EPIC
Institute of Technology
Powered by <epam>

# How the Web works

Introduction to Front-End Development

Leu Kanapayeu

February 26, 2024

EPIC Institute of Technology
Powered by <epam>

## Contents

EPIC
Institute of Technology
Powered by <epam>

01

# Web standards

# Web standards

**Web standards** are the technologies we use to build websites. These standards exist as long technical documents called specifications, which detail exactly how the technology should work. These documents are not very useful for learning how to use the technologies they describe, but instead are intended to be used by software engineers to implement these technologies (usually in web browsers).

For example, the HTML Living Standard describes exactly how HTML (all the HTML elements, and other related technologies) should be implemented.

Web standards are created by standards bodies — institutions that invite groups of people from different technology companies to come together and agree on how the technologies should work in the best way to fulfill all of their use cases. The W3C is the best known web standards body, but there are others such as the WHATWG (who maintain the living standards for the HTML language), ECMA (who publish the standard for ECMAScript, which JavaScript is based on), Khronos (who publish technologies for 3D graphics, such as WebGL), and others.

# “Open” standards

One of the key aspects of web standards is that the web (and web technologies) should be free to both contribute and use. Therefore anyone can write the code to build a website for free, and anyone can contribute to the standards creation process, where the specs are written.

Because web technologies are created openly, in collaboration between many different companies, it means that no one company gets to control them, which is a really good thing. You wouldn't want a single company suddenly deciding to put the entire web behind a paywall, or releasing a new version of HTML that everyone has to buy to continue making websites, or worse still, just deciding they aren't interested any more and just turning it off.

This allows the web to remain a freely-available public resource.

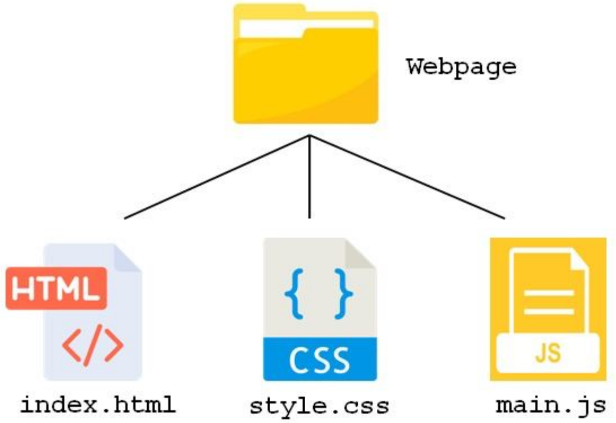
# Don't break the web

Another phrase you'll hear around open web standards is "don't break the web" — the idea is that any new web technology that is introduced should be backwards compatible with what went before it (i.e. old websites will still continue to work), and forwards compatible (future technologies in turn will be compatible with what we currently have).

