kita ingin melakukan sesuatu berulang kali. Misalnya, menulis kalimat sebanyak 10 kali atau menghitung dari 1 sampai 100. Kalau kita menulis satu per satu, pasti capek dan lama, kan?

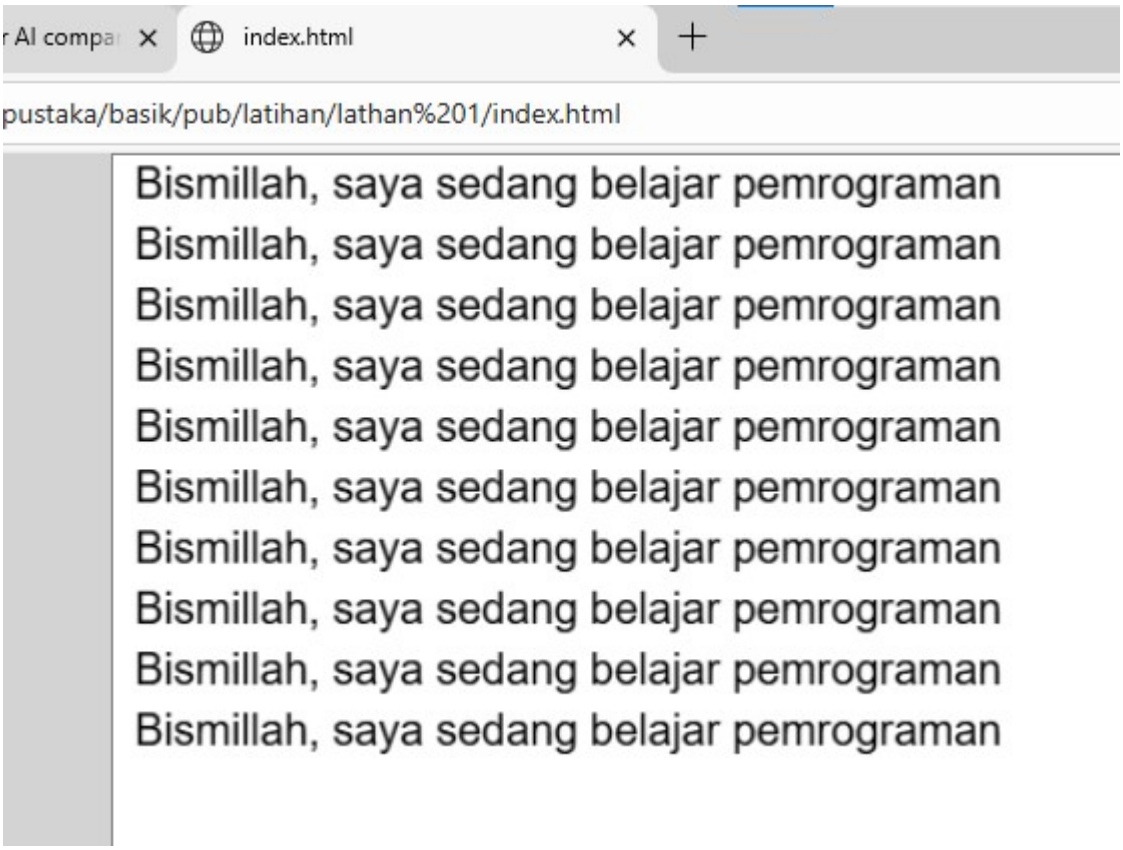
Nah, di JavaScript, ada cara pintar untuk mengulang perintah secara otomatis. Namanya **perulangan** atau **looping**. Salah satu jenis perulangan adalah `for`.

Dengan `for`, kita bisa membuat kode yang lebih pendek dan rapi!

📓 Contoh Perulangan

```
for (let i = 1; i <= 10; i++) {
  tulis("Bismillah, saya sedang belajar pemrograman");
}
```

📌 Hasilnya, perintah `tulis()` di atas akan ditulis sebanyak 10 kali!



Yuk kita pahami bagian-bagian dari perintah `for` ini:

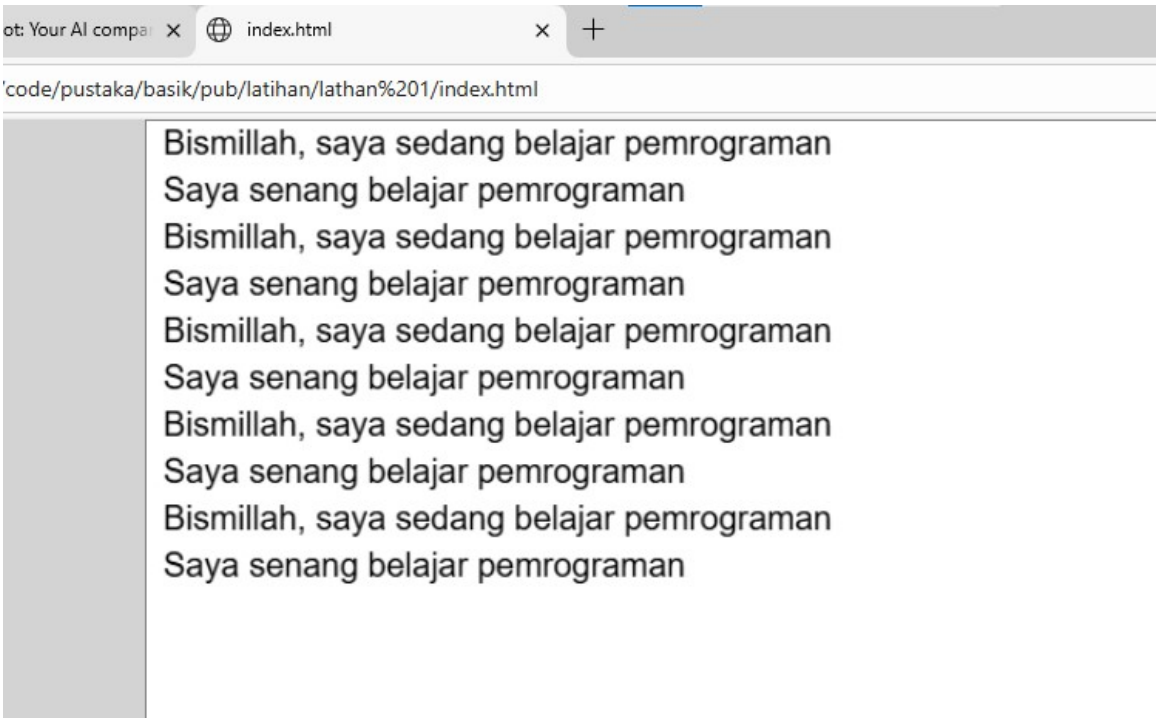
- `let i = 1` → Kita membuat sebuah **variabel** bernama `i`, dan kita isi dengan angka 1.
- `i <= 10` → Selama nilai `i` masih kurang dari atau sama dengan 10, perulangan akan terus berjalan.
- `i++` → Setelah satu kali perulangan selesai, nilai `i` akan bertambah 1. Ini sama seperti `i = i + 1`.
- `tulis(...)` → Ini adalah perintah yang akan dijalankan setiap kali perulangan terjadi.

