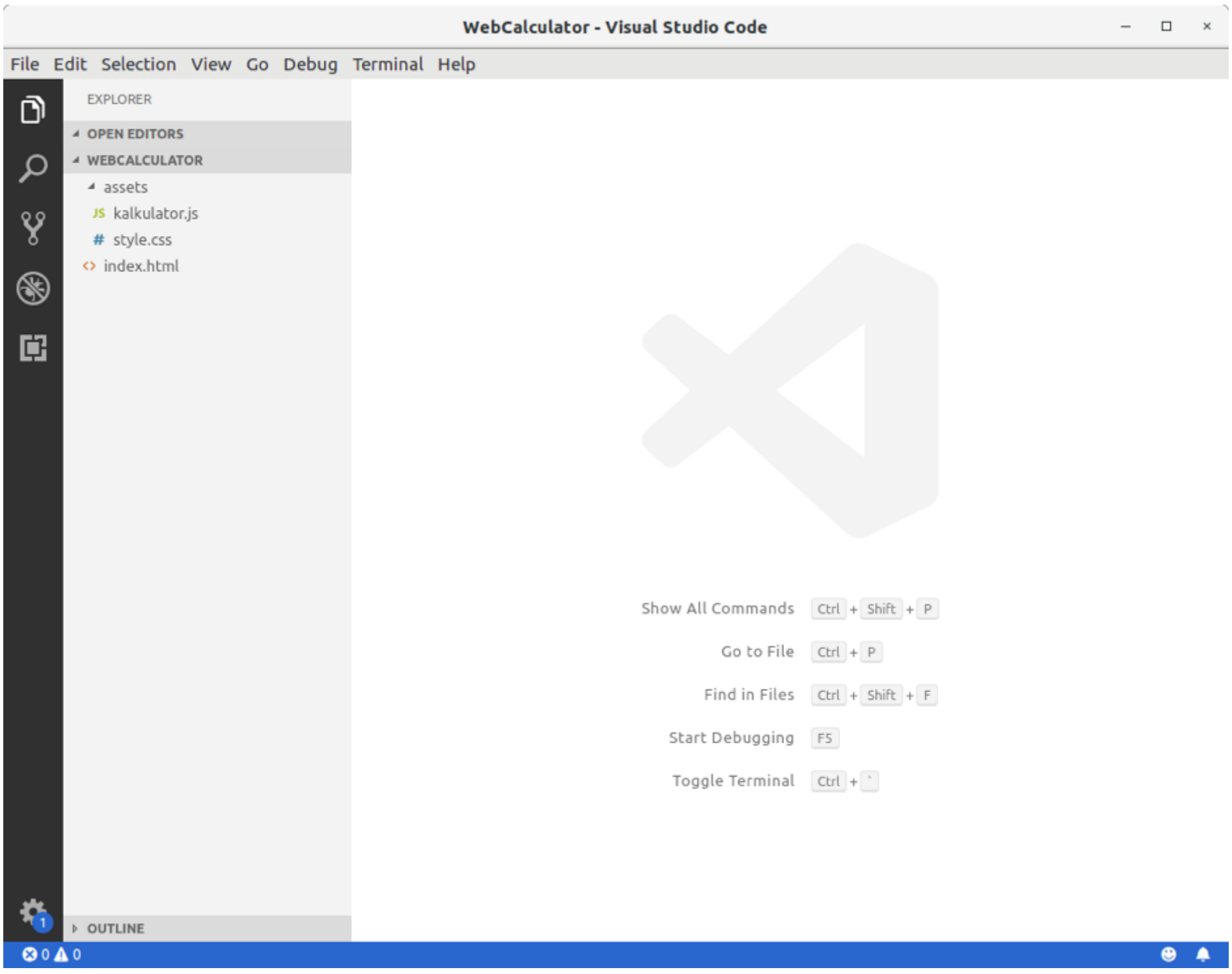
Menerapkan JavaScript pada Web Kalkulator

Sejauh ini kita sudah mempelajari bagaimana cara menggunakan JavaScript pada HTML, bagaimana menuliskan sintaks JavaScript, mengenal tipe data dan fungsi sintaks yang ada, hingga akhirnya memanipulasi dan memberikan event pada elemen HTML melalui DOM *Object*.

Dengan komponen yang sudah disebutkan tadi, sepertinya sekarang kita sudah cukup bekal untuk memberikan “nyawa” pada Web Kalkulator kita sehingga dapat berfungsi layaknya sebuah kalkulator pada umumnya. Let’s do it!

Silakan buka project Web Kalkulator dengan editor favorit kita.



Jika Anda mengikuti seluruh latihan sebelumnya, maka struktur projek tampak seperti gambar di atas. Kita juga sudah mencoba menghubungkan berkas JavaScript (***kalkulator.js***) dengan berkas HTML (***index.html***) lalu menuliskan sintaks dasar untuk menampilkan pesan pada console browser.

Selanjutnya kita akan bekerja full pada ***kalkulator.js***. Silakan hapus sintaks yang sudah kita buat sebelumnya.

```
1.  console.log("Selamat Anda berhasil menggunakan JavaScript pada Website")
```

Langkah pertama adalah buatlah sebuah objek dengan nama `calculator`. Di dalamnya terdapat properti yang menggambarkan data dan kondisi dari kalkulatornya, seperti `displayNumber`, `operator`, `firstNumber`, dan `waitingForSecondNumber`. Sehingga kodenya akan nampak seperti ini:

```
    displayNumber: '0',
    operator: null,
    firstNumber: null,
    waitingForSecondNumber: false
};
```

Kita gunakan objek ini sebagai tempat menyimpan data dan kondisi pada calculator, di mana angka yang muncul pada layar kalkulator selalu diambil dari data `calculator.displayNumber` .

