# Theory Assignment : CSS / CSS3

## What are the benefits of using CSS ?

Using CSS (Cascading Style Sheets) offers a range of benefits for web development:

- **Separation of Concerns:** CSS allows you to separate the structure (HTML) of a web page from its presentation (styling), making the code cleaner, easier to maintain, and more flexible.
- **Consistency:** By defining styles centrally in CSS, you can ensure consistent design across all pages of a website or web application. This consistency enhances the user experience and makes the site look more professional.
- **Faster Page Loading:** External CSS files can be cached by the browser, meaning that once a user has loaded your site, subsequent pages will load faster as they don't need to reload the CSS. This leads to faster overall page loading times.
- **Responsive Design:** CSS provides powerful tools for creating responsive layouts that adapt to different screen sizes and devices, ensuring that your website looks good on desktops, tablets, and smartphones.
- **Ease of Maintenance:** With CSS, you can make changes to the visual appearance of your entire website by simply editing one or a few CSS files. This makes updates and maintenance much quicker and more efficient compared to updating each HTML page individually.
- **Modularity and Reusability:** CSS allows you to define styles once and apply them to multiple elements across your site. This promotes code reusability and modularity, reducing redundancy and making it easier to scale your website.
- **Accessibility:** CSS provides features like color contrast adjustments, font resizing, and other accessibility enhancements that make it easier for users with disabilities to access and navigate your site.
- **Animations and Effects:** CSS allows you to create animations, transitions, and other visual effects without relying on JavaScript or Flash. This enhances the user experience and adds interactivity to your site.
- **Compatibility:** CSS is widely supported by modern web browsers, ensuring that your styles will display consistently across different platforms and devices.
- **Future-proofing:** Using CSS to style your web pages ensures compatibility with future web standards and technologies, making your website more future-proof and easier to adapt as the web evolves.

## What are the disadvantages of CSS?

While CSS offers many advantages for styling web pages, it also has some disadvantages:

- **Browser Compatibility:** Different browsers may interpret CSS rules differently, leading to inconsistencies in how a website looks across different browsers. This can require additional testing and sometimes workarounds to ensure cross-browser compatibility.
- **Learning Curve:** CSS can have a steep learning curve, especially for beginners. Understanding CSS selectors, specificity, and layout techniques can take time and practice to master.
- **Limited Layout Capabilities:** Although CSS has improved significantly over the years, achieving complex layouts can still be challenging compared to other layout methods like tables or flexbox. CSS Grid and Flexbox have made layout easier, but they may not cover all use cases.
- **Performance Impact:** Poorly optimized CSS, especially with large or complex stylesheets, can impact website performance. CSS file size, selector efficiency, and rendering speed can all affect page load times.
- **Global Scope:** CSS rules have a global scope by default, which can lead to unintended style conflicts and specificity issues, especially in larger projects or when working with third-party libraries.
- **Limited Dynamic Capabilities:** CSS is primarily designed for static styling and lacks advanced programming capabilities. While CSS animations and transitions exist, they have limitations compared to JavaScript-based animations.
- **Debugging Challenges:** Debugging CSS issues, especially in complex layouts or with browser-specific problems, can be time-consuming and require advanced troubleshooting skills.
- **Dependency on HTML Structure:** CSS is tightly coupled with HTML structure, so changes to the HTML structure can sometimes necessitate corresponding changes in the CSS, leading to maintenance overhead.
- **Accessibility Challenges:** While CSS can enhance accessibility, it can also introduce accessibility challenges if not used properly. For example, improperly applied styles can make content difficult to read or navigate for users with disabilities.
- **Limited Browser Support for New Features:** New CSS features may not be supported in older browsers, requiring fallbacks or polyfills to ensure consistent behaviour across different browser versions.

## What is the difference between CSS2 and CSS3?

CSS2 and CSS3 are different versions of the Cascading Style Sheets (CSS) language, each with its own set of features and capabilities. Here are the key differences between CSS2 and CSS3:

**Modules vs. Single Specification:**

- CSS2 was released as a single specification in 1998 and consisted of various modules.
- CSS3 is modularized, meaning it's divided into separate modules that can be updated and implemented independently. This modular approach allows for more frequent updates and easier adoption of new features.