EPIC Institute of Technology
Powered by <epam>

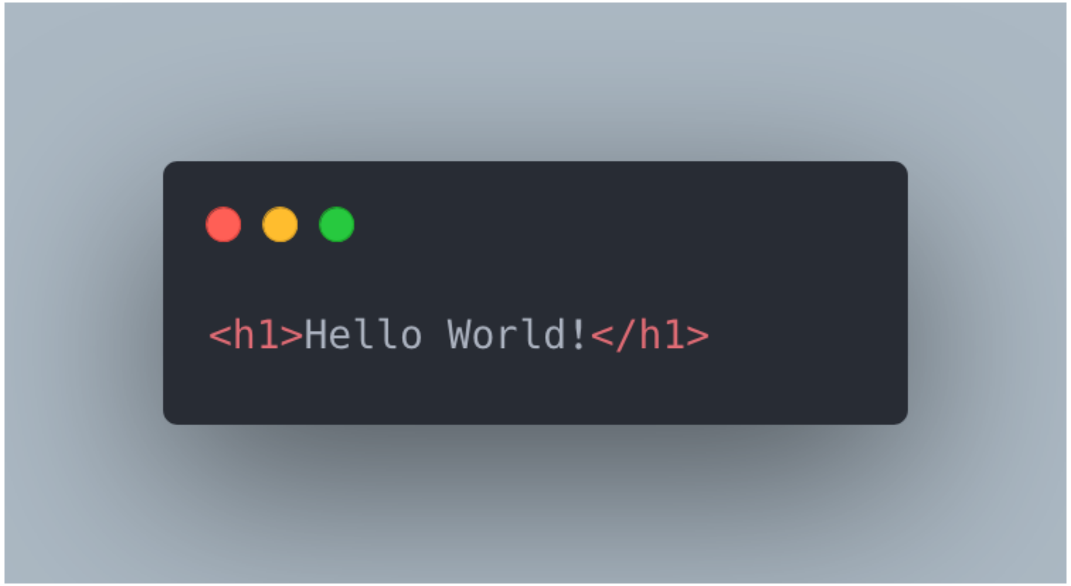
02

# Basic web technologies and tools



# HTML

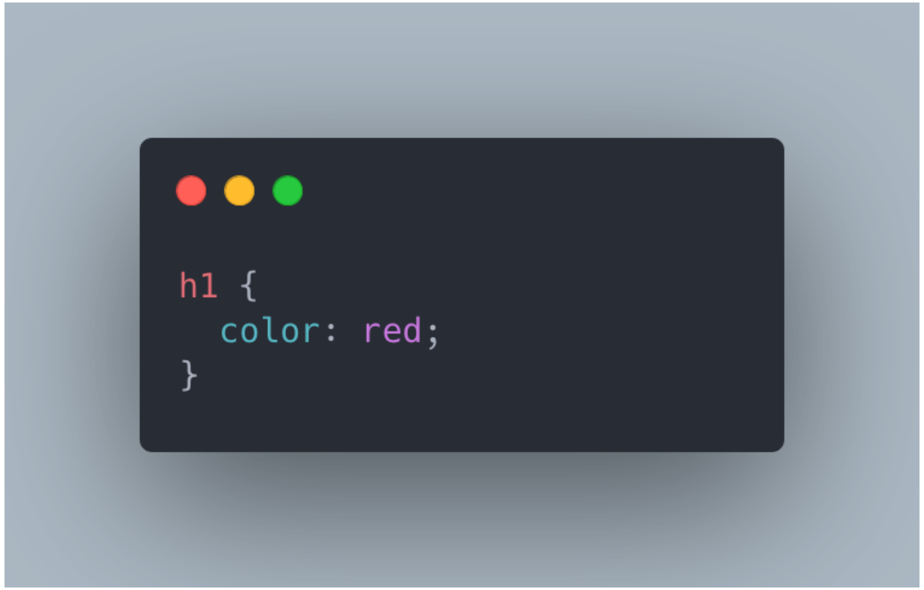
HTML (*HyperText Markup Language*) is the code that is used to **structure** a web page and its **content**. For example, content could be structured within a set of paragraphs, a list of bulleted points, or using images and data tables.



```html
<h1>Hello World!</h1>
```

# CSS

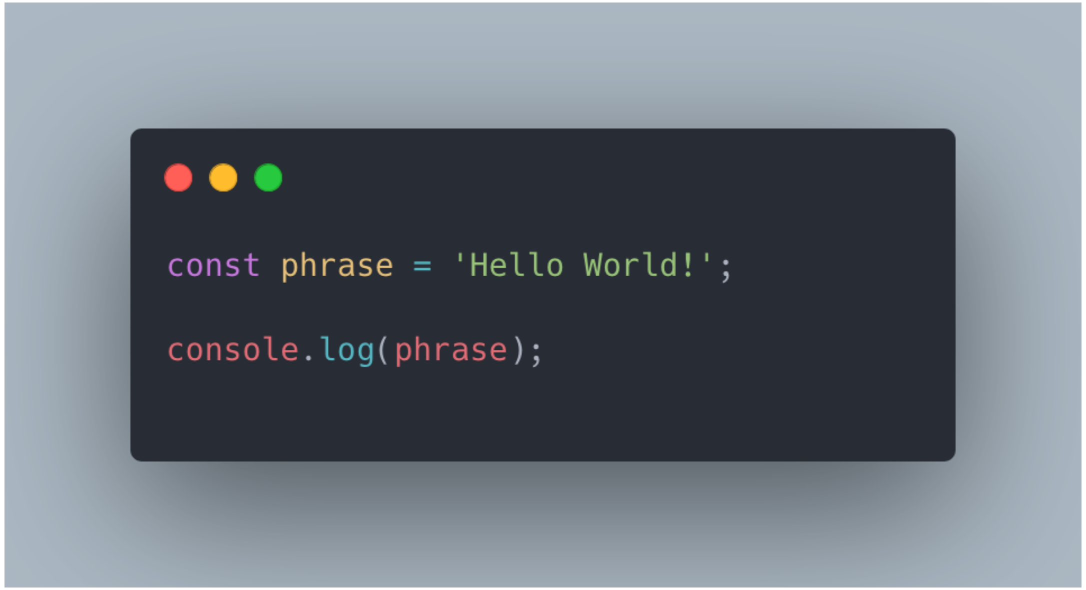
CSS (*Cascading Style Sheets*) is the code that **styles** web content. It is used to style and lay out web pages — for example, to change the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features.



```
h1 {
  color: red;
}
```