Properti operator dan `firstNumber` kita gunakan nilai null terlebih dahulu karena properti tersebut akan diberikan nilai ketika pengguna melakukan aksi.

Dan properti `waitingForSecondNumber` merupakan kondisi di mana kalkulator sedang menunggu pengguna menentukkan angka kedua dalam melakukan perhitungan.

Setelah membuat *object* `calculator` , selanjutnya kita buat fungsi - fungsi umum yang dilakukan kalkulator seperti meng-*update* angka pada layar dan menghapus data pada kalkulator.

```
function updateDisplay() {
    document.querySelector("#displayNumber").innerText = calculator.displayNumber;
}

function clearCalculator() {
    calculator.displayNumber = '0';
    calculator.operator = null;
    calculator.firstNumber = null;
    calculator.waitingForSecondNumber = false;
}
```

Lalu kita buat juga fungsi untuk memasukkan angka ke dalam nilai `displayNumber` kalkulator.

```
function inputDigit(digit) {
    calculator.displayNumber += digit;
}
```

Kemudian kita buat variabel `buttons` dengan menginisialisasikan nilai seluruh elemen button yang ada, dan berikan event **click** pada tiap elemennya.

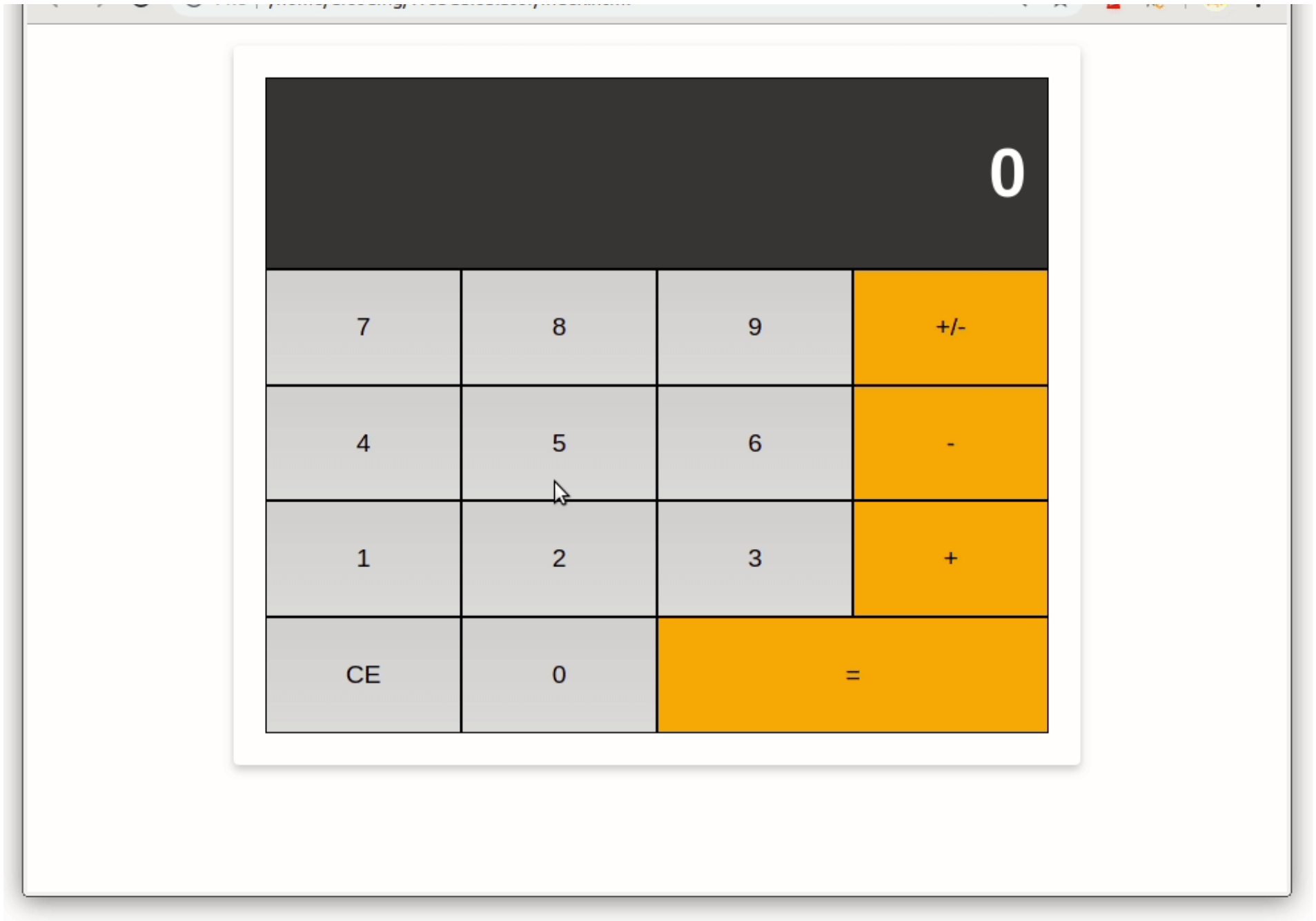
Untuk mendapatkan nilai seluruh elemen button kita gunakan `querySelectorAll(“.button”)` kemudian kita *looping* nilainya dan berikan event **click** pada tiap itemnya.

```
for (let button of buttons) {
    button.addEventListener('click', function(event) {

        // mendapatkan objek elemen yang diklik
        const target = event.target;

        inputDigit(target.innerText);
        updateDisplay()
    });
}
```

