

The background features a light gray grid with several overlapping circles in teal, lime green, orange, and pink. Dashed lines in various colors (teal, yellow, blue) form loops and paths across the page.

How to Build a Web Page

Chieh-Yang Huang

Chieh-Yang Huang



I am currently a third-year Ph.D. student in IST.

I am a research assistant in [Dr. Kenneth Huang's](#) amazing [CrowdAI Lab](#), working on **CrowdSourcing, Deep Learning, and Natural Language Processing** projects.

My work mainly focuses on building tools to help human's day-to-day life, especially writing and language-related works.

What is a web page?

- A web page

= Browser (HTML + CSS + JavaScript)



Compiler
Rendering



Structure



Style



Behavior
Functions

Browser

- Your browser will parse the HTML and render it into the web page you see.



Chrome



Firefox

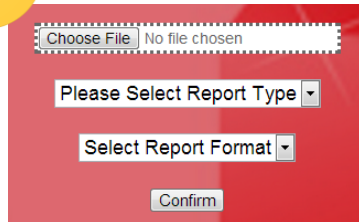
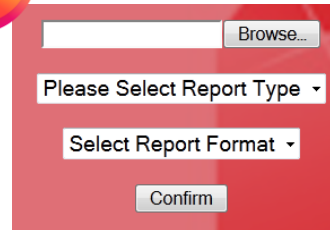


Edge

Browser

- Different browsers implement same function in different ways.
- Even the same HTML/CSS/JavaScript code can result in different layouts/behaviors.

➡ Make sure to test your interface in at least two browsers

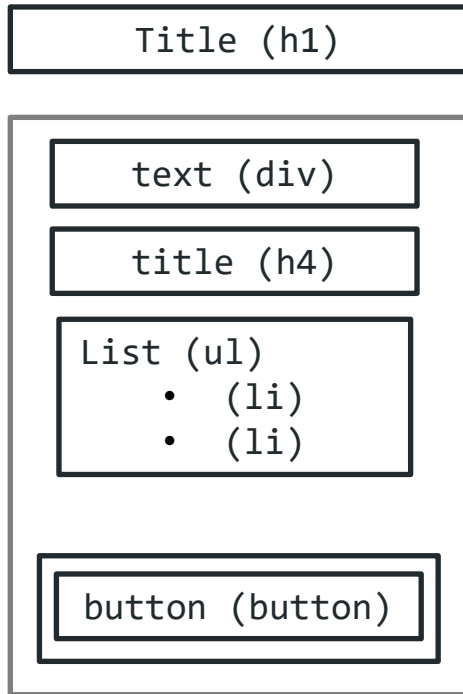
A screenshot of a web form rendered in Google Chrome. The form has a red background. At the top, there is a file selection area with a 'Choose File' button and the text 'No file chosen'. Below this is a dropdown menu labeled 'Please Select Report Type'. Underneath is another dropdown menu labeled 'Select Report Format'. At the bottom is a 'Confirm' button.A screenshot of the same web form rendered in Mozilla Firefox. The form has a red background. At the top, there is a file selection area with a 'Browse...' button. Below this is a dropdown menu labeled 'Please Select Report Type'. Underneath is another dropdown menu labeled 'Select Report Format'. At the bottom is a 'Confirm' button.

[when rendering the page on different browsers layout changes - Stack Overflow](#)

HTML

- Hyper Text Markup Language
- HTML describes the structure
- HTML elements tell the browser how to display the content
- Elements are presented as boxes (blocks)

HTML



```
<html>
<head>
  <title>Assignment 02</title>
  <link rel="stylesheet" href="static/main.css">
  <script src="static/main.js"></script>
  <link rel="shortcut icon" type="image/ico" href="static/favicon.ico">
</head>

<body>
  <h1 id="title">Write a Short Story Based on a Photo Sequence</h1>

  <div class="container instruction_panel" id="instruction_1">
    <div>
      In this task, there are <strong>two steps</strong> and in the end you will need to write
    </div>
    <h4>Step 1</h4>
    <ul>
      <li>Please <strong>watch the following images</strong> carefully for <strong>15</strong>
      <li>Feel free to click the thumbnail to see the large photo.</li>
    </ul>
    <div class="center">
      <button type="button" class="btn btn-info btn-block" id="go_btn">GO</button>
    </div>
  </div>
```

