Hyperiondev

TASK

Web Development Overview

Visit our website

Introduction

WELCOME TO THE WEB DEVELOPMENT OVERVIEW TASK!

In this task, you'll learn the basics of the web, including where it originated and some technical details about how it works. In this course, you will develop a web application that runs on the Internet, so background knowledge of the web is crucial to help you understand the context in which you will be working.

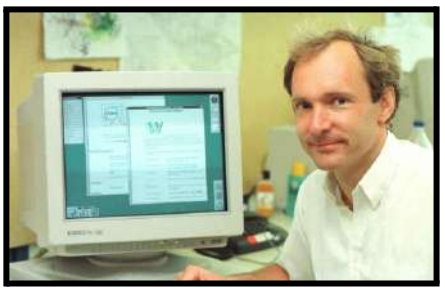
As you progress from someone who *uses* web applications to someone who actually *designs and builds* them, we will peel back the layers of the Internet to show you what really lies beneath the surface.

THE WORLD WIDE WEB

What is the World Wide Web really? It's a global information system consisting of web pages linked to each other using **hyperlinks**. These are the links that allow us to navigate from one page on a website to another. They also allow us to navigate to pages from other websites from around the world. It is this linking technology that creates the effect of an infinite web of information that we navigate daily.

A note from the
HyperionDev Team

Even though we can't imagine our lives without it, the World Wide Web is a rather recent invention. It was invented by Tim Berners-Lee, an English computer scientist, in 1989. Since its invention, it has expanded exponentially until it has become intricately interwoven into every part of our lives. Our work, entertainment, communication, and even our culture are strongly influenced by this powerful technology.



Tim Berners-Lee (ITU Pictures, 1990)

CORE COMPONENTS OF THE WORLD WIDE WEB

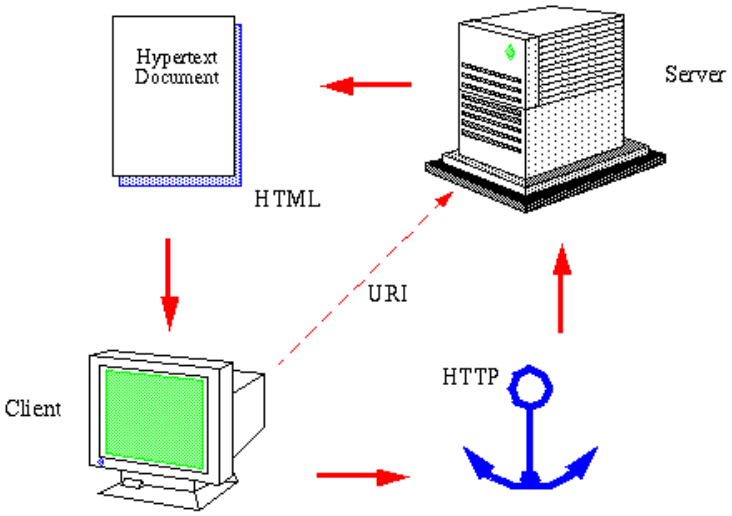
The World Wide Web consists of three key components. The first is called a **Uniform Resource Locator**, or URL. This is a unique identifier assigned to each page and resource on the web. It allows us to identify and retrieve the specific page or video, audio, etc. file that we want over the Internet.

The second technology is the language used to create web pages. As you know, this language is called **Hypertext Markup Language (HTML)**. Unlike other programming languages that allow us to create programs that actually perform tasks, this language is used to create the format of the page, i.e., what the contents are and how they are placed on the page. **Hypertext** is a way of organising information that lets you jump easily between different pieces of content. Instead of reading in a straight line, like in a book, you can click on links to move to related topics or pages.

The third technology is the protocol used to request and transfer web pages from one location to another. The Internet is a frantically busy and complex medium of communication, and in order for us to ensure that we transfer things successfully from one location to another, we must have rules of transfer (a protocol) for devices to adhere to. This protocol is called the **Hypertext Transfer Protocol**, better known as **HTTP**.

