Matriculation number: 200009834

1. Web Standards and Frameworks

(a):

1. First, someone becomes interested in a particular topic.

2. When the attention of this particular topic is getting higher and higher, the W3C director will announce the development of a proposal charter for one or more interest groups and working groups. W3C members review the proposal charter.

3. When someone within W3C supports investing resources on a topic of concern, the W3C director will approve the establishment of a working group, and the group will officially begin work.

4. The working group charter contains the deliverables expected by each group, and the working group will establish standards and guidelines in accordance with this expectation.

5. These standards and guidelines need to be revised and reviewed repeatedly, reviewed by W3C members and the public, and must meet implementation and interoperability requirements.

6. Finally, the advisory committee reviews the mature technical report, and if passed, it will be published as a standard.

The above is a brief process of standard production.

When a proposal was discussed and recognized by W3C, W3C established a Working Group starting from Editor's Draft:

1. Editor's Draft: represents the prototype and the latest attributes. This version can be updated at any time and runs through the entire standard life cycle. Now most of it is hosted on GitHub.
2. Working Draft (WD): The finished design will be extensively reviewed and iterated.
3. Candidate Recommendation (CR): indicates that the API design is basically completed, and implementation and testing have begun.
4. Proposed Recommendation (PR): Final review of specification documents.
5. Formal recommendation standard (W3C Recommendation, referred to as REC): A Web standard recommended by W3C for deployment in the final stage of technical

specifications, which has been unanimously recognized by members and W3C directors. Motivation: Based on extensive consensus and decision-making, comprehensive consideration of accessibility, privacy, security, and internationalization needs, reflecting the views of different industries and global stakeholders, balancing speed, fairness, accountability and quality, through W3C internal and external Extensive review by various groups to achieve uniform specifications and facilitate development.

(b):

Problems to be solved in mobile-friendly development:

1. The cross-platform problem of CSS and JavaScript.
2. Use of h5 and many new features.
3. Responsive layout and screen resolution issues.
4. Use of native interaction.
5. Debugging method.
6. Performance optimization.

Mobile-friendly web development also needs to solve the following common problems in traditional web pages:

1. Resource loading: Static resources such as HTML, JS, CSS, and pictures are stored on remote servers, which require dynamic asynchronous pull, and the initialization efficiency is very slow.
2. Rendering mechanism: In the design of the browser, the execution of JS and the layout of the page and Paint are all in the same main thread and cannot be parallelized.
3. Page switching: There is no concept of routing in the browser, which causes the switching experience between pages to completely rely on the capabilities provided by the browser Shell, which will be loaded repeatedly when the page is switched.
4. API capabilities: The browser's security mechanism is a sandbox mechanism based on the same-origin policy. This sandbox mechanism prevents front-end developers from using native system capabilities. You can only use the functions defined by the W3C standard. This is not a problem in the PC-side scenario, but on the mobile-side it is the opposite. Developers hope to be able to call the system interface to implement some more interactive scenarios.
5. Interactive performance: The browser's real-time interactive performance experience is poor, and large-scale rearrangement in complex interactive scenarios limits the UI frame rate. This limitation is particularly serious in low-end mobile devices.
6. Script language, dynamic analysis and execution JS is a JIT language, which means that it needs to be dynamically analyzed and executed. Compared with the AOT language that compiles machine code in advance, the execution performance of JS is much worse.

The difference between mobile development and traditional development.

1. The use of new technologies:
   As the mobile terminal is mainly based on the webkit core, it has better support for new technologies such as HTML5, so new technologies can be used in a wider range; while

the PC-side development requires compatibility with older browsers such as IE in many scenarios. Browser compatibility considerations limit the use of new technologies in some cases.

2. Adaptability of the page:
   Traditional PC-side page development generally chooses to set a fixed width for the page with blanks on both sides, but the mobile-side page usually chooses to display as much content as possible on the phone screen due to its carrier phone screen is much smaller than the PC. This requires mobile pages to be able to fully adapt to mobile phones of various screen sizes and make the most use. From this point of view, it is more difficult to adapt to mobile pages.
3. Page performance:
   The network situation on the PC side is generally relatively stable, and it is connected to the network through a network cable or Wi-Fi; but the mobile side is more complicated. In addition to Wi-Fi, there are 2G, 3G, 4G and even alternate switching among several different network connections. It also often happens that the challenge of unstable network connection to page performance is that the page resources of the mobile terminal cannot be too large, otherwise the page will be inaccessible under bad network conditions, which will seriously affect the user experience.
4. Frame selection
   Due to the instability of the mobile terminal network situation, a lightweight framework needs to be considered. For example, only 9.6K after compression like zepto.js can meet the needs of general business. If you want to build a more complex single-page application, you can choose Frameworks like vue.js have powerful functions, but only more than 20K after volume reduction.

