Supplementary Material

A Additional Details

A.1 Details of Rendering Parameters

As described in Sec. 3.1, we utilize rendering parameters to standardize CSS due to its code complexity. The examples in Fig. 8 demonstrate this complexity. As shown on the left side of Fig. 8, CSS can be utilized in different forms[10]: `Inline Styles` for direct HTML element styling via the "style" attribute; `Internal Style Sheets` using "<style>" tags within HTML documents; and `External Style Sheets` linking to CSS files externally. The middle of Fig. 8 showcases various CSS selectors[11], including simple `tag`, `class`, and `ID` selectors, as well as complex `attribute` and `descendant` selectors. Furthermore, CSS follows certain rules regarding inheritance and overrides[12]. An example on the right side of Fig. 8 shows how the `.highlight` class's red color is overridden by the more specific ID selector `#main-content` p, turning the color green.

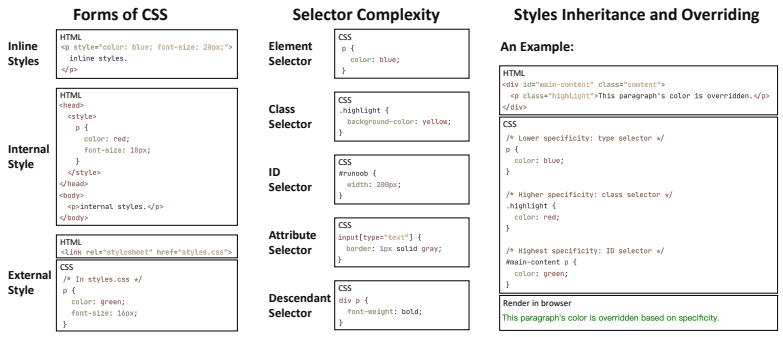

Fig. 8: Examples of CSS code complexity, showcasing various CSS forms (*left*), selector complexity (*middle*), and style inheritance and overrides (*right*).

The complexity of CSS makes direct generation of CSS impractical. Even parsing CSS code to obtain WebRPG task labels is challenging. Since browsers compute the final applied CSS property values (i.e., rendering parameters) for each element based on HTML and CSS to render web pages, we propose extracting each element's RPs directly from the browser, as described in Sec. 3.2. This approach bypasses the need to parse CSS code, achieving the standardization of CSS.

[10] `https://www.w3schools.com/css/css_howto.asp`
[11] `https://www.w3schools.com/css/css_selectors.asp`
[12] `https://developer.mozilla.org/en-US/docs/Web/CSS/Inheritance`

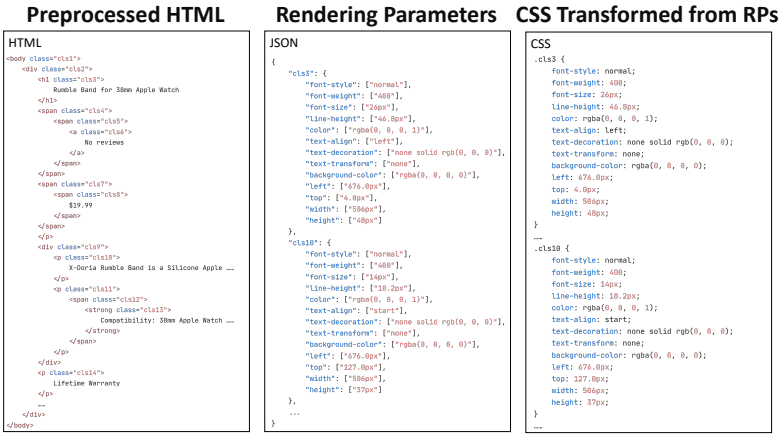

Fig. 9: A illustration case of rendering parameters organization, including preprocessed HTML (*left*), JSON-stored rendering parameters (*middle*), and the CSS transformed from those RPs (*right*).

Table 3: The complete vocabulary of rendering parameters including all categories, their index ranges, and selected examples.

