Internet Engineering

HTML 3

مهندسی اینترنت

Dr. Yaeghoobi

Ph.D. (Computer Science and Engineering), India

وزارت علوم تحقیقات و فناوری

دانشگاه علم و فرهنگ

Content

- HTML Iframe
- HTML Head
- HTML Forms
- HTML JavaScript
- HTML Layouts

HTML Iframe

HTML Iframe

- An iframe is used to display a web page within a web page.

```
<iframe src="URL"></iframe>
```

```
<iframe src="demo_iframe.htm" height="200" width="300"> </iframe>
```

```
<iframe src="demo_iframe.htm" style="height:200px; width:300px;">
</iframe>
```

Remove the Border

- **By default, an iframe has a border around it.**

```
<iframe src="demo_iframe.htm" style="border:none;"></iframe>
```

- Or Change the size and color of border

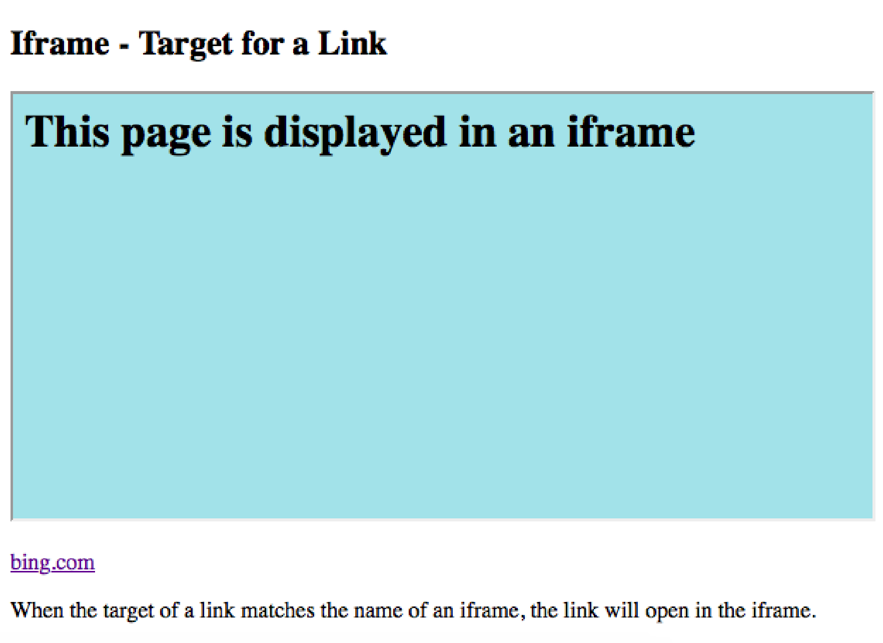
```
<iframe src="demo_iframe.htm" style="border:2px solid red;"></iframe>
```

Iframe - Target for a Link

- An iframe can be used as the target frame for a link.
- The target attribute of the link must refer to the name attribute of the iframe.

```
<iframe src="demo_iframe.htm" name="iframe_a"></iframe>

<p><a href="https://www.bing.com" target="iframe_a">
bing.com </a> </p>
```