🧠 Menambahkan Lebih Banyak Perintah

Kita bisa menambahkan lebih dari satu perintah di dalam perulangan. Semua perintah itu harus ditulis di antara tanda `{` dan `}`.

```
for (let i = 1; i <= 5; i++) {
  tulis("Bismillah, saya sedang belajar pemrograman");
  tulis("Saya senang belajar pemrograman");
}
```

📌 Setiap kali perulangan berjalan, kedua perintah `tulis()` akan dijalankan.

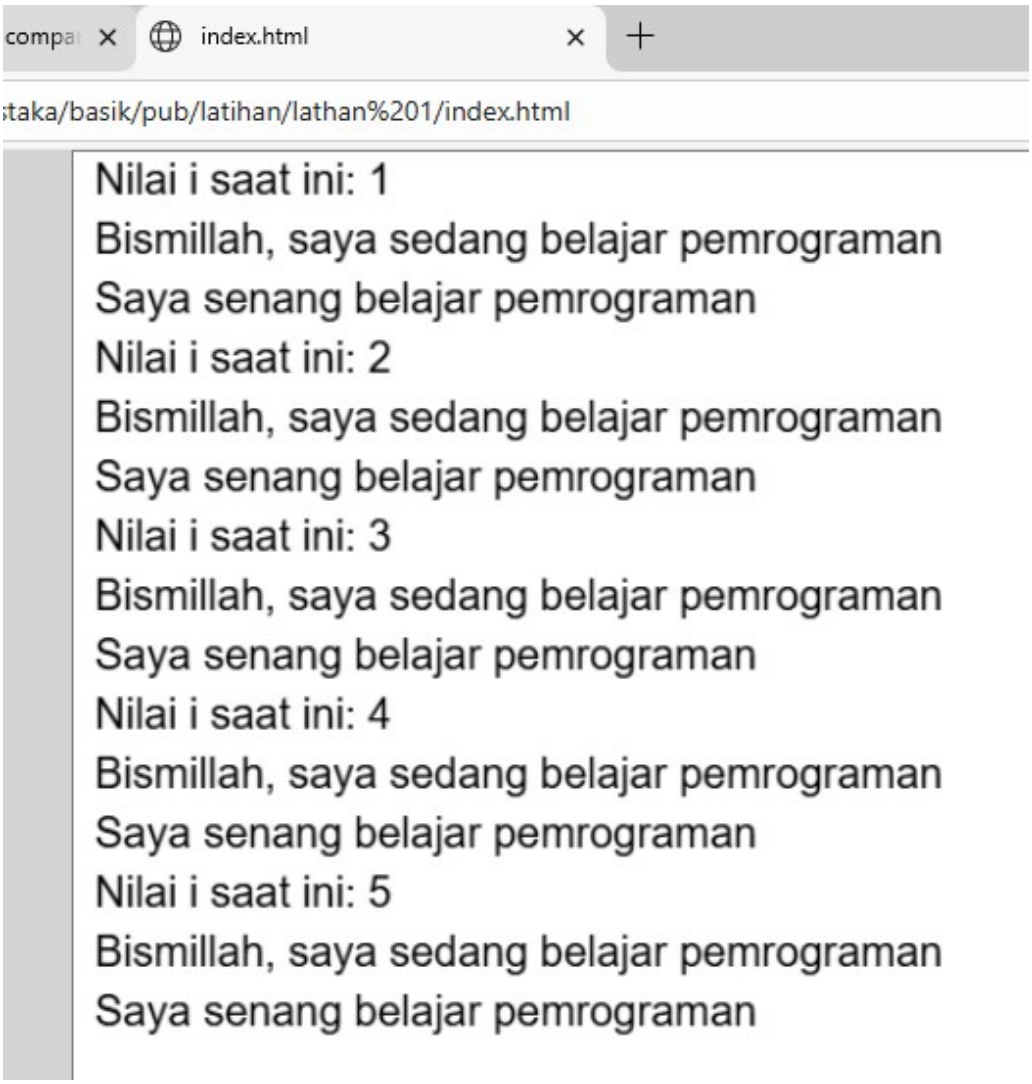


🔢 Menampilkan Nilai `i`

Kita juga bisa menampilkan nilai dari variabel `i` agar tahu sudah berapa kali perulangan berjalan.

```
for (let i = 1; i <= 5; i++) {
  tulis("Nilai i saat ini: " + i);
  tulis("Bismillah, saya sedang belajar pemrograman");
  tulis("Saya senang belajar pemrograman");
}
```

📌 Hasilnya:



✨ **Indentasi (Menjorok ke Dalam**)

Kalau kamu perhatikan, semua perintah di dalam {} ditulis agak menjorok ke kanan. Ini disebut **indentasi**.
Indentasi membuat kode lebih rapi dan mudah dibaca. Walaupun tidak wajib, sebaiknya kamu biasakan menulis seperti ini.
Kalau kamu pakai editor seperti **VS Code** atau **Notepad++**, biasanya indentasi akan dibuat otomatis.