Category	Index Range	Examples
Integer Pixel	0-1920	1px, 1052px, 1920px
Color	1921-1966	RGBA(153, 204, 0, 1), RGBA (255, 255, 255, 1)
Font Style	1967-1969	italic, oblique
Font Weight	1970-1978	100, 500, 900
Line Height	1979	normal
Text Align	1980-1985	start, center, end
Text Decoration	1986-1987	none, underline
Text Transform	1988-1991	uppercase, capitalize
PAD	1992	PAD

In practice, we follow the pre-order traversal order of the DOM tree to assign a unique ID to each element, achieved by modifying the class name, as shown on the left side of Fig. 9. We organize the rendering parameters using JSON, where the key is the element's ID, as illustrated in the middle of Fig. 9. RPs can also be transformed into CSS, utilizing class selectors only, as demonstrated on the right side of Fig. 9.

Additionally, the complete vocabulary of all rendering parameters is detailed in Tab. 3, and index ranges of each rendering parameter are presented in Tab. 4.

A.2 Details of Visual Complexity Metric

The Visual Complexity (VC) metric integrates three dimensions: color, size, and alignment. For any given web page, the three dimensions are defined as follows:

Table 4: Index ranges for each rendering parameter in the vocabulary.

Rendering Parameter	Index Range
left	0-1920
top	0-1920
width	0-1920
height	0-1920
font-style	1967-1969
font-weight	1970-1978
font-size	0-32
line-height	0-50, 1979
text-align	1980-1985
text-decoration	1986-1987
text-transform	1988-1991
color	1921-1966
background-color	1921-1966

Color: The color metric measures the richness of colors and is defined as:

$$VC_{color} = \frac{1}{2N}(C_c + C_{bg} - 2), \tag{7}$$

where N is the number of elements, and C_c and C_{bg} are the counts of unique *color* and *background-color* attributes respectively.

Size: The size metric measures the diversity of sizes among web page elements. In particular, it calculates the size diversity for all N' parent elements and then computes the average. The formula is as follows:

$$VC_{size} = \frac{1}{N'}\sum_{i=1}^{N'}\left(\frac{DS_i - 1}{NC_i}\right), \tag{8}$$

with NC_i and DS_i being the count of child elements and their distinct sizes for element i, respectively.

Alignment: The complexity of a web page inversely correlates with the number of pairwise alignments [15]. To simplify, this metric applies only to leaf nodes. The calculation formula is as follows:

$$VC_{alg} = 1 - \frac{1}{N_{leaf}(N_{leaf} - 1)}\sum_{j=1}^{N_{leaf}}\sum_{i\neq j}^{N_{leaf}} ALG_{ij}, \tag{9}$$

where N_{leaf} denotes the number of leaf node elements, and ALG_{ij} is a binary indicator of alignment (1) or misalignment (0) between elements i and j.

The overall VC is the sum of three metrics: $VC = VC_{\text{color}} + VC_{\text{alg}} + VC_{\text{size}}$.

A.3 Dataset Details

The distribution of Visual Complexity (Sec. A.2) values across all samples is illustrated in Fig. 10. In our dataset, samples with a VC value of less than 0.1

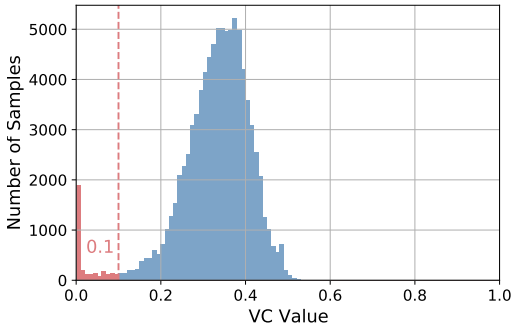

Fig. 10: Histogram showcasing Visual Complexity (Sec. A.2) value distribution across all samples. Red indicates samples are filtered out, while blue represents those retained in the dataset.

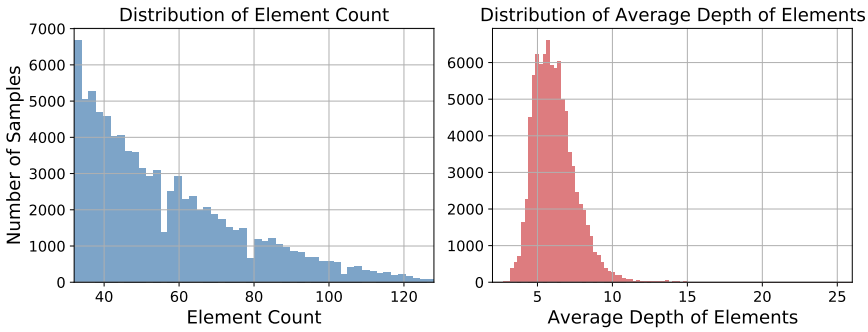

Fig. 11: Histograms showcasing element count and the average depth of elements distribution across all samples in the dataset.

are filtered out, resulting in a remaining subset where the VC distribution is relatively concentrated and approximates a normal distribution, thereby helping to mitigate the impact of extreme samples on training. Additionally, to further investigate our dataset, we visualize two crucial statistical values, element count and the average depth of elements, in Fig. 11. This visualization indicates that the dataset lacks samples containing a large number of elements or considerable element depths.