Example

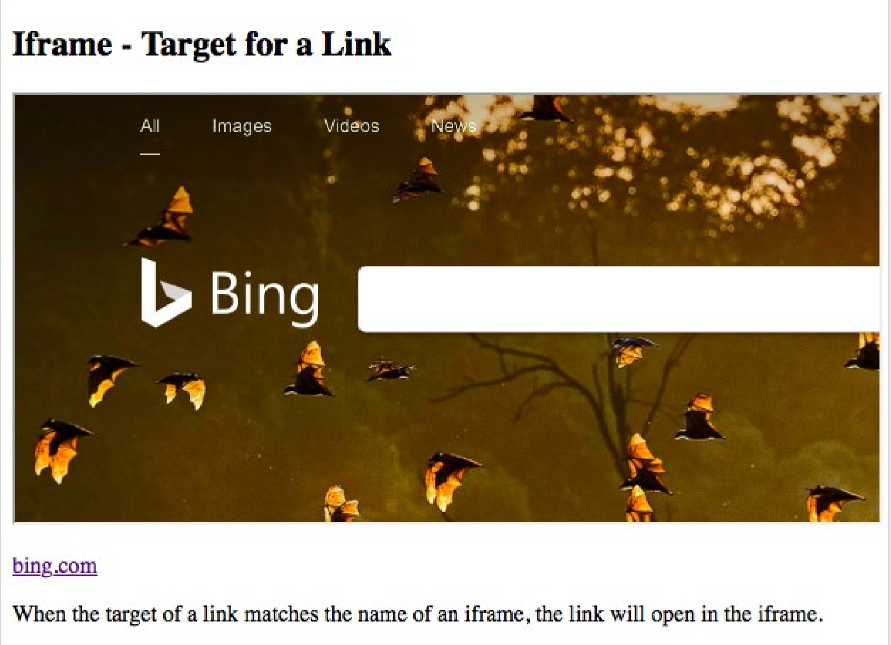


Iframe - Target for a Link

This page is displayed in an iframe

bing.com

When the target of a link matches the name of an iframe, the link will open in the iframe.



Iframe - Target for a Link

bing.com

When the target of a link matches the name of an iframe, the link will open in the iframe.

```
<!DOCTYPE html>
<html>
<body>

<h2>Iframe - Target for a Link</h2>

<iframe height="300px" width="100%" src="demo_iframe.htm” name="iframe_a">
</iframe>

<p><a href="https://www.bing.com" target="iframe_a">bing.com</a></p>

<p>When the target of a link matches the name of an iframe, the link will open in the
iframe.</p>

</body>
</html>
```

HTML Head

HTML Head

Tag	Description
<head>	Defines information about the document
<title>	Defines the title of a document
<base>	Defines a default address or a default target for all links on a page
<link>	Defines the relationship between a document and an external resource
<meta>	Defines metadata about an HTML document
<script>	Defines a client-side script
<style>	Defines style information for a document

Meta Tag

- HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.
- Meta tags allow you to **provide metadata** about your HTML pages.
- This can be **useful for search engines, browsers, and other applications trying to understand more about your page**.
- You can add metadata to your web pages by placing <meta> tags between the <head></head> tags.

Meta Tag …

Attribute	**Description**
name	Metadata name. The name specifies what aspect of metadata is being set.
content	Specifies the property's value.
charset	Specifies a character encoding declaration.
http-equiv	Used for http response message headers. For example http-equiv can be used to refresh the page or to set a cookie. Values include **content-type**, **expires**, **refresh** and **set-cookie**.

Meta Tag …

- Keywords:

```
<meta name="keywords" content="HTML, meta tags, metadata">
```

- Description:

```
<meta name="description" content="Contains info about meta tags">
```

- Author:

```
<meta name="author" content="Homer Simpson">
```

- Refresh the page every 10 seconds:

```
<meta http-equiv="refresh" content="10">
```

Setting The Viewport

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

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

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

HTML Forms

HTML Forms

- **Forms allow us to build more dynamic websites that allow our users to interact with it.**
- An HTML form is made up of any number of *form elements*.
- These elements enable the user to do things such as enter information or make a selection from a preset options.
- In HTML, a form is defined using the <form> </form> tags.