2. Presentation Logic

(a):

1. Inline style

 Sample

Advantage: It is easy to understand and intuitive. The writing method is relatively simple, applicable to a wide range, and has a high priority.

Disadvantage: Poor customization. There is no separation between content and styles, and styles are not reusable and only apply to the current element.

2. Internal/embedded style

```
<html>
    <head>
    <title>Sample</title>
        <style type="text/css">
        div {
            background-color:green;
            color: pink;
        }
        </style>
    </head>
    <body>
        <div>Sample</div>
    </body>
</html>
```

Advantage: Realize style reuse. The degree of reuse is not high and cannot be reused between multiple pages

Disadvantage: Style and content are not completely separated: multiple pages cannot be reused;

3. External style

In CSS file:

```
div {
    background-color: pink;
    color: black;
    font-size:20px;
    font-family:Serif;
    font-weight:5px;
    font-style:italic;
}
```

In HTML file:

```
<html>
    <head>
        <title>Sample</title>
        <link rel="stylesheet" type="text/css" href="div.css" />
    </head>
    <body>
        <div>SAMPLE </div>
    </body>
</html>
```

Advantage: html and CSS are completely separated; CSS is reused to the greatest extent.

Disadvantage: Need to switch frequently while editing.

4. Import

```
<html>
    <head>
        <title>Sample</title>
        <style type="text/css">
            @import url(div.css);
        </style>
    </head>
<body>
    <div>Sample</div>
</body>
</html>
```

Advantage: Clear understanding of file distribution.
Disadvantage: Very bad for maintenance.

5. Use frame

Link to Bulma frame:
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bulma@0.9.1/css/bulma.min.css">

Then follow official documentation to program.

```
<section class="section">
            <div class="container">
                <h1 class="title">
                  Hello World
                </h1>
                <p class="subtitle">
                  My first website with <strong>Bulma</strong>!
                </p>
             </div>
</section>
```

Advantage: Simple and easy to use.
Disadvantage: Deep customization is more difficult.

(b):

This is the CSS customization for the section tag, which will design the <section> tag in the HTML. The CSS transform property of the section is modified in detail, which allows the section to rotate, zoom, etc.

The same content is described below, but it needs to be defined for compatibility with different browsers.
1. -moz-transform for Firefox.
2. -webkit-transform for Safari and Chrome.
3. -o-transform for Opera.
4. -ms-transform for IE 9.
5. transform default

The Scale attribute changes the size of the section.
The rotate attribute will rotate the section.
The translateX attribute will translate the section along the x axis.
The skewX attribute will tilt the section along the x axis.

The web page will display nothing, because there is no defined height, width, etc.

(c):

First download or write the specified font icon resource.
Then put font icon resources into the project.
Create the corresponding font icon in CSS.
Use the following syntax to introduce custom fonts.

```
@font-face {
    font-family: 'YourWebFontName';

    src:url('fontName.eot');
    src:local(''),
        url('fontName.eot?#iefix') format('embedded-opentype'),
        url('fontName.woff') format('woff'),
        url('fontName.ttf') format('truetype'),
        url('fontName.svg#webfontOTINA1xY') format('svg');
        font-weight:normal;
        font-style:normal;
}
```

The font formats that can be supported are:

1. TureTpe (.ttf) format.
2. OpenType (.otf) format
3. Web Open Font Format (.woff) format.
4. Embedded Open Type (.eot) format.
5. SVG (.svg) format.

3. Optimisation and sustainability

(a):

1. According to business needs, choose VPS or dedicated hosting, these two hosting plans will significantly improve the website speed than shared hosting.

2. Use Content Delivery Network (CDN)
CDN can provide content to users more effectively because it is a collection of Web servers distributed in multiple locations around the world. CDN can save up to 60% of bandwidth, reduce the load of any single server, and can also protect websites from DDoS attacks.

3. Reduce HTTP requests
Too many HTTP requests will greatly reduce the speed of the website. For each page component of the website, an HTTP request is issued. Therefore, if you have a large number of page components on your website, the page rendering will be longer.

