vijfhart
IT-OPLEIDINGEN

RESPONSIVE **WEB DESIGN**

Introduction round

- Role / background
- Hobbies / interests / family / pets
- What do you hope to learn?

vijfhart
IT-OPLEIDINGEN

Introduction

- Welcome to the Responsive Web Design (RWD) workshop!
- Learn to create modern, responsive web pages that work on various screen sizes.
- Suitable for beginners in HTML, CSS, and JavaScript looking to develop frontend skills.
- Gain hands-on experience building responsive web pages during the workshop.
- Understand the importance and benefits of Responsive Web Design (RWD).
- Explore differences between responsive, adaptive, and fixed design approaches.

vijfhart
IT-OPLEIDINGEN

Importance and Benefits

- ❑ Enhanced User Experience
- ❑ Mobile Usage is Rising
- ❑ One Website, Multiple Devices
- ❑ Improved SEO Performance
- ❑ Cost-Effective and Efficient
- ~~❑ Future-Proofing~~

Responsive, Adaptive and Fixed Design

Responsive Design: Adapts fluidly to any screen size for optimal user experience.

Adaptive Design: Uses specific layouts for different screen sizes, but may not cover all device variations.

Fixed Design: Maintains a static layout, leading to potential usability issues on different devices

Responsive vs. Adaptive: Responsive design is more flexible and compatible with a wider range of devices.

Considerations: Content prioritization, performance, and development complexity influence design choices.

Balancing UX and Development: Responsive design strikes a good balance between usability and resources

vijfhart IT-OPLEIDINGEN

Mobile first vs desktop first approach

Mobile First Approach: Adapts fluidly to any screen size for optimal user experience.

Emphasizes Content and Performance: Focuses on delivering essential content and optimizing performance for smaller devices.

Progressive Enhancement: Builds upon the core mobile experience, adding more features for larger screens.

Designing for Constraints: Considers limited screen space, touch-based interactions, and slower network connections.

Desktop First Approach: Starts with designing for larger screens and then adapts to smaller devices.

Emphasizes Feature-Rich Experience: Prioritizes rich visuals, complex interactions, and advanced functionalities.

vijfhart
IT-OPLEIDINGEN

Basics of RWD

- **Fluid Grids:** Utilize flexible grid systems that automatically adjust the layout based on screen size.
- **Relative Units:** Use percent, em, and rem for sizing elements, allowing them to scale proportionally.
- **CSS Media Queries:** Apply conditional styles based on screen characteristics to adapt the layout and design.
- **Flexible Images and Media:** Ensure images and media adjust responsively using techniques like max-width and object-fit.
- **Scalable and Readable Text:** Set appropriate unit types, consider font scaling, and maintain optimal line-height.
- **Performance Optimization:** Implement techniques to enhance loading time, such as optimizing images and lazy loading.

HTML, CSS and JS

- **HTML (Hypertext Markup Language):** The standard markup language used to structure content on the web.
- **CSS (Cascading Style Sheets):** The style sheet language used to define the visual presentation of a web page.
- **JS (JavaScript):** The programming language that allows for dynamic and interactive elements on a web page.

HTML, CSS and JS

- **HTML Structure:** Organizes and structures the content of a web page using tags and elements.