🧠 Percabangan IF

Dalam pemrograman, kita biasanya menulis perintah dari atas ke bawah. Tapi kadang-kadang, kita ingin komputer memilih perintah yang dijalankan berdasarkan kondisi tertentu. Nah, di sinilah kita pakai percabangan IF.

Bayangkan kamu punya robot. Kamu bilang ke robot:

> "**Kalau** jam masih pagi, bilang 'Selamat pagi'. **Kalau tidak**, bilang 'Selamat siang'."

Robot akan mengecek dulu jamnya, lalu memilih perintah yang sesuai.

📜 Bentuk Dasar IF

```
if (kondisi) {
  // perintah yang dijalankan kalau kondisi benar
}
```

Kondisi biasanya berupa perbandingan, seperti:

- Apakah jam kurang dari 12?
- Apakah cuaca sama dengan "hujan"?

🌞 Contoh Sederhana

```
let jam = 8;
if (jam < 12) {
  tulis("Hari masih pagi");
}
```

📌 Karena jam = 8, maka kondisi jam < 12 adalah **benar**, jadi yang dijalankan adalah:
Hari masih pagi

🔀 IF + ELSE

Kalau kondisi tidak benar, kita bisa tambahkan else untuk memberi pilihan lain.

```
let jam = 15;
if (jam < 12) {
  tulis("Hari masih pagi");
} else {
  tulis("Hari sudah sore");
}
```

📌 Karena jam = 15, maka kondisi jam < 12 adalah salah, jadi yang dijalankan adalah:
Hari sudah sore

🌦️ IF + ELSE IF + ELSE

Kalau ada lebih dari dua pilihan, kita bisa pakai else if:

```
let cuaca = "mendung";
if (cuaca == "hujan") {
  tulis("Bawa payung");
} else if (cuaca == "mendung") {
  tulis("Siap-siap payung");
} else {
  tulis("Jalan santai");
}
```

📌 Karena cuaca = "mendung", maka yang dijalankan adalah:
Siap-siap payung

🎮 Kesimpulan

- IF = kalau kondisi benar, jalankan perintah tertentu
- ELSE = kalau kondisi salah, jalankan perintah lain
- ELSE IF = kalau ada banyak kondisi, pilih yang sesuai

📓 Function?

Function itu seperti **resep masakan** dalam dunia pemrograman. Kita bisa menuliskan langkah-langkah sekali, lalu memakainya berkali-kali tanpa harus menulis ulang semuanya. Praktis banget, kan?

✨ Kenapa Function Penting?

- Menghemat waktu
- Mengurangi kesalahan
- Membuat kode lebih rapi dan mudah dibaca

🛠️ Cara Membuat Function

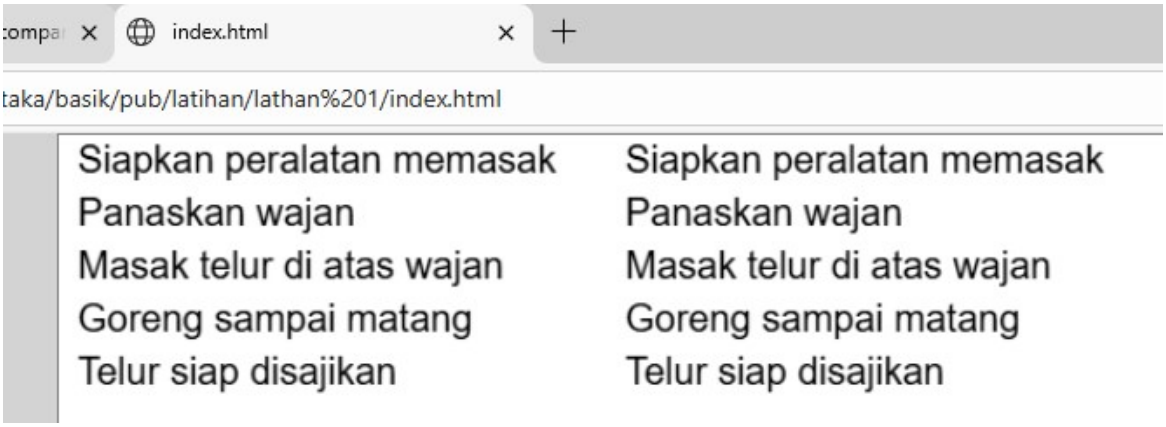
Kita mulai dengan **mendeklarasikan** function, yaitu menuliskan langkah-langkah yang ingin dijalankan.

Contohnya:

```
function resepGorengTelur() {
  tulis("Siapkan peralatan memasak");
  tulis("Panaskan wajan");
  tulis("Masak telur di atas wajan");
  tulis("Goreng sampai matang");
  tulis("Telur siap disajikan");
}

posisiTeks(10, 20)
resepGorengTelur();
posisiTeks(300,20)
resepGorengTelur();
```

📌 Setiap kali kita menulis `resepGorengTelur()`, semua langkah di dalam fungsi `resepGorengTelur` akan dijalankan. Kamu akan melihat dua tampilan yang sama.



