**JavaScript**

**Basics:**

* **console.log(“<contact>”) :** print contact into terminal part
* file extension: **.js**
* use in both backend and frontend part
* **Node.js** provides env for JS
* **const:** use to declare constant variable
* **var:** use to declare global variable
  1. if create variable using **var** in outside of function then you can also use it inside the function.
  2. If create variable using **var** inside the function then scope of that variable is within that function.
* **let:** use to declare local variable. Improved version of **var** keyword. Variable can’t access outside of the block.

****

**Practical of difference 2):**

// 1) const

const massage1 = "hello";

console.log(massage1);

// output: hello

// try to update variable first:

// massage1 = "How are you?";

// console.log(massage1);

// output : TypeError: Assignment to constant variable.

// we can't update variable value which declare using "const"

// 2) let:

let massage2 = "hello from let";

console.log(massage2);

// try to update variable value :

massage2 = "How are you?";

console.log(massage2);

// output: How are you?

// try to redeclare variable:

// let massage2 = "I am fine"; // SyntaxError: Identifier 'massage2' has already been declared

// 3) var:

var massage3 = "Hello from var";

console.log(massage3); // Hello from var

// try to update variable value:

massage3 = "How are you?";

console.log(massage3); // How are you?

// try to redeclare variable:

var massage3 = "I am fine";

console.log(massage3); // I am fine

**for difference 3):**

var first;

console.log(first); // output: undefined

let second;

console.log(second); // output: undefined

// const third; // output: SyntaxError: Missing initializer in const declaration

// console.log(third);

// => "var" and "let" : variable declared without intialization but "const" variable can't be declared without intialization

* **console.table([<attribute values]):** Print data into table format on terminal.
* **\* not use “var” for declare variable bcz of issue into black scope and functional scope.**
* **Undefined:** default value of variable declares using **let.**
* **“use strict”:** write in starting of code 🡪 after this code will be treated as newer version of JS
* **alert() :** run this type stmt in browser not in console.
* **Handle code readability**
* **Datatypes:** 
  1. **Primitive Datatypes:**
     + **string**
     + **number**
     + **Boolean**
     + **Bigint**
     + **Null:** It also standalone value in JS(One type of representation of empty value)
     + **Undefine:** Value which default gives to variable when we not assign any value into it.
     + **Symbol:** For find uniqueness.
  2. **Object:**
* **typeof :** use to identify any variable’s datatype.

**Statically vs Dynamically Typed**

Based on the provided search results, here’s a concise explanation of how to determine whether a language is statically typed or dynamically typed:

**Statically Typed Language:**

1. **Type checking occurs at compile-time**: The language checks the types of variables, function arguments, and return types before the code is executed.
2. **Variables must be declared with a specific data type**: The programmer must explicitly specify the data type for each variable before using it.
3. **Type inference is limited or absent**: The compiler does not automatically infer the data type of a variable based on its usage.

Examples of statically typed languages include:

* C
* C++
* Java
* Go

**Dynamically Typed Language:**

1. **Type checking occurs at runtime**: The language checks the types of variables, function arguments, and return types during the execution of the code.
2. **Variables do not require explicit data type declaration**: The programmer does not need to specify a data type for each variable before using it.
3. **Type inference is common**: The language infers the data type of a variable based on its usage, often using techniques like duck typing or structural subtyping.

Examples of dynamically typed languages include:

* Python
* JavaScript
* Ruby
* Perl
* PHP

To summarize:

* **Statically typed languages perform type checking at compile-time, require explicit data type declarations, and have limited type inference.**
* **Dynamically typed languages perform type checking at runtime, do not require explicit data type declarations, and often use type inference.**