A.4 Implementation details of FID model

As described in Sec. 6.1, the FID model is a binary classifier, incorporating a VAE described in Sec. 5.2, four transformer layers, and a classification header. A special *CLS* vector is utilized as the classification feature, representing all RPs. The rest of the input is the same as the model in Sec. 5.4. Three kinds of noise are designed to pollute the real data, namely perturbing the original values with

Table 5: The prompt template for GPT-4 experiment in Sec. 6.3.

Prompt	You are an exceptional web designer. Please create the corresponding CSS code based on the HTML code I have provided, so as to craft a well-designed visual presentation for the web page. You can only use the following CSS properties: "left", "top", "width", "height", "font-style", "font-weight", "font-size", "line-height", "color", "text-align", "text-decoration", "text-transform", "background-color". Please exercise caution in controlling the size of the image, as using the original image dimensions directly may result in excessive spatial occupation. Here are several demonstrations:{Demonstrates}. Below is the HTML code and do not reply with anything other than CSS code: {HTML_Code}	.
Slots	Demonstrates	The HTML-CSS pairs for three selected web page segments.
	HTML_Code	HTML code of given web page.

a fixed variance, randomly substituting elements with synthetic ones, and randomly swapping elements. The specific FID models for layout and style, namely FID_{layout} and FID_{style}, are trained by masking irrelevant inputs. Specifically, FID_{layout} processes only the layout, masking the style, and FID_{style} processes only the style, masking the layout. The FID models for overall, layout, and style, achieve classification accuracies of 88.8%, 95.5%, and 92.4%, respectively.

A.5 Implementation details of WebRPG Baselines

The backbone of WebRPG-AR consists of 6-layer transformers for both encoder and decoder, and WebRPG-DM is a 12-layer U-ViT. The mask scheduling function $\gamma(r)$ is a cosine function, the time steps T in diffusion follows [21] with a value of 1000, and λ_{KL} is set to 1e-6. For optimization, AdamW [45] is used with a learning rate of 1.2e-4, β_1 of 0.9, and β_2 of 0.99.

The prompt template for the LLMs experiment in Sec. 6.3 is detailed in Tab. 5. Due to the extensive length of textual representation for each element's RPs, as shown on the right side of Fig. 9, we opt to have LLMs directly generate the CSS code. The specific steps for conducting the LLMs experiment are:

1. Use the prompt to generate CSS code via LLMs.
2. Use a browser to render the web page with the given HTML and the CSS code generated by LLMs.
3. Extract the RPs for all elements, employing the method in Sec. 4.1.
4. Evaluate these RPs using the metrics in Sec. 6.1.