HTML

- Basic Structure

```
<html>
<head></head> // import .js & .css files
                // define metadata

<body></body> // the main content
                // put your tags here

</html>
```

```
<html>
<head>
  <title>Assignment 02</title>
  <link rel="stylesheet" href="static/main.css">
  <script src="static/main.js"></script>
  <link rel="shortcut icon" type="image/ico" href="static/favicon.ico">
</head>

<body>
  <h1 id="title">Write a Short Story Based on a Photo Sequence</h1>

  <div class="container instruction_panel" id="instruction_1">
    <div>
      In this task, there are <strong>two steps</strong> and in the
    </div>
    <h4>Step 1</h4>
    <ul>
      <li>Please <strong>watch the following images</strong> carefully
      <li>Feel free to click the thumbnail to see the large photo.</li>
    </ul>
    <div class="center">
      <button type="button" class="btn btn-info btn-block" id="go_bt">
    </div>
  </div>
```


HTML

- HTML consists of a series of elements
- HTML elements are represented by tags

```
<h1> This is Title! </h1>  
<div>  
    <button> Submit </button>  
</div>
```

This is Title!

Submit

HTML

- Attribute of the elements
 - ◎ Attribute provide additional information

src: the path to the image

Width: the width of the image

```

```

alt: alternate text for the image



HTML – Useful Tags

- Heading - <h1>, <h2>, <h3>

```
<h1>H1 Heading</h1>  
<h2>H2 Heading</h2>  
<h3>H3 Heading</h3>  
<h4>H4 Heading</h4>  
<h5>H5 Heading</h5>
```

H1 Heading

H2 Heading

H3 Heading

H4 Heading

H5 Heading

HTML – Useful Tags

- Text - `<p>`, `<pre>`, `
`

```
<p>
  Hello how are you?
  I am fine. Thank you.
</p>

<pre>
  Hello how are you?
  I am fine. Thank you.
</pre>

<p>
  Hello how are you? <br/>
  I am fine. Thank you.
</p>
```

Hello how are you? I am fine. Thank you.

Hello how are you?
I am fine. Thank you.

Hello how are you?
I am fine. Thank you.

HTML – Useful Tags

- List - `` `` ``

```
<ul>
  <li>This is important.</li>
  <li>This is also important.</li>
</ul>

<ol>
  <li>This is important.</li>
  <li>This is also important.</li>
</ol>
```

- This is important.
- This is also important.

1. This is important.
2. This is also important.

HTML – Useful Tags

- Division - <div> (Default to be a block)
- Span - (Default to be inline)

```
Hey
<div style="border: 1px solid black;">
  Hello!
</div>
Hey

<br><br><br>

Hey
<span style="border: 1px solid black;">
  Hello Again.
</span>
Hey
```

Hey
Hello!
Hey

Hey Hello Again. Hey

HTML – Useful Tags

- Form - <form>

Build a form for users to provide their inputs.

*** Remember to give all the “input” tag a name!!**

```
<form>
  <input type="text" name="account" placeholder="Account">
  <input type="password" name="password" placeholder="Password">
  <input type="submit">
</form>
```

HTML – Useful Tags

- Form - `<form>` `<input>` `<button>`

```
<div>
  <input type="text" placeholder="Text" name="input-text">
</div>

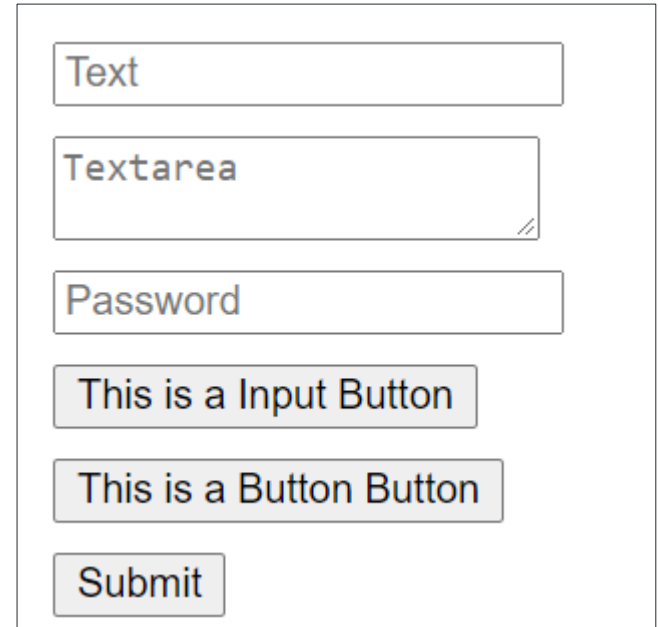
<div>
  <textarea placeholder="Textarea" name="textarea-text"></textarea>
</div>

<div>
  <input type="password" placeholder="Password" name="password">
</div>

<div>
  <input type="button" value="This is a Input Button">
</div>

<div>
  <button> This is a Button Button</button>
</div>

<div>
  <input type="submit">
</div>
```