# JavaScript

JavaScript is a programming language that adds **interactivity** to your website. It allows you to change the content and styles dependending on user’s actions, ex. when a user press switch button to change the website’s theme.



```
const phrase = 'Hello World!';

console.log(phrase);
```

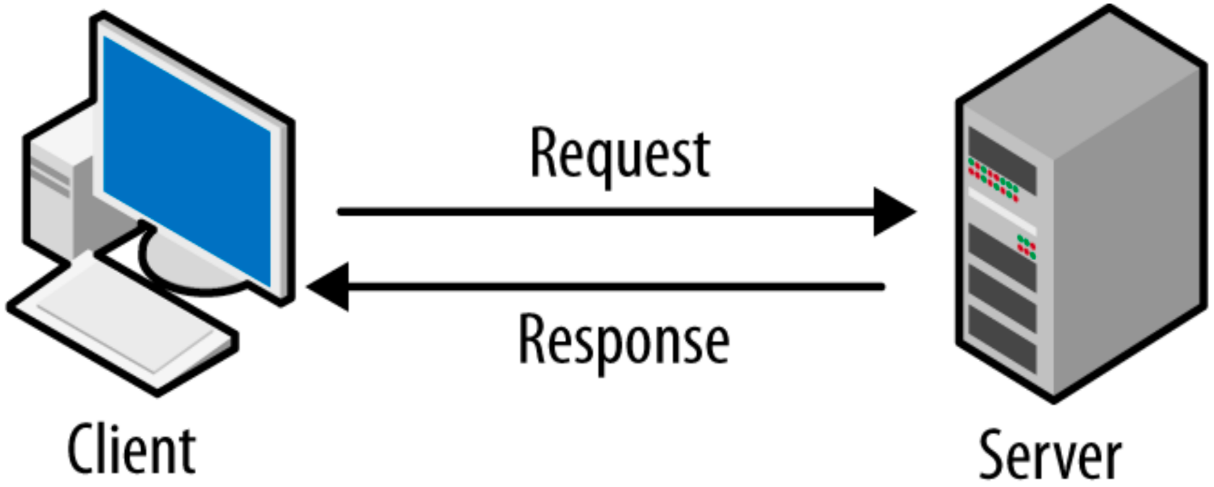
# Tools

Once you've learned the "raw" technologies that can be used to build web pages (such as HTML, CSS, and JavaScript), you'll soon start to come across various tools that can be used to make your work easier or more efficient. Examples include:

- The developer tools inside modern browsers that can be used to debug your code.
- Testing tools that can be used to run tests to show whether your code is behaving as you intended it to.
- Libraries and frameworks built on top of JavaScript that allow you to build certain types of website much more quickly and effectively.
- So-called "Linters", which take a set of rules, look at your code, and highlight places where you haven't followed the rules properly.
- Minifiers, which remove all the whitespace from your code files to make it so that they are smaller and therefore download from the server more quickly.