Sehingga keseluruhan kode pada ***kalkulator.js*** akan tampak seperti ini:

```
const calculator = {
    displayNumber: '0',
    operator: null,
    firstNumber: null,
    waitingForSecondNumber: false
};

function updateDisplay() {
    document.querySelector("#displayNumber").innerText = calculator.displayNumber;
}

function clearCalculator() {
    calculator.displayNumber = '0';
    calculator.operator = null;
    calculator.firstNumber = null;
    calculator.waitingForSecondNumber = false;
}

function inputDigit(digit) {
```

Sekarang kita coba jalankan ***index.html*** pada browser, dan lihat fungsi kalkulator untuk yang pertama kali.

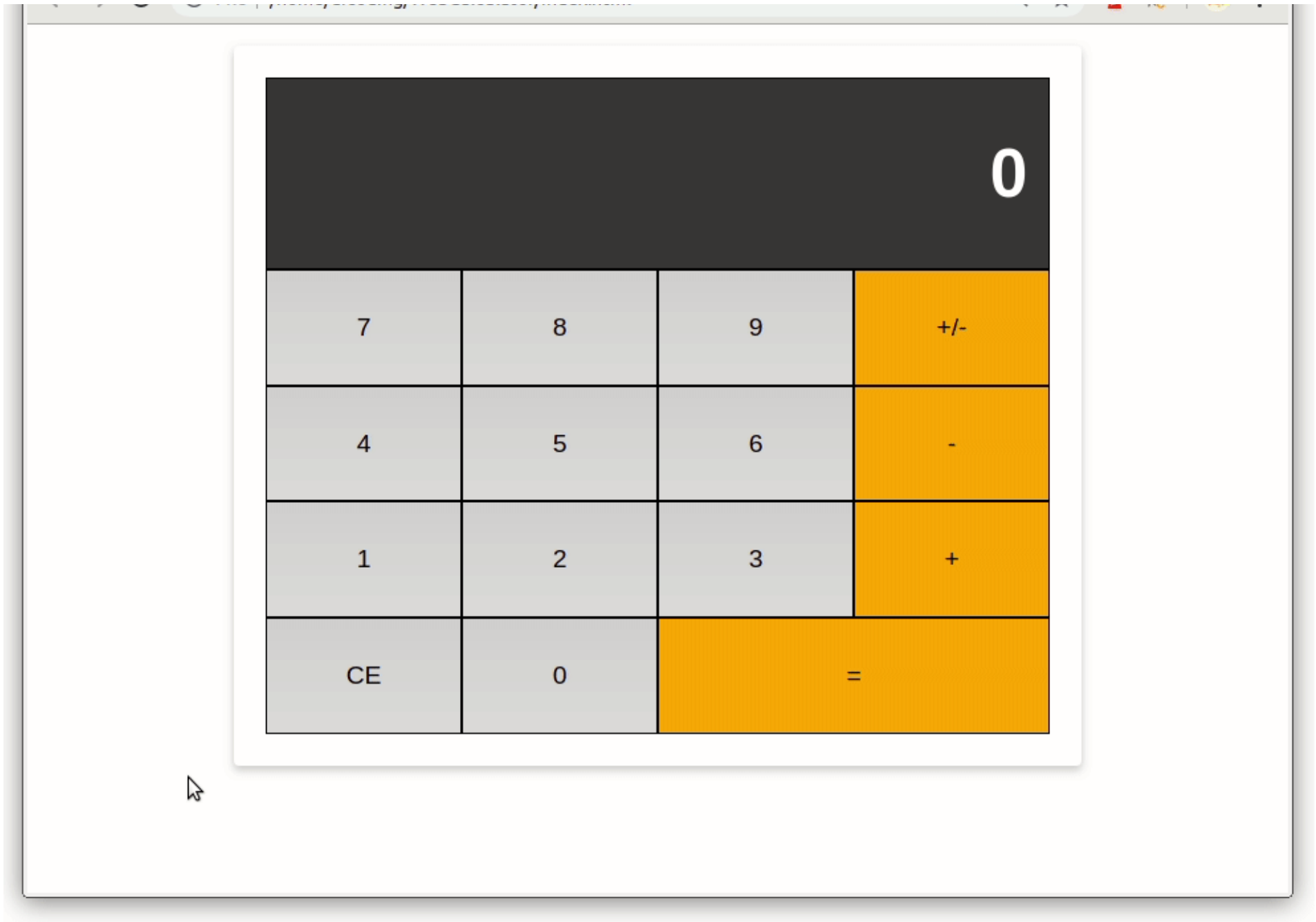


Uh sangat aneh bukan? Siapa yang ingin gunakan kalkulator seperti itu? Untuk saat ini tak apa. Setidanya kode yang kita tuliskan sudah berhasil berjalan dengan baik. Selanjutnya kita akan memperbaiki keanehan-keanehan yang ada satu per satu.

