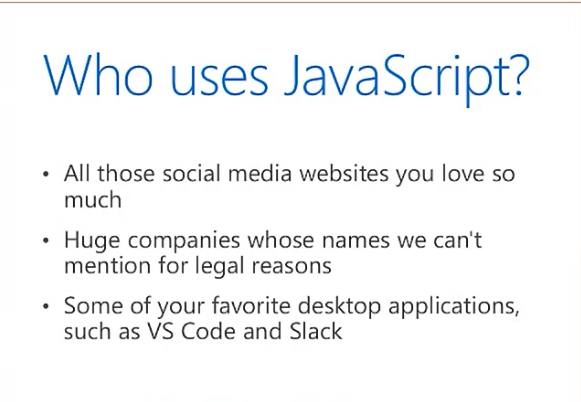
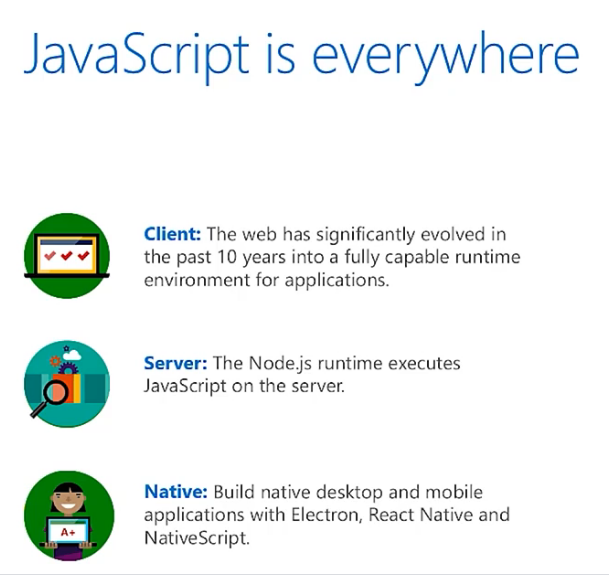
Introduction to JavaScript

(Penn, MS, Harvard, W3Cschools.com)

{Beginners Series to JavaScript - 51 slides}

Sources:

* w3schools.com/Js/js\_intro.asp
* Microsoft
* Penn
* Berkeley
* freeCodeCamp.org
* HarvardX – CS50