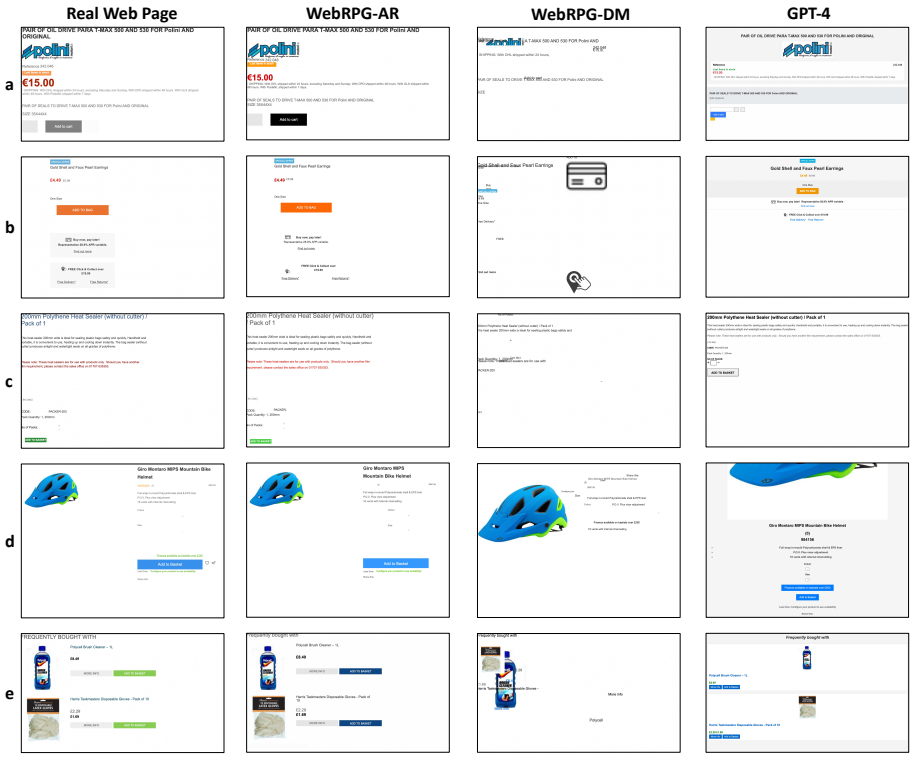

Fig. 12: Additional visualization of baseline-generated results. The screenshots focus on areas with elements.

B Additional Results

B.1 Additional Cases of Baseline-Generated Results

We present additional results from WebRPG baselines in Fig. 12. These results exhibit the performance of all baselines comparable to that outlined in Sec. 6.3. Additionally, Fig. 13 displays the web page variants generated by WebRPG-AR based on the same HTML, each produced through individual inferences. The differences in layout and style among these variants indicate that WebRPG-AR can generate diverse web pages while maintaining semantic coherence.

Table 6: FID on rendered web page screenshots.

	WebRPG-AR	GPT4	WebRPG-DM	Real Web Page
$\text{FID}_{\text{Screenshot}}$	3.2102	15.515	33.040	1.1156

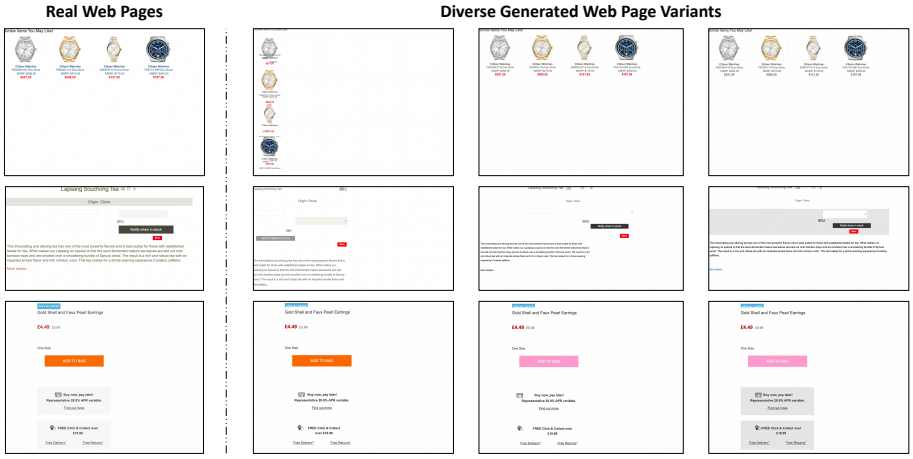

Fig. 13: The web page variants generated by WebRPG-AR based on the same HTML.

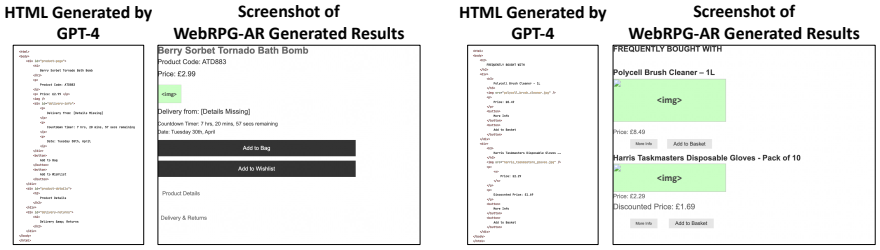

Fig. 14: The HTML code generated by GPT-4 and the corresponding web page visual results generated by WebRPG-AR. Screenshots use **green** <img> placeholders due to GPT-4 generates fictitious source addresses.

B.2 The FID on Screenshots of Rendered Web Pages

The FID on screenshots of rendered web pages is shown in Tab. 6.