Saat ini kalkulator masih dapat menampilkan angka 0 di awal bilangan, hal itu tentu aneh dan tidak pernah terjadi pada kalkulator manapun terkecuali dalam menampilkan bilangan desimal. Untuk memperbaikinya, pada fungsi `inputDigit()`, tambahkan sebuah kondisi dimana jika `displayNumber` bernilai '0', maka angka yang pertama dimasukkan pengguna akan menggantikan keseluruhan nilai `displayNumber` selain itu, lakukan seperti biasanya. Untuk melakukannya kita gunakan *if-else statement*.

```
function inputDigit(digit) {
    if(calculator.displayNumber === '0') {
        calculator.displayNumber = digit;
    } else {
        calculator.displayNumber += digit;
    }
}
```

Dengan begitu kalkulator tidak akan menampilkan angka 0 diawal bilangan lagi.



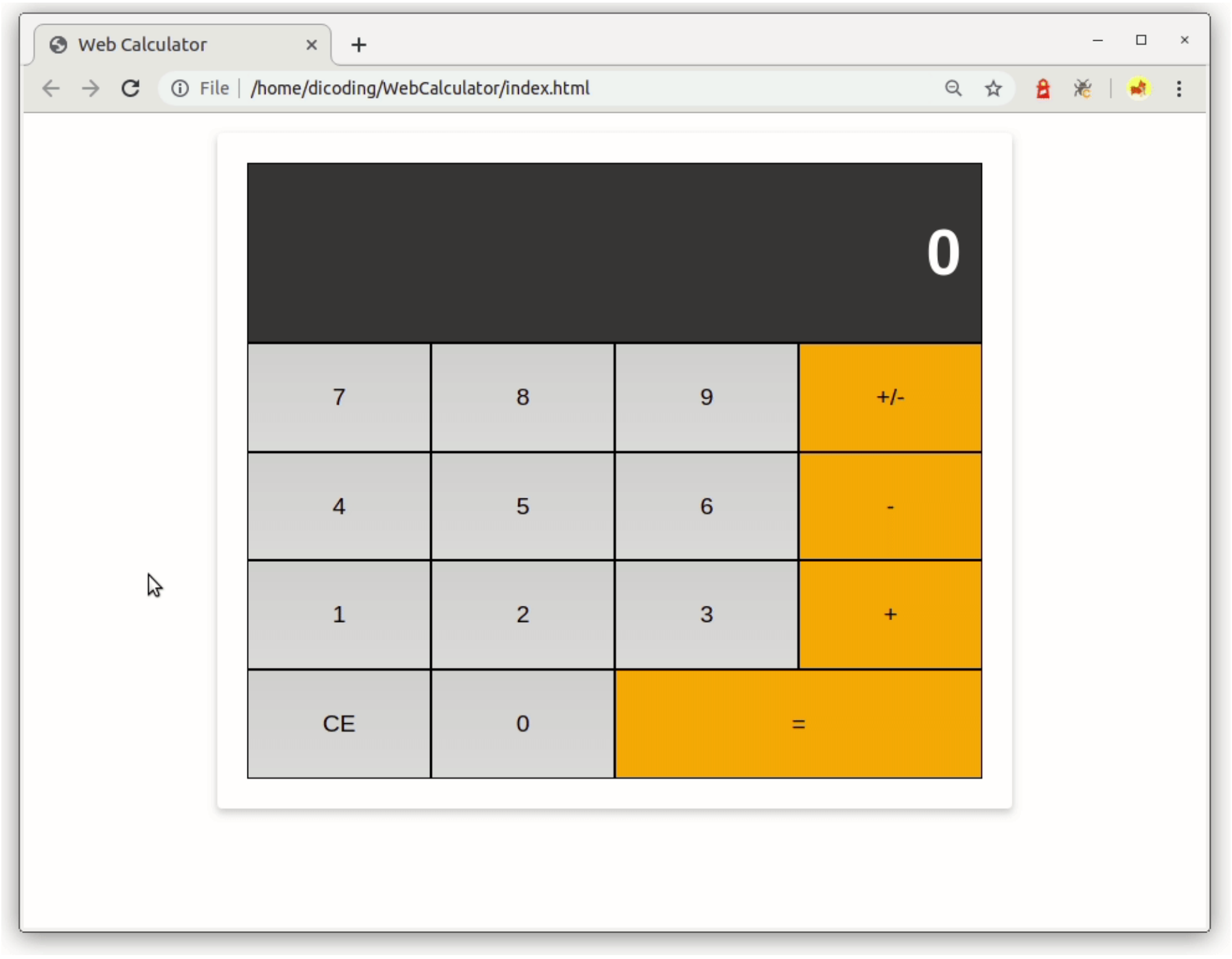
Kemudian kita akan membuat fungsi `clear` pada kalkulator berjalan dengan semestinya sehingga dalam mereset kalkulator kita tidak perlu melakukan reload pada browser seperti yang ditunjukkan pada gif di atas.