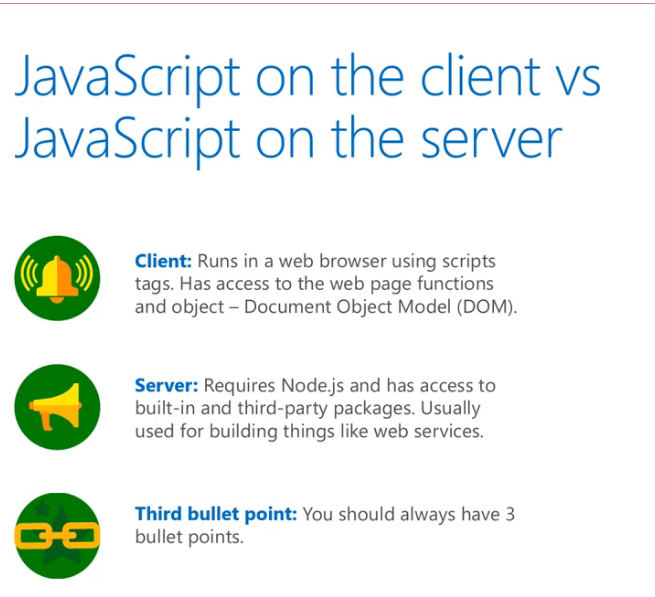




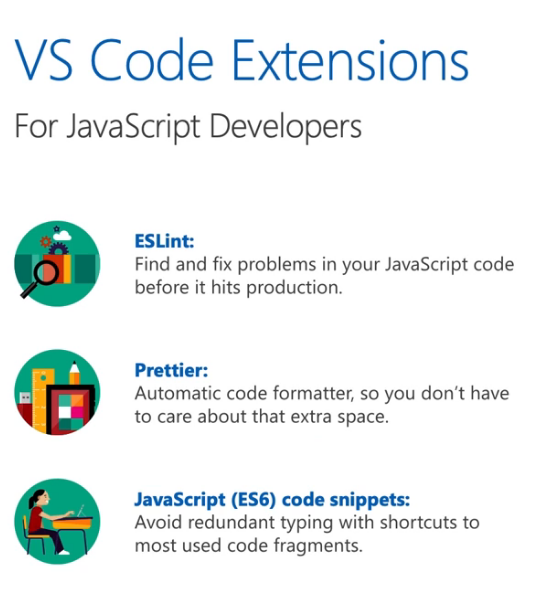
**JavaScript on a Server**

Need node JS

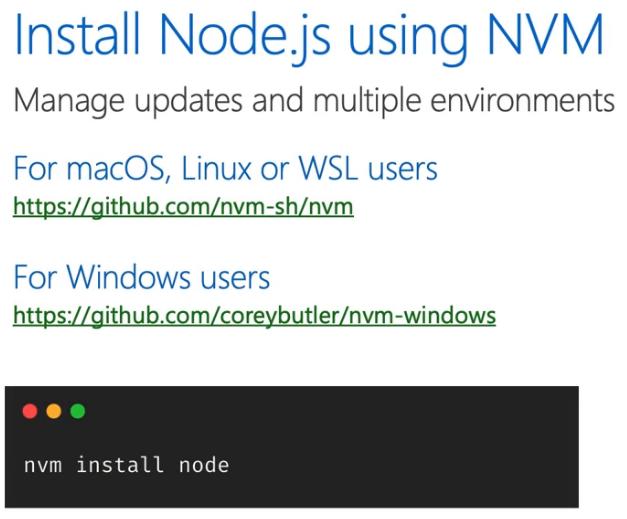












***Hello JavaScript!***

<!DOCTYPE html>

<html>

<body>

<h2>What Can JS Do?</h2>

<p id="demo">JavaScript can change HTML content.</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>Click Me!</button>

</body>

</html>

***Switch light bulb – Note: need images (on/off)***

<!DOCTYPE html>

<html>

<body>

<h2>What Can JavaScript Do?</h2>

<p>JavaScript can change HTML attribute values.</p>

<p>In this case JavaScript changes the value of the src (source) attribute of an image.</p>

<button onclick="document.getElementById('myImage').src='pic\_bulbon.gif'">Turn on the light</button>

<img id="myImage" src="pic\_bulboff.gif" style="width:100px">

<button onclick="document.getElementById('myImage').src='pic\_bulboff.gif'">Turn off the light</button>

</body>

</html>



The <script> Tag

In HTML, JavaScript code is inserted between <script> and </script> tags.

## JavaScript Functions and Events

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

**JavaScript in <head> or <body>**

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

**JavaScript in <head>**

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

### ***Example***

<!DOCTYPE html>  
<html>

<head>  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
</head>  
<body>

<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

## JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

### ***Example***

<!DOCTYPE html>  
<html>  
<body>  
  
<h1>A Web Page</h1>  
<p id="demo">A Paragraph</p>  
<button type="button" onclick="myFunction()">Try it</button>  
  
<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}  
</script>  
  
</body>  
</html>

## External JavaScript

Scripts can also be placed in external files:

**External file: myScript.js**

function myFunction() {  
  document.getElementById("demo").innerHTML = "Paragraph changed.";  
}

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

### ***Example***

<script src="myScript.js"></script>

## External References

External scripts can be referenced with a full URL or with a path relative to the current web page.

This example uses a full URL to link to a script:

***Example***

<script src="https://www.w3schools.com/js/myScript1.js"></script>

## File Path Examples:

|  |  |
| --- | --- |
| **Path** | **Description** |
| <img src="picture.jpg"> | The "picture.jpg" file is located in the same folder as the current page |
| <img src="images/picture.jpg"> | The "picture.jpg" file is located in the images folder in the current folder |
| <img src="/images/picture.jpg"> | The "picture.jpg" file is located in the images folder at the root of the current web |
| <img src="../picture.jpg"> | The "picture.jpg" file is located in the folder one level up from the current folder |
|  |  |
|  |  |
|  |  |

## HTML File Paths

A file path describes the location of a file in a web site's folder structure.