Function dengan Parameter

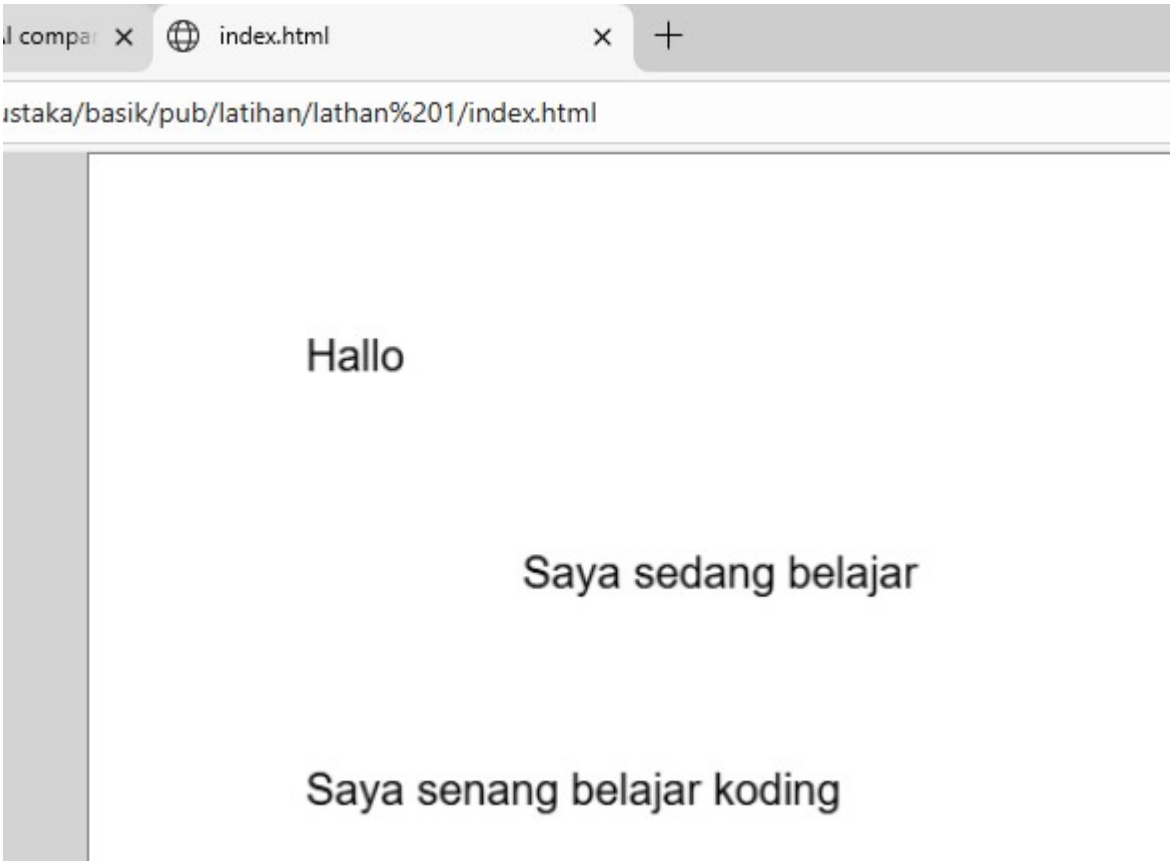
Kadang kita ingin function yang bisa dipakai untuk berbagai hal. Nah, di sinilah **parameter** berguna!

Contoh:

```
function tulisXY(x, y, teks) {
  posisiTeks(x, y);
  tulis(teks);
}

tulisXY(100, 100, "Hallo");
tulisXY(200, 200, "Saya sedang belajar");
tulisXY(100, 300, "Saya senang belajar koding");
```

📌 Jadi, meskipun function-nya sama, hasilnya bisa berbeda tergantung isi parameternya.



🎯 Kesimpulan

Function itu seperti alat bantu dalam coding:

- Bisa dipakai berulang-ulang
- Bisa disesuaikan dengan parameter
- Membuat program lebih efisien dan mudah dipahami

Function akan jadi sahabat terbaikmu dalam dunia koding!

🎮 Event

Event itu seperti **kejadian** yang terjadi saat aplikasi berjalan. Contohnya:

- Kamu gerakkan mouse 🖱️
- Kamu klik layar 💥
- Gambar selesai dimuat 🖼️

Nah, saat kejadian itu terjadi, kita bisa membuat program **merespons secara otomatis**. Seru, kan?

🧠 Cara Menangani Event di BASIC

Di BASIC, kita bisa menangani event dengan cara **membuat function khusus**. Nama function-nya sudah ditentukan, jadi tinggal kita isi dengan perintah yang ingin dijalankan.

