VOIS
Summer School

Introduction to **Frontend Development**

VOIS

Presenter

Mohammed Aqeed
Software Developer

VOIS

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

How the Web works | Concept

The Internet is a global network that offers a variety of services, such as email, file transfer, and web pages.

To uniquely identify the resources present on the Internet, the concept of a **URL (Uniform Resource Locator)** has been defined.

An important service on the Internet is the **Web service**. In simple terms, the Web service represents the totality of files, folders, and documents present on different computers on the Internet.

VOIS

How the Web works | Concept

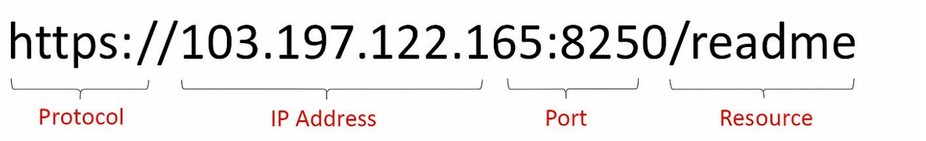
The most important protocol within the WWW (World Wide Web) is the **HTTP (Hyper Text Transfer Protocol)**. This protocol allows the transmission of data specific to the WWW service through the Internet.

Thus, websites on the Internet are accessed by browsers using the HTTP protocol. A browser is essentially an application that can connect to the Internet and, using the HTTP protocol, access a website, then display it graphically in a window on the user's computer.

VOIS

How the Web works | Concept

URLs are frequently used in web technologies because this is the main advantage of the Web – the interconnection of multiple resources across the Internet.

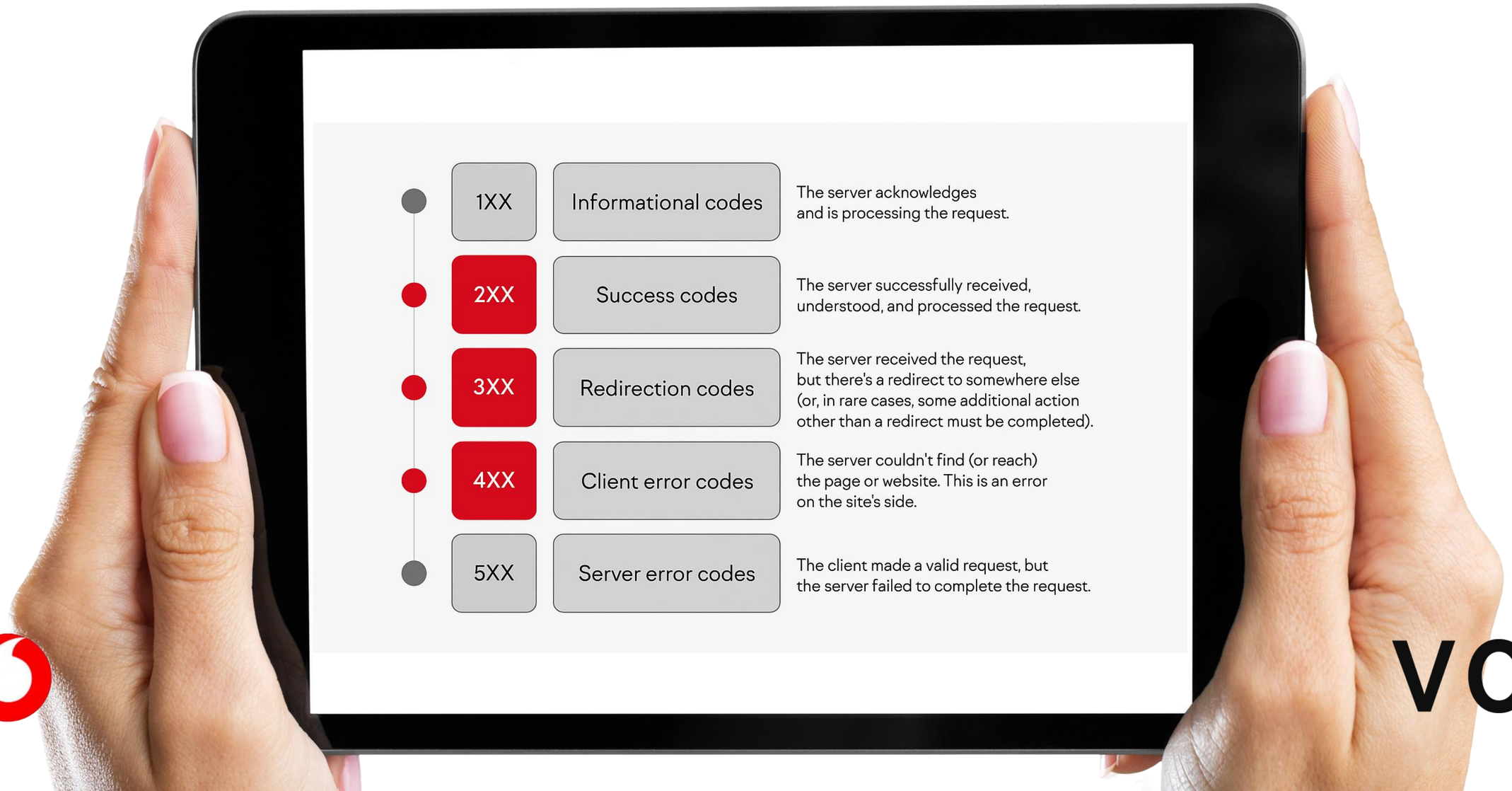


What Happens When You Visit a Website?

- The browser sends a request to the server hosting the website.
- The server responds with all necessary files
- The browser loads these files, displaying the website to the user.

VOIS

How the Web works | Informational Responses



1XX	Informational codes	The server acknowledges and is processing the request.
2XX	Success codes	The server successfully received, understood, and processed the request.
3XX	Redirection codes	The server received the request, but there's a redirect to somewhere else (or, in rare cases, some additional action other than a redirect must be completed).
4XX	Client error codes	The server couldn't find (or reach) the page or website. This is an error on the site's side.
5XX	Server error codes	The client made a valid request, but the server failed to complete the request.