**Selectors:**

- CSS2 introduced basic selectors like element selectors, class selectors, ID selectors, descendant selectors, etc.
- CSS3 expanded the selector capabilities with more advanced selectors like attribute selectors, pseudo-classes (:nth-child, :not, etc.), and pseudo-elements (::before, ::after, etc.), providing more precise targeting of elements.

**Box Model:**

- The box model in CSS2 was relatively simple and did not include features like border-radius, box-shadow, or text-shadow.
- CSS3 introduced properties like border-radius for rounded corners, box-shadow for drop shadows, and text-shadow for text effects, allowing for more sophisticated visual styling.

**Layout:**

- CSS2 relied mainly on floats, tables, and positioning for layout.
- CSS3 introduced advanced layout modules such as Flexbox and Grid, which provide more powerful and flexible ways to structure and position elements on a webpage.

**Media Queries:**

- CSS2 had limited support for media queries, primarily targeting different types of devices (screen, print, handheld, etc.).
- CSS3 introduced more robust media queries, allowing developers to apply styles based on various factors like screen size, resolution, orientation, and more, enabling responsive web design.

**Animations and Transitions:**

- CSS2 had no built-in support for animations or transitions.
- CSS3 introduced the @keyframes rule for defining animations and properties like transition for creating smooth transitions between different states of an element.

**Typography:**

- CSS2 provided basic typographic styling capabilities.
- CSS3 introduced features like web fonts (via @font-face), text shadows, and text overflow properties, allowing for more creative and expressive typography.

**Backgrounds and Borders:**

- CSS2 provided basic background and border properties.
- CSS3 introduced more options for backgrounds and borders, including multiple background images, background gradients, border images, and rounded borders with border-radius.

Overall, CSS3 introduced a wide range of new features and enhancements that greatly expanded the capabilities of CSS, allowing developers to create more dynamic, responsive, and visually appealing websites.

## Name a few CSS style components.

- Bootstrap
- Materialize CSS
- Tailwind CSS
- Foundation
- Bulma
- Semantic UI
- Pure CSS
- UI-Kit

## What do you understand by CSS opacity?

CSS opacity refers to the level of transparency of an element on a webpage. It determines how much an element allows light to pass through it.

In CSS, opacity is controlled using the opacity property, which accepts values between 0 and 1, where:

0 represents completely transparent (invisible)

1 represents fully opaque (not transparent)

For example:

```
.element {
   opacity: 0.5; /* 50% transparent */
}
```

This CSS rule sets the opacity of the element to 50%, meaning it will be semi-transparent.

It's important to note that CSS opacity affects the entire element and its contents, including any child elements. If you only want to make the background of an element transparent while keeping its content opaque, you can use RGBA colors or CSS background properties like background-color: rgba() to achieve the desired effect.

## How can the background color of an element be changed?

The background color of an element can be changed in CSS using the background-color property. Here's how you can use it:

```
.element {
    background-color: red; /* Change to any color value */
}
```

You can replace "red" with any valid color value, such as:

Hexadecimal: #RRGGBB (e.g., #FF0000 for red)

RGB: rgb(R, G, B) (e.g., rgb(255, 0, 0) for red)

RGBA: rgba(R, G, B, A) (e.g., rgba(255, 0, 0, 0.5) for semi-transparent red)

HSL: hsl(H, S%, L%) (e.g., hsl(0, 100%, 50%) for red)

HSLA: hsla(H, S%, L%, A) (e.g., hsla(0, 100%, 50%, 0.5) for semi-transparent red)

You can also use color names like "red", "blue", "green", etc.

For example:

```
.element {
    background-color: #00FF00; /* Green using hexadecimal */
}

.another-element {
    background-color: rgba(0, 0, 255, 0.5); /* Semi-transparent blue using RGBA */
}
```

These CSS rules will set the background color of the .element to green and the background color of the .another-element to semi-transparent blue.

## How can image repetition of the backup be controlled?

In CSS, you can control how an image is repeated as a background using the background-repeat property. Here are the possible values:

repeat: The background image will be repeated both horizontally and vertically to fill the element.

repeat-x: The background image will be repeated only horizontally.

repeat-y: The background image will be repeated only vertically.

no-repeat: The background image will not be repeated. It will be displayed only once.