**Ways to include JS into Code:**

1. **External JS:**

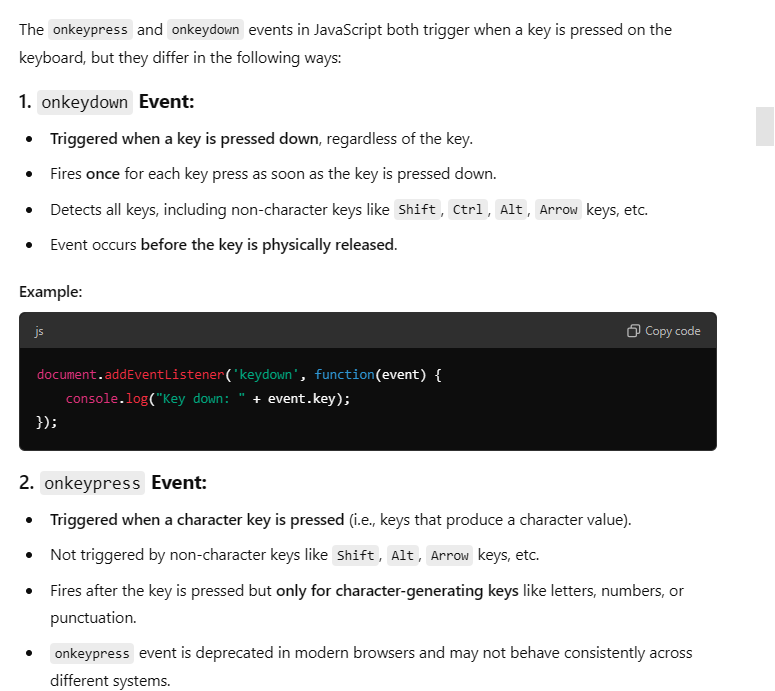
<script src="FILE.JS"></script>

1. **Internal JS:**

<script>DO SOMETHING</script>

1. **Inline JS:**

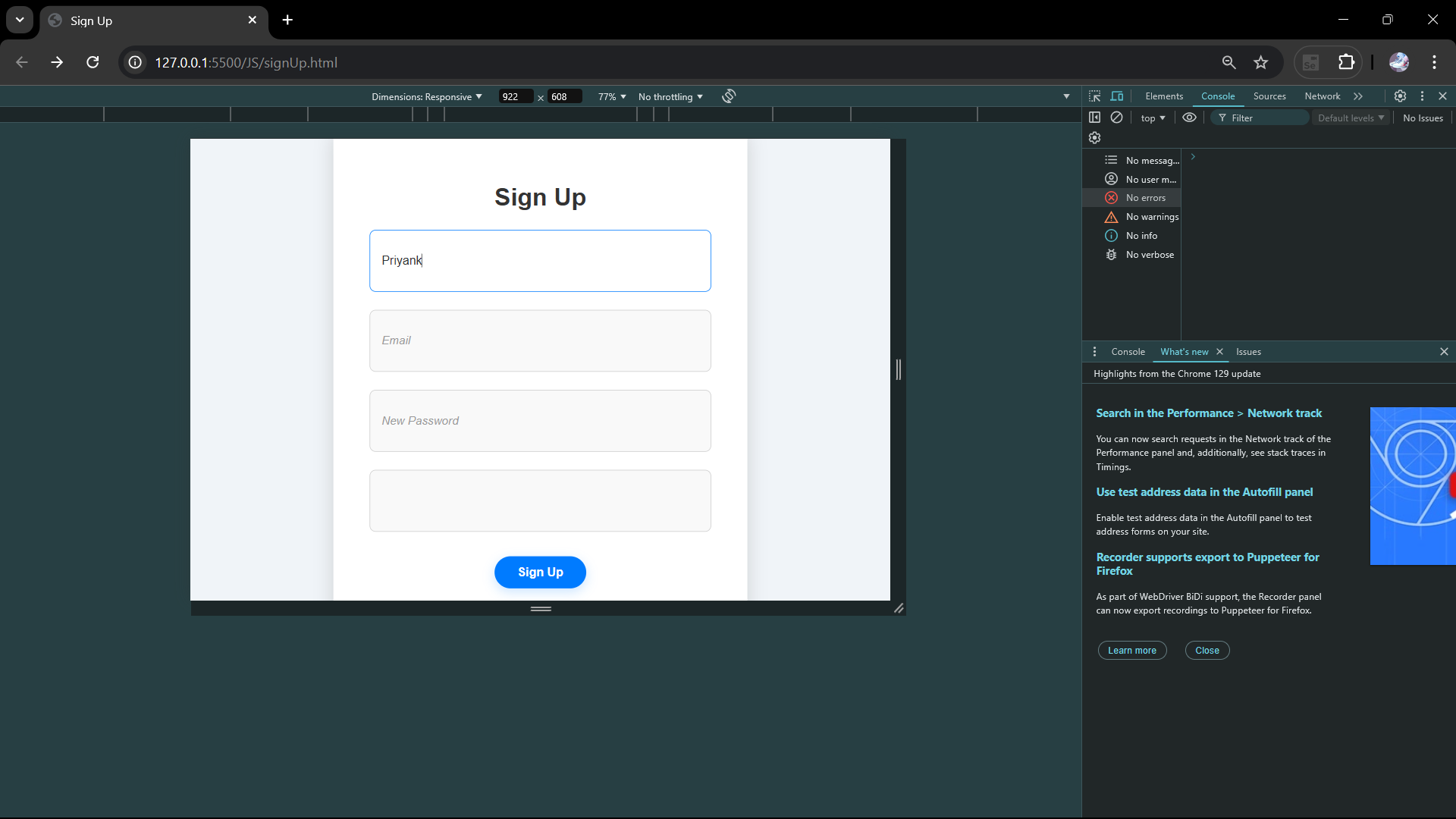
<input type="button" value="Test" onclick="FUNCTION()">