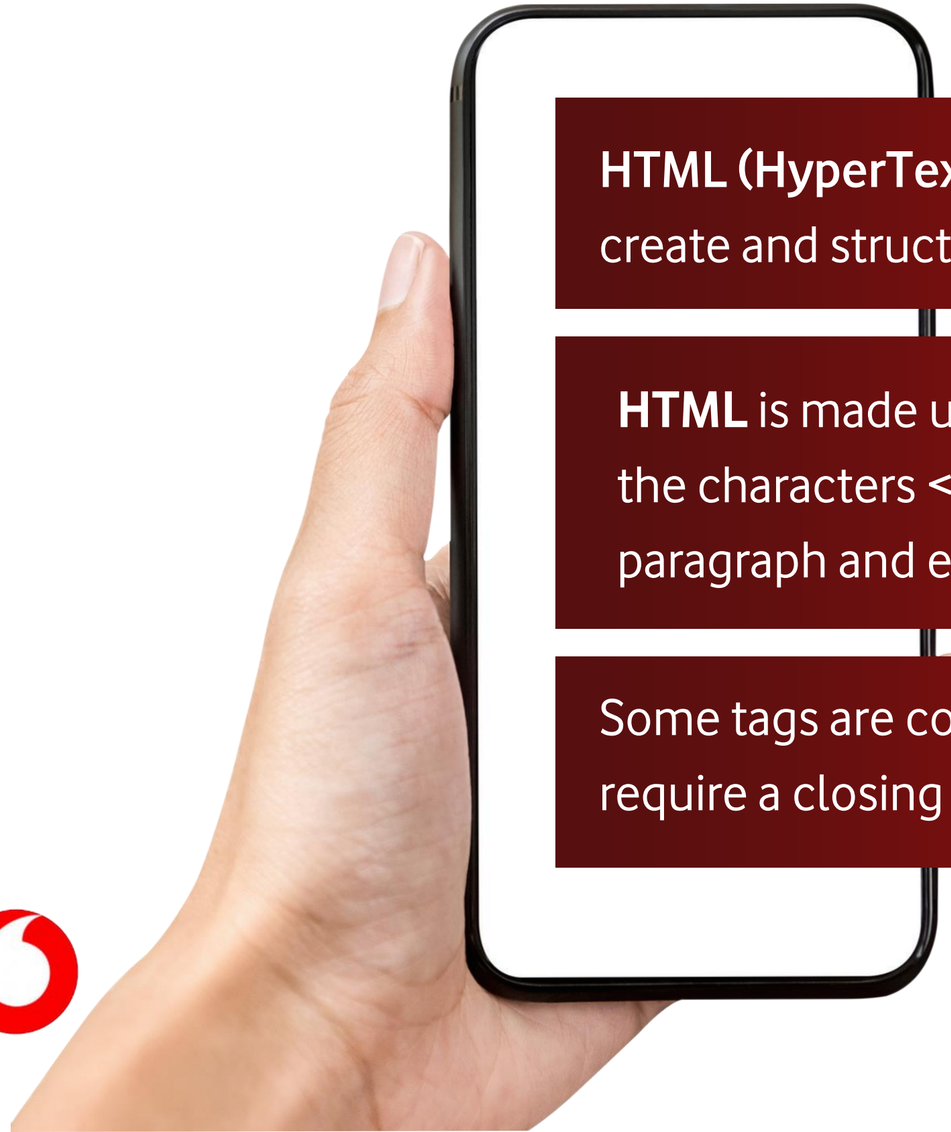
VOIS

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

HTML | HyperText Markup Language



HTML (HyperText Markup Language) is the standard language used to create and structure content on the web.

HTML is made up of elements (tags) which are keywords enclosed between the characters < and >. As a rule, tags come in pairs: and (start of paragraph and end of paragraph).

Some tags are considered empty (self-closing tags), meaning they don't require a closing tag **(
, <hr />).**

VOIS

HTML | Syntax

```html
<!DOCTYPE html>
<html>
    <head>
      <meta charset="UTF-8">
      <meta
        name="viewport"
        content="width=device-width, initial-scale=1"
      />
      <title>Titlul paginii</title>
    </head>
    <body>
        <h1>Acesta este un header</h1>
        <p>Iar aici avem un paragraf.</p>
    </body>
</html>
```

```
<html>
  <head>
    <title>Titlul paginii</title>
  </head>
  <body>
    <h1>Acesta este un header</h1>
    <p>Iar acesta este un paragraf.</p>
  </body>
</html>
```

VOIS

HTML | Common tags

– Inline Grouping

- Used to group text or inline elements without adding layout breaks.
- Ideal for applying styles to parts of a sentence.

```
<style>
  span { color: green; }
</style>
<p>Pretul este de doar <span>99.99</span> USD</p>
```

<a> – Hyperlink

- Creates links between pages or to external resources.
- Requires the href attribute for the destination.

```
<a href="https://www.example.com">Visit Example</a>
```

VOIS

HTML | Common tags

– Section Container

- Defines a generic block-level section.
- Can hold multiple elements: text, images, other s
- Commonly used with CSS to structure page layout.

```
<div>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</div>
```

<img> – Image Embedding

- Displays images in a page.
- Uses src for the image path and alt for description.

```
<img src="smiley.gif" alt="Smiley face" />
```

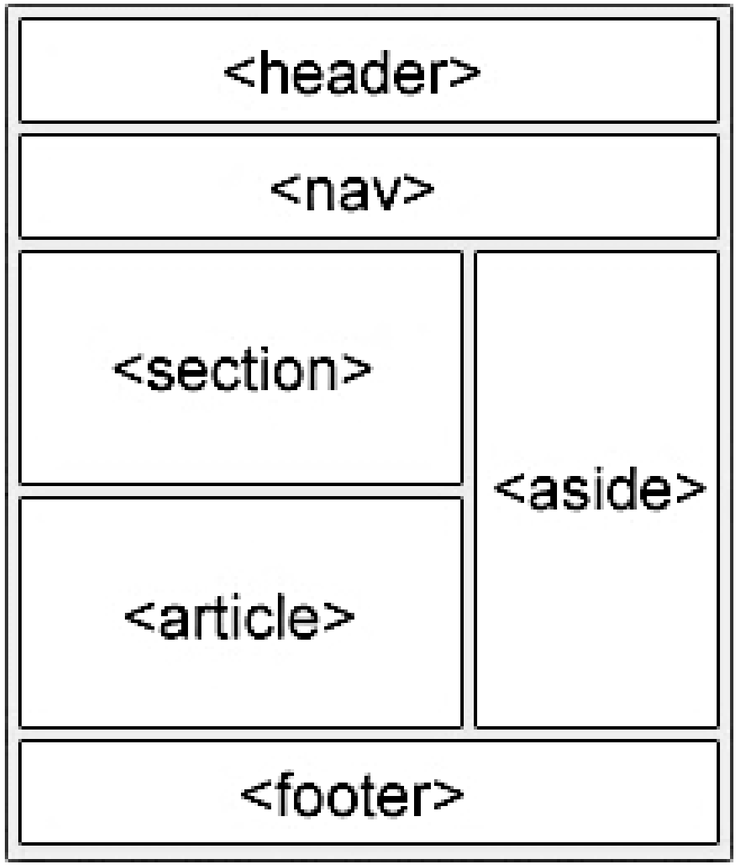
VOIS

HTML5 | Semantic HTML

Semantic HTML refers to the use of semantic elements within a webpage. More precisely, it means using elements that have a specific meaning for the exact purpose implied by their meaning.

With HTML5, a number of new elements were introduced that have a well-defined semantic role:

- **<article>**
- **<aside>**
- **<details>**
- **<footer>**
- **<header>**
- **<main>**
- **<nav>**
- **<section>**