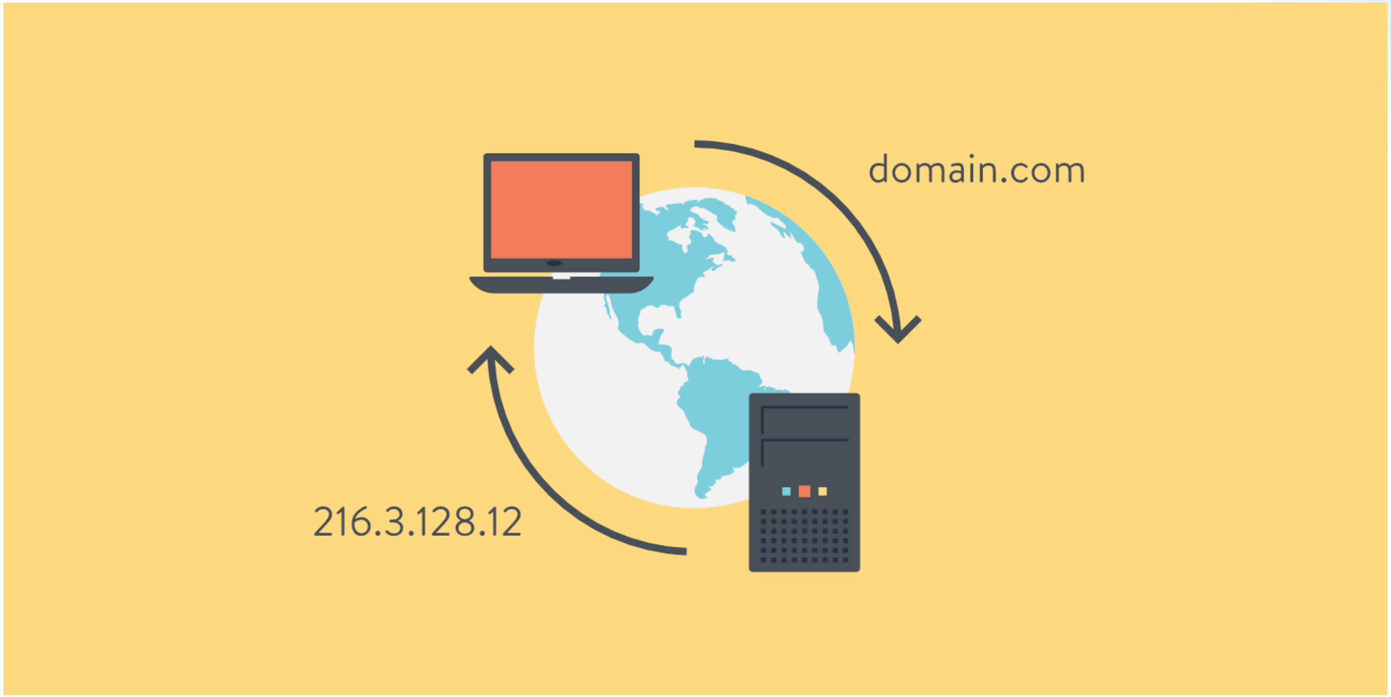
03

# Client-server model



EPIC Institute of Technology
Powered by <epam>

# DNS



# HTTP

The data is sent via the "HyperText Transfer Protocol" (known as "HTTP") - a standardized protocol that defines what a request (and response) has to look like, which data may be included (and in which form), and how the request will be submitted. And there also is HTTPS - it's like HTTP but encrypted.

# What are packets?

When data is sent across the web, it is sent in thousands of small chunks. There are multiple reasons why data is sent in small packets. They are sometimes dropped or corrupted, and it's easier to replace small chunks when this happens. Additionally, the packets can be routed along different paths, making the exchange faster and allowing many different users to download the same website at the same time. If each website was sent as a single big chunk, only one user could download it at a time, which obviously would make the web very inefficient and not much fun to use.

# File parsing

When browsers send requests to servers for HTML files, those HTML files often contain <link> elements referencing external CSS stylesheets and <script> elements referencing external JavaScript scripts. It's important to know the order in which those files are parsed by the browser as the browser loads the page:

- The browser parses the HTML file first, and that leads to the browser recognizing any <link>-element references to external CSS stylesheets and any <script>-element references to scripts.
- As the browser parses the HTML, it sends requests back to the server for any CSS files it has found from <link> elements, and any JavaScript files it has found from <script> elements, and from those, then parses the CSS and JavaScript.
- The browser generates an in-memory DOM (*Document Object Model*) tree from the parsed HTML, generates an in-memory CSSOM (*CSS Object Model*) structure from the parsed CSS, and compiles and executes the parsed JavaScript.
- As the browser builds the DOM tree and applies the styles from the CSSOM tree and executes the JavaScript, a visual representation of the page is painted to the screen, and the user sees the page content and can begin to interact with it.