Pada *event handler*, kita tambahkan kondisi dimana ketika *event target* merupakan elemen yang menerapkan class `clear` maka kita akan panggil fungsi `clearCalculator()` .

```
1.  button.addEventListener('click', function(event) {
2.
3.      // mendapatkan objek elemen yang diklik
4.      const target = event.target;
5.
6.      if(target.classList.contains('clear')) {
7.          clearCalculator();
8.          updateDisplay();
9.          return;
10.     }
11.
12.     inputDigit(target.innerText);
13.     updateDisplay()
14.  });
```

Kita bisa memanfaatkan event.classList untuk melihat nilai class apa saja dalam bentuk array yang ada pada element target, kemudian menggunakan `contains()` yang merupakan *method* dari array yang berguna untuk memastikan nilai yang terkandung di dalam array tersebut.

handler terhenti sehingga kode yang ada di bawahnya tidak ikut tereksekusi.

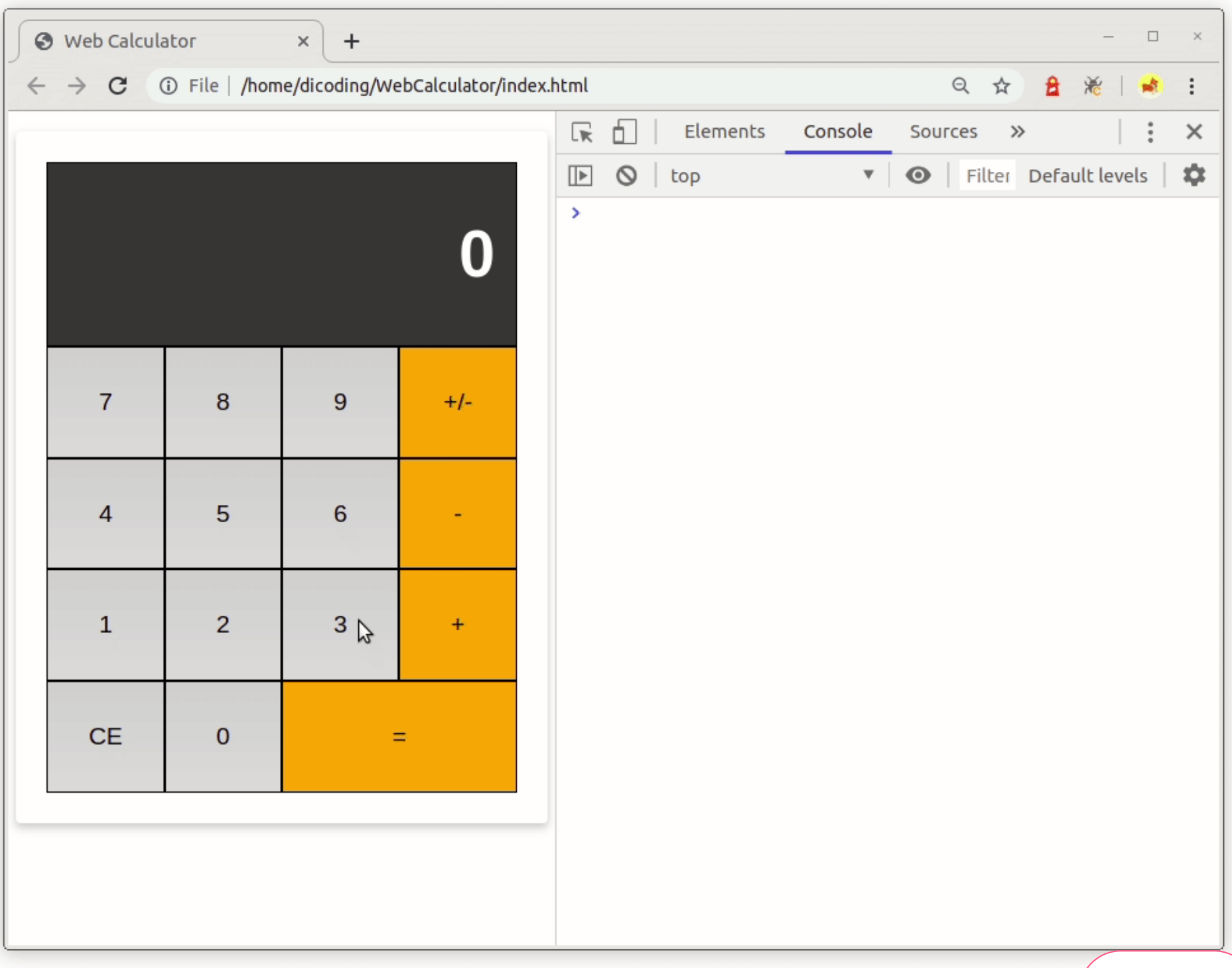


Setelah menerapkan kondisi tersebut maka seluruh kode pada ***kalkulator.js*** akan tampak seperti berikut:

```
const calculator = {
    displayNumber: '0',
    operator: null,
    firstNumber: null,
    waitingForSecondNumber: false
};

function updateDisplay() {
    document.querySelector("#displayNumber").innerText = calculator.displayNumber;
}

function clearCalculator() {
    calculator.displayNumber = '0';
    calculator.operator = null;
    calculator.firstNumber = null;
    calculator.waitingForSecondNumber = false;
}
```

```
button.addEventListener('click', function(event) {

        // mendapatkan objek elemen yang diklik
        const target = event.target;

        if (target.classList.contains('clear')) {
            clearCalculator();
            updateDisplay();
            return;
        }

        if(target.classList.contains('negative')) {
            inverseNumber();
            updateDisplay();
            return;
        }

        if(target.classList.contains('equals')) {
```