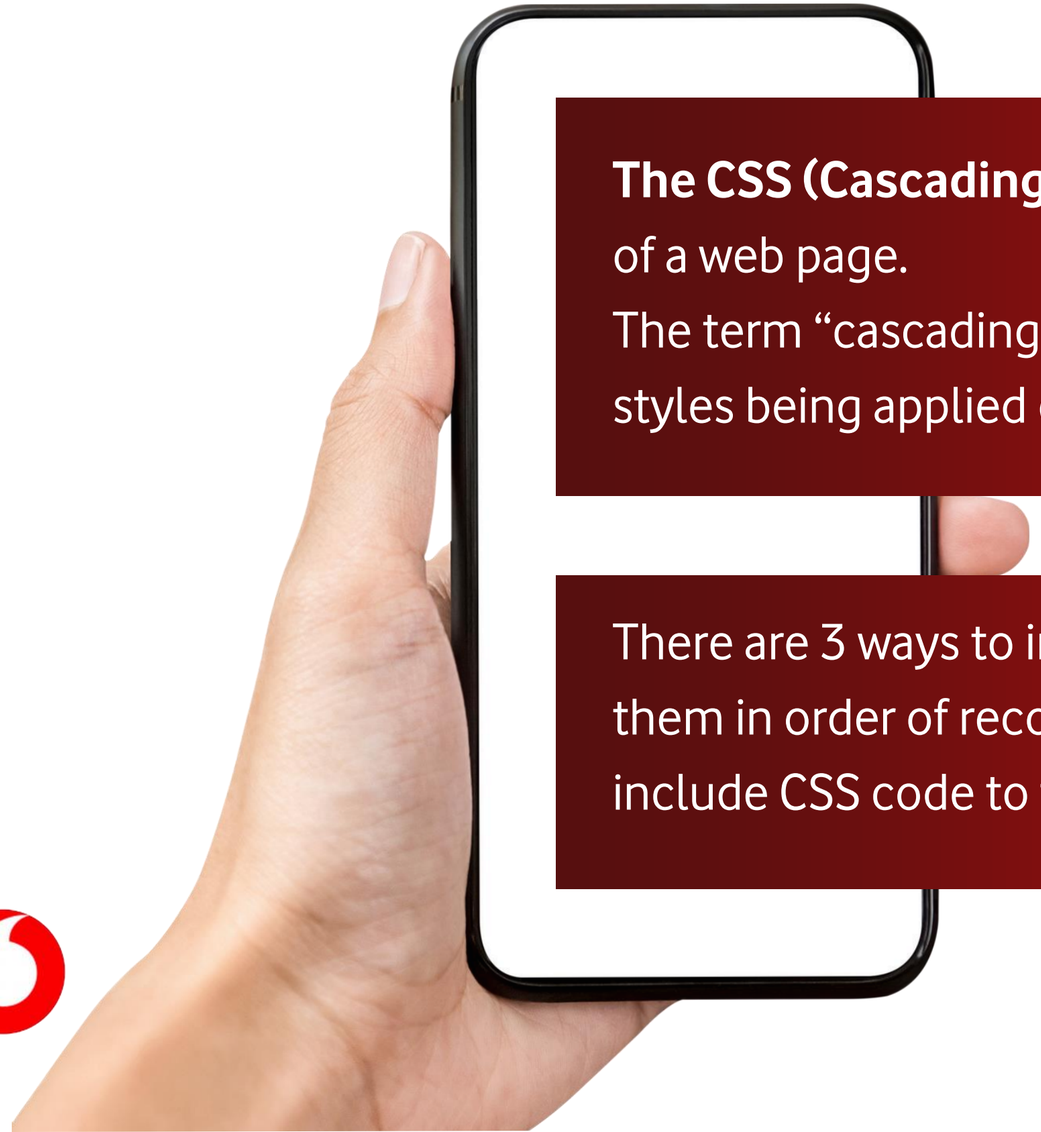
VOIS

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

CSS | Cascading Style Sheets



The CSS (Cascading Style Sheets) language allows for the complete styling of a web page.
The term “cascading “implies that the styles are applied gradually, certain styles being applied over others.

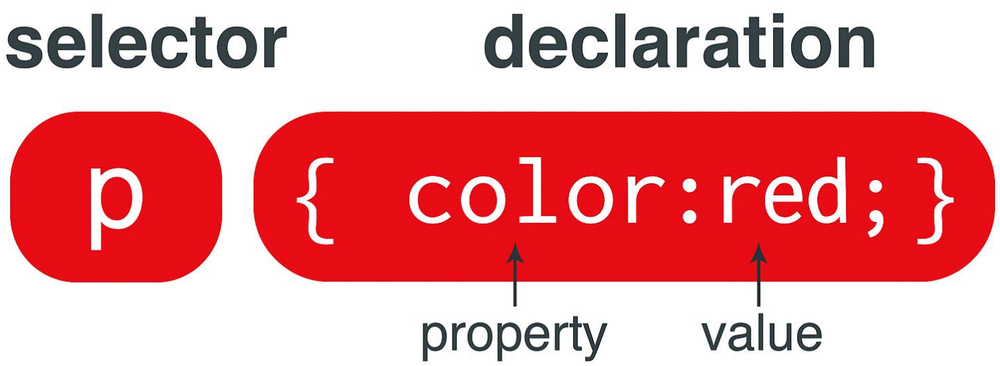
There are 3 ways to include CSS code within an HTML page. We present them in order of recommended use (from the most appropriate way to include CSS code to the least recommended way to write CSS code)

VOIS

CSS | Syntax

The CSS language consists of a series of style declarations. Each declaration has two components:

- the element(s) we are styling (the selector)
- the styles applied



Including CSS code through the <LINK>

CSS code located in a file can be included within an HTML page (or multiple pages, obviously) through the <LINK> tag.

```
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="fisierul_meu.css" />
    </head>
    <body>
    </body>
</html>
```

Including CSS code using the <STYLE> tag

Although it is recommended to always include CSS code using the <LINK> tag, HTML also allows you to include CSS code using the **<STYLE> tag.**

```
<html>
    <head>
        <style type="text/css">
        /* CSS code comes here */
        </style>
    </head>
    <body>
    </body>
</html>
```

VOIS

Including CSS code via the STYLE attribute

This method is the least recommended because it complicates the HTML code and is not an organized way to store CSS code.

```
<html>
    <head>
    </head>
    <body>
        <div style="border: 1px solid black;">
            <h1>Titlul div-ului</h1>
            <p>Aici este un paragraf in interiorul div-ului.</p>
        </div>
    </body>
</html>
```

Inheriting CSS properties

The C in CSS stands for Cascading. We learned that there are 3 ways to introduce CSS code.

In addition to those, there are two other sources of style:

- the browser's default style (you will notice that the same page looks different in different browsers)
- the style applied by the user accessing the page (some users can customize the way their pages are displayed).

CSS | Selectors

The class type selector - .classname

```
<style>
.red {
    color: red;
}
</style>
<p class="red"> This text is written in red. </p>
```

The asterisk selector – *

```
* {
  box-sizing: border-box;
  margin: 0;
  padding: 0;
}
```

The id type selector - #idelement

```
<style>
#header {
    background: yellow;
}
</style>
<div id="header">
  This header will be displayed with a yellow background.
</div>
```

The element selector - tag

```
<style>
p {
    font-style: italic;
}
</style>
<p>This text will be displayed in italic format</p>
```

VOIS

CSS | Box Model

The concept of the box model is fundamental in CSS because it relates to how CSS works and how HTML elements are managed.

In CSS, all HTML elements are considered rectangular boxes. Essentially, we can think of each HTML element as being enclosed in such a box, which has the following properties:

- **Margins**
- **Borders**
- **Padding**
- **Actual content**

VOIS

CSS | Flexbox Layout

A CSS layout model is designed for distributing space and aligning items in a single axis (horizontal or vertical).

Why use it?