Text

Textarea

Password

This is a Input Button

This is a Button Button

Submit

HTML – Useful Tags

- Form - radio, checkbox

```
<div>
  <input type="radio" id="radio-button-1" name="radio-button-group" value="radio-option-1">
  <label for="radio-button-1">Radio-button-1</label>
  <input type="radio" id="radio-button-2" name="radio-button-group" value="radio-option-2">
  <label for="radio-button-2">Radio-button-2</label>
  <input type="radio" id="radio-button-3" name="radio-button-group" value="radio-option-3">
  <label for="radio-button-3">Radio-button-3</label>
</div>

<div>
  <input type="checkbox" id="checkbox-button-1" name="checkbox-button-group" value="checkbox-option-1">
  <label for="checkbox-button-1">checkbox-button-1</label>
  <input type="checkbox" id="checkbox-button-2" name="checkbox-button-group" value="checkbox-option-2">
  <label for="checkbox-button-2">checkbox-button-2</label>
  <input type="checkbox" id="checkbox-button-3" name="checkbox-button-group" value="checkbox-option-3">
  <label for="checkbox-button-3">checkbox-button-3</label>
</div>
```

☐ Radio-button-1 ☐ Radio-button-2 ☒ Radio-button-3

☐ checkbox-button-1 ☒ checkbox-button-2 ☒ checkbox-button-3

HTML – Useful Tags

- Image -

```

```



HTML – Useful Tags

- Table - <table>, <tr>, <td>, <th>

```
<table>
  <tr>
    <th>Header-1</th>
    <th>Header-2</th>
    <th>Header-3</th>
  </tr>

  <tr>
    <td>(1, 1)</td>
    <td>(1, 2)</td>
    <td>(1, 3)</td>
  </tr>

  <tr>
    <td>(2, 1)</td>
    <td>(2, 2)</td>
    <td>(2, 3)</td>
  </tr>
</table>
```

Header-1	Header-2	Header-3
(1, 1)	(1, 2)	(1, 3)
(2, 1)	(2, 2)	(2, 3)

HTML - id & class

- Giving an element an id/class allows us to specify the style/behavior for that element.
- **id** is an **unique** value!!!
- **class** means the **group** has the same style/behavior.

Color: red

```
<button class="round_btn" id="red_btn">Red</button>
```

```
<button class="round_btn" id="blue_btn">Blue</button>
```

Same shape

Color: blue

HTML - id & class

- Important for CSS/JavaScript
 - (1) Specify the element using a selector (id/name/...)
 - (2) Specify the style/function

```
<div class="container">  
  <button class="round_btn" id="red_btn">Red</button>  
  <button class="round_btn" id="blue_btn">Blue</button>  
</div>
```



```
.round_btn {  
  border-radius: 5px;  
  background: ■ white;  
  border: 1px solid □ black;  
  width: 100px;  
  height: 30px;  
}  
#red_btn {  
  color: ■ red;  
}  
#blue_btn {  
  color: ■ blue;  
}
```

CSS

- Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
→ Styling

How to write CSS?

- Specify the element(s) by the selector.
- Set the property and the value.

- Selector

- Tell our browser WHO is the **target element**.
- # (id)
- . (class)
- <https://flukeout.github.io/>
(Learn more about the selector here)

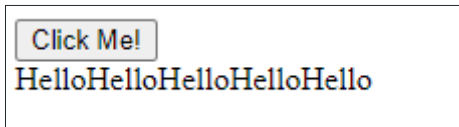
```
.round_btn {  
  border-radius: 5px;  
  background: ■ white;  
  border: 1px solid ■ black;  
  width: 100px;  
  height: 30px;  
}  
#red_btn {  
  color: ■ red;  
}  
#blue_btn {  
  color: ■ blue;  
}
```

JavaScript

- Define the behavior of the web page.
- How to write JavaScript?
 - Specify the element(s) by the selector.
 - Write the desired function.
- We can use **jquery.js** to help us manipulate the page.