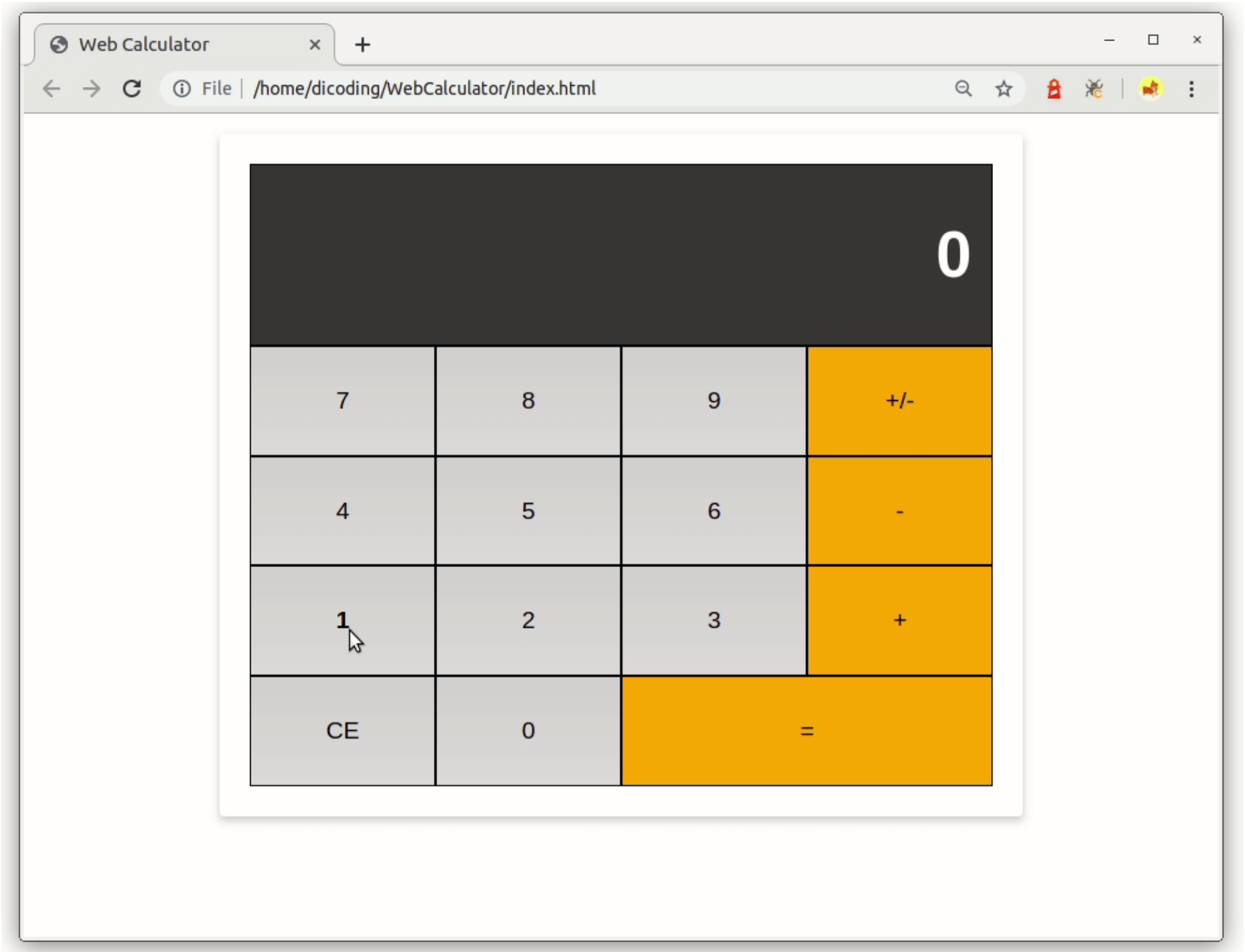
Jika kita membukanya sekarang, maka eror akan muncul ketika tombol - tombol fungsi dan operatornya ditekan.



Hal tersebut wajar karena kita belum mendefinisikan seluruh fungsi - fungsi yang kita dituliskan di atas. Dengan kita buat fungsinya mulai dari `inverseNumber()`

```
    if (calculator.displayNumber === '0') {
        return;
    }
    calculator.displayNumber = calculator.displayNumber * -1;
}
```

Fungsi inverseNumber() cukuplah simple karena kita hanya perlu melakukan perkalian displayNumber dengan -1, terkecuali jika displayNumber masih bernilai ‘0’ maka perkalian tidak akan dilakukan.