- Simplifies responsive layouts
- Handles alignment, spacing, and order easily
- Replaces float- or inline-block-based hacks

```
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

Core Concepts:

- Container properties: display: flex;, flex-direction, justify-content, align-items, flex-wrap
- Item properties: flex, align-self, order

VOIS

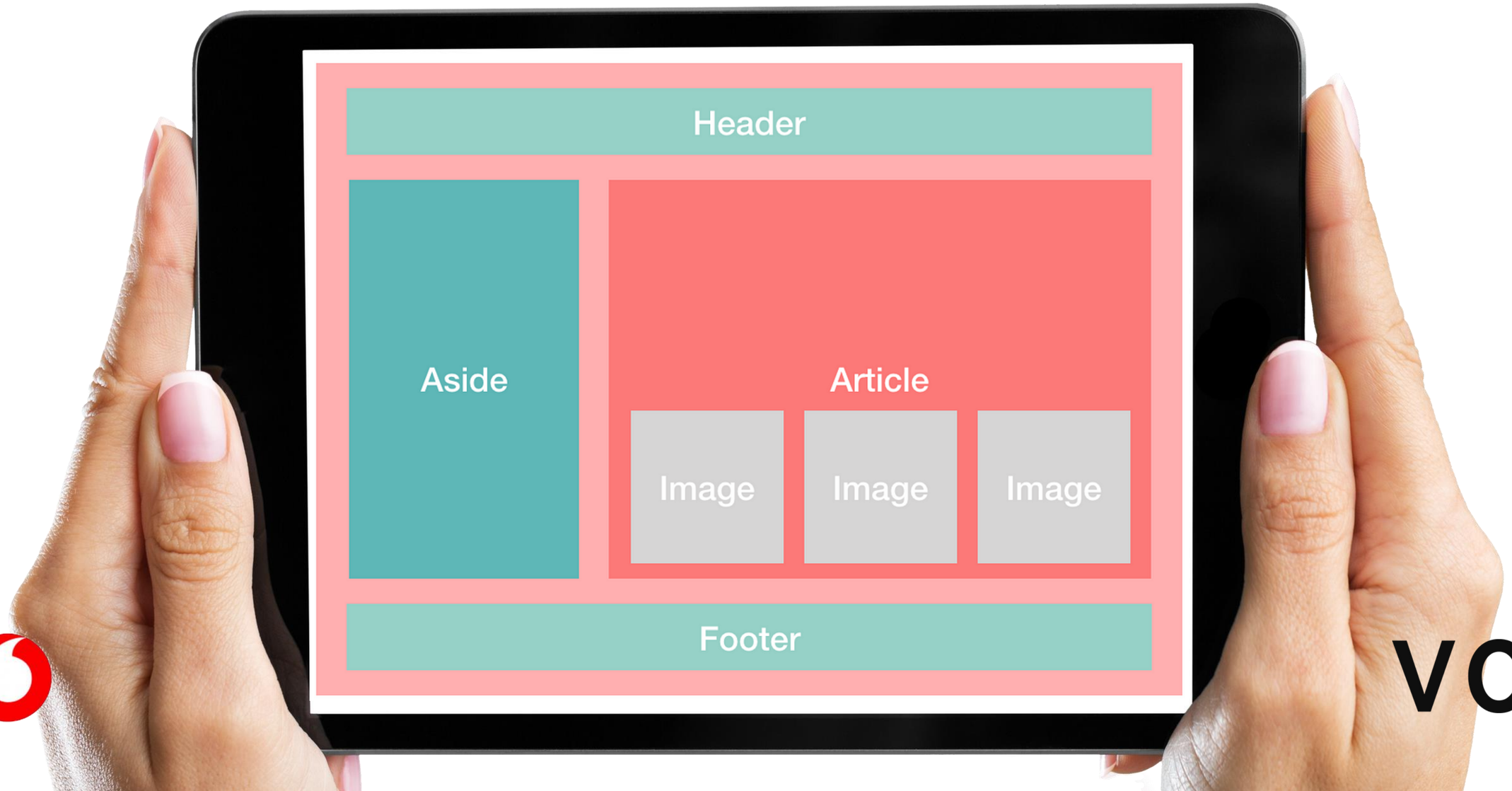
Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

Practice

Practice | Layout



VOIS

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

JavaScript

JavaScript is a programming language used to make web pages interactive. It runs in the browser and can change content, respond to user actions, and communicate with servers.

JavaScript helps us manipulate elements in the **DOM (Document Object Model)** and assign actions to them.

The **DOM** is like an internal database where the browser stores the structure of the page's tags along with each tag's individual attributes.

VOIS

JavaScript | Variables

let

- Used to declare block-scoped variables
- Can be reassigned

```
let score = 10;
score = 20;
```

const

- Also block-scoped
- Cannot be reassigned

```
const PI = 3.14;
// PI = 3.1415 ❌ Error
```

VOIS

JavaScript | Primitive Data Types

Type	Typeof return value	Example
Null	“object”	let something = null;
Undefined	“undefined”	let car; (value and type of car is undefined)
Boolean	“boolean”	true, false
Number	“number”	1, 1.5
BigInt	“bigint”	let x = BigInt("12345678901234567890123456789 0");
String	“string”	“this is a string”
Symbol	“symbol”	const value1 = Symbol('hello');

VOIS

JavaScript | Reference Data Types

Type	Key Characteristics	Example
Object	Key/value data structure	const person = {firstName:"John", lastName:"Doe"};
Array	Ordered, numerically indexed object	const cars = ["Toyota", "Logan", "Renault"];
Function	Callable object, can be stored in a variable	function oldFunction(p1, p2) { return p1 * p2; } const newFunction = (p1, p2) => p1 * p2;
Date	Object for calendar/date manipulation	const d = new Date("2024-06-17");
RegExp	Object for regular expressions	/abc/ or new RegExp("abc")
Map / Set	Advanced structures (unique keys/values, no duplicates)	new Map(), new Set()

VOIS

JavaScript | Operators

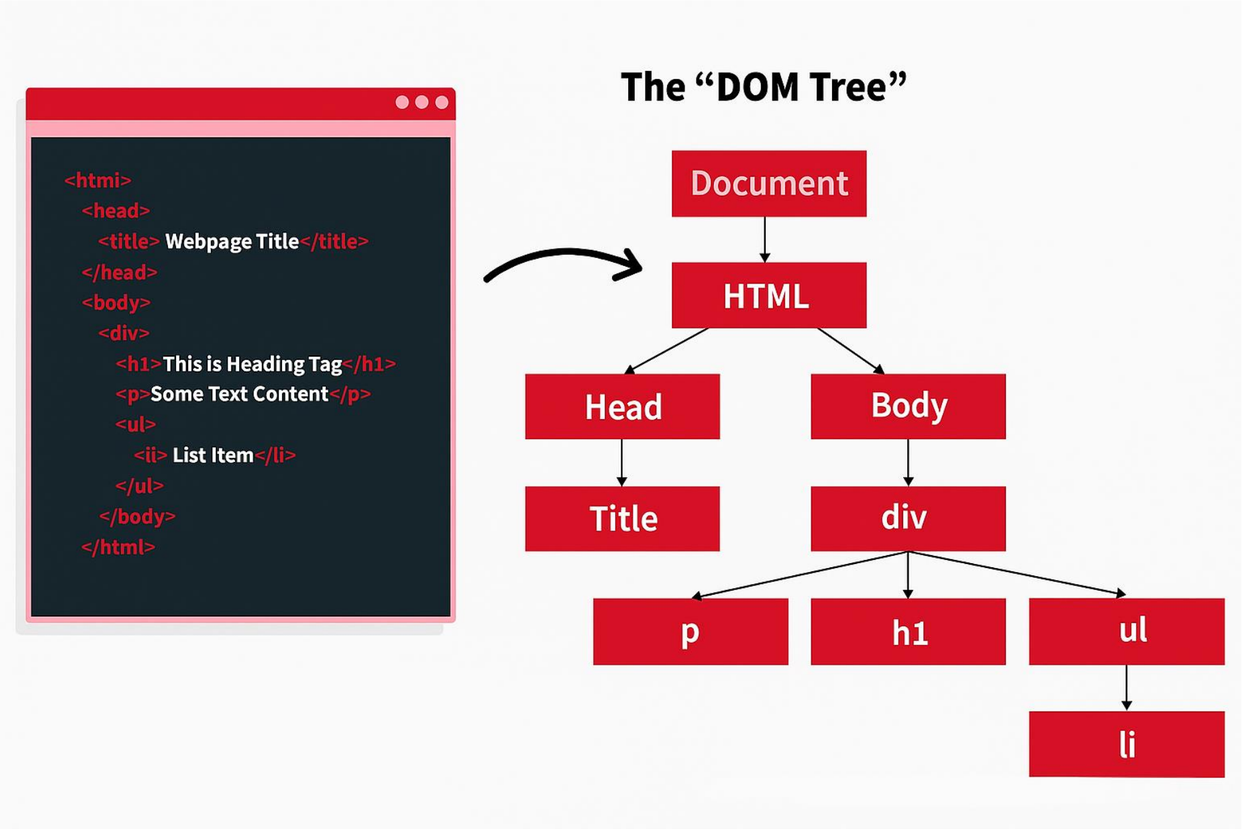
Category	Operator(s)	Example
Assignment	=, +=, -=, *=, /=, %=	Assign or update values (x += 2 → x = x + 2)
Arithmetic	+, -, *, /, %, **	Basic math (2 + 3, 4 ** 2 → 16)
Comparison	==, !=, ===, !==, >, <, >=, <=	Compare values (=== checks value & type)
Logical And	&&	isLoggedIn && showDashboard() Executes second expression only if the first is true
String	+	Concatenation ("Hello " + "world")
Nullish Coalescing	??	Default if null or undefined (a ?? "default")

VOIS

JavaScript | DOM

The DOM (Document Object Model) is a programming interface that represents a web page as a tree-like structure in memory.

When a browser loads a page, it builds a DOM tree of all elements (like titles, paragraphs, images, links), allowing scripts to access and manipulate them.



VOIS

JavaScript | DOM Methods

getElementById():