JQuery

- Add a function to a button.
 - Import jquery library
 - Define your button
 - Add an **id** to the button
 - In the `<script>` section, put everything inside **ready**
 - Specify the id and bind a **function** to the **click event**.



```
1 <html>
2 <head>
3   <title>Test JQuery</title>
4   <script src="https://code.jquery.com/jquery-3.6.0.m
5 </head>
6 <body>
7   <button id="click-btn">Click Me!</button>
8   <div id="target-div"></div>
9
10  <script>
11    $(document).ready(function() {
12
13      // (1) select element with id="click-btn"
14      // (2) add a function to the event "click"
15      $("#click-btn").click(function() {
16
17        // What does function do?
18        // (1) find target_div
19        // (2) add some texts to it
20        $("#target-div").append("Hello");
21      });
22    });
23  </script>
24 </body>
25 </html>
```

Building a Web Page

- Install an editor
→ Sublime / VS code / Notepad++ / Vim / Emacs
- Starting writing the web page!
→ HTML & CSS & JavaScript
- Upload your web page to a hosting server
→ GitHub Page

Example – Collecting Captions

- Task: show an image and collect the caption.
→ Writing a comprehensive instruction is important!

The diagram illustrates a user interface layout for a task where an image is shown and a caption is collected. It is structured as follows:

- Instruction**: A large rectangular box at the top, intended for displaying the task instructions.
- Image** and **Input Text**: Two rectangular boxes positioned side-by-side in the middle. The 'Image' box is on the left, and the 'Input Text' box is on the right, where the user would enter their caption.
- Submit Button**: A rectangular box at the bottom, centered horizontally, for submitting the collected caption.

Example – Collecting Captions

- Import JQuery to improve JavaScript.

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-  
/xUj+30JU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
```

- Import Bootstrap 4 to improve styling and some other functions.

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity  
="sha384-Gn5384xqQ1aowXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
```

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-  
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-  
JZR6Spejh4U02d8j0t6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
```

Example – Collecting Captions

- Build the basic structure

```
1  <html>
2  <head>
3    <title>Collecting Captions</title>
4    <meta charset="utf-8">
5    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
6    <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js">
7    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js">
8
9    <style>
10     /* Your own CSS code */
11   </style>
12
13   <script>
14     /* Your own JavaScript code */
15   </script>
16 </head>
17 <body>
18   <!-- Your own HTML code -->
19
20 </body>
21 </html>
```

Example – Collecting Captions

- Write down all the HTML tag

```
17 <body>
18   <!-- Your own HTML code -->
19   <div>
20     <h1> Instruction </h1>
21     <ul>
22       <li> Step 1. Bla Bla Bla </li>
23       <li> Step 2. Bla Bla Bla </li>
24     </ul>
25   </div>
26
27   <div>
28     
29     <textarea placeholder="Enter Caption Here"></textarea>
30   </div>
31
32   <div>
33     <button>Submit</button>
34   </div>
35
36 </body>
```

Instruction



Input Text

Submit Button

Instruction

- Step 1. Bla Bla Bla
- Step 2. Bla Bla Bla



Enter Caption Here

Submit

Example – Collecting Captions

- Add some styling using Bootstrap

```
<!-- Your own HTML code -->
<div class="container">
  <div class="row">
    <div class="col-12">
      <h1> Instruction </h1>
      <ul>
        <li> Step 1. Bla Bla Bla </li>
        <li> Step 2. Bla Bla Bla </li>
      </ul>
    </div>
  </div>

  <div class="row">
    <div class="col-4">
      
    </div>
    <div class="col-8">
      <textarea class="form-control" placeholder="Enter Caption Here" rows="11"></textarea>
    </div>
  </div>