HOW THE WEB WORKS

We all access the web using a **client**. A client can be a phone, laptop, desktop, or basically anything we can use to access the web. To get to a particular resource (web page, etc.) on the web, we often open a **browser** and use a **URL** to specify what we want to see.

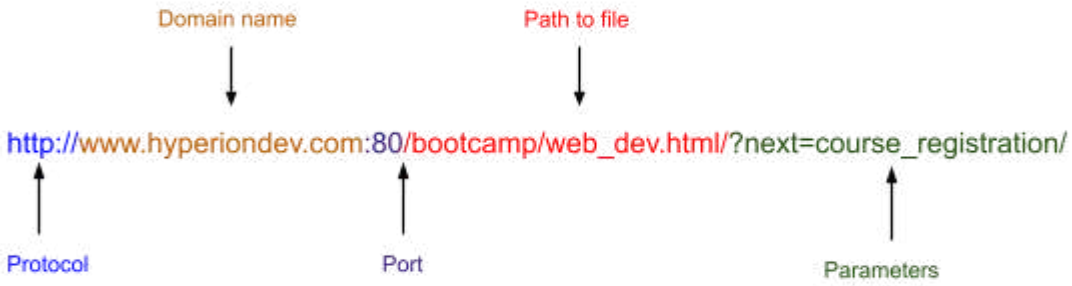


The World Wide Web Model (Frystyk, 1994)

A **web browser** is an application program on your web-accessing device that allows you to view websites. Chrome, Firefox, and Microsoft Edge are examples of web browsers. A browser takes a URL, which is the address of the website you want to visit, as input.

A URL is a type of **Uniform Resource Identifier**, or **URI**, which identifies the resource by specifying its location on the web. A URI is a way to give a unique name or address to resources on the internet, and a URL is a specific type of URI. This URL is a human-friendly address (for example, a page called **index.html**) for a particular resource on a particular web server somewhere in the world.

Consider the following fictional URL:



As you can see, the URL contains a lot of information:

1. It identifies the protocol being used to send information. In the example above, the protocol being used is HTTP.
2. It identifies the domain name of the web server on which the resource can be found, e.g., **www.hyperiondev.com**.
3. It identifies the port on the server. In this example, the port number is given as port 80. In reality, if the default HTTP ports are used, they don't have to be given in the URL (port 80 is the default for HTTP, port 443 for HTTPS).
4. It gives the path to the resource on the web server, e.g., **/bootcamp/web_dev.html**.
5. Parameters can be passed using the URL. Parameters are passed as key-value pairs (**?key=value&key2=value2**), e.g., **?next=course_registration**.

A web server is a computer that is set up to store and share many web resources, including the HTML files you will create, along with any images, videos, and CSS that you add to your HTML page. The function of the browser is to locate the server specified by the URL.

The browser will send an **HTTP** request for the web page to the server that stores it. The server receives requests from all over the world and sends an HTTP response with the requested resource back to the client's browser. After the server sends the page to the browser, the browser is then able to render the page on the screen for the user to view.

Web servers don't just contain static HTML pages. Back-end systems have been created due to the demand for more dynamic and responsive interaction with web applications. **Back-end development** has to do with writing code that sits on the server and dynamically builds resources that will be returned to the client. This code can be written using programming languages like F#, Python, etc. Many back-end applications interact with databases that store large amounts of data.

THE HTTP REQUEST–RESPONSE CYCLE

The HTTP request–response cycle is the fundamental process that occurs when a client communicates with a server over the web. This cycle involves a series of steps where the client sends a request to the server, and the server responds accordingly. The cycle provides us with an understanding of the way information flows through the web:

1. The user provides a client with a URL.
2. The client then builds a **request** for information.
3. The server receives this request and uses it to build a **response** that contains the requested information.
4. The response is sent back to the client in the requested format to be rendered by the client.

HTTP request

An HTTP request is a message sent from a client to a server to make a request for a particular resource or to perform a specific action. HTTP is the foundation of any data exchange on the web, with the most common types of HTTP requests being `GET` and `POST` requests.