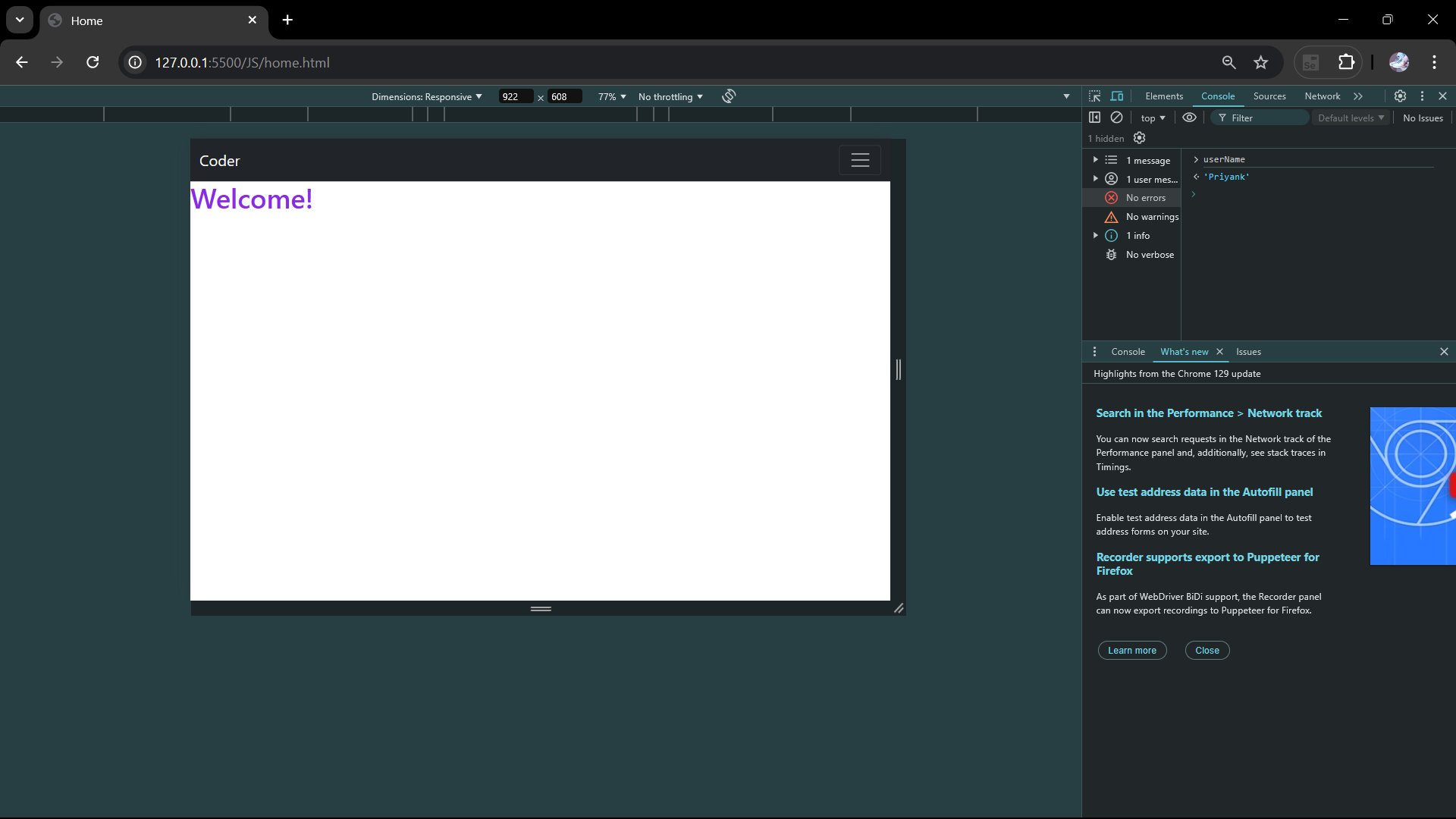
**How to get, stores and process on data from Form:**

* Data processed after click on submit button.
* By default, form data is sent as a GET request.
* use an id attribute on your input field, you can access the input field data and other values using the document method getElementByID('idName'):

Inout:



output:



Basic JS:

More operators

More validation like length etc.

Advance JS:

Normal property and object property difference?

Async and await

Can we use async w/o await visaversa?

JQuery for length check?

Filter fuction in Jquery

Callback functions

What is difference between localstorage and cookiestorage, session storage ?

Why use cookie?

When release cookies and browser?

Static properties?

Import export

Cache memory