<input> Element

Type	Description
<input type= "text">	Defines a one-line text input field
<input type= "radio">	Defines a radio button (for selecting one of many choices)
<input type= "submit">	Defines a submit button (for submitting the form)

Text Input

- <input type="text"> defines a one-line input field for **text input.**

```
<form>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname">
</form>
```

Radio Button Input

- Radio buttons let a user select ONE of a limited number of choices.

```
<form>
  <input type="radio" name="gender" value="male" checked> Male<br>
  <input type="radio" name="gender" value="female"> Female<br>
  <input type="radio" name="gender" value="other"> Other
</form>
```

The Submit Button

- <input type="submit"> defines a button for **submitting** the form data to a **form-handler**.
- The form-handler is typically a server page with a script for processing input data.
- The form-handler is specified in the form's **action** attribute

The Submit Button

```
<html>
<body>

<form action="welcome.php" method="post"> or <form action="welcome_get.php”
method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

welcome.php

"welcome_post.php"

```
<html>
<body>

Welcome <?php echo $_POST["name"];?><br>
Your email address is: <?php echo
$_POST["email"]; ?>

</body>
</html>
```

"welcome_get.php"

```
<html>
<body>

Welcome <?php echo $_GET["name"];?><br>
Your email address is: <?php echo
$_GET["email"]; ?>

</body>
</html>
```

```
Welcome John
Your email address is john@example.com
```

When to Use GET?

- The default method when submitting form data is GET.
- The submitted form data will be **visible in the page address field**.
- Appends form-data into the URL in name/value pairs
- The length of a URL is limited (2048 characters)
- Never use GET to send sensitive data! (will be visible in the URL)
- Useful for form submissions where a user wants to bookmark the result
- GET is better for non-secure data, like query strings in Google

- روش پیش فرض هنگام ارسال داده های فرم GET است.
- داده های فرم ارسال شده در قسمت آدرس صفحه قابل مشاهده خواهند بود.
- داده-فرم را به صورت نام / مقدار به URLاضافه می کند
- طول یک URLمحدود است (۲۰۴۸ کاراکتر)
- هرگز از GETبرای ارسال داده های حساس استفاده نکنید! (در URLقابل مشاهده خواهد بود)
- برای ارسال فرم در جایی که کاربر می خواهد نتیجه را علامت گذاری کند مفید است
- GETبرای داده های غیر ایمن مانند رشته های پرس و جو در Googleبهتر است

When to Use POST?

- Always use POST if the form data contains sensitive or personal information.
- The POST method does not display the submitted form data in the page address field.
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked.

- اگر داده های فرم حاوی اطلاعات حساس یا شخصی هستند ، همیشه از POST استفاده کنید.
- روش POSTداده های فرم ارسال شده را در قسمت آدرس صفحه نمایش نمی دهد.
- POSTهیچ محدودیتی در اندازه ندارد و می تواند برای ارسال مقادیر زیادی از داده ها استفاده شود.
- ارسالی فرم با POSTقابل نشانه گذاری نیست.

Action Attribute

- The action attribute defines the action to be performed when the form is submitted.
- Normally, the form data is sent to a web page on the server when the user clicks on the submit button.
- If the action attribute is omitted, the action is set to the current page.

Target Attribute

- The target attribute specifies if the submitted result will open in a new browser tab, a frame, or in the current window.
- The default value is "_self" which means the form will be submitted in the current window.
- To make the form result open in a new browser tab, use the value "_blank"

```
<form action="/action_page.php" target="_blank">
```

- Other legal values are "_parent", "_top", or a name representing the name of an iframe.

Name Attribute

- Each input field must have a name attribute to be submitted.
- If the name attribute is omitted, the data of that input field will not be sent at all.

```
<input type="text" name="name"><br>
```

Grouping Form Data with <fieldset>

- The <fieldset> element is used to group related data in a form.
- The <legend> element defines a caption for the <fieldset> element.

