3 Web Designs in 3 Weeks

Convert designs to websites

Juntao Qiu

3 Web Designs In 3 Weeks

Convert designs to websites

Juntao Qiu

This book is for sale at http://leanpub.com/3webdesignsin3weeks

This version was published on 2022-09-25

Contents

Preface (2nd edition)

Somehow eight years have passed since I wrote the first edition of `3 Pages in 3 Weeks`. Eight years is a considerablely long time in an era of rapid front-end technology development. Eight years ago, Internet Explorer was still popular, `flex` layout would not be adopted on most projects due to compatibility, and CSS variables were treated as a radical technology. Eight years ago, `float` layouts were a necessary skill for most advanced layouts, and to set up live refresh during development, developers had to do a lot of configuration locally.

While occasionally going through my mailbox recently, I realised it had been a long time since I updated this booklet. When I opened it up, I was surprised to find that much of it was completely outdated. Such as the fact that `float`, which was the defacto of complicated layout back then, is almost completely gone in the real world. `flex` and even `grid` layouts can easily be implemented in ways that developers hadn't even thought of before. And launching a live server in Visual Studio Code takes only a few seconds, not to mention that all modern browsers shipped with fancy DevTools that support all kinds of debugging and editing.

Structure of the book

Similar to the first edition, the book is divided into three main chapters. In fact, it is a process of implementing different styles of design, three projects in three weeks time.

The first chapter is about a typical landing page. It includes a navigation bar, a hero banner and a news list. Here we go through some environment setup, including the configuration of the tools and explanation of the basics. By the end of the first chapter, you will have learned how to deconstruct a design, write the corresponding HTML, and implement common layouts through `flex` layouts. Also you will be able to publish your website to the public and allow your friends or colleagues to access it.

Chapter two is a conceptual design for `Instagram`. It includes a vertical navigation bar, card design and more complex typography and layout. We will explore more about `flex` layout and `grid` layout. In addition, we will learn about the use of font icons and the priority of selectors in CSS.

Chapter three is a page with a responsive design. We will complete the interface on the small screen with a mobile-first strategy first, and then overrides styles to adapting to desktop version through media queries. In the process, we'll learn how to simplify HTML writing using `Emmet`, use CSS variables to eliminate duplicate code and much more. At the end of the project, we will also learn some foundamental design principles so that you can be more confident about how to implement a page on your own without a mockup.

Style changes of new edition

The first edition had two projects plus a hands-on (build-your-own) practice. The new edition has been modified to three different projects, placing the exercises section at the end of each chapter. Regarding the distribution of knowledge, the new edition is more evenly distributed to ensure that each chapter covers some important points.

In addition, when re-reading the first edition, I found many incorrect assumptions about beginners' knowledge. Many concepts unfamiliar to beginners were not explained enough, as well as problems such as the slightly large amount of code.

In rewriting it, I tried to add as much background knowledge as possible to make each step easier for starters.

I hope you can enjoy the new edition just as you did for the first one.

August 2022 in Melbourne

Preface (1st edition)

In November 2014, I organised a 3-week workshop at the Thoughtworks Xi'an office called 3 Pages in 3 Weeks. Participants were asked to implement a web page using HTML/CSS in each session.

The workshop was a bit like a drawing class, where everyone worked from a mockup (usually from a professional designer) like a landing page or product detail page and implemented it in HTML/CSS.

The participants were mixed with front-end developers, back-end developers, test engineers, UX designers, business analysts and recruiters. Most participants have had some prior exposure to CSS, but not to the point of proficiency; a small number of participants have no experience to web design/development at all, and the rest of them are specialised in web design/development.

In the workshop, I covered the basics of using the new HTML5 tags for more semantic documents and the new CSS3 features like shadows, animations, rounded corners, etc. In the second half of each session, there was a showcase where they could share their work and get feedback from others and the instructor (me). After the showcase, I demonstrated how to implement a small part of the page to help those participants who did not have a lot of basic knowledge.

As a result, the workshop was very well received, and participants put in a lot of efforts to complete the 'homework'. By the end of the three weeks, the majority achieved a solid foundation of HTML/CSS and could implement a working page from any given mockup. And according to the results of a post-course survey, most of them enjoyed the content as well as how it was delivered.

One day after the workshop, I realised that this might be a good topic that could easily be crystallised for others who had not attended the workshop. So I decided to put it all together and create a book, saving me the trouble of going over and over the lectures myself.

Some updates

3 Pages in 3 Weeks has received a surprising amount of support since it was posted on GitBook, from both my colleagues and the community. A lot of my colleagues at Thoughtworks have helped me to actively promote the book after its release, which has resulted in the book getting more attention, and I would like to thank them for that as well.

March 2015 in Shenzhen

Some testimonials about the first edition of the book

“I found out about your blog through the Three Weeks Three page in ThoughtWorks WeChat Channel, and it was very insightful to read it. The way you think about the issues and the way you explain them is fascinating and enjoyable to read, thanks.” – **Hongjiang Zhang**

“Mr Qiu: Hi, I was surprised to see the 3 weeks 3 pages project on your podcast by chance, and I have been looking for such a tutorial.” – **Mor1999**

“…I just came here after seeing your book and learned a lot from your book, thanks :)” – **Dai Zhi Lian**

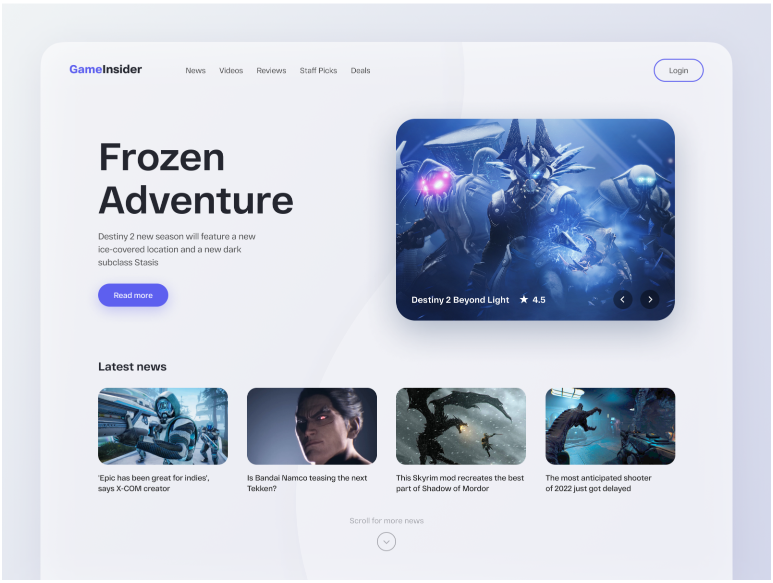
"...I purchased your 3 weeks 3 pages and have benefited a lot..." – **Douche**

"...I have received (gained) a lot from reading your 3 weeks 3 pages..." – **tkin**

"...I downloaded the book from gitbook and paid for another one from selfstore..." – **Evoque**

Week 1: Game News - Game Insider

In week one, we will work together to implement a page in HTML and CSS. This mockup[1] by designer Roman Zakhareko about game news was published on Dirbbble. It was chosen because it contains many essential elements, such as the navigation bar, the hero banner, a list with images and text, etc. Understanding how these elements are implemented in HTML and CSS will help us to implement any other more complex pages in the future.


GameInsider
News
Videos
Reviews
Staff Picks
Deals
Login
Frozen Adventure
Destiny 2 new season will feature a new ice-covered location and a new dark subclass Stasis
Read more
Destiny 2 Beyond Light ★ 4.5
Latest news
'Epic has been great for indies', says X-COM creator
Is Bandai Namco teasing the next Tekken?
This Skyrim mod recreates the best part of Shadow of Mordor
The most anticipated shooter of 2022 just got delayed
Scroll for more news

Design By Roman Zakhareko

In addition, this mockup is relatively simple in general and contains

[1] https://dribbble.com/shots/19038057-Game-News-Main-Page

a limited amount of elements. This allows us to intersperse the implementation of the page with an introduction to some useful tools. Such as how to select colours from the mockup, how to use icons in HTML, how to edit code in real-time. The use and configuration of these tools is also crucial to implementing our other two designs in the following two weeks.

After this chapter, you will be able to know: how to write HTML and CSS code in Visual Studio Code editor, how to debug styles in the browser, how to deconstruct a mockup, how to use `flex` layout and how to publish your work to the public.

