How Do You Improve "Time to Usable"?

Improving the "time to usable", or in other words the interval from when a user requests a page to when they can meaningfully interact with it, is crucial for enhancing user experience.

Let's explore effective strategies to achieve a faster time to usable.

Begin by focusing on the content that the user sees first. This means loading essential elements immediately and deferring non-critical components.

For example, you could try to implement lazy loading.

We have already covered lazy loading, but here is an example of lazy load images as a refresher:

```
<img src="image.jpg" loading="lazy" alt="Description">
```

All you need to do is add the `loading="lazy"` attribute to image tags. This defers the loading of images until they are needed, typically when they enter the user's viewport. By loading images and videos only when they enter the viewport, we are conserving bandwidth as well as causing the speed up of initial rendering.

Normally, a webpage loads all images at once, even those not visible on the screen. With lazy loading, images below the fold, or in other words off-screen, they are only loaded when the user scrolls down to them.

And here is an example of lazy load iframes:

```
<iframe src="video.html" loading="lazy"></iframe>
```

You can apply the same attribute that you did before, but this time to iframes, and the same concept will apply.

You can also use the `defer` attribute to delay non-critical JavaScript until after the initial page load. Here is an example of us doing just that:

```
<script src="non-critical.js" defer></script>
```

You could also consider minimizing render-blocking resources.

Render-blocking resources delay the page from becoming interactive.

To avoid this, you could try loading CSS asynchronously. For non-critical CSS, use the `media` attribute to load stylesheets conditionally. Here is an example of us doing just that:

```
<link rel="stylesheet" href="print.css" media="print">
```

By specifying media value as `print`, the `print.css` will only be loaded when the page is printed or previewed on screen. In this next example, by passing `min-width: 800px` as the value to the `media` attribute, we are saying we only want `desktop.css` to load for screens wider than 800px:

```
<!-- This stylesheet is only loaded on screens wider than 800px -->
<link rel="stylesheet" href="desktop.css" media="(min-width: 800px)">
```

This saves bandwidth by skipping unnecessary styles.

So, in conclusion, by implementing these strategies, you can significantly reduce the "time to usable" for your website, ensuring users can interact with your content promptly.

Questions

What is the best way to describe "time to usable"?

The time intervals in the `setTimeout` method.

The time from when a user requests a page to when they can meaningfully interact with it.

Correct!

The amount of time it takes for the server to process a user's request and return a response.

The period during which a user can only view content but not interact with it.

Which of these sentences is true?

Normally, a webpage loads only images that are visible on the screen.

Normally, a webpage loads only images that are not visible on the screen.

Normally, a webpage loads some images, even those not visible on the screen.

Normally, a webpage loads all images at once, even those not visible on the screen.

Correct!

What do Render-blocking resources do?

Render-blocking resources stop pages loading completely.

Render-blocking resources can crash websites.

Render-blocking resources delay the page from becoming interactive.

Correct!

Render-blocking resources can cause power outages.

Navigated to How Do You Improve "Time to Usable"?