Here's how you can use it:

```
.element {
    background-image: url('background-image.jpg');
    background-repeat: repeat; /* or repeat-x, repeat-y, or no-repeat */
}
```

For example:

```
.element {
    background-image: url('background-image.jpg');
    background-repeat: repeat-x; /* Only repeat horizontally */
}
```

This CSS rule sets the background image of the element to 'background-image.jpg' and specifies that it should only repeat horizontally.

If you don't want the image to repeat at all, you can use background-repeat: no-repeat;

## What is the use of the background-position property?

The background-position property in CSS is used to specify the starting position of a background image within its containing element. It determines where the background image is placed relative to the element's padding box.

The property takes two values:

Horizontal position: Specifies the horizontal position of the background image. This value can be specified using keywords (left, center, right) or length units (px, %, em, etc.).

Vertical position: Specifies the vertical position of the background image. This value can also be specified using keywords (top, center, bottom) or length units.

For example:

```
.element {
    background-image: url('background-image.jpg');
    background-position: center top;
}
```

In this example, the background image will be positioned in the center horizontally and at the top vertically within the .element.

You can also use keywords like left, right, center, top, and bottom individually, or specify both horizontal and vertical positions together.

```
.element {
    background-image: url('background-image.jpg');
    background-position: left 20px top 50px; /* Horizontal position 20px from the left, Vertical position 50px from the top */
}
```

This CSS rule sets the background image's starting position 20 pixels from the left and 50 pixels from the top within the .element.

## Which property controls the image scroll in the background?

The property that controls whether or not an image scrolls with the content of a webpage is background-attachment.

background-attachment: This property determines whether the background image scrolls with the rest of the content, or remains fixed in place while the content scrolls over it.

There are two possible values for background-attachment:

scroll: The background image scrolls along with the content. This is the default behavior.

fixed: The background image remains fixed in place while the content scrolls over it.

Here's an example:

```
.element {
    background-image: url('background-image.jpg');
    background-attachment: fixed;
}
```

In this example, the background image of the .element will remain fixed in place while the content inside .element scrolls over it.

If you want the background image to scroll along with the content, you can use the default value (scroll) or explicitly specify it:

```
.element {
    background-image: url('background-image.jpg');
    background-attachment: scroll; /* This is the default behaviour */
}
```

This ensures that the background image scrolls with the content as the user scrolls down the page.

## Why should background and color be used as separate properties?

Using background and color as separate properties provides more flexibility and control over the styling of elements. Here's why it's beneficial to use them separately:

- **Modularity:** Separating background and color properties allows for more modular CSS code. You can change the background or text color of an element independently without affecting other properties.
- **Clarity and Readability:** Using separate properties makes your CSS code easier to read and understand. It's clear at a glance which property is responsible for setting the background and which one is for setting the text color.
- **Override Specificity:** Separating background and color makes it easier to override specific styles when needed. If you set both background color and text color using the background property, it becomes more difficult to override one of them without affecting the other.
- **Accessibility:** Separating color from background ensures that both foreground and background colors can be adjusted independently to maintain good contrast, which is crucial for accessibility.
- **Ease of Maintenance:** When background and text color are set separately, making changes to one property doesn't require touching the other. This reduces the risk of unintentional side effects and makes maintenance easier.
- **Consistency:** Separating background and color encourages consistent styling practices across your codebase. It's a common convention in CSS, so other developers working on the project will find it familiar and predictable.

In summary, using background and color as separate properties promotes better code organization, readability, maintainability, and flexibility in styling your web elements.

## How to center block elements using CSS1?

In CSS1, which is the earliest version of CSS, there is no direct way to center block elements horizontally or vertically. However, you can achieve horizontal centering of block elements using a combination of properties:

Auto Margins: You can use auto margins to horizontally center block-level elements within their containing block. This technique works by setting both left and right margins to auto.

Here's how you can do it:

```
.element {
    width: 200px; /* Set the width of the element */
    margin-left: auto;
    margin-right: auto;
}
```

This will horizontally center the block-level element with the class .element within its containing block.