```
let element = document.getElementById("myElementId");
```

getElementsByTagName():

```
let elements = document.getElementsByTagName("div");
```

getElementsByClassName():

```
let elements = document.getElementsByClassName("myClassName");
```

querySelector() and querySelectorAll():

```
let element = document.querySelector("#myElementId");
let elements = document.querySelectorAll(".myClassName");
```

VOIS

Important DOM Events

onClick Event

The onClick event occurs when a user clicks on an element, such as a button, link, or any clickable item. It's commonly used to trigger some JavaScript code when the element is clicked.

```
<button id="myButton">Click Me!</button>

<script>
  const button = document.getElementById('myButton');

  button.onclick = function() {
    alert('Button was clicked!');
  };
</script>
```

onChange Event

The onChange event occurs when the value of an input element changes and loses focus (for inputs like text fields, selects, checkboxes, etc.). It's often used to detect when a user changes a form input.

```
<select id="mySelect">
  <option value="red">Red</option>
  <option value="green">Green</option>
  <option value="blue">Blue</option>
</select>

<script>
  const select = document.getElementById('mySelect');

  select.onchange = function() {
    console.log('Selected color:', this.value);
  };
</script>
```

JavaScript | Control Structures - Conditional

Structure	Syntax Example	Purpose
If / else	if (x > 0) { ... } else { ... }	Run code based on conditions
else if	if (...) { } else if (...) { } else { }	Multiple branching paths
switch	switch(day) { case "Mon": ... break; }	Choose from multiple options
Ternary	isAdult ? "Yes" : "No"	Short inline condition

VOIS

JavaScript | Control Structures - Loops

Structure	Syntax Example	Purpose
for	for (let i = 0; i < 5; i++) { ... }	Repeat with counter
while	while (condition) { ... }	Repeat while condition is true
do...while	do { ... } while (condition);	Run at least once, then test
for...of	for (let item of array) { ... }	Loop through iterable values
for...in	for (let key in object) { ... }	Loop through object keys

VOIS

JavaScript | Asynchronous

Async and await in JavaScript are used to work with asynchronous operations, allowing you to write asynchronous code in a clearer and more readable way.

A Promise is an object used to represent the eventual completion or failure of an asynchronous operation, and the resulting value from that operation.

Async functions are functions that return a Promise. They allow the use of the await keyword inside them to wait for a Promise to complete.

VOIS

JavaScript | async/await Example

```javascript
// Define an async function
async function fetchUserData() {
  try {
    // Await a simulated API call (could be fetch, axios, etc.)
    const response = await fetch('https://api.example.com/user');

    // Check if response is OK
    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    // Parse the JSON data
    const data = await response.json();

    // Use the data (e.g., display in UI or log)
    console.log('User data:', data);

  } catch (error) {
    // Catch any errors (network issues, JSON parsing, etc.)
    console.error('Error fetching user data:', error.message);
  }
}

// Call the async function
fetchUserData();
```

VOIS

JavaScript | ES6

ES6, officially known as ECMAScript 2015, is a major revision to the JavaScript language specification, released in 2015.

It introduced a wide range of features that make JavaScript more powerful, modular, and developer-friendly.

✅ Improves code structure with classes, modules, and block scoping

⚡ Enables cleaner syntax through arrow functions, destructuring, and template literals

🔄 Modernizes asynchronous programming with Promises and async/await support (in later versions)

🌐 Supports modular development, making code easier to maintain and reuse in large applications

VOIS

JavaScript | ES6 – Spread Operator {...}

The spread operator in JavaScript, introduced in ES6, is represented by three dots (...) and is used to unpack elements from arrays, objects, or other iterable structures into individual elements. It allows you to create copies, merge data, or pass elements where multiple arguments or values are expected.

```
let years = [2023, 2024, 2025];
let allYears = [2021, 2022, ...years];
// [2021, 2022, 2023, 2024, 2025]
```

```
const fruits = ["apple", "orage"];
const vegetables = ["carrot", "broccoli"];
const result = ["bread", ...fruits, "chicken", ...vegetables];
console.log("Result:", result);
// ["bread", "apple", "orage", "chicken", "carrot", "broccoli"]
```

VOIS

JavaScript | ES6 – Destructuring

Destructuring allows you to extract values from arrays or properties from objects and assign them to variables using a concise syntax.

```
// ES5 Code
var myObject = { one: 'a', two: 'b', three: 'c' };
var one = myObject.one;
var two = myObject.two;
var three = myObject.three;
```

Instead of accessing each value manually, destructuring lets you "unpack" data from complex structures (like arrays, objects, or function returns) directly into distinct variables — making code more readable and expressive.

```
// ES6 Code
const myObjectES6 = { one: 'a', two: 'b', three: 'c' };
const { one, two, three } = myObjectES6;
```

VOIS

ES6 – Arrow Functions

Arrow functions are a shorter and more concise way to write function expressions in JavaScript.

They use the => (fat arrow) syntax and are especially useful for writing small, anonymous functions.