File paths are used when linking to external files, like:

* Web pages
* Images
* Style sheets
* JavaScripts

## Absolute File Paths

An absolute file path is the full URL to a file:

***Example***

<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">

## Relative File Paths

A relative file path points to a file relative to the current page.

In the following example, the file path points to a file in the images folder located at the root of the current web:

***Example***

<img src="/images/picture.jpg" alt="Mountain">

In the following example, the file path points to a file in the images folder located in the current folder:

***Example***

<img src="images/picture.jpg" alt="Mountain">

In the following example, the file path points to a file in the images folder located in the folder one level up from the current folder:

Example

<img src="../images/picture.jpg" alt="Mountain">

## Best Practice

It is best practice to use relative file paths (if possible).

When using relative file paths, your web pages will not be bound to your current base URL. All links will work on your own computer (localhost) as well as on your current public domain and your future public domains.

The HTML <head> element is a container for the following elements: <title>, <style>, <meta>, <link>, <script>, and <base>.

## The HTML <head> Element

The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag.

HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, scripts, and other meta information.

## The HTML <title> Element

The <title> element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The <title> element is required in HTML documents!

The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

The <title> element:

* defines a title in the browser toolbar
* provides a title for the page when it is added to favorites
* displays a title for the page in search engine-results

So, try to make the title as accurate and meaningful as possible!

A simple HTML document:

***Example***

<!DOCTYPE html>  
<html>  
<head>  
  <title>A Meaningful Page Title</title>  
</head>  
<body>  
  
The content of the document......  
  
</body>  
</html>

## The HTML <style> Element

The <style> element is used to define style information for a single HTML page:

***Example***

<style>  
  body {background-color: powderblue;}  
  h1 {color: red;}  
  p {color: blue;}  
</style>

## The HTML <link> Element

The <link> element defines the relationship between the current document and an external resource.  
  
The <link> tag is most often used to link to external style sheets:

***Example***

<link rel="stylesheet" href="mystyle.css">

## The HTML <meta> Element

The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.

The metadata will not be displayed on the page, but are used by browsers (how to display content or reload page), by search engines (keywords), and other web services.

## *Examples*

**Define the character set used:**

<meta charset="UTF-8">

**Define keywords for search engines:**

<meta name="keywords" content="HTML, CSS, JavaScript">

**Define a description of your web page:**

<meta name="description" content="Free Web tutorials">

**Define the author of a page:**

<meta name="author" content="John Doe">

**Refresh document every 30 seconds:**

<meta http-equiv="refresh" content="30">

**Setting the viewport to make your website look good on all devices:**

<meta name="viewport" content="width=device-width, initial-scale=1.0">

***Examples***

<meta charset="UTF-8">  
<meta name="description" content="Free Web tutorials">  
<meta name="keywords" content="HTML, CSS, JavaScript">  
<meta name="author" content="John Doe">

**Setting The Viewport**

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following <meta> element in all your web pages:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

## The HTML <script> Element

The <script> element is used to define client-side JavaScripts.

The following JavaScript writes "Hello JavaScript!" into an HTML element with id="demo":

***Example***

<script>  
function myFunction() {  
  document.getElementById("demo").innerHTML = "Hello JavaScript!";  
}  
</script>

## The HTML <base> Element

The <base> element specifies the base URL and/or target for all relative URLs in a page.

The <base> tag must have either an href or a target attribute present, or both.

There can only be one single <base> element in a document!