Text Alignment (For Inline Blocks): If the block element is displayed as an inline block, you can use text alignment to center it horizontally within its containing block.

```
.element {
    display: inline-block; /* Display the element as an inline block */
    text-align: center; /* Center the content horizontally */
}
```

However, this technique only works for inline or inline-block elements, not for block-level elements.

Fixed Width: If you know the exact width of the block element, you can set its left and right margins to auto, and the element will be centered horizontally.

```
.element {
    width: 200px; /* Set the width of the element */
    margin-left: auto;
    margin-right: auto;
}
```

These methods provide basic horizontal centering for block elements in CSS1. For vertical centering or more advanced layout techniques, you would typically need to use CSS2 or later versions.

## How to maintain the CSS specifications?

Maintaining CSS specifications involves staying updated with the latest changes, understanding how different features and modules work, and ensuring that your CSS code adheres to the current standards. Here are some steps you can take to maintain CSS specifications:

- Read Official Documentation: Regularly review the official documentation provided by the World Wide Web Consortium (W3C) for CSS. This includes the CSS specifications and any updates or drafts.
- Follow W3C Updates: Subscribe to W3C newsletters or follow their announcements to stay informed about new developments, updates, and changes to CSS specifications.
- Stay Updated with Browser Support: Keep track of browser updates and their support for CSS features. Websites like caniuse.com provide information on browser compatibility for different CSS properties and features.
- Use Vendor Prefixes Wisely: Understand which CSS properties require vendor prefixes and when to use them. Keep track of vendor prefix usage guidelines and updates from browser vendors to ensure compatibility.
- Utilize Modern Tools: Use tools like CSS preprocessors (e.g., Sass, Less) and postprocessors (e.g., Autoprefixer) to streamline your CSS workflow and ensure cross-browser compatibility.
- Test in Multiple Browsers: Regularly test your CSS code in different browsers to ensure consistency and compatibility. Use browser developer tools to debug and identify any issues.
- Follow Best Practices: Adhere to best practices for writing CSS, including using meaningful class names, organizing stylesheets, avoiding excessive specificity, and optimizing performance.
- Consider Accessibility: Keep accessibility in mind when writing CSS, ensuring that your styles are usable and accessible to all users, including those with disabilities.
- Stay Informed about CSS Frameworks and Libraries: Keep an eye on popular CSS frameworks and libraries and understand how they implement and maintain CSS specifications. This can provide insights into best practices and new techniques.
- Engage with the Community: Participate in online forums, communities, and social media platforms related to web development and CSS. Engage in discussions, ask questions, and share knowledge with others to stay informed and up-to-date with the latest trends and practices.
- By following these steps, you can effectively maintain CSS specifications and ensure that your CSS code remains current, compatible, and optimized for modern web development.

## What are the ways to integrate CSS as a web page?

There are several ways to integrate CSS into a web page. Here are the most common methods:

- **External CSS:**

Create a separate CSS file with the .css extension.

Link the external CSS file in the <head> section of your HTML document using the <link> element.

```
<head>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

This method is recommended for larger projects as it keeps the HTML and CSS separate, making it easier to manage and maintain.

- **Internal CSS:**

Write CSS directly within the <style> element in the <head> section of your HTML document.

```
<head>
  <style>
    /* CSS rules */
    body {
      background-color: lightblue;
    }
  </style>
</head>
```

This method is useful for smaller projects or when you want to apply styles specific to one page only.

- **Inline CSS:**

Apply styles directly to individual HTML elements using the style attribute.

```
<div style="background-color: lightblue; color: white;">Content</div>
```

This method is suitable for making quick style changes to specific elements, but it can clutter your HTML and make maintenance difficult.

- **CSS Preprocessors:**

Use CSS preprocessors like Sass or Less to write CSS in a more efficient and organized way, and then compile it into regular CSS.

Include the compiled CSS file in your HTML using one of the methods mentioned above.

CSS Frameworks:

Use CSS frameworks like Bootstrap, Foundation, or Bulma, which provide pre-written CSS styles and components.

Link the framework's CSS file in your HTML document to utilize its styling.