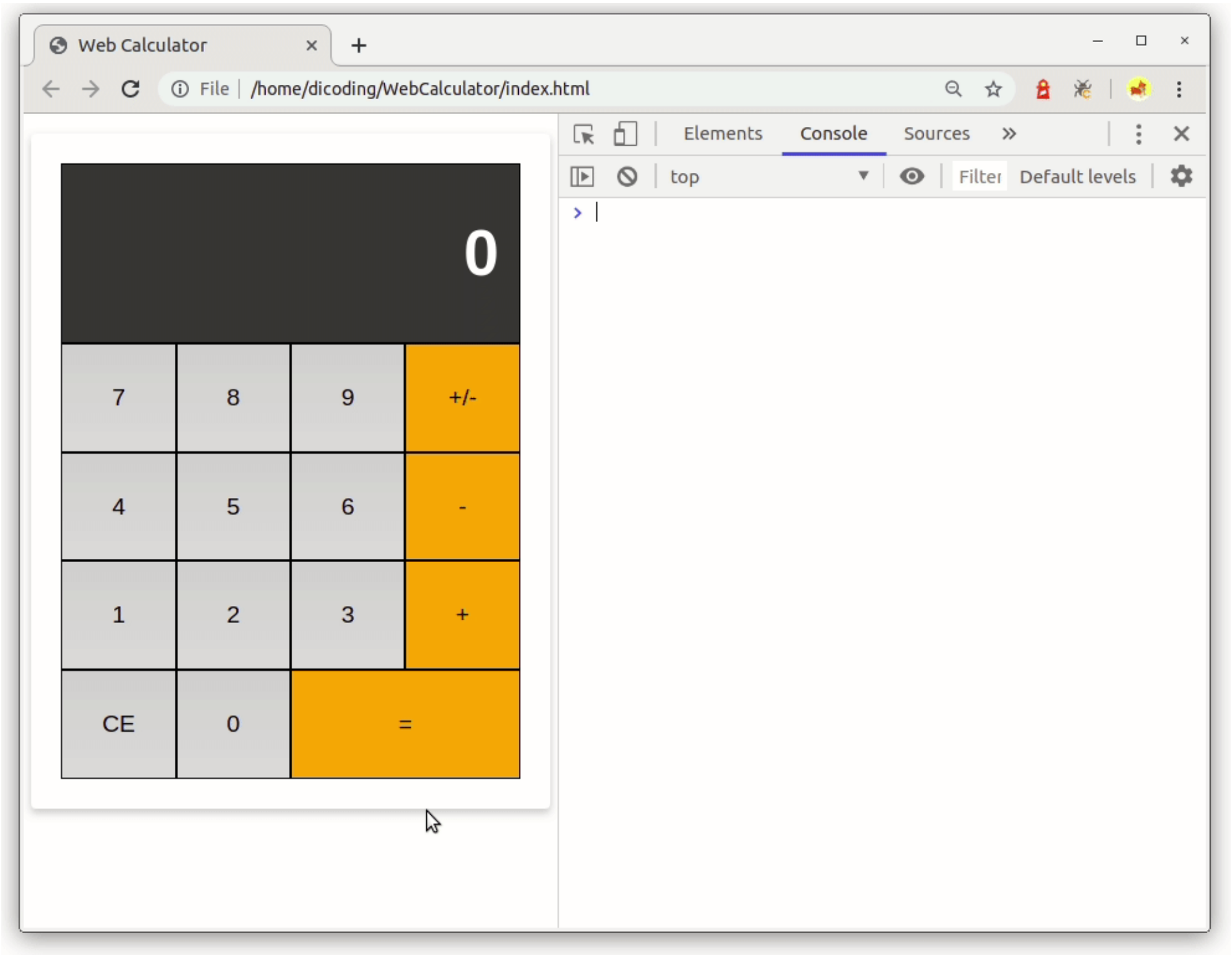


Sekarang tombol "+/-" sudah berfungsi dengan baik, selanjutnya kita akan membuat fungsi untuk menetapkan sebuah operator, baik itu + atau - pada kalkulator. Tuliskan fungsi berikut:

```
function handleOperator(operator) {
    if (!calculator.waitingForSecondNumber) {
        calculator.operator = operator;
        calculator.waitingForSecondNumber = true;
        calculator.firstNumber = calculator.displayNumber;

        // mengatur ulang nilai display number supaya tombol selanjutnya dimulai dari angka pertama lagi
        calculator.displayNumber = '0';
    } else {
        alert('Operator sudah ditetapkan')
    }
}
```

firstNumber dengan nilai displayNumber saat ini pada *object* calculator , hanya jika properti waitingForSecondNumber bernilai false . Namun jika waitingForSecondNumber bernilai true , browser akan menampilkan alert() dengan pesan **"Operator sudah ditetapkan"**

Voila! Sekarang tombol operator sudah dapat menetapkan nilai operator pada *object calculator*.



Kita bisa lihat pada console bahwa sekarang nilai properti operator dan firstNumber tidak lagi null , begitu pula dengan properti waitingForSecondNumber yang sudah berubah menjadi true .

Kita buat fungsi terakhir yakni performCalculation() . Fungsi ini digunakan untuk melakukan kalkulasi terhadap nilai - nilai yang terdapat pada objek calculator , sehingga pastikan kalkulator sudah memiliki nilai operator dan firstNumber ketika fungsi ini dijalankan.

```
    if (calculator.firstNumber == null || calculator.operator == null) {
        alert("Anda belum menetapkan operator");
        return;
    }

    let result = 0;
    if (calculator.operator === "+") {
        result = parseInt(calculator.firstNumber) + parseInt(calculator.displayNumber);
    } else {
        result = parseInt(calculator.firstNumber) - parseInt(calculator.displayNumber)
    }

    calculator.displayNumber = result;
}
```