```
<form action="/action_page.php">
 <fieldset>
  <legend>Personal information:</legend>
  First name:<br>
  <input type="text" name="firstname"><br>
  Last name:<br>
  <input type="text" name="lastname" ><br><br>
  <input type="submit" value="Submit">
 </fieldset>
</form>
```

<select> Element

- The <select> element defines a **drop-down list.**

```
<form action="/action_page.php">
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
<input type="submit">
</form>
```

Multiple Selections

```
<select name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

<textarea> Element

- The <textarea> element defines a multi-line input field (**a text area**).
- The rows attribute specifies the visible number of lines in a text area.
- The cols attribute specifies the visible width of a text area.

```
<form action="/action_page.php">
<textarea name="message" rows="10" cols="30">
</textarea>
<input type="submit">
</form>
```

<textarea> Element …

- Define the size of the text area by using CSS

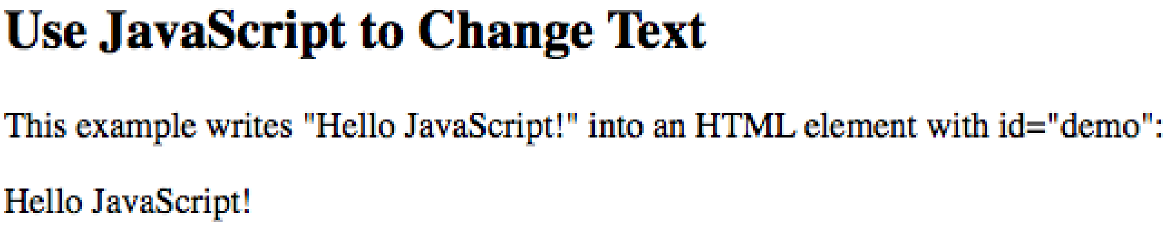
```
<textarea name="message" style="width:200px;
height:600px;"> </textarea>
```

HTML JavaScript

HTML JavaScript

- JavaScript makes HTML pages more dynamic and interactive.
- The <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains script statements, or it points to an external script file through the src attribute.
- Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
- To select an HTML element, JavaScript most often uses the document.getElementById() method.

```
<!DOCTYPE html>
<html>
<body>

<h2>Use JavaScript to Change Text</h2>
<p>This example writes "Hello JavaScript!" into an HTML element with
id="demo":</p>

<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>

</body>
</html>
```

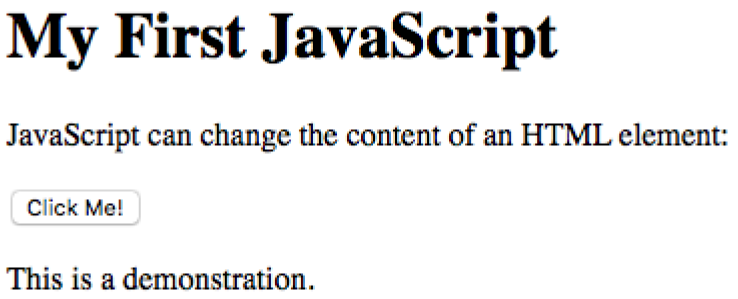


Use JavaScript to Change Text

This example writes "Hello JavaScript!" into an HTML element with id="demo":

Hello JavaScript!

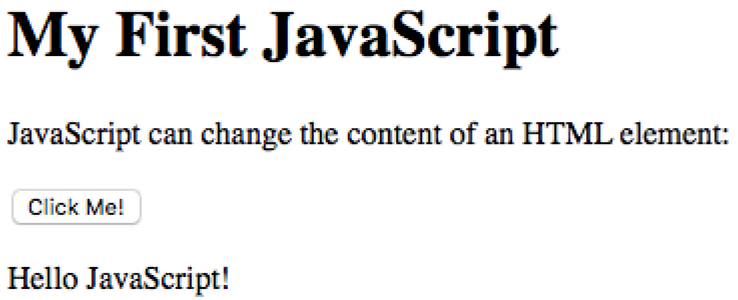
Example



My First JavaScript

JavaScript can change the content of an HTML element:

This is a demonstration.