```
// ES5 code
var multiplyES5 = function(a, b) {
  return a * b;
};

// ES6 code
const multiplyES6 = (a, b) => a * b;
```

VOIS

JavaScript | ES6 – map(), reduce(), filter()

map() – Transform Values

- Returns a new array by applying a function to each element
- Does not mutate the original array

filter() – Select Elements

- Returns a new array with only elements that pass a condition
- Useful for searching, filtering data

reduce() – Accumulate Results

- Reduces the array to a single value (number, string, object, etc.)
- Ideal for summing, counting, or merging

```
const nums1 = [1, 2, 3];
const doubled = nums1.map(n => n * 2); // [2, 4, 6]


const nums2 = [1, 2, 3, 4];
const even = nums2.filter(n => n % 2 === 0); // [2, 4]


const nums3 = [1, 2, 3, 4];
const sum = nums3.reduce((acc, curr) => acc + curr, 0); // 10
```

VOIS

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

What is React?

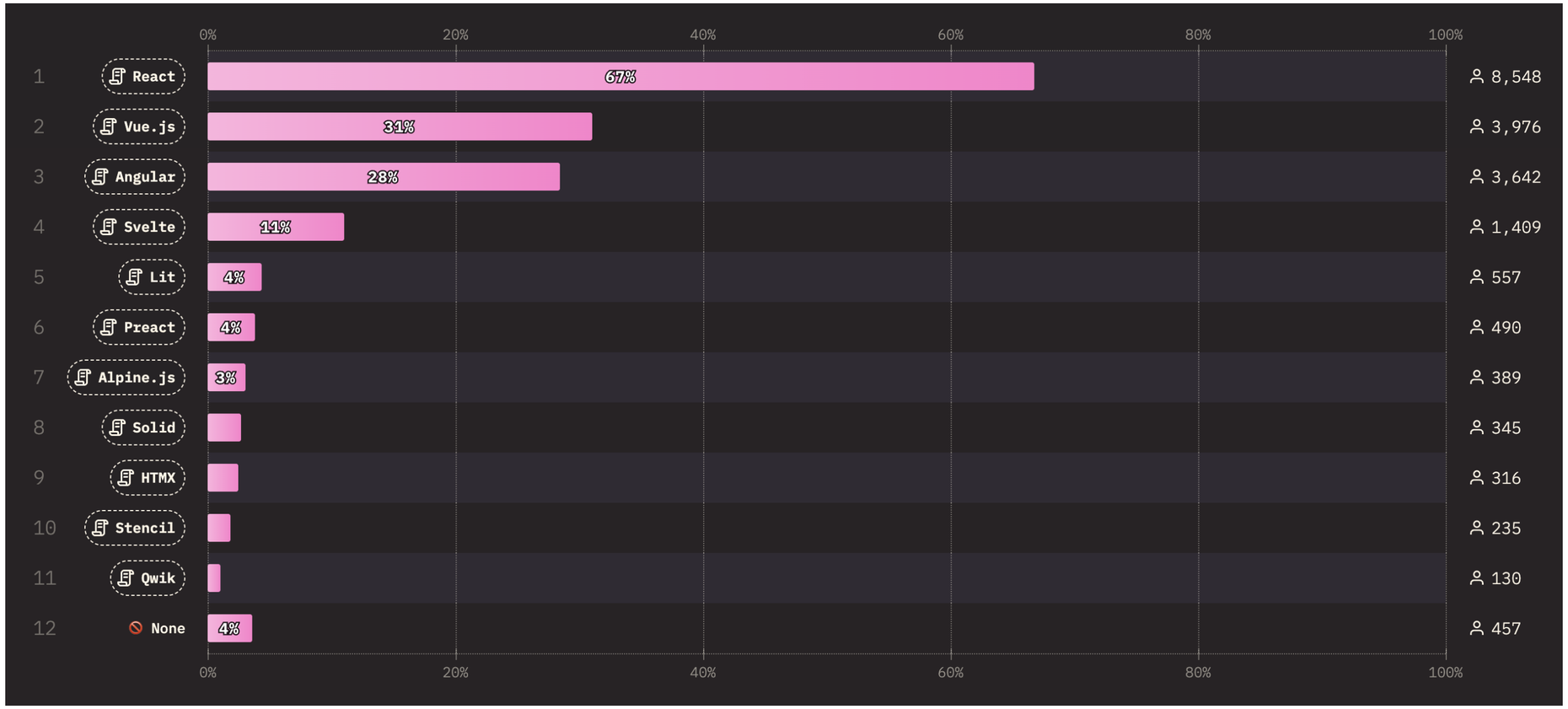
As stated in the official documentation, **React** is a JavaScript library for building user interfaces.

React is a very popular, declarative, component-based, and state-driven JavaScript library for creating user interfaces. It was developed by Facebook.

- Component-based ⟶ React revolves around components. Everything is about components
- Declarative ⟶ Each component must have all the information it needs in order to be rendered.

VOIS

React | Used at work



VOIS

Why React? Why not Vanilla JS?

Until around 2010, all websites were server-rendered — this is called server-side rendering.

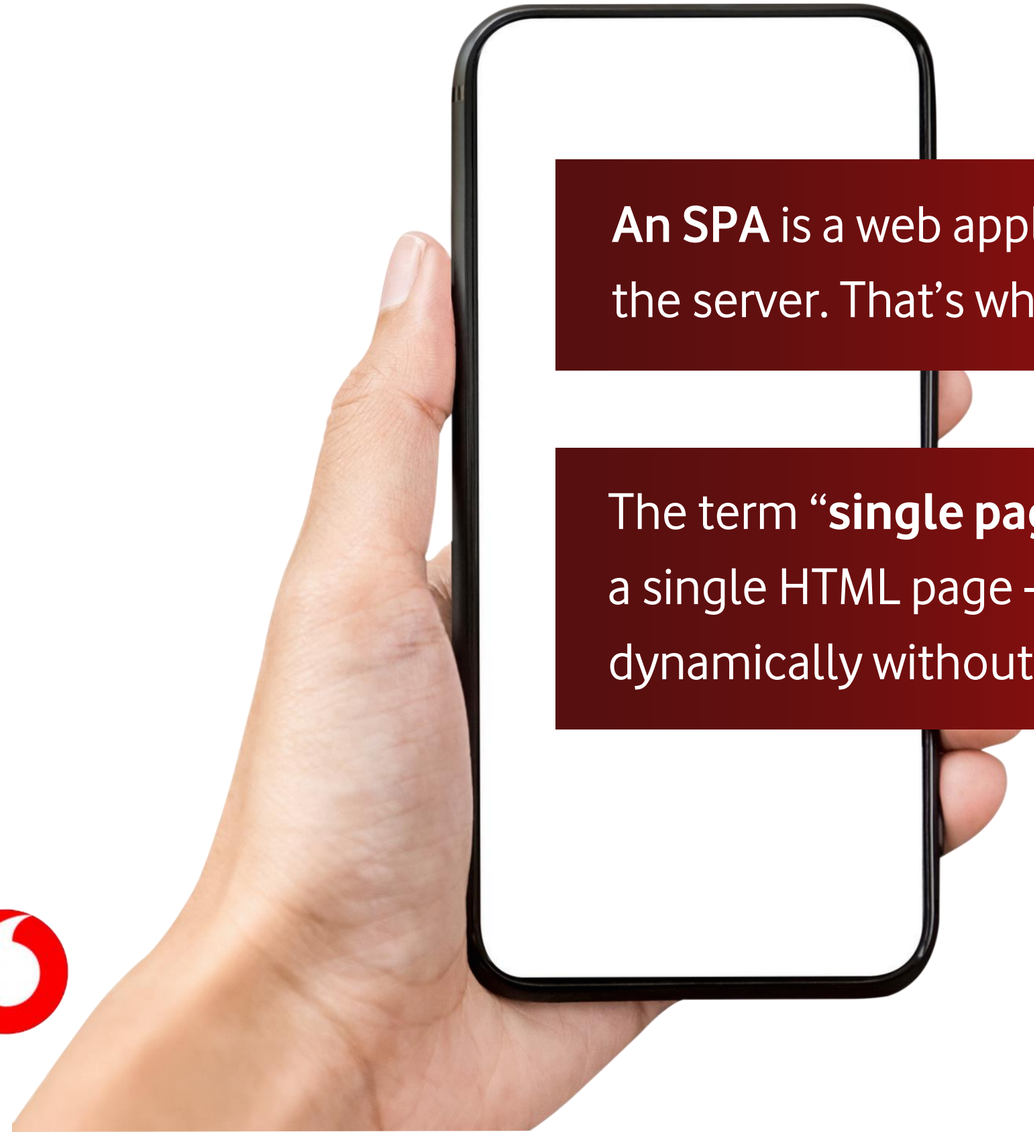
The files were assembled in the backend and sent to the browser.

The browser would then interpret the code and display the website.

Back then, JS was mostly used for styling purposes: displaying an animation, hiding some data, etc.

Websites grew more complex, creating a need for cleaner, leading to the rise of SPAs

What is an SPA?



An SPA is a web application that is rendered on the client side, not on the server. That's why they're called web apps, not just web pages.

The term "**single page**" comes from the fact that the app is technically a single HTML page — only the components inside that page change dynamically without reloading the entire page.

VOIS

Real World Example

React | React vs VanillaJS

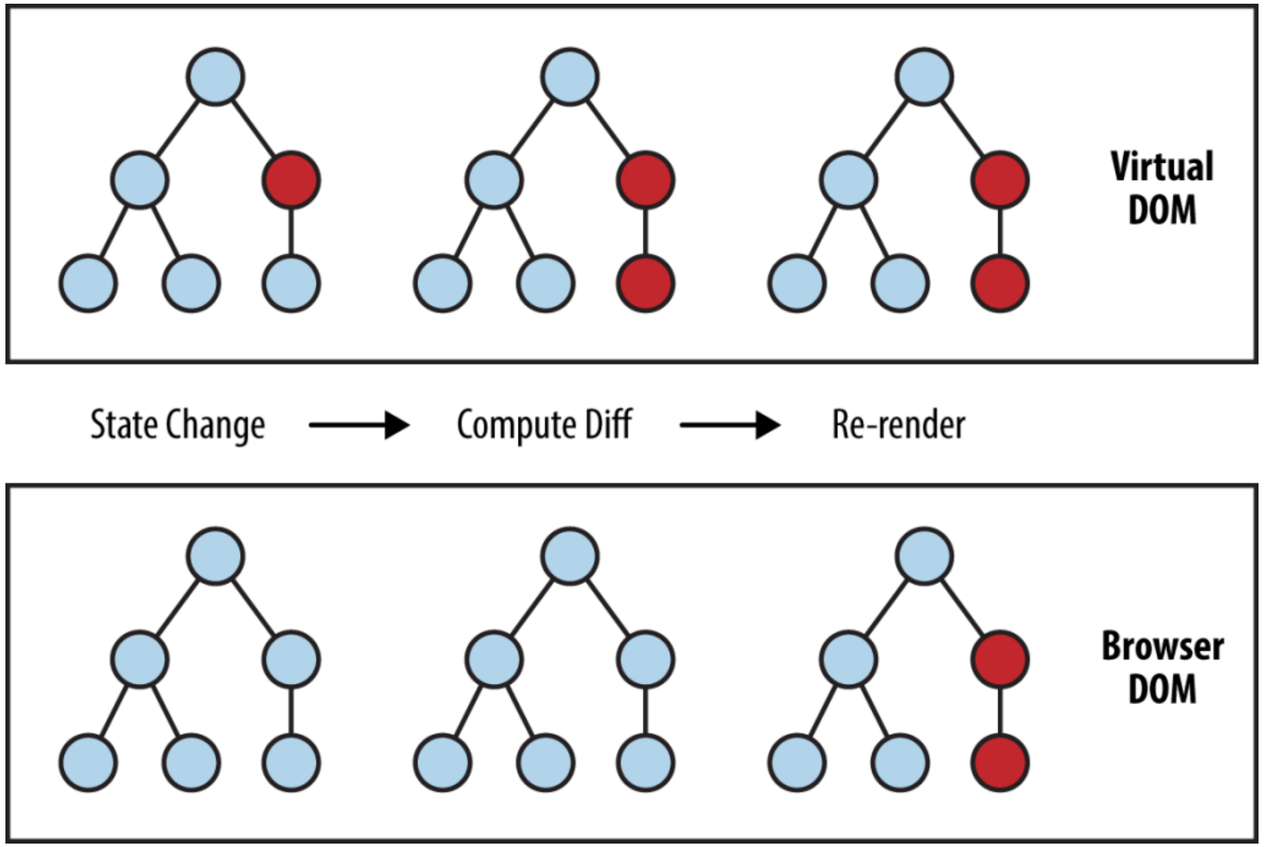
```react.jsx
import React from 'react';