```
<head>
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
```

Then, use the classes provided by the framework to style your HTML elements.

Each of these methods has its advantages and use cases, so choose the one that best fits your project requirements and preferences.

## What is embedded style sheets?

Embedded stylesheets, also known as internal stylesheets, are CSS styles that are written directly within an HTML document using the <style> element.

Here's how you can use embedded stylesheets in HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Embedded Stylesheet Example</title>
    <style>
        /* Embedded CSS styles */
        body {
            font-family: Arial, sans-serif;
        }

        h1 {
            color: blue;
        }

        p {
            font-size: 16px;
            line-height: 1.5;
        }
    </style>
</head>
<body>
    <h1>This is a heading</h1>
```

```
    <p>This is a paragraph with some text.</p>
</body>
</html>
```

In this example, the CSS styles are defined within the <style> element in the <head> section of the HTML document. These styles will be applied to the corresponding HTML elements within the document.

Embedded stylesheets are useful for small projects or when you want to apply styles that are specific to a particular HTML document. However, for larger projects, it's generally recommended to use external stylesheets to keep the CSS separate from the HTML for better organization and maintainability.

## What are the external style sheets?

External style sheets are separate CSS files that contain styles for web pages. These CSS files are linked to HTML documents using the <link> element in the <head> section of the HTML document.

Here's how external style sheets work:

Create the CSS File:

Create a new file with a .css extension (e.g., styles.css).

Write your CSS styles within this file.

```
/* styles.css */
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}

h1 {
    color: blue;
}

p {
    font-size: 16px;
    line-height: 1.5;
}
```

Link the CSS File to HTML:

In the <head> section of your HTML document, add a <link> element to link the external CSS file.

```
<head>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

Replace "styles.css" with the path to your CSS file.

Apply Styles:

Once linked, the styles defined in the external CSS file will be applied to the HTML elements throughout the document.

External style sheets offer several advantages:

**Modularity:** CSS styles are separated from HTML, making it easier to manage and maintain.

**Reusability:** Styles defined in an external CSS file can be applied across multiple HTML documents, ensuring consistency.

**Efficiency:** The browser can cache external CSS files, leading to faster page loading times for subsequent visits.

**Collaboration:** Multiple developers can work on CSS and HTML files simultaneously without interfering with each other's work.

Using external style sheets is a best practice for most web projects, especially larger ones, as it promotes better organization, scalability, and maintainability of the codebase.

## What are the advantages and disadvantages of using external style sheets?

Using external style sheets offers several advantages and disadvantages:

**Advantages:**

**Modularity:** External style sheets allow you to separate your CSS code from your HTML, promoting cleaner, more organized code. This separation makes it easier to maintain and update your styles, as changes can be made in one central location and applied to multiple HTML documents.

**Reusability:** Styles defined in an external CSS file can be reused across multiple HTML documents, ensuring consistency in design and layout throughout your website. This saves time and effort by eliminating the need to duplicate styles in each HTML file.

**Efficiency:** External style sheets can be cached by the browser, reducing page load times for subsequent visits to your website. Once the CSS file is cached, it doesn't need to be re-downloaded, resulting in faster loading times and a better user experience.

**Accessibility:** External style sheets can improve accessibility by allowing you to apply consistent styling and ensure proper contrast, font sizes, and other design considerations across your website.

**Easy Collaboration:** External style sheets make it easier for multiple developers to work on the same project simultaneously. Each developer can work on different parts of the CSS file without interfering with each other's work.

**Disadvantages:**

**Additional HTTP Request:** Each external CSS file requires an additional HTTP request, which can slightly increase page load times, especially if there are many separate CSS files to download.

**Potential Overhead:** If your website uses multiple external CSS files, the cumulative size of these files can add up, leading to increased download times for users, particularly on slower connections.

**Dependency:** External style sheets create a dependency between the HTML and CSS files. If the CSS file fails to load or is deleted, the styling of the HTML document may be compromised.

**Difficulty with Inline Styling:** If inline styles (styles applied directly within HTML elements) are required for specific cases, managing and maintaining consistency can become more challenging when using external style sheets.

**Cascading Issues:** The cascade and specificity of CSS rules can sometimes lead to unexpected behavior, especially when dealing with large or complex style sheets. Proper organization and planning are necessary to avoid these issues.

Overall, the advantages of using external style sheets, such as modularity, reusability, and efficiency, often outweigh the disadvantages, making them a preferred choice for most web projects.

## What is the meaning of the CSS selector?

A CSS selector is a pattern used to select and target specific HTML elements on a web page to apply styles or perform other actions. CSS selectors allow you to define rules that determine which elements in the HTML document should be styled in a certain way.

Here's a breakdown of the parts of a CSS selector:

Element Selector: Selects HTML elements based on their element type.