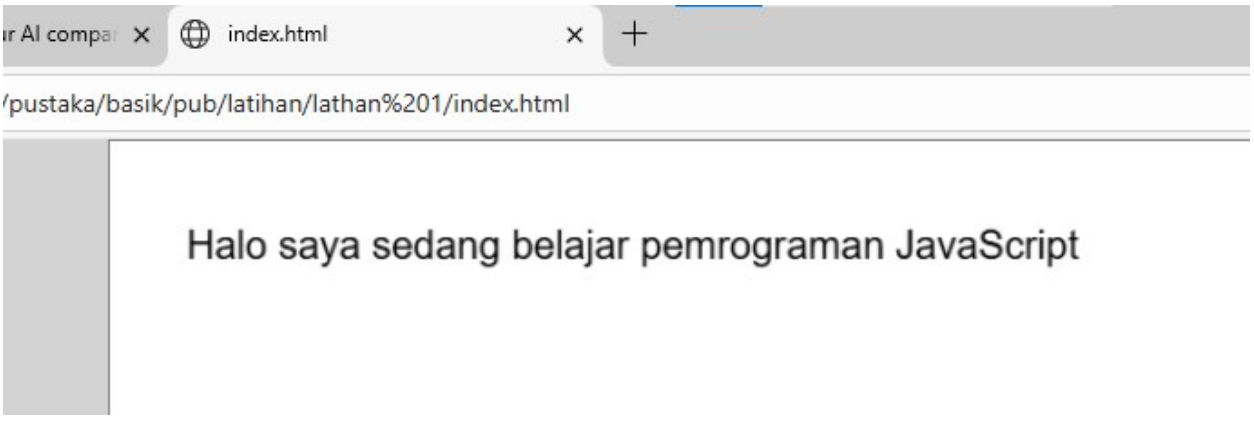
🔄 Contoh Event: update

Event Update terjadi **berulang kali setiap detik** (sekitar 30–60 kali). Cocok banget untuk membuat animasi atau efek yang terus bergerak.

Contoh:

```
function update() {
  bersihkanLayar(); // Bersihkan layar
  posisiTeks(mouseX(), mouseY()); // Ambil posisi mouse
  tulis("Halo saya sedang belajar pemrograman JavaScript");
}
```

📌 Hasilnya: Tulisan akan mengikuti posisi mouse ke mana pun kamu gerakkan!



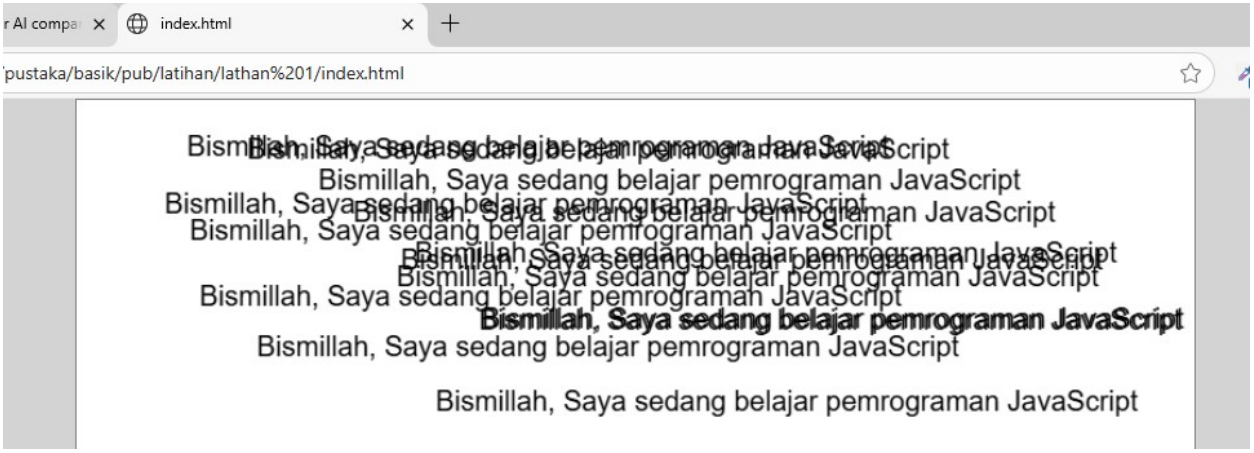
🖱️ Contoh Event: Klik Mouse

Kalau kamu ingin program merespons saat kamu **klik layar**, kamu bisa pakai function `tombolDitekan`.

Contoh:

```
function mouseDitekan() {
  posisiTeks(mouseX(), mouseY());
  tulis("Bismillah, Saya sedang belajar pemrograman JavaScript");
}
```

📌 Hasilnya: Saat kamu klik, tulisan muncul di tempat kamu klik! Klik sebanyak mungkin dimana saja dan lihat hasilnya.



🎯 Kesimpulan

Event membuat aplikasi jadi **interaktif dan hidup**. Kamu bisa:

- Menangani gerakan mouse
- Menangani klik
- Menangani update layar
- Dan banyak lagi!

Kalau kamu suka bikin game atau aplikasi interaktif, belajar event itu wajib banget!

🎨 Object Gambar

Dalam dunia coding, kita bisa menampilkan gambar seperti latar belakang, karakter, atau benda-benda lain. Di dalam BASIK, gambar diwakili oleh object bernama Gambar, dan kita bisa **mengatur posisi, ukuran, bahkan membuatnya bergerak atau berputar**!

🖼️ Cara Memuat Gambar

Untuk menampilkan gambar, kita gunakan fungsi muatGambar():

```
let bg = muatImage("bg_bintang.jpg");
```

Artinya: kita memuat gambar bernama `bg_bintang.jpg` dan menyimpannya ke dalam variabel bg.

Gambar bg_bintang.jpg ini berada di folder bernama **asset**. Kamu bisa membuka folder ini untuk melihat gambar apa saja yang tersedia, kamu juga bisa menambahkan gambar kamu sendiri.

🧠 Menampilkan Gambar ke Layar

Setelah gambar dimuat, kita harus menggambarnya ke layar dengan:

```
stempel(bg);
```

Biasanya kita taruh ini di dalam function `update()` supaya gambar terus muncul saat program berjalan.

