

3_HTML_2

Agenda

- block and inline
 - div and class, id
 - span and styling
- inputs , labels
- forms and validation

Div vs span

span

- span : It is a tag that takes only the required space

```
<span>I am a span</span>  
<span>I am a span also</span>
```

I am a span I am a span also

- **use case of span**: The `` element is used for styling parts of inline content.

```
<p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Officia quod  
ullam dolores voluptatibus,  
  
<span style="color: blue; font-weight: bold;">soluta tempora</span>  
  
quos maxime quaerat aut totam consequuntur repudiandae ipsa neque sed  
molestiae fugiat placeat perspiciatis  
  
officiis!  
  
</p>
```

Div

- div individually: It is a tag that takes full width of the and allow next element to appear after it

```
<div>I am div</div>  
<div>I am also a div</div>
```



use case of div

- wrapping up text -> for better code readability

```
<div>  
  <h1>I am heading</h1>  
  <h2>I am heading</h2>  
  <!-- card level heading -->  
  <h3>I am heading</h3>  
  <h4>I am heading</h4>  
  <h5>I am heading</h5>  
  <h6>I am heading</h6>  
</div>  
  
<!-- paragraph -->  
<div>  
  <p>Paragraph :dshjbfjhdsbfjhdbfhj</p>  
  <p>Paragraph :dshjbfjhdsbfjhdbfhj</p>  
  <p>Paragraph :dshjbfjhdsbfjhdbfhj</p>  
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. In  
eos delectus molestiae corporis vitae dolore.  
    Eum  
    deleniti labore nostrum? Tenetur quos mollitia natus magni  
corporis similique exercitationem? Molestias,
```

```

        iste?
        Quidem?</p>
</div>

<!-- button -->
<div >
    <button>I am a button</button>
    <button>I am a button2</button>
    <button>I am a button3</button>
</div>

...

```

****Problem Statement**** : let's say you have use case of wrapping similar type of elements -> like group of images.

```

```html

```

```

<div>

 <!-- local file -->

</div>

<div>

</div>

```

## classes and id

in this case we have two html attributes that comes to rescue

- classes -> to give that common name or group these two
- id -> to provide the unique name to a block of html code

let's use these two wrap it efficiently

```
<div id="first-set" class="images">

 <!-- local file -->

</div>
<div id="second-set" class="images">

</div>
```

## Practical use case of id

We have these kinds of navigation bar where once a user clicks on a particular link . they are navigated

to a particular section. let's see how this can be build easily using ids and links

# JavaScript basics

[Previous](#)[Overview: Getting started with the web](#)[Next](#)

JavaScript is a programming language that adds interactivity to your website. This happens in games, in the behavior of responses when buttons are pressed or with data entry on forms; with dynamic styling; with animation, etc. This article helps you get started with JavaScript and furthers your understanding of what is possible.

## What is JavaScript?

[JavaScript](#) is a powerful programming language that can add interactivity to a website. It was invented by Brendan Eich.

JavaScript is versatile and beginner-friendly. With more experience, you'll be able to create games, animated 2D and 3D graphics, comprehensive database-driven apps, and much more!

JavaScript itself is relatively compact, yet very flexible. Developers have written a variety of tools on top of the core JavaScript language, unlocking a vast amount of functionality with minimum effort. These include:

- Browser Application Programming Interfaces ([APIs](#)) built into web browsers, providing functionality such as dynamically creating HTML and setting CSS styles; collecting and manipulating a video stream from a user's webcam, or generating 3D graphics and audio samples.
- Third-party APIs that allow developers to incorporate functionality in sites from other content providers, such as [Disqus](#) or Facebook.

### In this article

[What is JavaScript?](#)[A "Hello world!" example](#)[Language basics crash course](#)[Supercharging our example website](#)[Conclusion](#)[See also](#)

## Example

Assuming we have these `sections` and a on top that we have a `navbar` usually navbar `is a list of links so we will be using ul, li and anchor to achieve it`.