EPIC
Institute of Technology
Powered by ‹epam›

04

# Web best practices

EPIC Institute of Technology Powered by <epam>

- **Cross-browser compatibility** is the practice of trying to make sure your webpage works across as many devices as possible. This includes using technologies that all the browsers support, delivering better experiences to browsers that can handle them (progressive enhancement), and/or writing code so that it falls back to a simpler but still usable experience in older browsers (graceful degradation). It also involves a lot of testing to see if anything fails in certain browsers, and then more work to fix those failures.
- **Responsive web design** is the practice of making your functionality and layouts flexible so they can automatically adapt to different browsers. An obvious example is a website that is laid out one way in a widescreen browser on the desktop, but displays as a more compact, single-column layout on mobile phone browsers.

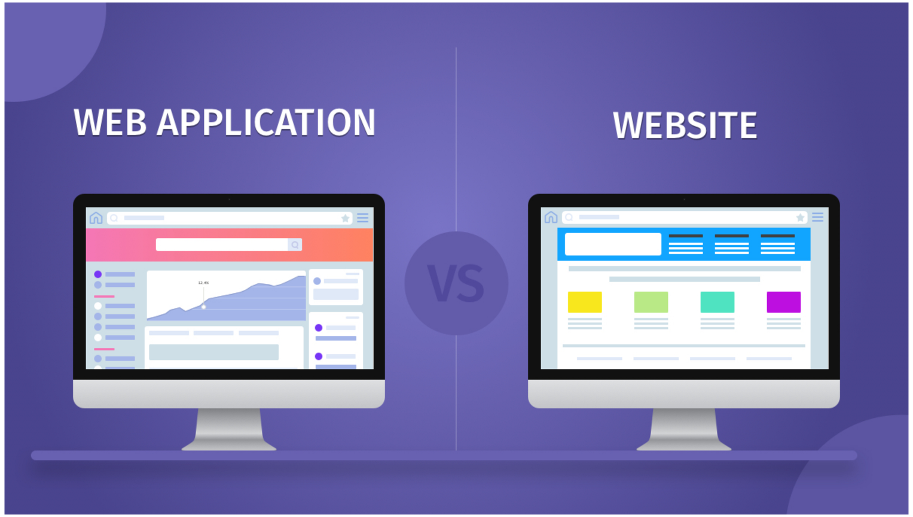
- **Performance** means getting websites to load as quickly as possible, but also making them intuitive and easy to use so that users don't get frustrated and go somewhere else.
- **Accessibility** means making your websites usable by as many different kinds of people as possible (related concepts are diversity and inclusion, and inclusive design). This includes people with visual impairments, hearing impairments, cognitive disabilities, or physical disabilities. It also goes beyond people with disabilities — how about young or old people, people from different cultures, people using mobile devices, or people with unreliable or slow network connections?

- **Internationalization** means making websites usable by people from different cultures, who speak different languages to your own. There are technical considerations here (such as altering your layout so that it still works OK for right-to-left, or even vertical languages), and human ones (such as using simple, non-slang language so that people who have your language as their second or third language are more likely to understand your text).
- **Privacy & Security.** These two concepts are related but different. Privacy refers to allowing people to go about their business privately and not spying on them or collecting more of their data than you absolutely need to. Security refers to constructing your website in a secure way so that malicious users cannot steal information contained on it from you or your users.

EPIC
Institute of Technology
Powered by ‹epam›

05

# Website vs Web application



06

# Summary

How the Web works

1. The URL gets resolved
2. A Request is sent to the server of the website
3. The response of the server is parsed
4. The page is rendered and displayed

# References

1. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works
2. https://academind.com/tutorials/how-the-web-works/
3. https://medium.com/@essentialdesign/website-vs-web-app-whats-the-difference-e499b18b60b4
4. https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/The_web_and_web_standards