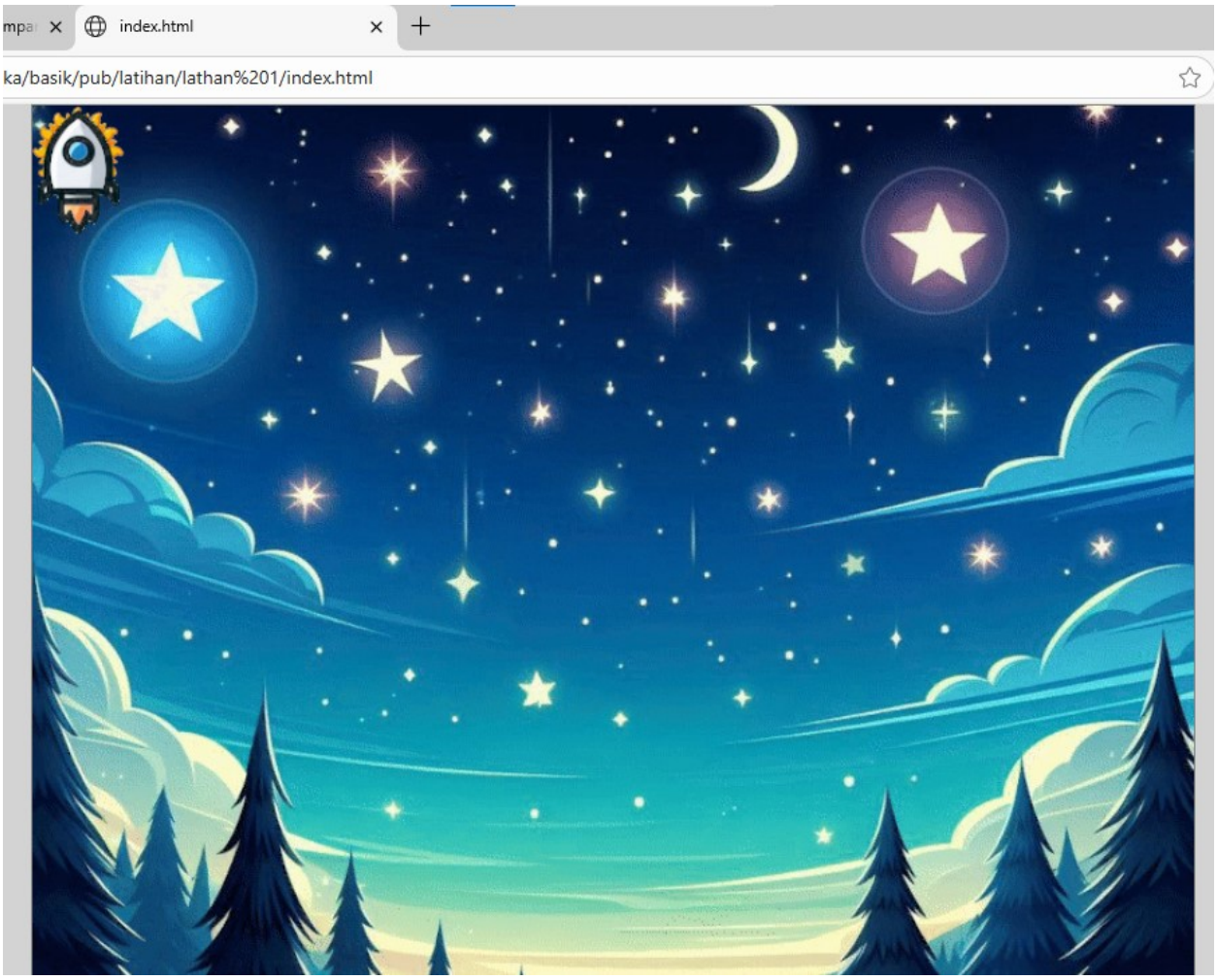
🌞 Menambahkan Banyak Gambar

Kita bisa memuat lebih dari satu gambar:

```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

function update() {
  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
}
```

📌 Hasilnya:



🗂️ Gambar sebagai Object?

Gambar yang kita muat adalah **object**. Object itu seperti kotak ajaib yang punya:

- **Property** (sifat): seperti posisi, ukuran, rotasi
- **Method** (aksi): seperti bergerak, berputar, dll

Contoh mengatur posisi gambar:

```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

matahari.x = 142;
matahari.y = 142;

function update() {
  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
}
```

📌 Artinya: kita geser gambar matahari ke kanan dan ke bawah.