Tools

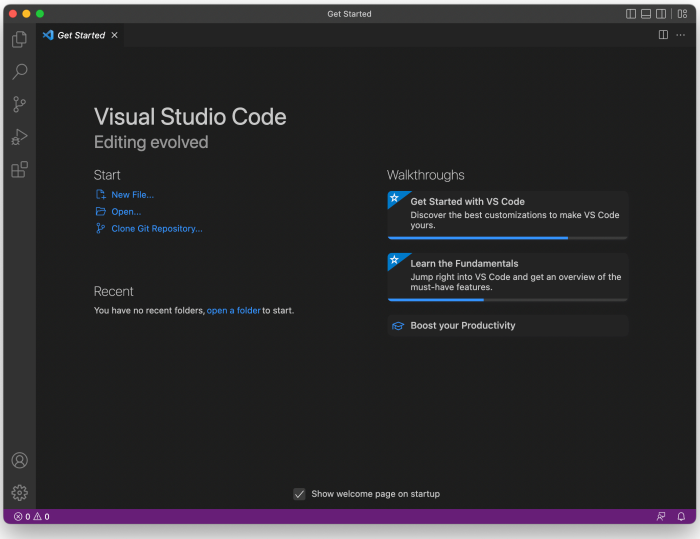
> To do a good job, one must first use the right tools

Editors

In this book, we will be using Visual Studio Code[2] (hereafter referred to as Code) for writing and debugging code. While there are countless editing tools out there, I have found that Code with a few popular extensions makes writing HTML and CSS easy and fun. More importantly, it is free and small enough that you don't need expensive hardware to run it. Also, you can install extensions to make Code even more efficient and customisable just for you.

After downloading the platform-specific installation package and installing it, you will get an interface that looks like this. (I will demonstrate with the Mac version in this book.)

[2]https://code.visualstudio.com/



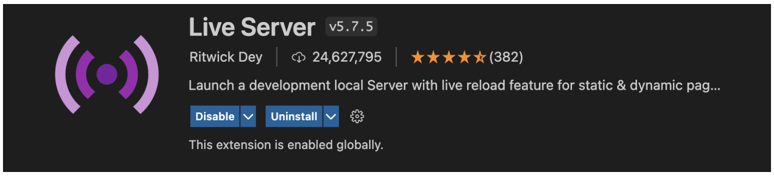
You have no recent folders, open a folder to start.

Discover the best customizations to make VS Code yours.

Jump right into VS Code and get an overview of the must-have features.

Visual Studio Code

Before we can start, we need to install one Code extension: Live Server[3] (by Ritwick Dey). Once installed, Live Server will launch a local HTTP server and monitor your HTML/CSS changes so that the corresponding page in your browser will automatically refresh as soon as an edit is made.



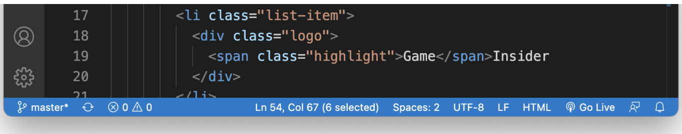
Launch a development local Server with live reload feature for static & dynamic pag...

This extension is enabled globally.

Live Server

After installation, you will see a new icon in the status bar.

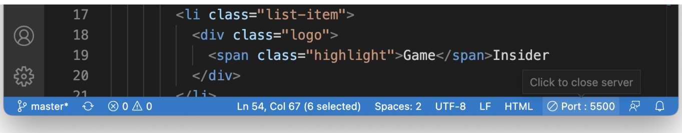
[3]https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer



```
<li class="list-item">
  <div class="logo">
    <span class="highlight">Game</span>Insider
  </div>
</li>
```

Live Server Installed

Clicking on this icon will start a local HTTP service, and Live Server will launch your default browser and load the page.

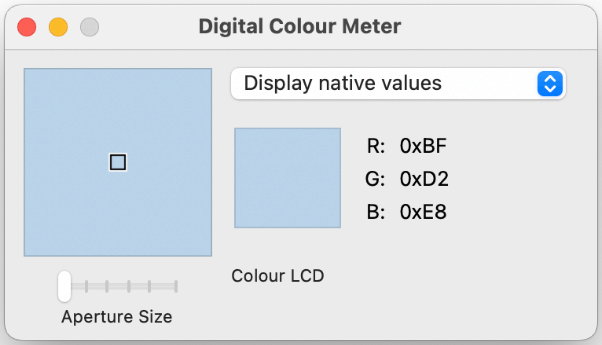


```
<li class="list-item">
  <div class="logo">
    <span class="highlight">Game</span>Insider
  </div>
</li>
```

Live Server Launched

Colour picker (optional)

We will need a colour picker as the mockup doesn't come with the colors. Depending on the platform, you may need to install different applications. On Mac, the built-in tool `Digital Colour Meter` is pretty good and does the job.



Color Picker

In practice, designers usually would share the colour palette with developers so that you don't have to inspect yourself. But at other times, such as when you wish to do some exploration or when you only have the mockup on hand, the colour picker is a could be handy.

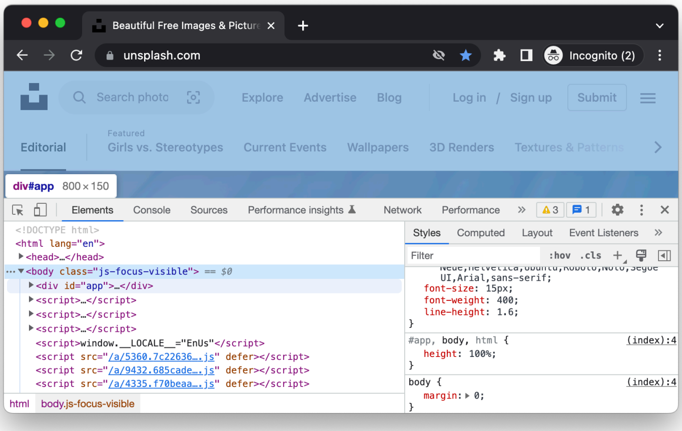
Well, that's all we need. Let's get started on our first page.

Browsers

Today's browsers are extremely versatile. It can be a video player, a music player, a reader, an image viewer or an IDE! That's right, an `IDE`, and I'm not talking about an editor such as codepen[4], but inside it is a development environment itself.

For example, in Chrome on Mac, you can open the DevTool by right-clicking on any web page and click `inspect` or just pressing `Cmd+Shift+I`.

[4]https://codepen.io/



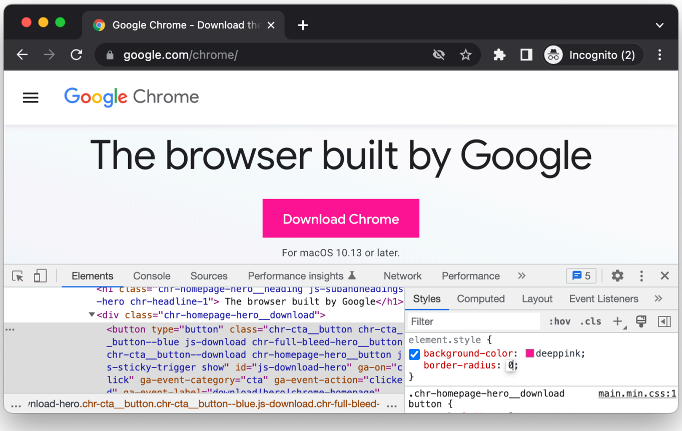
```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
…▼<body class="js-focus-visible"> == $0
   ▶<div id="app">…</div>
   ▶<script>…</script>
   ▶<script>…</script>
   ▶<script>…</script>
    <script>window.__LOCALE__="EnUs"</script>
    <script src="/a/5360.7c22636….js" defer></script>
    <script src="/a/9432.685cade….js" defer></script>
    <script src="/a/4335.f70beaa….js" defer></script>
```

```
     UI,Arial,sans-serif;
   font-size: 15px;
   font-weight: 400;
   line-height: 1.6;
}
#app, body, html {                    (index):4
   height: 100%;
}
body {                                (index):4
   margin: ▶ 0;
}
```

Chrome Devtool

Once in inspect mode, you can click on any element on the page to see the code. On the right-hand side, you can see the style used for the element, and you can modify them in place. Pretty amazing, right?