const books = [
  { id: 1, title: 'The Great Gatsby', author: 'F. Scott Fitzgerald' },
  { id: 2, title: 'To Kill a Mockingbird', author: 'Harper Lee' },
  { id: 3, title: '1984', author: 'George Orwell' }
];

function BookList() {
  return (
    <div>
      <h1>Book List (React Example)</h1>
      <ul>
        {books.map(book => (
          <li key={book.id}>{`${book.title} by ${book.author}`}</li>
        ))}
      </ul>
    </div>
  );
}

export default BookList;
```

```vanilla.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Vanilla JavaScript Example</title>
</head>
<body>
  <div id="app"></div>

  <script>
    const books = [
      { id: 1, title: 'The Great Gatsby', author: 'F. Scott Fitzgerald' },
      { id: 2, title: 'To Kill a Mockingbird', author: 'Harper Lee' },
      { id: 3, title: '1984', author: 'George Orwell' }
    ];


    function renderBooks() {
      const app = document.getElementById('app');
      const ul = document.createElement('ul');

      books.forEach(book => {
        const li = document.createElement('li');
        li.textContent = `${book.title} by ${book.author}`;
        ul.appendChild(li);
      });

      app.appendChild(ul);
    }

    renderBooks();
  </script>
</body>
</html>
```

VOIS

React | Virtual DOM

The Virtual DOM is a lightweight, in-memory representation of the real DOM (Document Object Model). It is a key concept in React that helps improve performance and efficiency when updating the user interface.



Instead of directly modifying the real DOM every time something changes, React updates the Virtual DOM first. It then compares this updated Virtual DOM to a previous version using a process called "diffing", and finally applies only the minimal set of changes to the real DOM.

VOIS

JSX Syntax and Rules

In React, we use JSX — a declarative syntax that looks like a mix of HTML and JavaScript. It allows us to write UI elements directly inside JavaScript code.

Each component must return a single parent element. If you try to return multiple siblings without a wrapper, you'll get an error. You can wrap elements using a or a Fragment (<> </>).