***Example***

Specify a default URL and a default target for all links on a page:

<head>  
<base href="https://www.w3schools.com/" target="\_blank">  
</head>  
  
<body>  
<img src="images/stickman.gif" width="24" height="39" alt="Stickman">  
<a href="tags/tag\_base.asp">HTML base Tag</a>  
</body>

## Chapter Summary

* The <head> element is a container for metadata (data about data)
* The <head> element is placed between the <html> tag and the <body> tag
* The <title> element is required and it defines the title of the document
* The <style> element is used to define style information for a single document
* The <link> tag is most often used to link to external style sheets
* The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings
* The <script> element is used to define client-side JavaScripts
* The <base> element specifies the base URL and/or target for all relative URLs in a page

## HTML head Elements

|  |  |
| --- | --- |
| **Tag** | **Description** |
| [<head>](https://www.w3schools.com/tags/tag_head.asp) | Defines information about the document |
| [<title>](https://www.w3schools.com/tags/tag_title.asp) | Defines the title of a document |
| [<base>](https://www.w3schools.com/tags/tag_base.asp) | Defines a default address or a default target for all links on a page |
| [<link>](https://www.w3schools.com/tags/tag_link.asp) | Defines the relationship between a document and an external resource |
| [<meta>](https://www.w3schools.com/tags/tag_meta.asp) | Defines metadata about an HTML document |
| [<script>](https://www.w3schools.com/tags/tag_script.asp) | Defines a client-side script |
| [<style>](https://www.w3schools.com/tags/tag_style.asp) | Defines style information for a document |

# **HTML Layout Elements and Techniques**

## HTML Layout Elements

HTML has several semantic elements that define the different parts of a web page:

|  |  |
| --- | --- |
| HTML5 Semantic Elements | * <header> - Defines a header for a document or a section * <nav> - Defines a set of navigation links * <section> - Defines a section in a document * <article> - Defines an independent, self-contained content * <aside> - Defines content aside from the content (like a sidebar) * <footer> - Defines a footer for a document or a section * <details> - Defines additional details that the user can open and close on demand * <summary> - Defines a heading for the <details> element   You can read more about semantic elements in our [HTML Semantics](https://www.w3schools.com/html/html5_semantic_elements.asp) chapter. |

## HTML Layout Techniques

There are four different techniques to create multicolumn layouts. Each technique has its pros and cons:

* CSS framework
* CSS float property
* CSS flexbox
* CSS grid

**CSS Frameworks**

If you want to create your layout fast, you can use a CSS framework, like [W3.CSS](https://www.w3schools.com/w3css/default.asp) or [Bootstrap](https://www.w3schools.com/bootstrap/default.asp).

**CSS Float Layout**

It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work. **Disadvantages:** Floating elements are tied to the document flow, which may harm the flexibility. Learn more about float in our [CSS Float and Clear](https://www.w3schools.com/css/css_float.asp) chapter.

## CSS Flexbox Layout

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

## CSS Grid Layout

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

# **HTML Responsive Web Design**

Responsive web design is about creating web pages that look good on all devices!

A responsive web design will automatically adjust for different screen sizes and viewports.

## What is Responsive Web Design?

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

## Setting The Viewport

To create a responsive website, add the following <meta> tag to all your web pages:

***Example***

<meta name="viewport" content="width=device-width, initial-scale=1.0">

This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.

## Responsive Images

Responsive images are images that scale nicely to fit any browser size.

### **Using the width Property**

If the CSS width property is set to 100%, the image will be responsive and scale up and down:

***Example***

<img src="img\_girl.jpg" **style="width:100%;"**>

Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

### **Using the max-width Property**

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

### ***Example***

<img src="img\_girl.jpg" style="**max-width:100%;**height:auto;">

### **Show Different Images Depending on Browser Width**

The HTML <picture> element allows you to define different images for different browser window sizes.

Resize the browser window to see how the image below change depending on the width:

### ***Example***

<picture>  
  <source srcset="img\_smallflower.jpg" media="(max-width: 600px)">  
  <source srcset="img\_flowers.jpg" media="(max-width: 1500px)">  
  <source srcset="flowers.jpg">  
  <img src="img\_smallflower.jpg" alt="Flowers">  
</picture>

## Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width".