For example, on the `Chrome` download page, you can inspect the download button and change the rounded blue button to pink and remove the rounded corners.



```
<h1 class="chr-homepage-hero__heading js-subandheadings
-hero chr-headline-1"> The browser built by Google</h1>
▼<div class="chr-homepage-hero__download">
  <button type="button" class="chr-cta__button chr-cta_
  _button--blue js-download chr-full-bleed-hero__button
  chr-cta__button--download chr-homepage-hero__button j
  s-sticky-trigger show" id="js-download-hero" ga-on="c
  lick" ga-event-category="cta" ga-event-action="clicke
  d" ga-event-label="download|hero|chrome-homepage"
```

```
... ›nload-hero.chr-cta__button.chr-cta__button--blue.js-download.chr-full-bleed- ...
```

```
element.style {
  background-color: deeppink;
  border-radius: 0;
}
.chr-homepage-hero__download        main.min.css:1
button {
```

Chrome Download

During our process, we will frequently use the developer tools for debugging to see the effect in real-time. Once we are happy with our changes, we can then copy them back into the editor.

Getting started with HTML and CSS

This section will introduce you to HTML and CSS in the simplest way possible. If you have some basic knowledge of HTML and CSS, you are welcome to skip this section.

All the different pages you see in your browser every day, whether a KFC order page or a news portal, a Douban book review or a Wikipedia page, a personal blog or a document editor, are based on HTML – the Hyper Text Markup Language. Today HTML is used on the web far beyond what it was originally designed for, but understanding and mastering it is essential for you to implement any complex page.

However, what is described in HTML is only a part of the story:

it's for the content part only. If you want your pages to be visually appealing, then a technique to be able to customise styles of the content is needed: CSS - Cascading Style Sheets.

With CSS, we can use selectors to find one element or a class of HTML elements and apply styles to them. Here the style can be visual elements such as background colours, font colours, borders, shadows, and logical layouts such as inner spacing (called padding)and outer spacing (called margin).

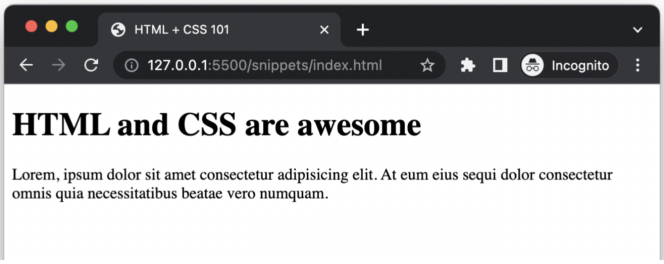
In short, we describe the content through HTML and change the style through CSS. HTML describes an unpolished house, while CSS corresponds to finishes (yes, it is sometimes possible to knock out **non-load bearing walls** to change the layout of the interior of the house).

We can discuss the relationship between the two with a small example below.

```
<div class="container">
  <h1>HTML and CSS are awesome</h1>
  <p>
    Lorem, ipsum dolor sit amet consectetur adipisicing e\
lit. At eum eius sequi
    dolor consectetur omnis quia necessitatibus beatae ve\
ro numquam.
  </p>
</div>
```

Inside a `div` (short for `divider`, an invisible container for other elements), we defined an `h1` (meaning heading level 1) tag and then a `p` tag (meaning paragraph).

By default, the browser would render it as something like this:



HTML and CSS are awesome

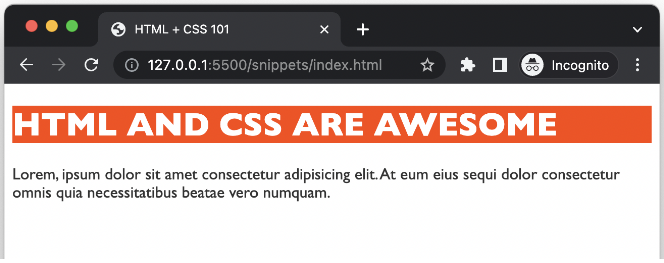
Lorem, ipsum dolor sit amet consectetur adipisicing elit. At eum eius sequi dolor consectetur omnis quia necessitatibus beatae vero numquam.

HTML Only

And if we define some CSS to modify the style:

```css
.container {
  max-width: 800px;
  margin: 0 auto;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Tre\
buchet MS", sans-serif;
}

h1 {
  text-transform: uppercase;
  background-color: orangered;
  color: white;
}

p {
  color: #333;
}
```

We could get something more appealing like this:



HTML AND CSS ARE AWESOME

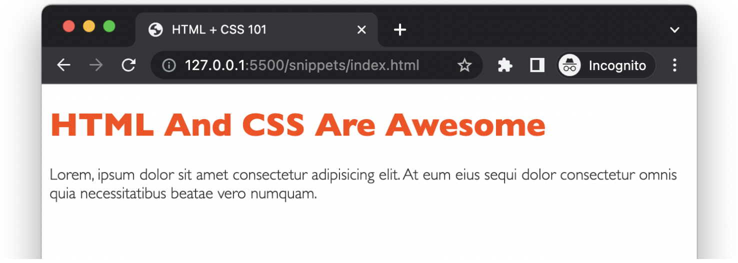
Lorem, ipsum dolor sit amet consectetur adipisicing elit. At eum eius sequi dolor consectetur omnis quia necessitatibus beatae vero numquam.

With Style One

Don't worry if you don't understand everything at the moment, I'll have you covered. If we apply another style set like:

```css
.container {
  max-width: 800px;
  margin: 0 auto;
  font-family: "Gill Sans", "Gill Sans MT", Calibri, "Tre\
buchet MS", sans-serif;
}

h1 {
  text-transform: capitalize;
  color: orangered;
}

p {
  color: #333;
  font-weight: 100;
}
```

With no changes to the HTML, we get a completely different picture of the presentation simply by changing the CSS.



HTML And CSS Are Awesome

Lorem, ipsum dolor sit amet consectetur adipisicing elit. At eum eius sequi dolor consectetur omnis quia necessitatibus beatae vero numquam.

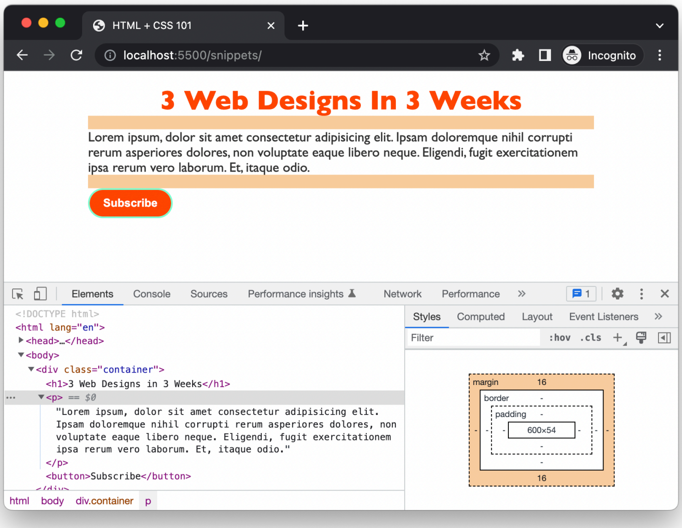
With Style Two

Here I have deliberately ignored the place of JavaScript in our projects (which is crucial in any serious web applications), as this book is primarily concerned with the combination of HTML and CSS. Additionally, `JavaScript` alone has become very rare in web development, and people are more or less need libraries or frameworks such as React or AngularJS for web applications.

Box model

One of the core functions of CSS is to provide layout for page elements, and at the heart of layout is determining the size of an element. To understand the size of an element, we need to talk a little bit about to the box model.

When viewing a particular element in your browser's DevTool, you may have noticed a small set of boxes in the bottom right-hand corner of the page:



3 Web Designs In 3 Weeks

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Ipsam doloremque nihil corrupti rerum asperiores dolores, non voluptate eaque libero neque. Eligendi, fugit exercitationem ipsa rerum vero laborum. Et, itaque odio.

```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▼<div class="container">
      <h1>3 Web Designs in 3 Weeks</h1>
     ▼<p> == $0
        "Lorem ipsum, dolor sit amet consectetur adipisicing elit.
        Ipsam doloremque nihil corrupti rerum asperiores dolores, non
        voluptate eaque libero neque. Eligendi, fugit exercitationem
        ipsa rerum vero laborum. Et, itaque odio."
      </p>
      <button>Subscribe</button>
```

Bbox Model

The space that an element occupies on the page is made up of several components: the content of the element itself, the distance from the content to the border: the inner spacing (padding), the width of the border, and the outer spacing between the border and other elements (margin).