The first line in the HTTP request method is referred to as the **request line**, and the following lines are called the **header lines**. The request line consists of three fields: the method field, the URL field, and the HTTP version field. For subsequent use cases, the HTTP 1.1 version will be utilised. See the example below:

```
GET /somedir/page.html HTTP/1.1
```

The method field can take on several different values.

Method	Description
`GET`	Used when a browser requests an object specified in the URL.
`POST`	The `POST` method is often used when a user enters form data, which requests a web page from the server. However, the contents of the page are dependent on the data entered by the user.
`DELETE`	Allows a user or application to delete an object on a web server.
`PUT`	Creates or replaces a resource to a web server. It's often used by applications that need to upload objects to web servers.
`PATCH`	Applies partial modifications to a resource without changing the whole data.

Header lines provide additional information about the request or the client making the request, and they are represented as key-value pairs separated by a colon. The below example specifies the host on which the object resides, the type of connection (in this case, a non-persistent connection), the browser type of the client, and the preferred language of the client.

```
Host: www.someschool.edu
Connection: close
User-Agent: Mozilla/5.0
Accept-language: en
```

The final component of the HTTP request message is the **entity body** or **message body**. This contains data related to the request, typically used in `POST`, `PUT`, or `PATCH` requests, but not all requests have a message body.

HTTP response

An HTTP response message is sent by a server to a client as a result of an HTTP request made by the client. The response contains information about the status of the request, along with optional data in the message body. The response message format consists of three components: a status line, six header lines, and an entity body.

The status line consists of three fields: the protocol version field, a status code, and a corresponding status message. The example below the status line indicates the server is utilising HTTP/1.1, with a status code of 200 and corresponding message "OK", which tells us the server has found and is sending the requested object to the client.

```
HTTP/1.1 200 OK
```

Analysing the header lines below the first line in the code below tells us that the TCP connection to the server is closed after sending the message. The "Date" line captures the time the response message was created and sent by the server, the "Server" line indicates which version of Apache web server is being run by the server, and the "Last-Modified" line tells us when the resource object was created or last updated. Furthermore, the "Content-Length" field indicates the size in bytes of the object being sent, and the "Content-Type" field indicates the type of object being sent in the entity body is HTML text.

```
Connection: close
Date: Tue,  14 Nov 2023 15:35:05 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 10 Oct 2023 14:23:46 GMT
Content-Length: 6821
Content-Type: text/html
```

The entity body contains the requested object itself and is stored in the format specified by the "Content-Type" field. In this case, the HTML content is JSON data in the entity body.

```
<!DOCTYPE html>
<html>
<head>
  <title>Example Page</title>
</head>
<body>
  <h1>Hello, World!</h1>
</body>
</html>
```

HTTP response status codes

HTTP status codes are three-digit numbers that are returned by a server in response to a client's request made to the server. These codes are grouped into different classes, each class having a specific meaning. Here are some of the common HTTP status code classes with some examples of each:

1. Informational responses (100–199):
    - 100 Continue
2. Successful responses (200–299):
    - 200 OK: The request succeeded, and the information was returned in the response.
3. Redirection responses (300–399):
    - 301 Moved Permanently: The requested object has been permanently moved, and the new URL is specified in the “Location” header of the response message. This will automatically be retrieved by the client.
4. Client error responses (400–499):
    - 400 Bad Request: Generic error code indicating that the request could not be understood by the server.
    - 404 Not Found: The requested document does not exist on this server.
5. Server error responses (500–599):
    - 505 HTTP Version Not Supported: The requested HTTP version is not supported by the server.

THE EVOLUTION OF THE WEB

The evolution of the Web can generally be divided into two phases: Web 1.0 and **Web 2.0**. The first generation of websites were basically one-way communication channels. The author would create a web page with some text and images, primarily to communicate information to customers. There was no means of interacting with the web page; all you could do was browse the pages on the website. Since there was no means of changing the web page that was being viewed without actually taking it off the Internet and editing the HTML, we can think of these web pages as being static. In a Web 1.0 site, the user sits back and consumes the content. For example, check out this classic **Web 1.0 site**.