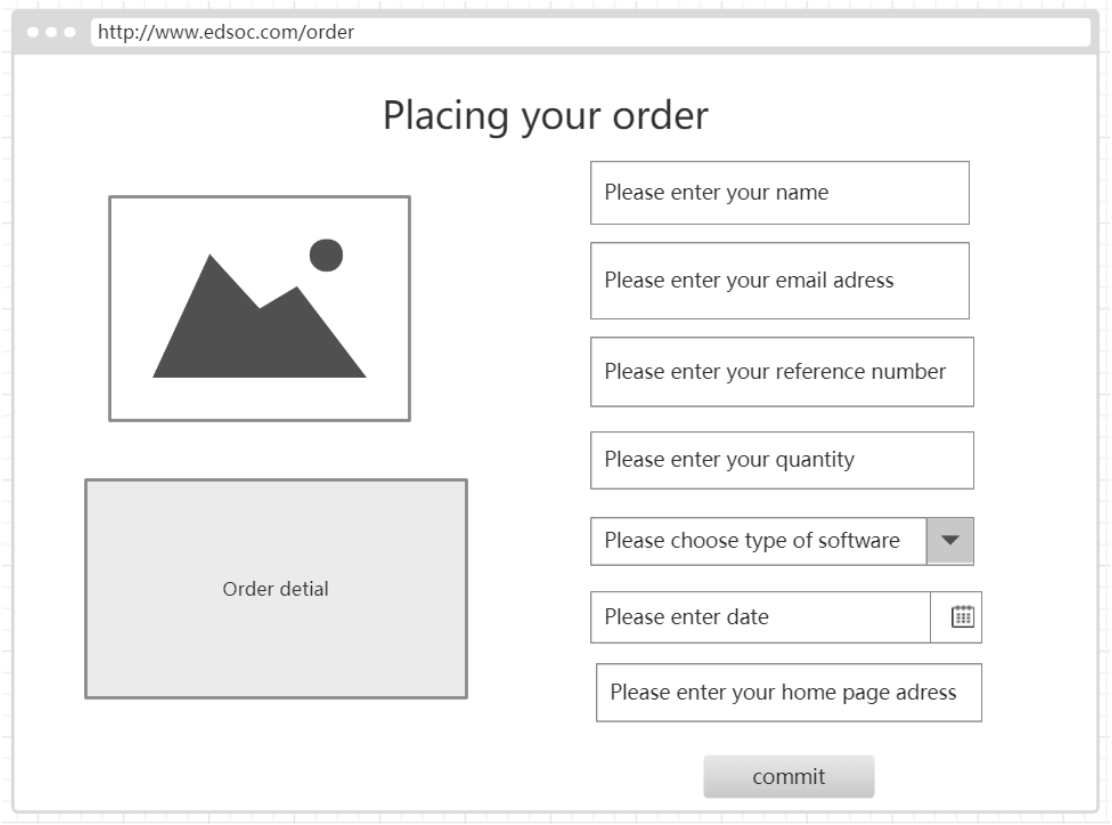
4. Image optimization
Large image size has a great impact on website speed, so it is necessary to optimize the image by compressing the image, cropping the image, or changing the resolution of the image.

5. Enable caching
Enabling caching for website can improve the user experience because it stores website version on the user's browser and provides the same version before website is updated.

6. CSS and JS optimization
Every time a user visits a website, HTML, CSS, and JavaScript files increase the number of requests made by the website, which affects the speed of the website. This number can be reduced by shrinking and combining files, using asynchronous loading for CSS and JavaScript files, and delaying JavaScript loading.

7. DNS lookup
The speed of page loading depends on the time it takes for DNS lookups. If you are using a slow DNS, the time for the browser to find your website will increase. To speed up this process, it is necessary to switch to a faster DNS provider.

(b):




```
<form id="placeOrder" name="placeOrder" method="POST" action="process_order.php">
    <p>
        <label for"username">Username</label>
        <input type="text" name="username" id="username">
    </p>
    <p>
        <label for"email">E-mail</label>
        <input type="email" name="email" id="email">
    </p>
    <p>
        <label for"refeNo">Reference number</label>
        <input type="number" name="refNo" id="refNo" min=0>
    </p>
    <p>
        <label for"quantity">quantity</label>
        <input type="number" name="quan" id="quan" min=0>
    </p>
    <p>
        <label for"type">Type of software</label>
        <select>
```

```
            <option value="none">none</option>
            <option value="ide">IDE</option>
            <option value="matlab">MatLab</option>
            <option value="search">Search</option>
            <option value="word">Word</option>
          </select>
    </p>
    <p>
        <label for”date”>Date</label>
        <input type="date" name="date" id="date">
    </p>
    <p>
        <label for”url”>Home page adress</label>
        <input type="password" name="url" id="url">
    </p>
    <p>
        <input type="submit" name="button" id="button" value="submit" >
    </p>
</form>
```

The user fills in the information step by step according to the prompts of the UI, and sends a post request to the server at the end.

(c):

Perceivable: plug-ins should be added to allow blind people to listen to the content of the page. Also, for color-blind users, images should have text counterpart descriptions.

Operational: Users should be able to place orders and interact with web pages without using a mouse, and navigate through screen readers.

Understandable: The order website needs to be simplified, the text should not be more complicated than it needs, and the website should behave in a predictable way.

Robustness: All services provided should not be restricted by the platform or operating system. This can prevent people from providing some imperfect services, which will be unavailable to most people due to hardware/software limitations. For example, browsers on different devices can use the order website together, and the navigation should be consistent.