We can set these different parts independently in CSS, for example, `padding-left` to control the left inner spacing, `margin-top` to control the upper outer spacing, and `border-right` to control the right border, etc.

I could list many similar trivialities, but to keep you from getting bored, let's start implementing the mockup, where we will learn a lot about the syntax of HTML tags and CSS.

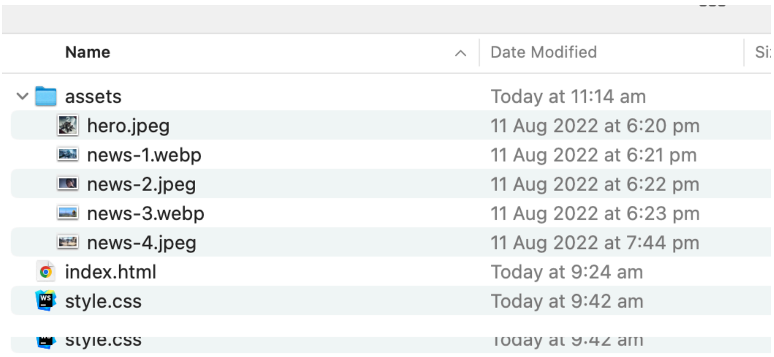
Task of our project

Whether you're just starting to write your very first page or already experienced, the first step should be to think about the content of the page and decompose the mockup into small pieces. Generally speaking, we can disassemble the page into multiple modules in a top-to-bottom order, gradually describe each module with HTML, and finally add styles with CSS.

You can either write all the HTML at once or implement one module, add styles to it, and move on to the next one. Here, we'll implement the first page in small chunks at a time to prevent learning too much at once.

Deconstructing the mockup

As seen from the mockup, the navigation part of it is a relatively independent component, so we can start from it. Let's do some preparatory work first, create a directory `week-1`, and create two files `index.html` and `style.css` in it. Finally, we create a subdirectory of `assets` in `week-1` for image resources.



Name	Date Modified	Si:
assets	Today at 11:14 am	
hero.jpeg	11 Aug 2022 at 6:20 pm	
news-1.webp	11 Aug 2022 at 6:21 pm	
news-2.jpeg	11 Aug 2022 at 6:22 pm	
news-3.webp	11 Aug 2022 at 6:23 pm	
news-4.jpeg	11 Aug 2022 at 7:44 pm	
index.html	Today at 9:24 am	
style.css	Today at 9:42 am	

Folder Structure

In `index.html` we need to set some basic HTML code, aka boilerplate HTML:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" \
/>
    <meta name="viewport" content="width=device-width, in\
itial-scale=1.0" />
    <link rel="stylesheet" href="style.css" />
    <title>Week 1</title>
  </head>
  <body>
    <!-- our code goes here -->
  </body>
</html>
```

If you're unsure of what's described here, don't worry, we'll barely touch this part. The `Week 1` defined in the `title` tag will appear in your browser tab, so give it a cool name if you like.

We'll be working in the `body` area throughout this chapter (including the entire book), all of our HTML code will be written inside it (almost, we may need to add external CSS links, but we will talk about it later).

Additionally, the statement `<link rel="stylesheet" href="style.css">` indicates that we wish to use an external file in this HTML document: `style.css`. Yes, this is where we define the styles, but this file doesn't have anything yet.

Now uou can start the Live Server in Code (by clicking the Go Live button on the status bar). Once the page is loaded in browser, type some text into `body`, and you should be able to see the browser page refreshed with the edits. By default, There will be a one second delay.

Implement the navigation bar

Let's start with the HTML of navigation bar. `HTML` is a very loose, inclusive language compared to other programming languages. Looseness means that there are many different ways to implement a component. For example, you can use `div` to represent a logical container on the page, and you can also use `article` or `section` to achieve the same purpose.

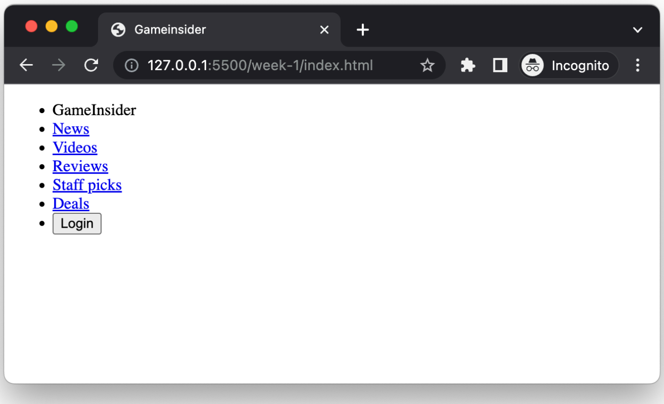
In practice, however, we need to use semantic tags, such as `h1`-`h7` for titles, `p` for paragraphs, and `article` to hold a relatively independent item and so on.

For the navigation bar, HTML has a `nav` tag available, and for the list content in `nav`, we can use an unordered list `ul` or an ordered list `ol`. Since `nav` is in the header of the page, we can use `header` as the container for the whole navigation.

```
<header>
  <nav>
    <ul class="nav-list">
      <li class="list-item">
        <div class="logo"><span class="highlight">Game</s\
pan>Insider</div>
      </li>
      <li class="list-item">
        <a href="#">News</a>
      </li>
      <li class="list-item">
        <a href="#">Videos</a>
      </li>
      <li class="list-item">
        <a href="#">Reviews</a>
```

```html
        </li>
        <li class="list-item">
          <a href="#">Staff picks</a>
        </li>
        <li class="list-item">
          <a href="#">Deals</a>
        </li>
        <li class="list-item">
          <button class="button-primary">Login</button>
        </li>
      </ul>
    </nav>
  </header>
```

There are seven items in the navigation bar. Five hyperlinks are in the middle, and the first one is an icon and the last one is a button. To help CSS to select specific elements, we need to add the attribute like `class= "xxx"` (and use `.xxx` to refer it) to some elements.

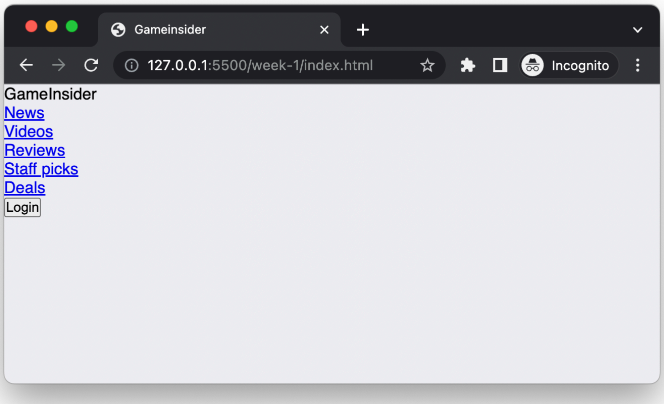


- GameInsider
- News
- Videos
- Reviews
- Staff picks
- Deals
- Login

Without Style

By default, browsers add a bullets to all `li`s, as well as some spacing to `ul` for easier reading. Browsers always assume that the page is usable without any CSS. We need to override these default styles via CSS:

```css
* {
  margin: 0;
  padding: 0;
}

body {
  font-size: 16px;
  font-family: "Open Sans", sans-serif;
}

li {
  list-style: none;
}
```



GameInsider
News
Videos
Reviews
Staff picks
Deals
Login

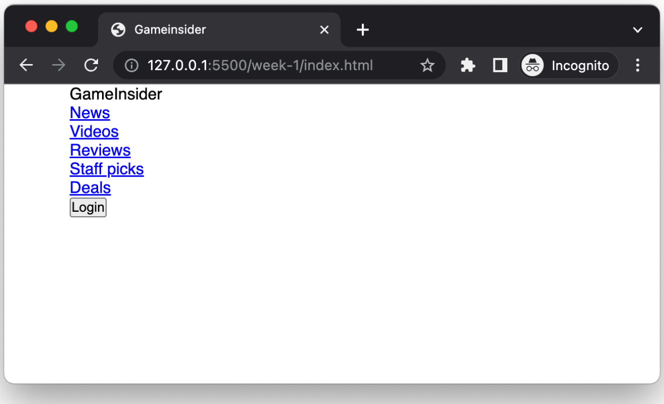
Reset Default Styles

Here, `*` is a special selector, which means to select all HTML elements on the page. And for all these selected elements, we apply two rules, `margin:0` to eliminate the outer space, that is, no outer padding; `padding:0` means eliminate padding, that is, there is no space between the element content and the border.

Similarly, we use the tag type `body` to select the main content area, set the font size to 16 pixels, and use the font `Open Sans` (if `Open Sans` is unavailable, the system default sans-serif font sans-serif is used) .

Additionally, we want the entire navbar to be centred on the page and have a width of 80% of the available area:

```
nav {
  width: 80%;
  margin: 0 auto;
}
```



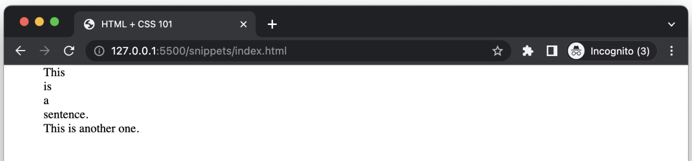
GameInsider
News
Videos
Reviews
Staff picks
Deals
Login

Center Aligned

Next, we want the seven elements in the navigation to be arranged horizontally, not vertically. Here we have to mention two different elements in HTML: `block-level` elements and `inline` elements (strictly speaking, there are more than two in HTML, but we will only describe the two most common ones here). Block-level elements occupy a single line by default, no matter how wide their actual content is. Inline elements keep the content on the same line.

```html
<div class="container">
  <p>This</p>
  <p>is</p>
  <p>a</p>
  <p>sentence.</p>
  <span>This</span>
  <span>is</span>
  <span>another</span>
  <span>one.</span>
</div>
```

For example, since `p` is a block-level element by default, it will be on its own line. And `span` will stay on the same line. Since most HTML elements are block-level elements, they render as many lines by default.



```
This
is
a
sentence.
This is another one.
```

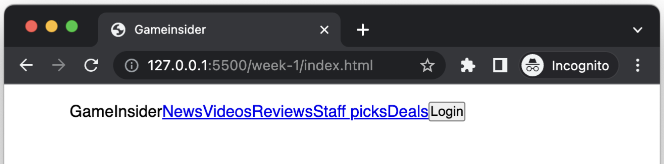
Block and Inline

We can use `flex` layout to override this default behaviour. `flex` one of the layout machenism in CSS. I'm not going to cover all `flex` properties here (because that is almost a book in itself), and we will only cover a few common ones here.

First, to use the `flex` layout, we need a container element, and then set the container's `display` to `flex`:

```css
.nav-list {
  display: flex;
}
```

Immediately, all elements became one line like a magic!



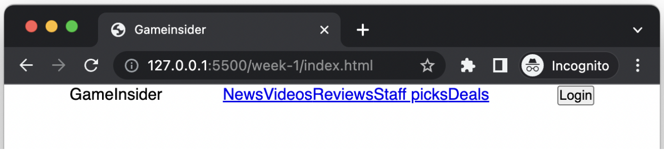
GameInsiderNewsVideosReviewsStaff picksDealsLogin

Flex Layout

Next, we make some spacing adjustments, making sure the `Login` button is on the right, and the logo is on the left, and the other links are relatively centred:

```css
.list-item:last-child {
  margin-left: auto;
}

.list-item:first-child {
  margin-right: 32px;
}
```

The `.list-item:last-child` selector here will select the last child node with the class `.list-item`, and similarly, `:first-child` will select the first child.



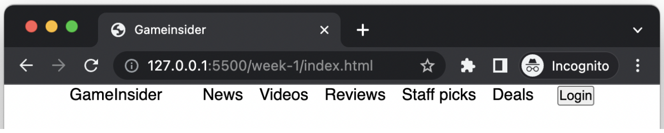
GameInsider NewsVideosReviewsStaff picksDeals Login

With Space in Between

For hyperlinks, browsers will add a lot of styles by default, such as the underline and bright font colours. According to the mockup, we need to override these styles as well:

```css
.list-item a {
  padding: 0 8px;
  text-decoration: none;
  color: black;
  white-space: nowrap;
}
```

The `.list-item a` selector means to look for the `a` (hyperlink) tag in the child nodes of `.list-item`. CSS selectors are parsed from right to left, this selector when interpreted will first look for all `a`, then see if there is `.list-item` in its ancestor node, and then decide whether to apply these CSS rules or not.



Anchors

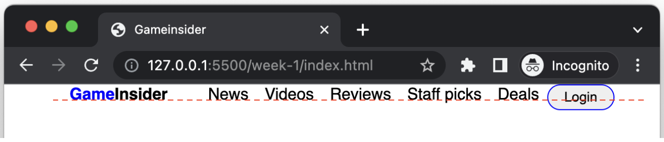
Next, let's customise the button, which has rounded corners and some white space between the text and its border. White space is very important in design. Enough white space can make reading easier and make the page more attracting.

```css
.button-primary {
  appearance: none;
  border: 1px solid blue;
  padding: 4px 16px;
  border-radius: 16px;
}
```

Usually, the logo will be an image element, but we simply use some texts to simulate it here. We need to use a thicker font and highlight the `Game` colour:

```css
.logo {
  font-weight: bold;
}

.highlight {
  color: blue;
}
```

In this way, the logo looks the same as the mockup.



Button

But if you look closely at the button, it doesn't align with other elements (like the red line in the image above). Alignment is another extremely important principle in design (we'll cover that in chapter three), and our brains prefer neat arrangements over disorder. In fact, horizontally aligning elements inside a container has historically been super difficult, but this problem is easily solved using `flex` layout.

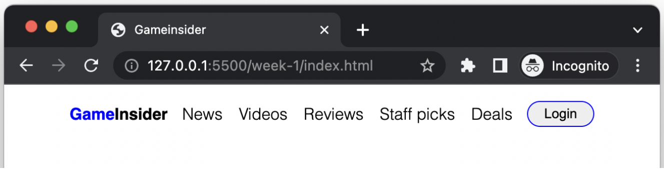
On a `flex` container, just use `align-items`, and set it as `center`:

```css
.nav-list {
  display: flex;
  align-items: center;
}
```

Also, let's add some extra whitespace to the navigation bar to give him more breathing room:

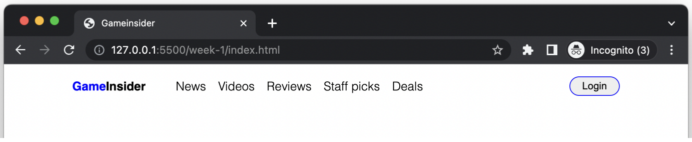
```css
.nav-list {
  display: flex;
  align-items: center;
  padding: 16px 0;
}
```

Here is how the "final" navigation bar looks like now:



Navigation List

We are now very close to the mockup, although it's not pixel prefect. Even better, using the `flex` layout, our navbar will still work well on wider screens, like the button will always be on the right side of the page, pretty cool right?



In Wider Screen

Implementing the Hero Banner

Next, we can look at the hero banner section. Almost everyone website has such a section that is spacial, usually with large fonts, eye-catching pictures, and a button (Call To Action) to guide users to action (like, click it).

According to the mockup, we can divide the entire area into left and right parts. The title and descriptive text are on the left, followed by a call-to-action button. The right side is a little more complicated, containing an image and some description and rating placed on top of the image.

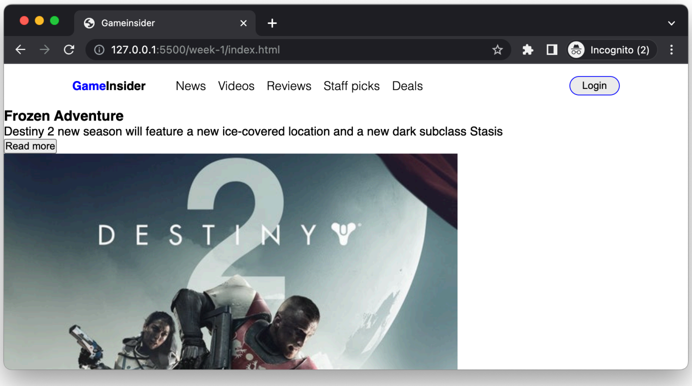
As mentioned above, let's write the HTML content first. It is worth noting that when writing HTML, we need to assume that the page is usable without any CSS. The title should be readable, hyperlinks is clickable and can navigate users to the new address they need, etc.

