1. Semantic Elements

Elements with a Meaning. A semantic element clearly describes its meaning to both the browser and the developer. Clearly defines its content.

Example:

<article>,<aside>,<details>,<figcaption>,<figure>,<header>,<footer>,<main>,<mark>,<nav>,<section>,<summary>,<time>.

2. **Iframe Tag**

HTML Iframe is used to display a nested webpage (a webpage within a webpage). The HTML <iframe> tag defines an inline frame, hence it is also called as an Inline frame.An HTML iframe embeds another document within the current HTML document in the rectangular region.

The webpage content and iframe contents can interact with each other using JavaScript.

Syntax:

<iframe> Content </iframe>

3. Z-index

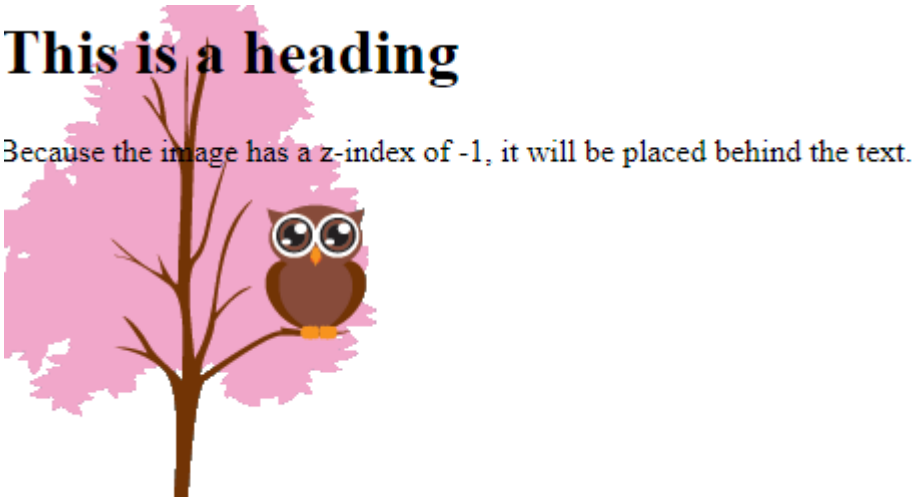
When elements are positioned, they can overlap other elements.The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).An element can have a positive or negative stack order.

Example:

```
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<img src="img_tree.png">
<p>Because the image has a z-index of -1, it will be placed behind the text.
</p>
```

Output:



4. Canvas Tag

The <canvas> tag is used to draw graphics, on the fly, via scripting (usually JavaScript.The <canvas> tag is transparent, and is only a container for graphics, you must use a script to actually draw the graphics.Any text inside the <canvas> element will be displayed in browsers with JavaScript disabled and in browsers that do not support <canvas>

Example:

```
<h1>The canvas element</h1>

<canvas id="myCanvas">
Your browser does not support the canvas tag.
</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "red";
ctx.fillRect(20, 20, 75, 50);

//Turn transparency on
ctx.globalAlpha = 0.3;
ctx.fillStyle = "blue";
ctx.fillRect(50, 50, 75, 50);
ctx.fillStyle = "orange";
ctx.fillRect(80, 80, 75, 50);
</script>
```

5. Hoisting

In JavaScript, a variable can be declared after it has been used.In other words, a variable can be used before it has been declared.

```
x = 5; // Assign 5 to x

elem = document.getElementById("demo"); // Find an element
elem.innerHTML = x;                     // Display x in the element

var x; // Declare x
```

6. var, let, const difference

var	let	const
The scope of a *var* variable is functional scope.	The scope of a *let* variable is block scope.	The scope of a *const* variable is block scope.
It can be updated and re-declared into the scope.	It can be updated but cannot be re-declared into the scope.	It cannot be updated or re-declared into the scope.
It can be declared without initialization.	It can be declared without initialization.	It cannot be declared without initialization.
It can be accessed without initialization as its default value is "undefined".	It cannot be accessed without initialization, as it returns an error.	It cannot be accessed without initialization, as it cannot be declared without initialization.

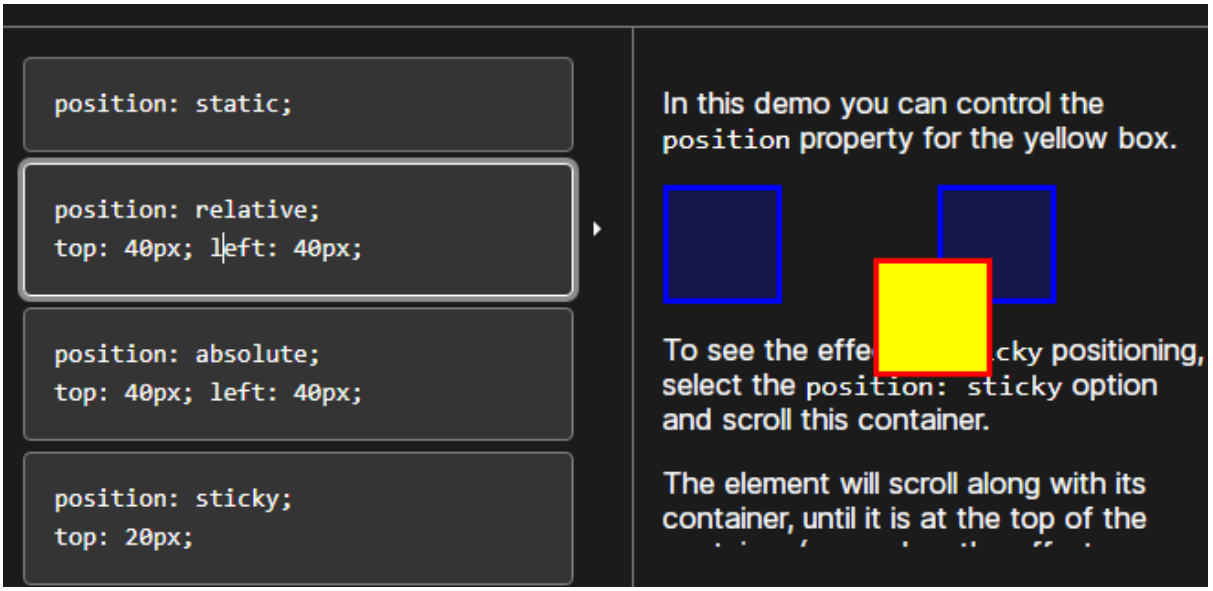
7. Position

The Position Css property sets how an element is positioned in a document. The top, right, bottom and left properties determine the final location of positioned elements.

Syntax:

Static, relative, absolute, fixed, sticky

Example:



Web Hosting

Hosting (also known as Web site hosting, Web hosting, and Webhosting) is **the business of housing, serving, and maintaining files for one or more Web sites**. More important than the computer space that is provided for Web site files is the fast connection to the Internet.