```
p {
    color: blue;
}
```

This rule selects all <p> elements and sets their text color to blue.

Class Selector: Selects HTML elements based on their class attribute.

```
.button {
    background-color: green;
}
```

This rule selects all elements with the class "button" and sets their background color to green.

ID Selector: Selects a single HTML element based on its ID attribute.

```
#header {
    font-size: 24px;
}
```

This rule selects the element with the ID "header" and sets its font size to 24 pixels.

Attribute Selector: Selects HTML elements based on their attributes.

```
input[type="text"] {
    border: 1px solid black;
}
```

This rule selects all <input> elements with the type "text" and sets their border to 1px solid black.

Pseudo-classes and Pseudo-elements: Selects elements based on their state or position within the document.

```
a:hover {
    color: red;
}

p::first-line {
    font-weight: bold;
}
```

The :hover pseudo-class selects links when they are being hovered over and changes their color to red.

The ::first-line pseudo-element selects the first line of a paragraph and makes it bold.

Combination and Grouping: You can combine multiple selectors to create more specific rules, or group them to apply the same styles to multiple selectors.

```
h1, h2 {
    color: blue;
}

.container p {
    margin-bottom: 20px;
}
```

The first rule applies the same color to both <h1> and <h2> elements.

The second rule applies margin-bottom to <p> elements inside elements with the class "container".

## What are the media types allowed by CSS?

CSS supports various media types, which allow you to apply different styles to different types of devices or media. Here are the media types allowed by CSS:

**all:** Applies to all media types.

```
@media all {
  /* styles here apply to all media types */
}
```

**print:** Applies when the document is printed.

```
@media print {
  /* styles here apply when the document is printed */
}
```

**screen:** Applies to computer screens, tablets, smart-phones, etc. (default if the media type is not specified).

```
@media screen {
  /* styles here apply to screen media */
}
```

**speech:** Applies to speech synthesizers.

```
@media speech {
  /* styles here apply to speech media */
}
```

**tty:** Applies to media using a fixed-pitch character grid, such as teletypes or terminals.

```
@media tty {
  /* styles here apply to tty media */
}
```

**tv:** Applies to television-type devices.

```
@media tv {
  /* styles here apply to TV media */
}
```

**projection:** Applies to projectors.

```
@media projection {
   /* styles here apply to projection media */
}
```

**embossed:** Applies to paged Braille printers.

```
@media embossed {
   /* styles here apply to embossed media */
}
```

These media types allow you to create styles tailored to different output devices or media, ensuring that your web pages look and function correctly across various platforms and environments. You can also combine media types with other features like media queries to create responsive designs that adapt to different screen sizes and devices.

## What is the rule set?

In CSS, a rule set is a combination of a selector and a declaration block. It defines how HTML elements selected by the selector should be styled.

Here's the general syntax of a CSS rule set:

```
selector {
    property1: value1;
    property2: value2;
    /* more properties and values */
}
```

Selector: Specifies the HTML elements to which the styles should be applied. It can be an element selector, class selector, ID selector, or other types of selectors.

Declaration Block: Consists of one or more property-value pairs enclosed within curly braces {}. Each property-value pair defines a style rule for the selected elements.

For example:

```
h1 {
    color: blue;
    font-size: 24px;
}

.button {
    background-color: green;
    color: white;
    padding: 10px 20px;
}

#header {
    font-family: Arial, sans-serif;
```

```
    font-size: 18px;
}
```

In this example:

- The first rule set applies styles to all <h1> elements, setting their color to blue and font size to 24 pixels.
- The second rule set applies styles to elements with the class "button", setting their background color to green, text color to white, and padding.
- The third rule set applies styles to the element with the ID "header", setting its font family to Arial or sans-serif and font size to 18 pixels.
- Rule sets are the building blocks of CSS, allowing you to define the appearance and layout of HTML elements across your web pages.