B.3 Further Cases of Integrating LLM with WebRPG Model

Fig. 14 showcases more cases of WebRPG-AR creating visual presentations of web pages based on HTML code generated by GPT-4. The prompt template for automatically generating HTML is in Tab. 7. The prompt encompasses human-authored descriptions of web design ideas, with an example shown in Tab. 8.

B.4 Human Evaluation

We conduct a human evaluation using pairwise comparisons. We randomly select 100 test samples and generate visual presentations using WebRPG-AR,

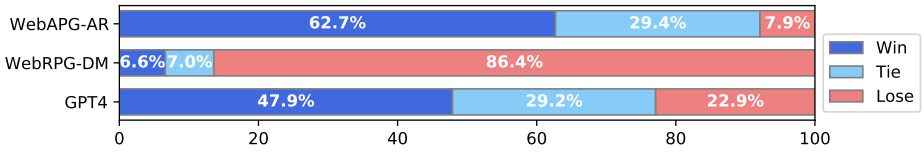

Fig. 15: Human pairwise comparison evaluation results.

Table 7: The prompt template for automatically generating HTML.

Prompt	You are a web developer. Please generate the HTML code for a web page with a caption of {Deign_Idea}.
Slot	Deign_Idea Human-authored descriptions of web design ideas, with an example shown in Tab. 8.

WebRPG-DM, and GPT-4. Five human annotators evaluate each pair to determine the superior presentation or if there is a tie. The results, shown in Fig. 15, align with the objective evaluations in Tab. 1.

Table 8: An example of web design ideas described by humans.

This web page showcases the "Rumble Band for 38mm Apple Watch," offered at $19.99. It's identified as the X-Doria Rumble Band and is noted for its compatibility with the 38mm Apple Watch Series 1, 2, 3, and Nike Edition. Highlighted on the page are customer assurances including a lifetime warranty, complimentary shipping on all orders, and a 30-day hassle-free return policy. A conspicuous "Add to Cart" button is prominently displayed. The product's image is designed to highlight its appearance and design features.

C An Example Explanation of SC Score

Fig. 16 provides an example to explain the SC Score further. The elements representing price (marked with a green box, hereafter termed as price elements) on the real web page W and on generated web page 1 $\hat{W}_1$ have differing styles in terms of font color and size. However, these differences do not affect the perception of price elements, as their style remains consistent within each individual web page. In contrast, the generated web page 2 $\hat{W}_2$ changes just one price element, which leads to confusion when perceiving the price elements. Although $\hat{W}_2$ seems more visually similar to W because of only one differing element, from a semantic perspective, $\hat{W}_1$ is more coherent. Therefore, the SC Score evaluates whether elements that share a style on the real web page maintain that consis-

Fig. 16: An example for visualizing style consistency. Notably, $\hat{W}_1$ and $\hat{W}_2$ are artificially created for demonstration purposes.

Fig. 17: A visualization of the style consistency subset based on a real web page. The style consistency subset is defined in Sec. 6.1.

tency on the generated page, beyond just visual similarity. Additionally, Fig. 17 provides a visualization of the style consistency subset for a real web page.

D Further Discussion on the Performance of LLM in WebRPG Task

As described in Sec. 6.2, we employ GPT-4 as a representative for LLMs. Due to the complexity of CSS code practices and the noise in actual web pages, directly fine-tuning LLMs is not feasible. Consequently, we do not conduct fine-tuning experiments. Moreover, to further explore the performance of GPT-4 in WebRPG tasks, we conduct two qualitative experiments. Tab. 9 details the prompt templates. The first experiment inputs HTML and the captions from the original web page screenshots. The second experiment comprises HTML, these captions, and the screenshots themselves. It's noteworthy that the additional data comprised visual information from the original web pages, serving essentially as a form of ground truth. The second experiment and the generation of web page screenshot captions both leverage the multimodal capabilities of GPT-4V[13]. Fig. 18 presents visualizations of selected cases, showing that additional data does not enhance GPT-4's performance. Given that these two qualitative experiments involve ground truth inputs, we do not include them in the main text or conduct quantitative experiments.