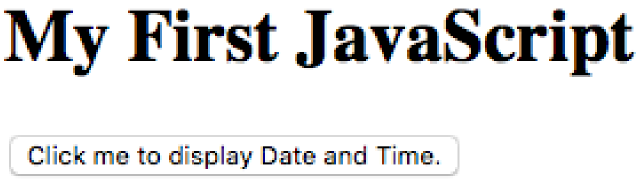
My First JavaScript

JavaScript can change the content of an HTML element:

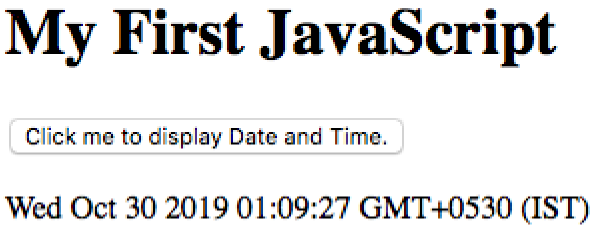
Hello JavaScript!

```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>

<p>JavaScript can change the content of an HTML element:</p>

<button type="button" onclick="myFunction()">Click Me!</button>

<p id="demo">This is a demonstration.</p>

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

</body>
</html>
```

Example (Date and Time)



My First JavaScript



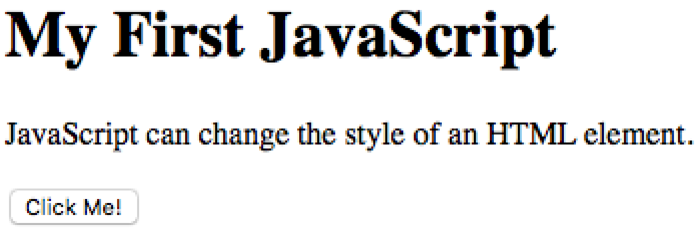
My First JavaScript

Wed Oct 30 2019 01:09:27 GMT+0530 (IST)

Example (Date and Time)

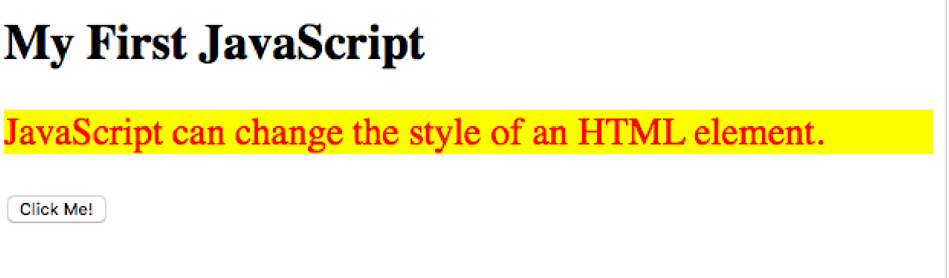
```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>

<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>

</body>
</html>
```

Example (Change HTML Styles)



My First JavaScript

JavaScript can change the style of an HTML element.



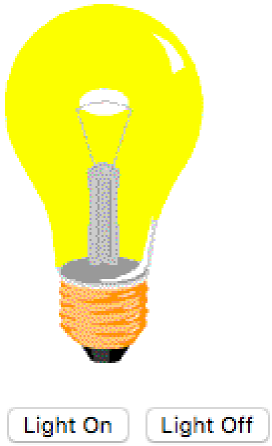
My First JavaScript

JavaScript can change the style of an HTML element.

```
<!DOCTYPE html>
<html>
<body>

<h1>My First JavaScript</h1>

<p id="demo">JavaScript can change the style of an HTML element.</p>

<script>
function myFunction() {
  document.getElementById("demo").style.fontSize = "25px";
  document.getElementById("demo").style.color = "red";
  document.getElementById("demo").style.backgroundColor = "yellow";
}
</script>

<button type="button" onclick="myFunction()">Click Me!</button>

</body>
</html>
```

Example (Change HTML Attributes)