That way the text size will follow the size of the browser window:

***Example***

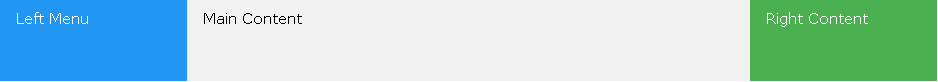
<h1 style="**font-size:10vw**">Hello World</h1>

## Media Queries

In addition to resize text and images, it is also common to use media queries in responsive web pages.

With media queries you can define completely different styles for different browser sizes.

Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stacked vertically on small screens:

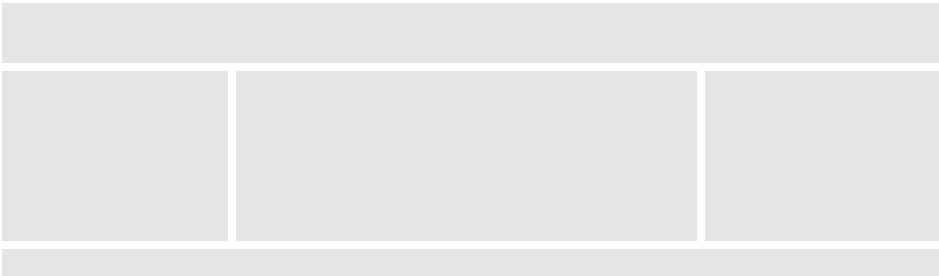


***Example***

<style>  
.left, .right {  
  float: left;  
  width: 20%; /\* The width is 20%, by default \*/  
}  
  
.main {  
  float: left;  
  width: 60%; /\* The width is 60%, by default \*/  
}  
  
/\* Use a media query to add a breakpoint at 800px: \*/  
@media screen and (max-width: 800px) {  
  .left, .main, .right {  
    width: 100%; /\* The width is 100%, when the viewport is 800px or smaller \*/  
  }  
}  
</style>

## Responsive Web Page - Full Example

A responsive web page should look good on large desktop screens and on small mobile phones.



## Responsive Web Design - Frameworks

All popular CSS Frameworks offer responsive design.

They are free, and easy to use.

## W3.CSS

W3.CSS is a modern CSS framework with support for desktop, tablet, and mobile design by default.

W3.CSS is smaller and faster than similar CSS frameworks.

W3.CSS is designed to be a high-quality alternative to Bootstrap.

W3.CSS is designed to be independent of jQuery or any other JavaScript library.



### ***Example***

<!DOCTYPE html>  
<html>  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">  
<body>  
  
<div class="w3-container w3-green">  
  <h1>W3Schools Demo</h1>  
  <p>Resize this responsive page!</p>  
</div>  
  
<div class="w3-row-padding">  
  <div class="w3-third">  
    <h2>London</h2>  
    <p>London is the capital city of England.</p>  
    <p>It is the most populous city in the United Kingdom,  
    with a metropolitan area of over 13 million inhabitants.</p>  
  </div>  
  
  <div class="w3-third">  
    <h2>Paris</h2>  
    <p>Paris is the capital of France.</p>  
    <p>The Paris area is one of the largest population centers in Europe,  
    with more than 12 million inhabitants.</p>  
  </div>  
  
  <div class="w3-third">  
    <h2>Tokyo</h2>  
    <p>Tokyo is the capital of Japan.</p>  
    <p>It is the center of the Greater Tokyo Area,  
    and the most populous metropolitan area in the world.</p>  
  </div>  
</div>  
  
</body>  
</html>

## Bootstrap

Another popular CSS framework is Bootstrap. Bootstrap uses HTML, CSS and jQuery to make responsive web pages.