```
function Welcome() {
  return (
    <div>
      <h1>Hello, world!</h1>
      <p>This is a React component using JSX.</p>
    </div>
  );
}

export default Welcome;
```

React | JSX Syntax and Rules

- **Use **className**** instead of class when adding CSS classes, because class is a reserved keyword in JavaScript.
- **Inline styles** can be written in JSX using two sets of curly braces:
  - The first {} tells React you're writing JS code.
  - The second {} contains the actual JavaScript object with style properties.
- In inline styles, CSS properties like background-color or font-size must be written in **camelCase** (backgroundColor, fontSize, etc.).

```
function Welcome() {
  return (
    <div>
      <h1 className="main-title">Welcome to React!</h1>
      <div style={{ backgroundColor: "lightblue" }}>
        This box is styled using inline styles.
      </div>
    </div>
  );
}

export default Welcome;
```

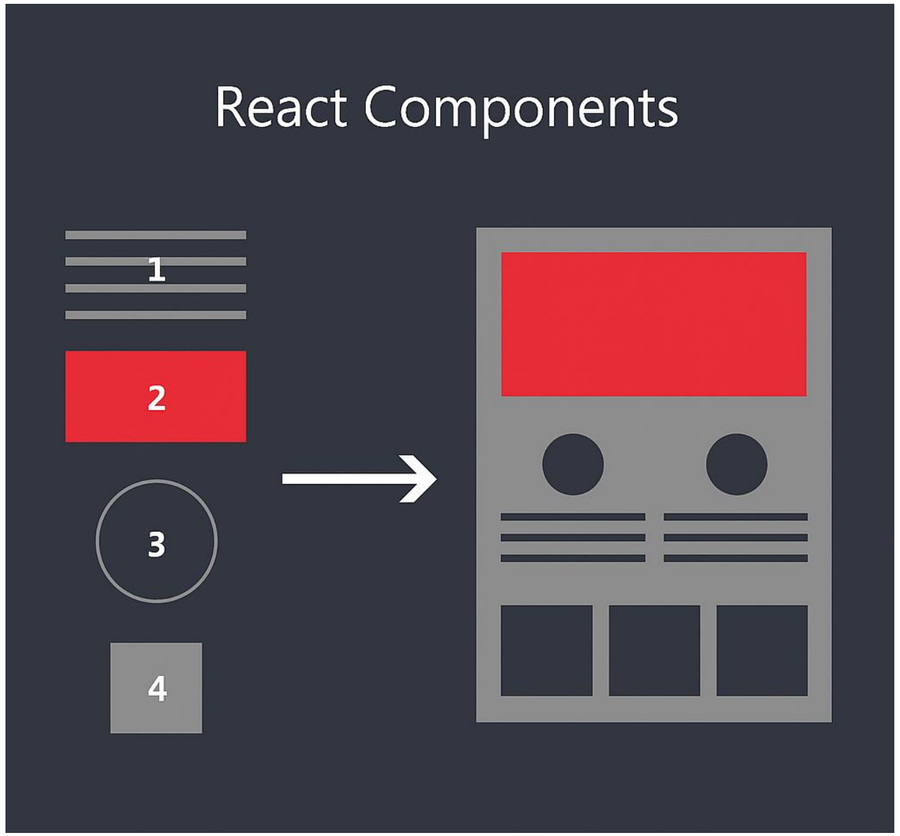
VOIS

React | What are react components?

ReactJS follows a component-based architecture, where UIs are composed of reusable building blocks called components.

A React component is a JavaScript function or class that returns a JSX representation of the UI.

It encapsulates the UI logic and state, making it reusable and modular.



VOIS

React | What are react props?

- “Props” = properties passed from parent to child component.
- Used to customize child components with dynamic data.
- Props are immutable inside the child.
- Props enable unidirectional data flow (parent → child only).

```
// App component
function App() {
  return <Greeting name="Alice" age={30} />;
}

// Greeting component
function Greeting(props) {
  return <p>Hello {props.name}, age {props.age}</p>;
}
```

VOIS

React | Hooks

React Hooks are built-in functions that let you use React's features — like state management and side effects — inside functional components.

They were introduced to eliminate the need for class components by giving functional components the ability to:

- Hold local state
- Respond to lifecycle events
- Access context, refs, and more

Hooks simplify component logic by keeping related behavior in one place, and they enable code reuse through custom hooks.

VOIS

React | Must-Know Hooks

01
useState

02
useEffect

03
useContext

04
useRef

05
useMemo

VOIS

React | Hooks - useState

Allows functional components to maintain internal state.

Returns a state variable and a function to update it.

Key Points

- useState is used to manage dynamic data in components
- State updates trigger a re-render
- Can use multiple useState calls in one component

Note: State updates are asynchronous; don’t rely on immediate value changes after calling setState.

```
import React, { useState } from "react";

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Current count: {count}</p>
      <button onClick={() => setCount(prevCount => prevCount + 1)}>Increment</button>
    </div>
  );
}

export default Counter;
```

VOIS

React | Component Lifecycle

Managed using the **useEffect() Hook**.

Handles side effects like data fetching, subscriptions.

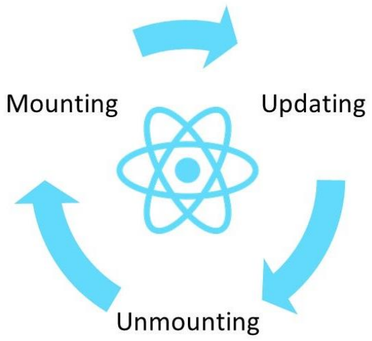
Mimics lifecycle stages from class components.

Lifecycle Phases via useEffect

- Mounting - useEffect with empty dependency array [] runs once.
- Updating - useEffect runs when dependencies change.
- Unmounting - Cleanup function in useEffect is triggered on unmount.

```
useEffect(() => {
  // 🕒 Start a timer when the component mounts
  const intervalId = setInterval(() => {
    console.log("Tick");
  }, 1000);

  // 🧹 Clear the timer when the component unmounts
  return () => {
    clearInterval(intervalId);
    console.log("Timer cleared on unmount");
  };
}, []); // ✅ Empty dependency array ensures it runs only on mount/unmount
```

React Component Lifecycle



VOIS

React | React Router

A powerful routing library for React that enables seamless navigation in single-page applications (SPAs).

It maps URLs to components and keeps the UI in sync with the browser — without full page reloads.

Why use React Router?

- Enables dynamic navigation within a React app
- Keeps UI in sync with the URL
- Improves user experience by eliminating full reloads

```
import { Routes, Route, Link } from 'react-router-dom';
import Home from './Home';
import About from './About';

function App() {
  return (
    <div>
      {/* 🔗 Navigation Links */}
      <nav>
        <Link to="/">Home</Link> | <Link to="/about">About</Link>
      </nav>

      {/* 🧭 Route Definitions */}
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
      </Routes>
    </div>
  );
}

export default App;
```

VOIS

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

Practice

Agenda

- How the Web works
- HTML
- CSS
- Practice
- JavaScript
- React
- Practice
- Closing Speech

VOIS

Resources

Recommended Udemy Courses

Top Learning Websites

VOIS

Any Questions?

Thank You