[13] `https://openai.com/research/gpt-4v-system-card`

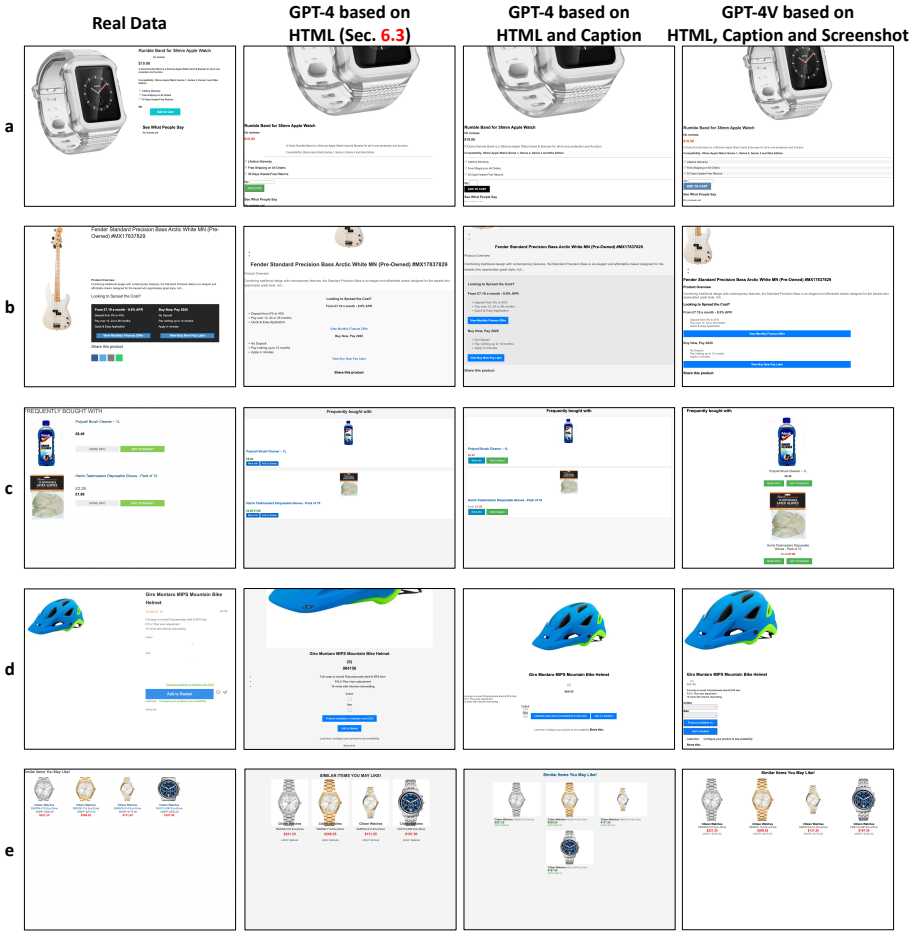

Fig. 18: Further qualitative evaluation of GPT-4's performance in WebRPG task. Notably, the "GPT-4 based on HTML" group is the experiment in Sec. 6.3.

Table 9: The prompts for Sec. D. "H.", "C.", and "S." denote "HTML", "caption" and "screenshot", respectively.

Information	H.+C.	You are an exceptional web designer. Please create the corresponding CSS code based on the HTML code I have provided, so as to craft a well-designed visual presentation for the web page. **Furthermore, for better comprehension of the original web page design, here is a detailed caption: {Caption}.** You can only use the following CSS properties: "left", "top", "width", "height", "font-style", "font-weight", "font-size", "line-height", "color", "text-align", "text-decoration", "text-transform", "background-color". Please exercise caution in controlling the size of the image, as using the original image dimensions directly may result in excessive spatial occupation. Here are several demonstrations:{Demonstrates}. Below is the HTML code and do not reply with anything other than CSS code: {HTML_Code}.
	H.+C.+S.	You are an exceptional web designer. Please create the corresponding CSS code based on the HTML code and **screenshot** I have provided, so as to craft a well-designed visual presentation for the web page. **Furthermore, for better comprehension of the original web page design, here is a detailed caption: {Caption}.** You can only use the following CSS properties: "left", "top", "width", "height", "font-style", "font-weight", "font-size", "line-height", "color", "text-align", "text-decoration", "text-transform", "background-color". Please exercise caution in controlling the size of the image, as using the original image dimensions directly may result in excessive spatial occupation. Here are several demonstrations:{Demonstrates}. Below is the HTML code and do not reply with anything other than CSS code: {HTML_Code}.
Slots	Caption	Captions from the original web page screenshots.
	HTML_Code	HTML code of given web page.
	Demonstrates	The HTML-CSS pairs for three selected web page segments.