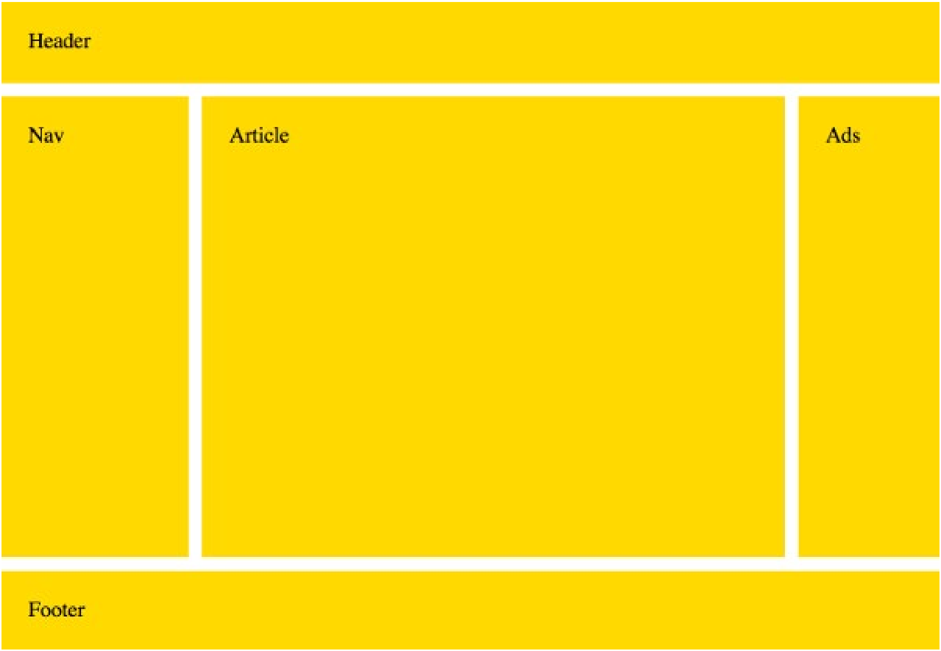
```
<!DOCTYPE html>
<html>
<body>
<script>
function light(sw) {
  var pic;
  if (sw == 0) {
    pic = "pic_bulboff.gif"
  } else {
    pic = "pic_bulbon.gif"
  }
  document.getElementById('myImage').src = pic;
}
</script>
<img id="myImage" src="pic_bulboff.gif"
width="100" height="180">

<p>
<button type="button" onclick="light(1)">Light
On</button>
<button type="button" onclick="light(0)">Light
Off</button>
</p>

</body>
</html>
```



HTML Layouts

HTML Layout

- Most modern websites and blogs consist of a header, footer, navbar, perhaps another sidebar, and the main content area.



HTML Layout

- HTML has a number of elements that can be used to define each of these areas. These include the main, header, footer, nav, aside and article elements.
- The div element is a generic block level element that can be used for grouping HTML elements.

```
<!DOCTYPE html>
<title> My Example </title>
<header> Header </header>
<article> Article </article>
<nav> Nav </nav>
<div> Ads </div>
<footer> Footer </footer>
```

CSS Grid Layout

- It works on a two-dimensional grid system, where you specify which elements go to which parts of the grid.
- the grid-template-areas property with a kind of "ASCII art" syntax to specify where each element goes.
- Then tie each element to each of those grid areas using the grid-area property.
- The rest of the code deals with sizing, gutters, general aesthetics, etc.

```
<!DOCTYPE html>
<title> Example</title>
<style>
body {
  display: grid;
  grid-template-areas:
    "header header header"
    "nav article ads"
    "footer footer footer";
  grid-template-rows: 60px 1fr 60px;
  grid-template-columns: 20% 1fr 15%;
  grid-gap: 10px;
  height: 100vh;
  margin: 0;
  }
#pageHeader {
  grid-area: header;
}
#pageFooter {
  grid-area: footer;
}
#mainArticle {
  grid-area: article;
  }
#mainNav {
  grid-area: nav;
  }
#siteAds {
  grid-area: ads;
  }
header, footer, article, nav, div {
  padding: 20px;
  background: gold;
}
</style>
<body>
  <header id="pageHeader">Header</header>
  <article id="mainArticle">Article</article>
  <nav id="mainNav">Nav</nav>
  <div id="siteAds">Ads</div>
  <footer id="pageFooter">Footer</footer>
</body>
```

Why Adding IDs?