  <div class="row">
    <div class="col-12">
      <button type="button" class="btn btn-outline-primary btn-block">Submit</button>
    </div>
  </div>
</div>
```

Instruction

- Step 1. Bla Bla Bla
- Step 2. Bla Bla Bla



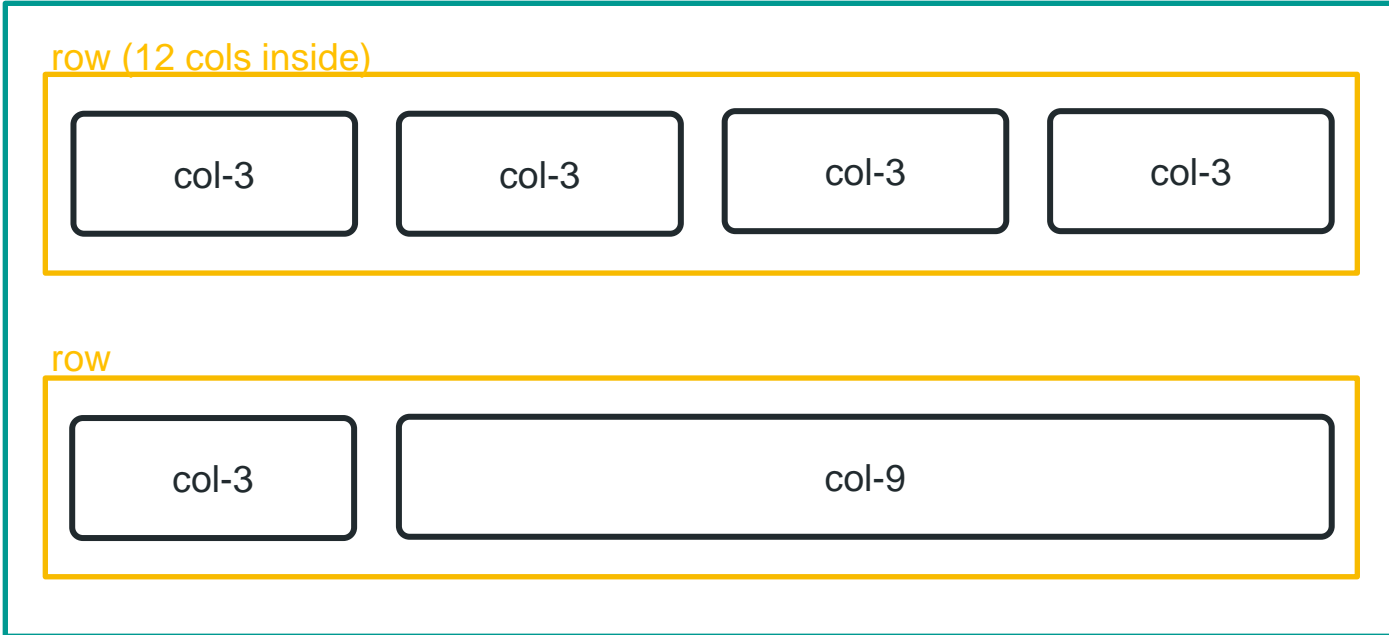
Enter Caption Here

Submit

Example – Collecting Captions

- Bootstrap's layout system

container



Example – Collecting Captions

- Add a `<form>` tag to include all your inputs and remember to give all `<input>` tag a name!

```
<!-- Your own HTML code -->
<div class="container">
  <form>
    <div class="row">
      <div class="col-12">
        <h1> Instruction </h1>
        <ul>
          <li> Step 1. Bla Bla Bla </li>
          <li> Step 2. Bla Bla Bla </li>
        </ul>
      </div>
    </div>

    <div class="row">
      <div class="col-4">
        
      </div>
      <div class="col-8">
        <textarea name="caption" class="form-control" placeholder="Enter Caption Here" rows="11"></textarea>
      </div>
    </div>

    <div class="row" id="submission_block">
      <div class="col-12">
        <button type="button" class="btn btn-outline-primary btn-block">Submit</button>
      </div>
    </div>
  </form>
</div>
```

Example – Collecting Captions

- Add more CSS code to style the web page

```
<h1> Instruction </h1>
<ul>
  <li>
    In this task you will see a figure.
    Please write a caption <span class="underline"> to describe what happens in the figure.</span>
  </li>
  <li> Please write at least <span class="highlight">20 words</span>. </li>
</ul>
```

```
<div class="row" id="submission_block">
  <div class="col-12">
    <button type="button" class="btn btn-outline-primary btn-block">Submit</button>
  </div>
</div>
```