Web 2.0 is all about allowing the user to interact with and contribute content to the website. This is done by providing some means for the user to enter a comment, upload a picture, or “like” something that has been added by someone else. This transition from passive consumption to active contribution to the content of the web page characterises the evolution of the web from 1.0 to 2.0. Because we can

actually change the details of the web pages we access, we can think of these web pages as dynamic. Examples of ways to engage with a dynamic website include:

- Posting a comment on someone's Facebook wall.
- Creating a page on Wikipedia or editing one that has already been added.
- Creating an investment account online using your bank's website.

Web 2.0 allows for the personalisation of our user experience for any given site. For example, after we log onto a social networking site like Facebook or X (formerly Twitter), we see information that is specific to our own personal user account. This is possible because, instead of a single pre-written web page being sent to your browser when you request a page, your request is processed by a program behind the scenes. The program then extracts data relating to you, which is then used to personalise your page.

USER ACCOUNTS

This is where the concept of a user account comes in. The program that runs behind the scenes needs to know the person it is dealing with in order to personalise a template specifically for them. Each person who uses a web application has a user account with a unique username that identifies them. When you log in to Gmail, your email address is used to identify you and link you with the personal data that will be presented to you. The user account is a core concept in dynamic web development.

Extra resource

This task has provided a very basic overview of how the web works. There's much more to explore beyond this overview, and as you continue in this bootcamp, you'll have opportunities to deepen your understanding. However, if you are interested in a little more detail on the fundamental protocols and infrastructure that make the web work, we highly recommend these additional readings:

1. **How Does the Internet Work?**
2. **Web Basics and Overview**

Spot check

Let's see what you can remember from this section.

1. What are the three key components of the World Wide Web?
2. What are the main differences between Web 1.0 and Web 2.0?

Take note

The task below is **auto-graded**. An auto-graded task still counts towards your progression and graduation. Give it your best attempt and submit it when you are ready.

When you select "Request Review", the task is automatically complete, and you do not need to wait for it to be reviewed by a mentor.

You will then receive an email with a link to a model answer, as well as an overview of the approach taken to reach this answer.

Take some time to review and compare your work against the model answer. This exercise will help solidify your understanding and provide an opportunity for reflection on how to apply these concepts in future projects.

In the same email, you will also receive a link to a survey, which you can use to self-assess your submission.

Once you've done that, feel free to progress to the next task.

Auto-graded task

Follow these steps:

- Create a document titled **WebFundamentals** (in a Google doc, Word doc, or anything similar that you can save to PDF format when you are finished). Briefly answer the following questions in your document:

    1. What is Web 3.0, and how is it different from Web 1.0 and Web 2.0? Read **this article** for more information.

    2. What are the functional differences between the front end of a web application and its back end? See **this article** for more information.

    3. In your own words, explain the process that takes place from when you type a URL into the address bar in your browser until you finally view the page you have requested. Include the HTTP request–response cycle in your answer and provide an example of the HTTP request and response messages. Watch this **TED-Ed video** to help you understand this better.

- Convert your document to PDF format and submit.

Important: Be sure to upload all files required for the task submission inside your task folder and then click "Request review" on your dashboard.

Spot check answers

1. Question 1's answer: The Uniform Resource Locator (URL), Hypertext Markup Language (HTML), and Hypertext Transfer Protocol (HTTP).
2. Question 2's answer:

    a. Web 1.0: One-way communication channels where the web page presented content to the consumer and the consumer had no way of engaging. These were static web pages.

b. Web 2.0: The consumer can interact with the website through communication channels like uploads, “liking”, and leaving comments. These are dynamic web pages. Dynamic web pages can be personalised to each individual consumer through user accounts.

Share your thoughts

Please take some time to complete this short feedback **form** to help us ensure we provide you with the best possible learning experience.

Reference list

ITU Pictures. (1994, July 11). *Tim Berners-Lee*. Flickr. **https://www.flickr.com/photos/itupictures/16662336315**

Frystyk, H. (1994, July). *The World-Wide Web*. W3C. **https://www.w3.org/People/Frystyk/thesis/WWW.html**