```html
<section class="hero-section">
  <section class="brief">
    <h1>Frozen Adventure</h1>
    <p>
      Destiny 2 new season will feature a new ice-covered\
 location and a new
      dark subclass Stasis
    </p>
    <button class="button-secondary">Read more</button>
  </section>

  <section class="media">
    <div class="game-cover">
      <img src="assets/hero.jpeg" alt="destiny 2" />
      <p class="title">Desitny 2 beyond light<span class=\
"rating">4.5</span></p>
    </div>
  </section>
</section>
```

Here you can use either a `div` or `section` as the container tag of the section. I usually prefer `section` as a container for content, and `div` as an auxiliary tag.

Also, don't use `div` unless necessary. Usually, though, the two are used interchangeably in many cases.

Before any styles, this part looks like this:



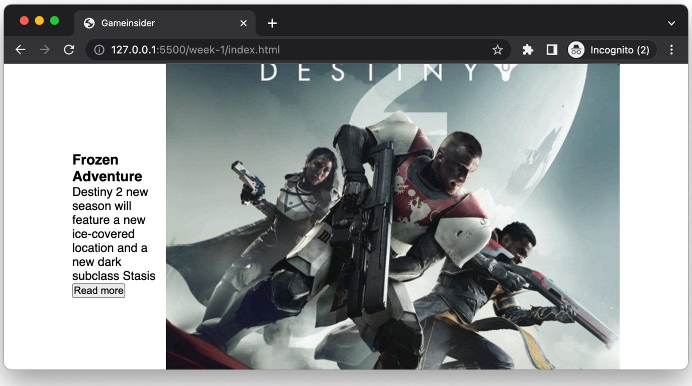
Frozen Adventure
Destiny 2 new season will feature a new ice-covered location and a new dark subclass Stasis
Read more

HTML of Hero Section

To make the two `block-level` elements `brief` and `media` line up horizontally, what method should we use? That's right, like the navbar, we can set the container `hero-section` to be a `flex` container:

```css
.hero-section {
  width: 80%;
  margin: 2rem auto;
  display: flex;
  align-items: center;
}
```

Well, the horizontal arrangement is fine, but the width of the two parts seems to be uneven. The image part occupies more space, and the text is pushed to the left. We need them be half and half distributed:



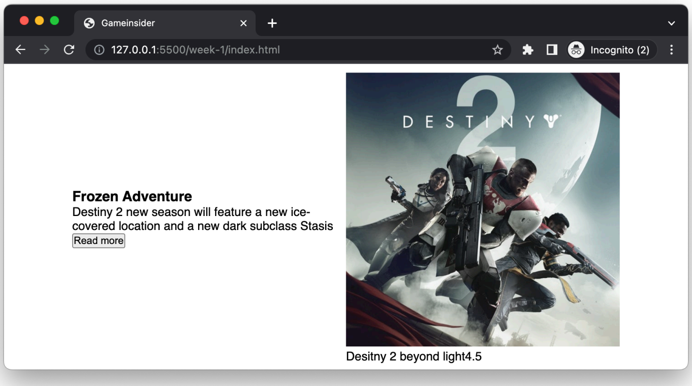
Frozen Adventure
Destiny 2 new season will feature a new ice-covered location and a new dark subclass Stasis

Flex Initial Attemp

At this time, we need to use the `flex:1` rule for both elements in the container and set the width of `100%` for the image instead of the default width so that the image will fill the available width of its container (the `.game-cover`):

```css
.hero-section > section {
  flex: 1;
}

.game-cover img {
  width: 100%;
}
```

Note there is a new CSS syntax here: `>`. This symbol represents the immediate child node of the selected `.hero-section`. Without this direct operator, `.hero-section section` would select all `sections` inside `.hero-section`, no matter how deep it is.



Frozen Adventure
Destiny 2 new season will feature a new ice-covered location and a new dark subclass Stasis

Desitny 2 beyond light4.5

Flex Evenly

The `flex:1` here needs some extra explanation too. This is a typical CSS abbreviation, and its full form is:

```css
flex-grow: 1;
flex-shrink: 1;
flex-basis: 0%;
```

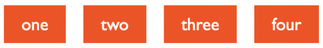
`flex-grow` is the growth factor, that is, in a flex container, how to allocate the remaining space to the element. It is only valid if all the elements themselves are smaller than the container size. `flex-shrink` refers to the shrink factor, that is, when the width of all flex elements exceeds the width of the container, in what proportion each element should be shrunk. `flex-basis` refers to the default size of flex elements.

Flex layout

We can illustrate the relationship between these attributes through a concrete example.

```html
<div class="container">
  <div class="box">one</div>
  <div class="box">two</div>
  <div class="box">three</div>
  <div class="box">four</div>
</div>
```

In the beginning, we set `container` as a flex container, and their child nodes `box` are automatically flex elements. But since the length of each word is not the same, the arrangement of the four boxes is like this:



Default settings

```css
.container {
  display: flex;
  align-items: center;
  gap: 1rem;
}

.box {
}
```

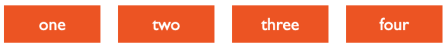
This is because by default the CSS for flex elements is set to `flex:0`, i.e.:

```css
flex-grow: 0;
flex-shrink: 1;
flex-basis: 0%;
```

We set the `flex-grow` of all divs with `box` class to 1, which means they have the same growth factor if the container has enough space:

```css
.box {
  flex-grow: 1;
  flex-shrink: 1;
  flex-basis: 0%;
}
```

At this point, they will fill the entire container and share the space evenly:

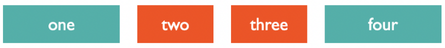


Distributed Evenly

If we set different growth factor for some elements:

```css
.box:nth-child(1) {
  background-color: lightseagreen;
  flex-grow: 2;
}

.box:nth-child(4) {
  flex-grow: 2;
  background-color: lightseagreen;
}
```

Then when the container space is sufficient, the first and fourth elements will grow faster than the other elements (as they have large `flex-grow` proportion).

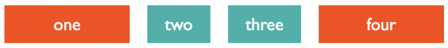


Grow faster

`flex-shrink` is quite the opposite of `flex-grow`. When the width of the flex container is less than the sum of the widths of all flex items, `flex-shrink` defines the ratio by which each element shrinks.

```css
.box {
  flex-grow: 1;
  flex-shrink: 1;
  flex-basis: 50%;
}

.box:nth-child(2) {
  background-color: lightseagreen;
  flex-shrink: 2;
}

.box:nth-child(3) {
  background-color: lightseagreen;
  flex-shrink: 2;
}
```

For example, in this example, we set `flex-basis` to 50%, that is, each flex element shrinks or stretches from 50% of the parent element. Obviously, because there are four elements in the container, the overall length exceeds the container, so `flex-shrink` takes effect, and the result is that the second and third elements shrink faster than the other two and are, therefore smaller:



Shrink Faster

Polishing the homepage banner

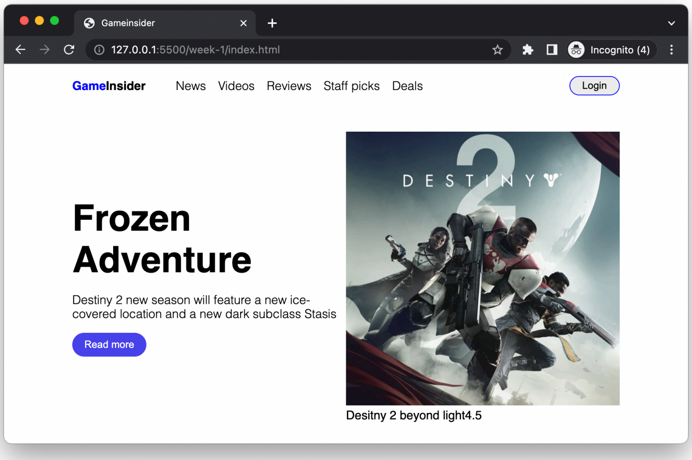
Well, our banner has a half and half layout now, and then we need to do some typography. Generally speaking, we can emphasise some elements through visual elements like different fonts, colours, etc., to make the subject more eye-catching and the page more balanced.

```
h1 {
  font-size: 48px;
  font-weight: bold;
}

.brief p {
  font-weight: lighter;
  margin: 16px 0;
}
```

For customising the button and the navigation bar, we used the colour picker to get the purple value from the mockup: #4A43EB.