```

 Section 1
 Section 2
 Section 3

<div>
 <h2>I am section-1</h2>
 <p>
 <!-- assume we have a lot of text here -->
 </p>
</div>
```

```
<div>
 <h2>I am section-2</h2>
 <p>
 <!-- assume we have a lot of text here -->
 </p>
</div>
```

```
<div>
 <h2>I am section 3</h2>
 <p>
 <!-- assume we have a lot of text here -->
 </p>
</div>
```

You can easily achieve it by providing unique ids to each section and adding the similar ids to anchors .

```

 Section 1
 Section 2
 Section 3

<div id="section-1">
 <h2>I am section-1</h2>
 <p>
 <!-- assume we have a lot of text here -->
 </p>
</div>
<div id="section-2">
 <h2>I am section-2</h2>
 <p>
 <!-- assume we have a lot of text here -->
 </p>
</div>
<div id="section-3">
 <h2>I am section 3</h2>
 <p>
 <!-- assume we have a lot of text here -->
 </p>
```

```
</div>
```

## Practical use case of classes

Usually there is common styling given to multiple elements -> to reuse it -> you can give them common class

### Example

```
<style>

 .section-firstline {
 color: blue;
 font-weight: bold;
 }
</style>

<body>
<h1>Blog Title</h1>
 <div>
 <h2>I am section-1</h2>
 <p>

 Welcome message with paragraphs and inline spans.

 </p>
 </div>
 <div >
 <h2 >I am section-2</h2>
 <p>

 Welcome message with paragraphs and inline spans.

 </p>
 </div>
 <div >
 <h2 >I am section 3</h2>
 <p>
```

```


 Welcome message with paragraphs and inline spans.

 </p>
</div>

</body>

```

## Block and inline elements

### Inline elements :

They only take the required width and allow the next inline element to be placed beside them if space is available

#### Example

Inline elements -> span , imgs, btns, anchors

### Block level elements

They only take the **full width of the parent** and next element will be placed on the next line .

let's understand it in greater depth with help of an example

```

<div class="heading" style="border:5px solid red">
 <!-- page title -->
 <h1 style="border: 5px solid green;">I am heading</h1>
</div>

```

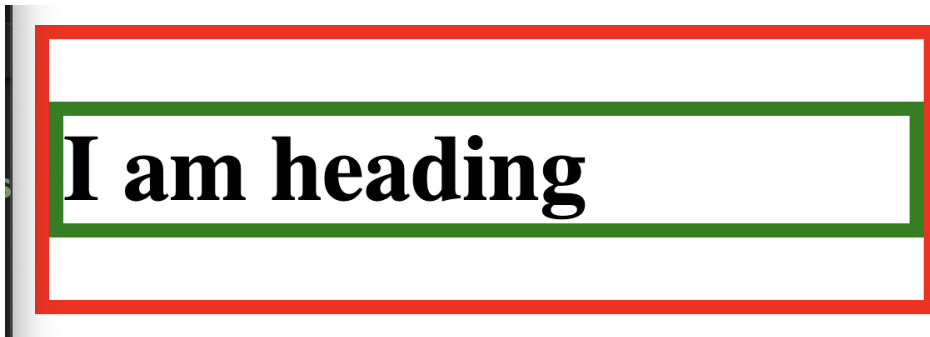
**I am heading**

You can see the outer div that wraps our h1 -> h1 has width equal to the div shown by the border

now if i will reduce the width of div h1's width will also decrease



```
<div class="heading" style=" width :300px ;border:5px solid red">
 <!-- page title -->
 <h1 style="border: 5px solid green;">I am heading</h1>
 <!-- section level heading -->
</div>
```



## HTML Form

### 1. Text Input

The `<input type="text">` element is used to create a single-line text input field.

#### Example:

```
<input type="text" id="name" placeholder="Enter your name">
```

### 2. Label

The `<label>` element is used to define a label for an `<input>` element, improving accessibility by associating the label with the input.

#### Example:

```
<label for="name">Name:</label>
<input type="text" id="name">
```

### 3. `for` Attribute

The `for` attribute in a `<label>` element binds it to an `<input>` element with a matching `id` attribute.

### Example:

```
<label for="nickname">Nick Name:</label>
<input type="text" id="nickname">
```

## 4. `id` Attribute

The `id` attribute uniquely identifies an `<input>` element on a page, enabling labels and JavaScript to reference it.

### Example:

```
<input type="text" id="nickname">
```

## 5. Placeholder

The `placeholder` attribute provides a hint to the user about what to enter in the input field.

### Example:

```
<input type="text" id="name" placeholder="Enter your name">
```

## 6. Other Input Types

HTML provides various input types to handle different types of data.

### Example:

- Number Input:

```
<label for="my-num">Number: </label>
<input type="number" id="my-num" min="10" max="100" step="4"
required>
```

- Email Input:

```
<label for="email">Email: </label>
<input type="email" id="email" value="ravi@gmail.com">
```

- **Password Input:**

```
<label for="password">Password: </label>
<input type="password" id="password" minlength="8" maxlength="14">
```

## 7. Use Case: Form for Email Validation

A form with an email input ensures the user provides a valid email address.

### Example:

```
<form action="/submit">
 <label for="email">Email: </label>
 <input type="email" id="email" required>
 <input type="submit">
</form>
```

## 8. Validation Attributes

- **minlength**: Sets the minimum number of characters for text inputs.
- **maxlength**: Sets the maximum number of characters for text inputs.
- **min**: Sets the minimum value for numeric inputs.
- **max**: Sets the maximum value for numeric inputs.
- **step**: Specifies the increment for numeric inputs.

### Example:

```
<input type="password" id="password" minlength="8" maxlength="14">
<input type="number" id="my-num" min="10" max="100" step="4">
```

## 9. Properties

- **value**: Sets the initial value of the input.
- **readonly**: Makes the input field uneditable while still submitting the value.

## Example:

```
<input type="text" id="nickname" readonly value="some value">
```

## Complete Form Example

```
<form action="/submit">
 <!-- Text Input -->
 <label for="nickname">Nick Name:</label>
 <input type="text" id="nickname" readonly value="some value">

 <label for="name">Name: </label>
 <input type="text" id="name" placeholder="Enter your name">

 <!-- Number Input -->
 <label for="my-num">Number: </label>
 <input type="number" id="my-num" min="10" max="100" step="4"
required>

 <!-- Email Input -->
 <label for="email">Email: </label>
 <input type="email" id="email" value="ravi@gmail.com" required>

 <!-- Password Input -->
 <label for="password">Password: </label>
 <input type="password" id="password" minlength="8" maxlength="14"
required>

 <input type="submit">
</form>
```

## Additional Input Types

### Date Input:

```
<label for="date">Date:</label>
<input type="date" id="date">
```

- **Purpose:** Allows users to select a date from a date picker.

### Time Input:

```
<label for="time">Time:</label>
<input type="time" id="time">
```

- **Purpose:** Allows users to select a time.

### Color Input:

```
<label for="color">Color:</label>
<input type="color" id="color">
```

- **Purpose:** Opens a color picker to select a color.

### File Input:

```
<label for="file">File:</label>
<input type="file" id="file" name="file">
```

- **Purpose:** Allows users to select a file from their device.

### Example HTML for Additional Inputs:

```
<form action="/submit">
 <!-- Date Input -->
 <label for="date">Date:</label>
 <input type="date" id="date">

 <!-- Time Input -->
 <label for="time">Time:</label>
 <input type="time" id="time">


```

```
<!-- Color Input -->
<label for="color">Color:</label>
<input type="color" id="color">

<!-- File Input -->
<label for="file">File:</label>
<input type="file" id="file" name="file">

<!-- Search vs Text Input -->
<label for="search">Search:</label>
<input type="search" id="search">

<label for="name">Name:</label>
<input type="text" id="name">

</form>
```

## Comparison: Input Type `text` vs `search`

### Text Input:

```
<label for="name">Name:</label>
<input type="text" id="name">
```

- **Purpose:** General text input for short data such as names or addresses.
- **Features:** Basic text input.
- **Usage:** Suitable for most single-line text entry needs.

### Search Input:

```
<label for="search">Search:</label>
<input type="search" id="search">
```

- **Purpose:** Specifically designed for search queries.
- **Features:** May include additional browser features like a clear button to reset the input, though these features depend on the browser.

- **Usage:** Ideal for search fields where users enter search terms.