```
<style>
  /* Your own CSS code */
  #submission_block {
    margin-top: 20px;
  }
  .highlight {
    background-color: yellow;
    color: red;
    font-weight: bold;
  }
  .underline {
    font-weight: bold;
    text-decoration: underline;
  }
</style>
```

Example – Collecting Captions

- Let's add some functions to make sure workers behave as what you expect!
 - ⦿ Time lock:
Make sure workers spend a certain time on the task
 - ⦿ Minimum word count:
Make sure workers write at least 20 words.
 - ⦿ Disable copy-and-paste functions
- Make sure you **display the information** to them instead of simply rejecting the submission!

Example – Collecting Captions

- Time Lock
 1. Disable the submission button
 2. Use setInterval(function, XXXX) as a timer
 - call the function every XXXX ms
 3. Display information to workers
 4. After 30 seconds, set disable=false

Example – Collecting Captions

```
<button disabled id="submit-btn" type="button" class="btn btn-outline-primary btn-block">Submit</button>
```

```
// Time Lock
// (1) initial time_left
var time_left = 30;

// (2) create setInterval function.
var interval_id = setInterval(function() {
    // count down 1 second
    time_left -- 1;

    // if time_left == 0 => turn on the submit-btn
    //                      => remove time information on the button
    //                      => stop setInterval function
    if (time_left == 0) {
        $("#submit-btn").prop("disabled", false);
        $("#submit-btn").text("Submit");
        clearInterval(interval_id);
    } else {
        // if time_left => display time information to users
        $("#submit-btn").text("Submit (" + time_left + ")");
    }
}, 1000);
```

disable=true

Submit (27)

disable=false

Submit

Example – Collecting Captions

- Minimum word count
 1. Compute word count by space characters.
 2. Display word count information.
 3. Examine word count before submission.

```
<textarea id="caption" name="caption" class="form-control" placeholder="Enter Caption Here" rows="11"></textarea>
<div>Word Count: <span id="word-count" class="red">0</span> </div>
```

```
// Minimum Word Count
// (1) init min_word_count
var min_word_count = 20;
var current_word_count = 0;

// (2) add a function to the change event on the "textarea" element
$("#caption").change(function() {

    // (3) get the input text
    var text = $("#caption").val();

    // (4) split using space character (" ", "\t", "\n")
    var words = text.split(/\s+/);

    // (5) keep the current word count
    current_word_count = words.length;

    // (6) display information to users
    $("#word-count").text(String(words.length));

    // (7) if #words >= min_word_count => change to black
    //     if #words < min_word_count => change to red
    if (words.length >= min_word_count) {
        $("#word-count").removeClass("red").addClass("black");
    } else {
        $("#word-count").removeClass("black").addClass("red");
    }
});
```

hello how are you?

Word Count: 5

Check the difference between
“change” event and “keypress” event!

Verifying before submission

- Custom the behavior of your submission button!
- Make sure to **pop out the reason** when rejecting the submission!

```
<form id="worker-form">
```

```
<button
  disabled
  id="submit-btn"
  type="button"
  class="btn btn-outline-primary btn-block"
>
  Submit
</button>
```

```
// Add submission function
$("#submit-btn").click(function() {
  // Verify word counts!
  if (current_word_count < min_word_count) {
    alert("Please write at least 20 words");
    return false;
  }

  // Verify other constraints!!
  // ***** //

  // Submit
  $("#worker-form").submit();
});
```


Example – Collecting Captions

- Disable copy-and-paste function in the textarea
 1. Bind new function to “onpaste” event
 2. Bind new function to “ondrop” event
 3. We can simply do this in HTML tag

```
<textarea
  id="caption"
  name="caption"
  class="form-control"
  placeholder="Enter Caption Here"
  rows="11"
  onpaste="return false"
  ondrop="return false"
></textarea>
```

Example – Collecting Captions

- Depending on your task, there are many different locks /checks we can do before allowing workers to submit the task.
 - ⦿ Read-all lock
 - Check if user scroll to the bottom of the page
 - ⦿ Validate if the input value satisfied the constraint
 - Number / Range / Type
 - ⦿ Think carefully when designing your own interface!

Thanks!



Any questions?

The Sample Code is here:

<https://github.com/appleternity/IST597-Crowd-AI>