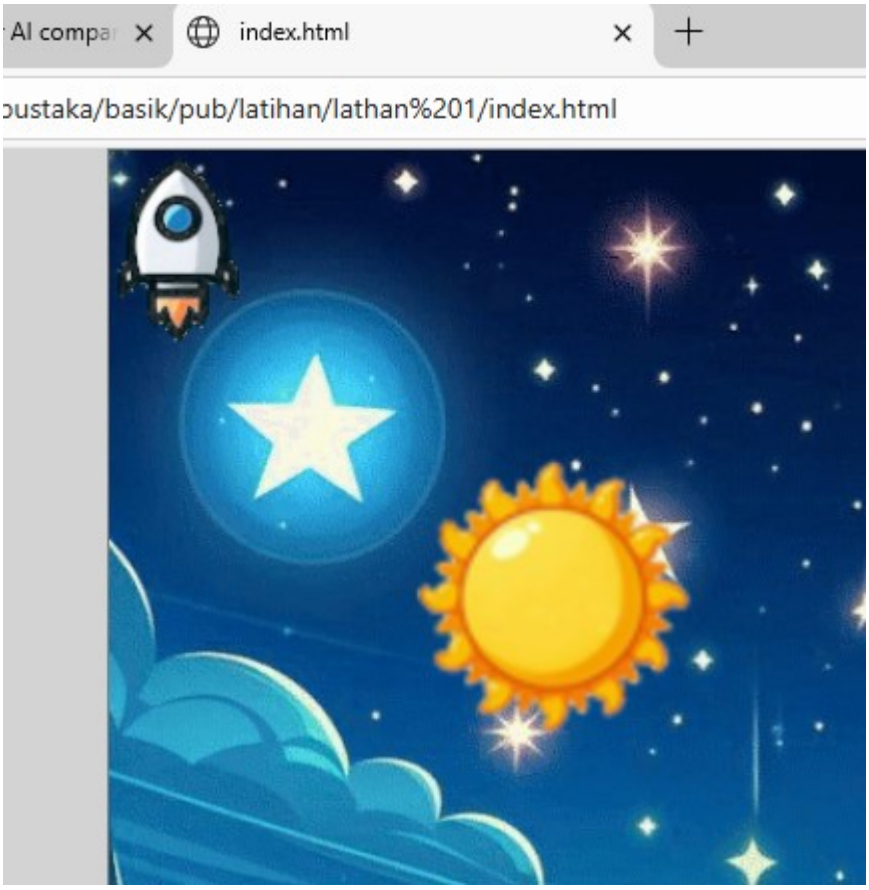
📏 Mengubah Ukuran Gambar

Kita bisa ubah ukuran gambar dengan:

```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

matahari.x = 142;
matahari.y = 142;
matahari.panjang = 128;
matahari.lebar = 128;

function update() {
  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
}
```



🚀 Membuat Gambar Bergerak Ikuti Mouse

Kita bisa buat gambar roket mengikuti posisi mouse:

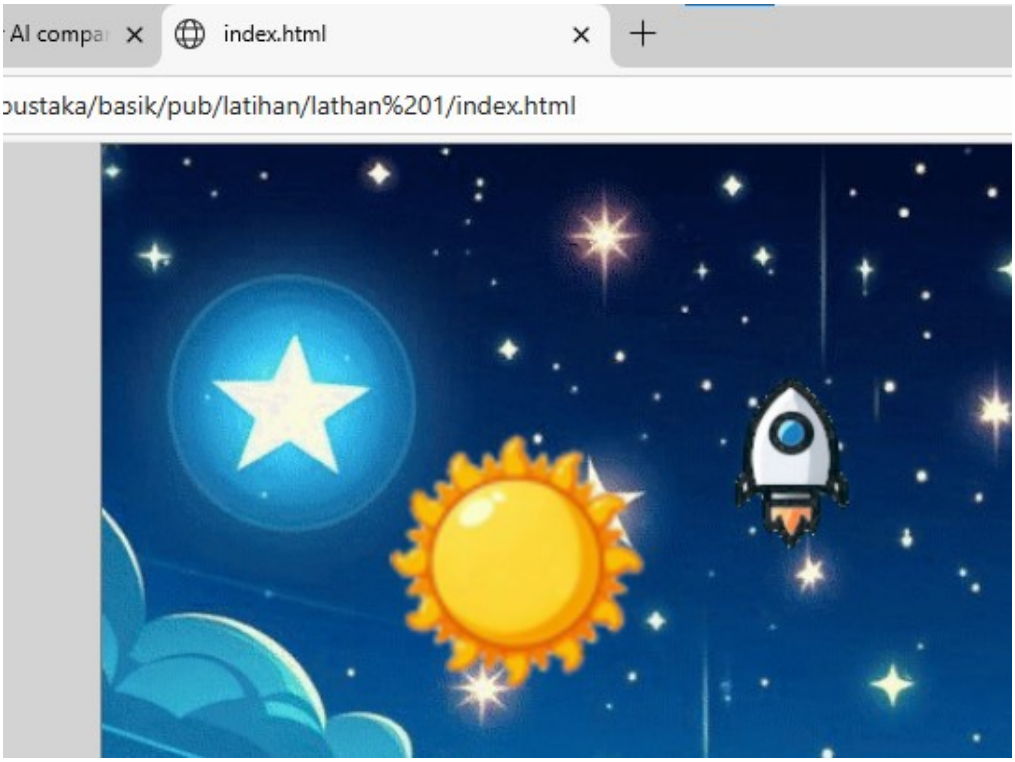
```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

matahari.x = 142;
matahari.y = 142;
matahari.panjang = 128;
matahari.lebar = 128;

function update() {
  roket.x = mouseX();
  roket.y = mouseY();

  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
}
```

📌Jika dijalankan, maka roket akan mengikuti kemana mouse bergerak.



Tapi... posisi roket mungkin tidak pas karena pusat gambarnya ada di pojok. Kita bisa ubah pusatnya ke tengah:

```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

matahari.x = 142;
matahari.y = 142;
matahari.panjang = 128;
matahari.lebar = 128;

roket.pusatX = 32;
roket.pusatY = 46

function update() {
  roket.x = mouseX();
  roket.y = mouseY();

  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
};
```

📌 Sekarang roketnya berada di tengah-tengah kursor


index.html
taka/basik/pub/latihan/lathan%201/index.html

Membuat Gambar Berputar

Kita bisa buat gambar berputar dengan properti rotasi:

```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

matahari.x = 142;
matahari.y = 142;
matahari.panjang = 128;
matahari.lebar = 128;

roket.pusatX = 32;
roket.pusatY = 46

function update() {
  roket.x = mouseX();
  roket.y = mouseY();

  matahari.rotasi++;

  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
}
```

Tapi supaya rotasinya bagus, kita harus atur pusat rotasinya ke tengah gambar:

```
let bg = muatImage("bg_bintang.jpg");
let matahari = muatImage("matahari.png");
let roket = muatImage("roket.png");

matahari.x = 142;
matahari.y = 142;
matahari.panjang = 128;
matahari.lebar = 128;
matahari.pusatX = 64;
matahari.pusatY = 64

roket.pusatX = 32;
roket.pusatY = 46

function update() {
  roket.x = mouseX();
  roket.y = mouseY();

  matahari.rotasi++;

  gambarImage(bg);
  gambarImage(matahari);
  gambarImage(roket);
};
```

🎯 Kesimpulan

Dengan Image, kita bisa:

- Menampilkan gambar ke layar
- Mengatur posisi dan ukuran
- Membuat gambar bergerak dan berputar
- Membuat tampilan aplikasi jadi keren dan interaktif!