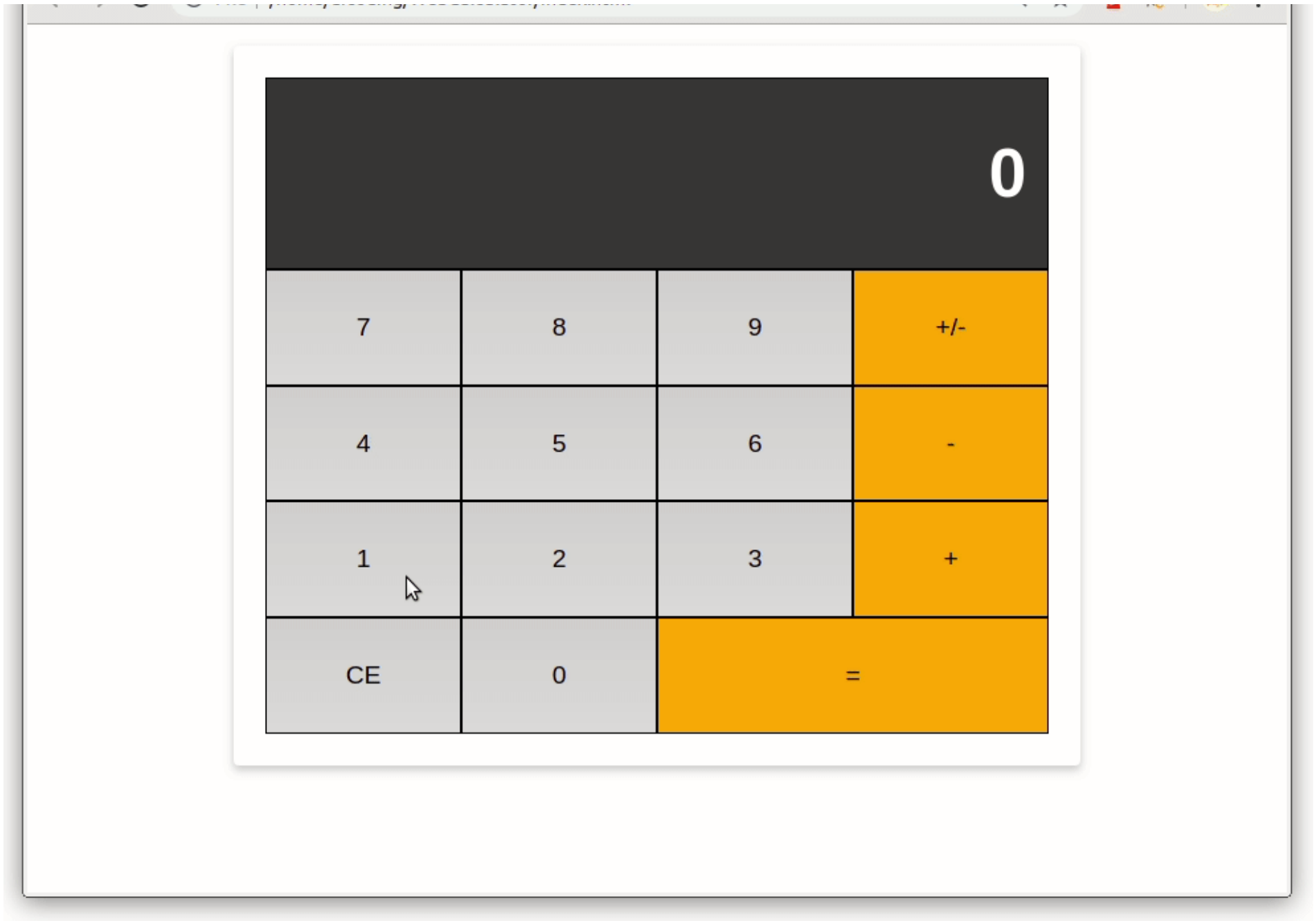
Fungsi tersebut diawali dengan pengecekan nilai-nilai yang dibutuhkan untuk melakukan kalkulasi. Jika tidak terpenuhi maka proses akan dihentikan. Namun jika terpenuhi kalkulasi akan dilakukan.

Dalam melakukan kalkulasi terdapat pengecekan tipe operator apa yang akan dilakukan. Kita juga menggunakan `parseInt()` untuk mengubah nilai **string** menjadi **number**. Mengapa konversi tipe data dibutuhkan? Sejatinya kita menggunakan string dalam menampilkan nilai pada jendela browser, namun untuk proses kalkulasi kita membutuhkan number.

Setelah menambahkan fungsi tersebut, maka seluruh struktur kode pada berkas `kalkulator.js` akan tampak seperti ini:

```
const calculator = {
    displayNumber: '0',
    operator: null,
    firstNumber: null,
    waitingForSecondNumber: false
};

function updateDisplay() {
    document.querySelector("#displayNumber").innerText = calculator.displayNumber;
}

function clearCalculator() {
    calculator.displayNumber = '0';
    calculator.operator = null;
    calculator.firstNumber = null;
    calculator.waitingForSecondNumber = false;
}

function inputDigit(digit) {
```

Jika kita buka ***index.html*** sekarang, seharusnya kalkulator sudah dapat melakukan kalkulasi.



Selamat! Sejauh ini kita sudah bisa membuat aplikasi web dengan memanfaatkan pengetahuan yang sudah kita pelajari. Selanjutnya kita akan mengenal salah satu Web API yang menarik untuk diterapkan pada project kalkulator kita.

Sebelumnya

Selanjutnya

Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Decode Ideas
Discover Potential

Tentang Kami

Blog
Reward
Showcase
Hubungi Kami
FAQ

Penghargaan

image
click bila
belum muncul

DIBANTU