***Example***

<!DOCTYPE html>  
<html lang="en">  
<head>  
<title>Bootstrap Example</title>  
<meta charset="utf-8">  
<meta name="viewport" content="width=device-width, initial-scale=1">  
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">  
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>  
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>  
</head>  
<body>  
  
<div class="container">  
  <div class="jumbotron">  
    <h1>My First Bootstrap Page</h1>  
  </div>  
  <div class="row">  
    <div class="col-sm-4">  
      ...  
    </div>  
    <div class="col-sm-4">  
      ...  
    </div>  
    <div class="col-sm-4">  
    ...  
    </div>  
  </div>  
</div>  
  
</body>  
</html>

https://www.w3schools.com/html/html\_computercode\_elements.asp

***Notes from freeCodeCamp.org***

Var x = 5; //comment goes here.

/\* or multi-line comment goes here \*/

/\* Data Types:

undefined, null, Boolean, string, symbol, number, and object

\*/

// Declaring with

var myNmae = “Brian”; /changeable.

let bugx = “bugs are beautiful”; /within scope only.

const pi = 3.1415; /never changes.

You can *declare* a variable and *assign* a variable.

var a;

var b = 2;

// See things in the console.

console.log(x)

/ Declare & initialize a variable.

var a = 9;

//Declarations

var studylyCapVar;

var properCamelCase;

studylyCapVar = 10;

propeCamelCase = “this sucks”

// Case sensitive.

var StUdLyCapVar NOT EQUAL studylyCapVar

function wordBlanks(myNoun, myAdjective, myVerb, myAdverb) {

// Your code below this line

var result = “”;

result += “The “ + myAdjective + “ “ + myNoun + “ “ +  
myVerb + “ to the store “ + myAdverb + “.”

// Your code above this line

return result;

}

console.log(wordBlanks(“dog”, “big”, “ran”, “quickly”));

console.log(wordBlanks(“bike”,”slow”, “flew“, “slowly));

// Arrays

var ourArray = [“John”, 23];

var myArray = [“Quincy”, 1];

// Nested Array

var ourArray = [[“the universe”, 42], [“everything”, 101010]];

var myArray = [[“Bulls”, 23], [“White Sox”, 47]];

// array

var ourArray = [50,60,70];

var ourData = ourArray[0]; // = 50

var myList = [[“cereal”, 3], [“milk”, 2], [“bananas”, 7], [“juice”, 2], [“eggs”, 12];

function ourTrueOrFalse(isItTrue) {

if(isItTrue) {

return “Yes, it’s true”;

}

Function trueOrFalse(wasThatTrue) {

}

// Example

function ourReusableFunction() {

Console.log(“Hayya, World”);

}



Why JavaScript?

JavaScript is an essential programming language, almost compulsory to learn for students or software developers that are gravitated towards web development. Wondering why? Here’s the answer:

* Javascript is the most popular programming language in the world and that makes it a default choice for web development. There are many frameworks available which you can use to create web applications once you have learned JavaScript.
* JavaScript offers lots of flexibility. You can create stunning and fast web applications with tons of customizations to provide users with the most relevant graphical user interface.
* JavaScript is now also used in mobile app development, desktop app development, and game development. This opens many possibilities for you as a Javascript developer.
* Due to the high demand in the industry, there are tons of job growth opportunities and high pay for those who know JavaScript.
* The incredible thing about Javascript is that you can find tons of frameworks and libraries already developed, which can be used directly in web development. That reduces the development time and enhances the graphical user interface.

What is JavaScript Used For?

JavaScript is used in various fields from the web to servers, and here’s a quick list of the significant areas it’s used in:

* Web Applications: JavaScript is used for adding interactivity and automation to websites. So, if you want your web application to be anything more than just a static page of contents, you’ll probably need to do some “JavaScript’ing.”
* Mobile Applications: JavaScript isn’t just for developing web applications; it is also used for developing applications for phones and tablets. With frameworks like React Native, you can develop full-fledged mobile applications with all those fancy animations.
* Web-based Games: If you’ve ever played a game directly on the web browser, JavaScript was probably used to make that happen.
* Back-end Web Development: JavaScript has traditionally been used for developing the front-end parts of a web application. However, with the introduction of NodeJS, a prevalent back-end JavaScript framework, things have changed. And now, JavaScript is used for developing the back-end structure also.