LAMPIRAN: Daftar Perintah BASIK

Perintah Umum

buatKanvas(panjang, lebar, kanvas, mode)

Mulai aplikasi. Anda harus memanggil perintah ini sebelum perintah lain.

Parameter:

Nama	Tipe	Default	Deskripsi
panjang	angka	320	Panjang kanvas yang diinginkan. nilai 0 akan diabaikan
lebar	angka	240	Lebar kanvas yang diinginkan. Nilai 0 akan diabaikan
kanvas	HTMLCanvasElement	null	(opsional) elemen kanvas. Jika kanvas tidak tersedia, kanvas baru akan dibuat dan ukurannya mengikuti ukuran yang diinginkan
mode	boolean	1	(default true) Gunakan mode layar penuh. Pada mode layar penuh, kanvas akan otomatis memenuhi layar dan menjaga rasio aspek.

kanvas() → { HTMLCanvasElement }

Mengembalikan referensi ke kanvas yang sedang aktif

setKanvas(c)

Set kanvas aktif.

Parameter:

Nama	Tipe	Deskripsi
c	HTMLCanvasElement	kanvas aktif yang baru

bersihkanLayar(x, y, x2, y2)

Membersihkan kanvas

Parameter:

Nama	Tipe	Default	Deskripsi

Nama	Tipe	Default	Deskripsi
x	number	0	(opsional) posisi x pertama
y	number	0	(opsional) posisi y pertama
x2	number	lebar-kanvas	(opsional) posisi x kedua
y2	number	tinggi kanvas	(opsional) posisi y kedua

Perintah Gambar

semuaGambarSelesaiDimuat() → {boolean}

Cek apakah semua gambar sudah dimuat

buatGambar(panjang, lebar) → {Gambar}

Membuat gambar kosong

Parameter:

Nama	Tipe	Deskripsi
panjang	number	lebar
lebar	number	tinggi

stempel(gbr)

Menggambar gambar ke layar

Parameter:

Nama	Tipe	Deskripsi
gbr	Gambar	Gambar yang akan di stempel ke layar

hapusGambar(gbr)

Hapus gambar dan semua sumber daya yang digunakan

Parameter:

Nama	Tipe	Deskripsi
gbr	Gambar	gambar yang akan dihapus

gambarTabrakan(gbr1, gbr2) → {boolean}

Cek apakah dua gambar bertabrakan. Menggunakan box untuk deteksi tabrakan. Memperhitungkan rotasi juga.

Parameter:

Nama	Tipe	Deskripsi
gbr1	Image	gambar pertama
gbr2	Image	gambar kedua

poinDidalamGambar(gbr, x, y) → {boolean}

Cek apakah gambar bertabrakan dengan titik tertentu

Parameter:

Nama	Tipe	Deskripsi
gbr	Gambar	gambar
x	number	posisi x yang diuji
y	number	posisi y yang diuji

muatGambar(namaFile) → {Gambar}

Muat gambar dari nama file

Parameter:

Nama	Tipe	Deskripsi
namaFile	string	Nama file gambar yang akan dimuat

Keyboard

tombolDitahan(tombol) → {boolean}

Cek apakah sebuah tombol sedang ditekan

Parameter:

Nama	Tipe	Deskripsi
tombol	string	tombol yang dicek

tombolEvent() → {string}

Mengembalikan tombol terakhir pada event tombol

Mouse

mouseDragAwalX() → {number}

posisi x awal drag

mouseDragAwalY() → {number}

posisi y awal drag

mouseDragX() → {number}

panjang drag horizontal dihitung dari posisi awal drag

mouseDragY() → {number}

panjang drag vertikal dihitung dari posisi awal drag

mouseDitahan() → {boolean}

status mouse ditekan

mouseDidrag() → {boolean}

mouse sedang didrag

mouseGerakX() → {number}

mengembalikan pergerakan mouse horizontal terakhir

mouseGerakY() → {number}

mengembalikan pergerakan mouse vertikal terakhir

MouseX() → {number}

posisi x mouse

MouseY() → {number}

posisi y mouse

Event

Fungsi ini akan dieksekusi otomatis ketika tersedia dan event tertentu terjadi

Nama	Deskripsi
keyboardDitekan	Pengguna mulai menekan keyboard
keyboardDilepas	Keyboard dilepas
mouseDitekan	Pengguna mulai menekan tombol mouse
mouseDilepas	Tombol mouse dilepas
mouseMulaiDrag	Drag dimulai
mouseSelesaiDrag	Drag selesai
mouseGerak	Mouse bergerak
mouseKlik	Klik
update	Update aplikasi
suaraSelesai	Suara selesai dimainkan

Objek

Gambar

Properti:

Nama	Tipe	Deskripsi
x	number	posisi x
y	number	posisi y
rotasi	number	rotasi dalam derajat
alpha	number	alpha (0 - 100)
panjang	number	lebar yang diinginkan
lebar	number	tinggi yang diinginkan
pusatX	number	posisi handle x
pusatY	number	posisi handle y
ubin	boolean	gambar direnderer sebagai ubin

Nama	Tipe	Deskripsi
didrag	boolean	gambar sedang didrag
ditekan	boolean	gambar ditekan
dimuat	boolean	gambar sudah selesai dimuat