```css
.button-secondary {
  appearance: none;
  padding: 8px 16px;
  border-radius: 16px;
  border: none;
  background-color: #4a43eb;
  color: white;
}
```

We use `appearance: none` here to reset the browser's default style and then add rounded corners, background colour and font colour to it.



Frozen Adventure

Destiny 2 new season will feature a new ice-covered location and a new dark subclass Stasis

Desitny 2 beyond light4.5

Left Side of Hero Area

For the right side of the banner, we noticed that rating content overlaid on the image in the mockup. This effect of **cascading** of content requires a little explanation.

When the browser renders HTML, it arranges the elements in order, and there is no overlay (after all, overlay means occlusion, that is, some content cannot be seen). If we need some elements to **to**

jump out of the arrangement process and free up the position that originally belonged to it, then we need to set the element's `position` to `absolute`.

This way, the element jumps out of the normal document flow and leaves the positioning to the developer. At this time, we can use `top`, `left`, `bottom` and `right` to control the absolute positioning of the element, the default coordinate origin of the upper left corner of the screen.

But most of the time, we don't want to take the upper left corner of the screen as the origin but a certain element. In CSS, we can set the `position` of an element to `relative` so that its child nodes will have their origin at the top left corner (instead of the screen top left corner).

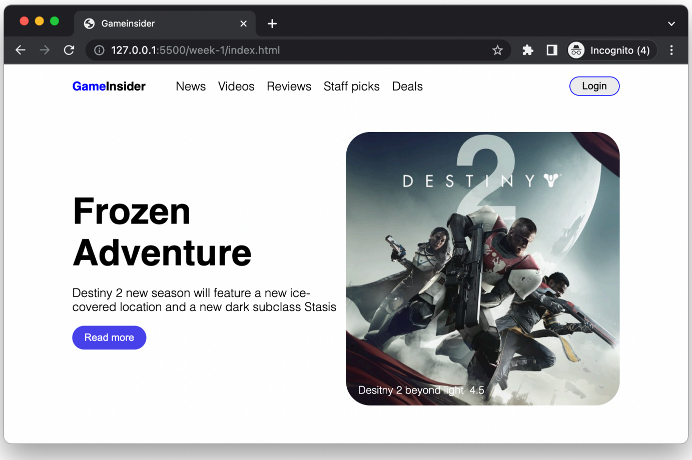
In this example, we want to use the `.game-over` as a baseline and then place the description text on top of it:

```css
.game-cover {
  position: relative;
}

.media .title {
  position: absolute;
  bottom: 16px;
  left: 16px;

  color: white;
  font-size: 14px;
  font-weight: lighter;
}

.media span.rating {
  margin-left: 8px;
}
```

Finally, we made a little adjustment to the corners of the image to make the implementation closer to the mockup:

```css
.game-cover img {
  border-radius: 32px;
}
```

This results in an image with rounded corners, and a circle with a radius of 32 pixels:


Gameinsider
127.0.0.1:5500/week-1/index.html
Incognito (4)
GameInsider
News
Videos
Reviews
Staff picks
Deals
Login
Frozen Adventure
Destiny 2 new season will feature a new ice-covered location and a new dark subclass Stasis
Read more
DESTINY 2
Desitny 2 beyond light 4.5

Final Touch

Congratulations, we have completed the second major part of the page. Here we learned some details of `flex` layout to achieve an even distribution of elements in a flex container, and we also learned to use absolute positioning (`position: absolute`) and relative positioning (`position: relative`) to make elements Layer up to make it more visually pleasing.

Recent News

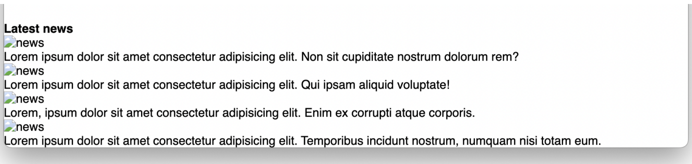
Next comes the last section of the page: the news list. What is the first step, again? That's right, write the HTML. When writing HTML, we can roughly analyse which elements need to be marked with `class`. You don't need to be perfect at this stage.

Because when using CSS selectors later, if we find some elements that need special attention, we can always go back and add them on the tag.

The news list is relatively straightforward, consisting of a title and four articles, each with an image and a text description.

```html
<section class="latest-news">
  <h4>Latest news</h4>
  <ul class="news-container">
    <li class="news">
      <article>
        <header>
          <img src="assets/news-1.webpx" alt="news" />
        </header>
        <p>
          Lorem ipsum dolor sit amet consectetur adipisic\
ing elit. Non sit
          cupiditate nostrum dolorum rem?
        </p>
      </article>
    </li>
    //...other items
  </ul>
</section>
```

Note the new tag we use here, `article`. `article` represents a standalone entity, such as a piece of news, a tweet, a blog, etc. When writing HTML, we also need to consider how to make the content semantic, meaning one can read the HTML code directly in the editor. Many web pages were filled with redundant and meaningless tags in the early days (and today as well). When we write a page, we need to consider the readability and maintainability from the very beginning.



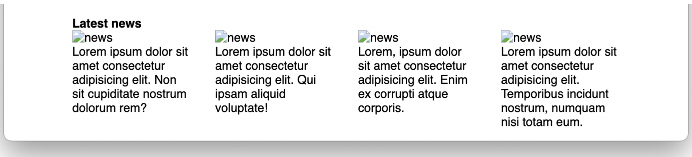
Latest news
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Non sit cupiditate nostrum dolorum rem?
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Qui ipsam aliquid voluptate!
news
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Enim ex corrupti atque corporis.
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus incidunt nostrum, numquam nisi totam eum.

HTML Done

Well, that doesn't look very appealing. Let's add some styles to it.

```css
.latest-news {
  width: 80%;
  margin: 1rem auto;
}

.news-container {
  display: flex;
  gap: 2rem;
}

.news {
  flex: 1;
}
```

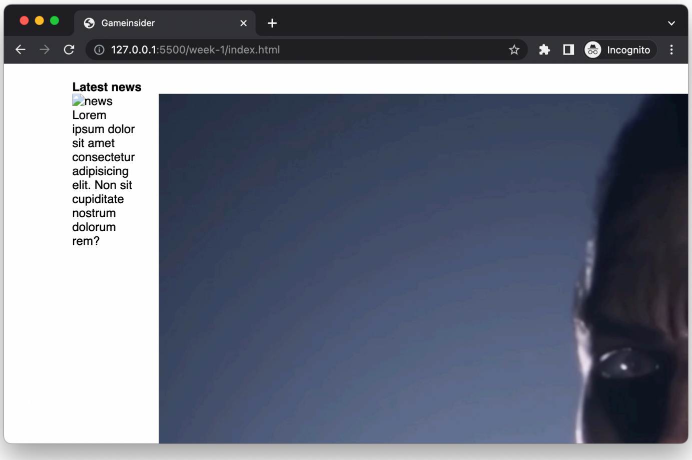
After adding `flex:1`, the news lines up correctly:



Latest news
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Non sit cupiditate nostrum dolorum rem?
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Qui ipsam aliquid voluptate!
news
Lorem, ipsum dolor sit amet consectetur adipisicing elit. Enim ex corrupti atque corporis.
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus incidunt nostrum, numquam nisi totam eum.

Without images

Now let's find some real game covers to make it looks authentic.



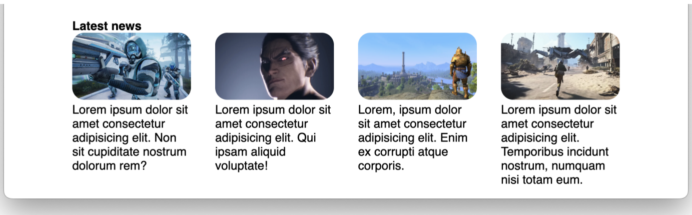
Latest news
news
Lorem ipsum dolor sit amet consectetur adipisicing elit. Non sit cupiditate nostrum dolorum rem?

With images

Oops, the whole page seems to be messed up! But don't worry. You only need to specify the size of the image as a percentage so that the image will automatically scale to its container size:

```css
.news img {
  width: 100%;
  border-radius: 1rem;
}
```

The picture scales correctly now:



Latest news

Lorem ipsum dolor sit amet consectetur adipisicing elit. Non sit cupiditate nostrum dolorum rem?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Qui ipsam aliquid voluptate!

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Enim ex corrupti atque corporis.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus incidunt nostrum, numquam nisi totam eum.

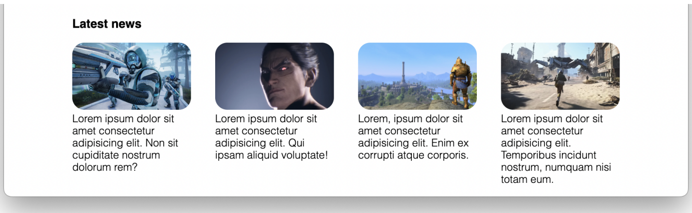
Resizing image

In addition, we noticed that the font was too thick and larger than in the mockup, also the spacing between the title and the image was too close.

Let's fix it up:

```css
.news p {
  font-size: 14px;
  font-weight: lighter;
}

h4 {
  margin: 1rem 0;
}
```

And now the final result is like:



Latest news

Lorem ipsum dolor sit amet consectetur adipisicing elit. Non sit cupiditate nostrum dolorum rem?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Qui ipsam aliquid voluptate!

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Enim ex corrupti atque corporis.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Temporibus incidunt nostrum, numquam nisi totam eum.

News Section

Looks much better, doesn't it? This completes most of the content of this page. There are some parts that I have deliberately omitted, such as the use of icons, the shadows of buttons. These topics will be covered in conjunction with design in the following chapters.

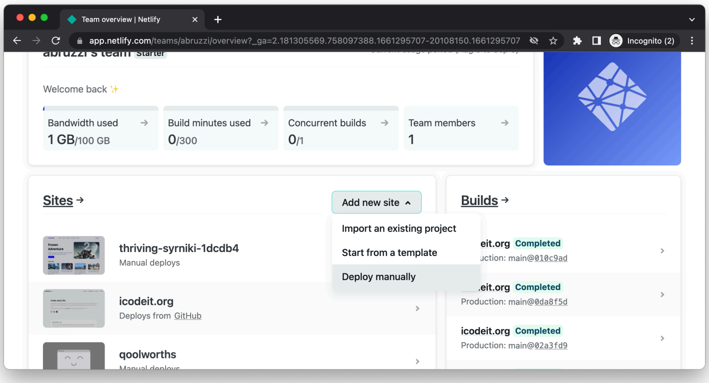
Publish your design

Congratulations, you have completed your first reproduction from the mockup. I believe you are very satisfied with your work, you really should. So let us share this joy with your friends!

How to publish your page on a public platform, so anyone anywhere in the world can access it? We can do this with tools like Netlify[5] or Surge[6].

If you don't want to install some extra software on your machine, then `Netlify` will be easier. After registering an account, you only need to drag and drop the local directory `week-1` to its website. After a few minutes `Netlify` will complete the deployment and provide you with a URL that you can share with others, pretty straightforward.

[5] https://www.netlify.com/
[6] https://surge.sh/



Drag and Drop in Netlify

Additionally, we'll cover a little bit about using `Surge` to publish your work here. First, you need to install `Node.js`[7]. After the installation, you will have the `npm` package manager in your command line.

Now, let's install the command line tool of `surge`. You need to execute it in the Terminal program:

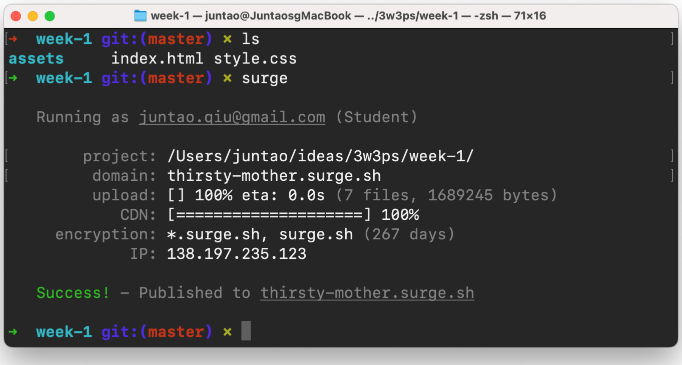
```
npm install --global surge
```

Then go to your working directory, which is `week-1`:

```
cd ~/tutorial/3w3ps/week-1
surge
```

At this time, `surge` will prompt you to log in or create an account, and then it will randomly assign a domain name to you and complete the deployment.

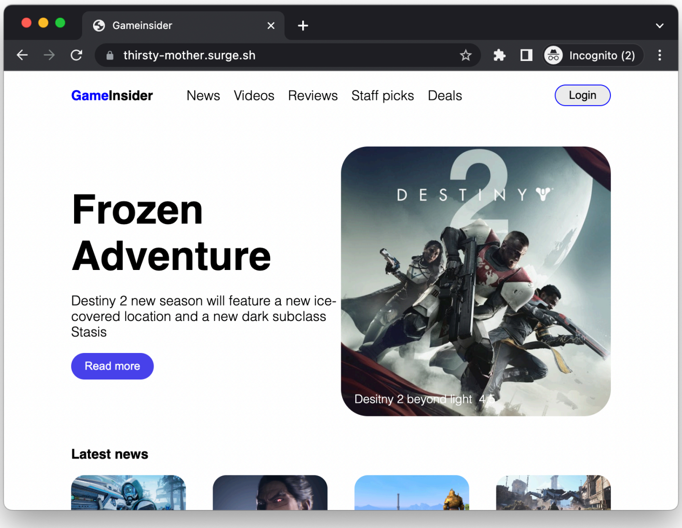
[7]https://nodejs.org/en/


week-1 — juntao@JuntaosgMacBook — ../3w3ps/week-1 — -zsh — 71×16

```
➜  week-1 git:(master) ✗ ls
assets      index.html style.css
➜  week-1 git:(master) ✗ surge

   Running as juntao.qiu@gmail.com (Student)

        project: /Users/juntao/ideas/3w3ps/week-1/
         domain: thirsty-mother.surge.sh
         upload: [] 100% eta: 0.0s (7 files, 1689245 bytes)
            CDN: [====================] 100%
     encryption: *.surge.sh, surge.sh (267 days)
             IP: 138.197.235.123

   Success! - Published to thirsty-mother.surge.sh

➜  week-1 git:(master) ✗ 
```

Run Surge Command Line Tool

You can now access the URL in public network or share it to others.



Our Site is Published

Summary

This week, we learned how to deconstruct a mockup and write corresponding HTML documents, learned how to use the `flex` layout to implement the navigation bar, and how to control the even distribution of elements, and learned how to use absolute positioning to elements that out of the document flow, as well as some other common typography techniques (like using `padding` and `margin`), etc.

I'm sure you can't wait to do something with your new achieved skills, so here are some interesting challenges.

Challenges

Challenge 1

Find a website you visit frequently, and use Chrome's DevTool to inspect the site's navigation bar. See what knowledge we discussed above and what knowledge is new to you. In addition, you can try to reimplement the navigation bar by the method we introduce here.

1. Write semantic HTML tags
2. Layout with `flex`
3. Adjust typography with margin and padding

Challenge 2

Inspect your favourite website's Landing Page, analyse its font and layout, then try to replicate it locally. You might want to pay attention to these:

1. Write semantic HTML tags

2. Use of fonts
3. The combination of background colour and foreground colour
4. Use `flex` layout
5. Adjust typography with margin and padding

In the next week, we will learn more about `flex` layout, in the vertical direction, using grids for more complex layouts, and layouting elements of different heights. We'll also cover how to use external fonts and icons to make pages more interesting.

Week 2: Social Media - New Instagram Interface

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Deconstruct the page

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Implementing Navigation Bar

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Using font icons

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Implementing Content Area

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Implement the Posts

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Grid Layout

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Card Implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

The icing on the cake - Shadows

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Challenges

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Challenge 1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Challenge 2

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Week 3: Mobile First - Creative Agency

Deconstruct the mockup

Emmet Plugin

Simulate Mobile Device in Chrome

CSS Variables

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Desktop version

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Media query

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Design principles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Characteristics of good design

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Alignment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Contrast

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Proximity

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Repetition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

How to organize content

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Hierarchy in typing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Colors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Summarize

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Challenges

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Challenge 1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

Challenge 2 and the Final one

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/3webdesignsin3weeks.

What’s next?