- In this case we did actually **change the markup slightly by adding IDs to the elements**.
- By doing this, we ensure that the grid areas will be occupied only by the elements with the correct ID. If we didn't do this, we could run into major problems if we ever add another element of the same name (for example, another header element) to the page.

- در این حالت ما در واقع با اضافه کردن شناسه به عناصر، نشانه را کمی تغییر دادیم.
- با این کار اطمینان حاصل می کنیم که مناطق شبکه فقط توسط عناصر با شناسه صحیح اشغال می‌شوند. اگر این کار را نکردیم، اگر عنصر دیگری با همین نام (مثلاً عنصر هدر دیگر) را به صفحه اضافه کنیم، می توانیم به مشکلات اساسی برخورد کنیم.

fr (Fraction Unit)

- **Can mix fr values with fixed and percentage values.**
- The fr values will be divided between the space that’s left after what’s taken by the other values.

```
grid-template-columns: 1fr 1fr 40px 2fr;
grid-template-rows: 100px 200px 100px;
```

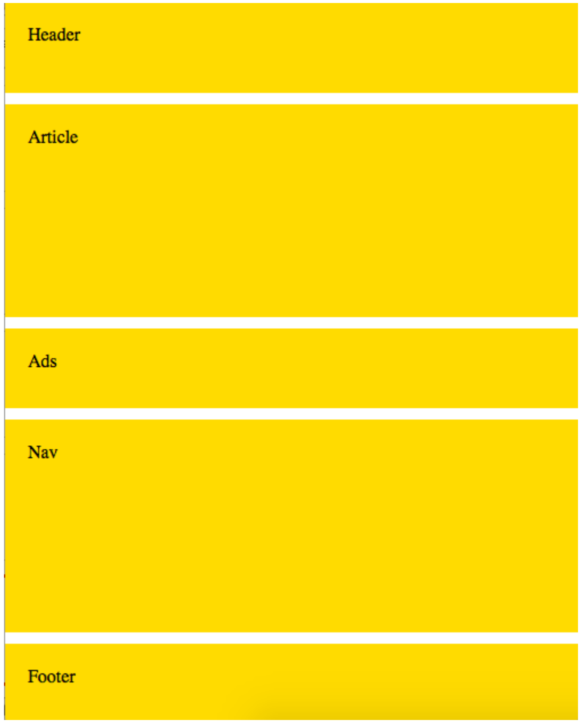
Responsive Layouts

- Responsive layouts adjust according to the screen size being used to view the website.
  - This means that your website will probably look different on a mobile phone vs a tablet vs a desktop computer.
  - The website adjusts itself to provide the best layout for the screen size.
- To do this, add a media query to test for the size of the screen. If it's smaller than a certain width, we show them the new layout.

Responsive Layouts …

```
@media all and (max-width: 575px) {
 body {
   grid-template-areas:
     "header"
     "article"
     "ads"
     "nav"
     "footer";
   grid-template-rows: 80px 1fr 70px 1fr 70px;
   grid-template-columns: 1fr;
 }
}
```

All the elements stacked upon each other
(in the order we specify)



Non-Grid Layout

- In order to support non-grid browsers, need to use other methods, such as CSS floats (with the float property) and/or flexbox (with flex and related properties).
- CSS was designed for presentation. CSS float property would be applied to block elements that needed to sit next to each other.
  - This enabled developers to continue with their three-column layouts while keeping the presentation separate to the content.
- However, flexbox is a one-dimensional system. It's perfect for lining up elements beside each other, or piled up on top of each other, but not both.

از توجه شما به این بخش سپاسگزارم