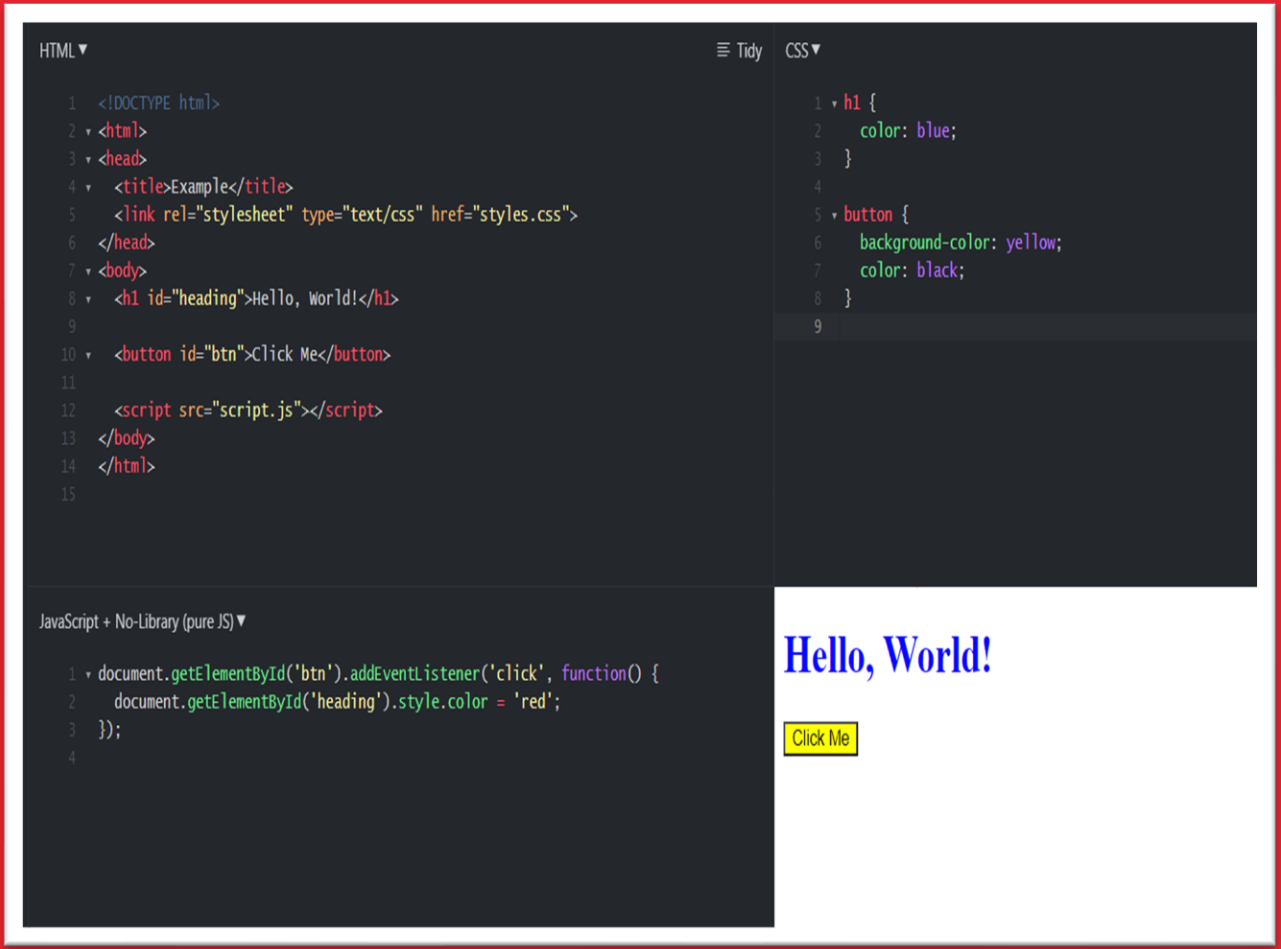
- **CSS Styling:** Controls the appearance of HTML elements, including layout, colors, fonts, and animations.

- **JS Interactivity:** Adds interactivity to web pages by enabling dynamic content, event handling, and client-side functionality.

- **Integration:** HTML, CSS, and JS work together to create a seamless and engaging user experience.

EXAMPLE

How can you create an HTML, CSS, and JavaScript code that displays the text "Hello, World!" and includes a button labeled "Click Me"? When the button is clicked, how can you make the text color of "Hello, World!" change dynamically?



```html
<!DOCTYPE html>
<html>
<head>
  <title>Example</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1 id="heading">Hello, World!</h1>

  <button id="btn">Click Me</button>

  <script src="script.js"></script>
</body>
</html>
```

```css
h1 {
  color: blue;
}

button {
  background-color: yellow;
  color: black;
}
```

```javascript
document.getElementById('btn').addEventListener('click', function() {
  document.getElementById('heading').style.color = 'red';
});
```

Exercise 1

"Write an HTML, CSS, and JavaScript code to create a web page with a heading that says 'Hello, World!' and a button labeled 'Click Me'. When the button is clicked, change the color of the heading text to red. Additionally, include a second button labeled 'Reset'. When this 'Reset' button is clicked, restore the original color of the heading text to its initial state. Test your code to ensure both buttons function correctly."

Fluid grids and relative units (percent, em, rem)

- **Fluid Grids:** Designing layouts using a flexible grid system that adjusts proportionally to different screen sizes.
- **Percentage (%) Units:** Using percentage values to define widths, heights, and margins, allowing elements to scale relative to their parent container.
- **EM Units:** Using the "em" unit to define sizes based on the current font size of the parent element, enabling scalable and relative sizing.
- **REM Units:** Utilizing the "rem" unit to define sizes based on the root (document) font size, providing consistent and scalable sizing across the entire document.
- **Responsive Typography:** Implementing relative units like "em" or "rem" for font sizes to ensure text adjusts proportionally to the screen size.
- **Media Queries:** Combining fluid grids and relative units with CSS media queries to create responsive layouts that adapt to different devices.

vijfhart
IT-OPLEIDINGEN

Exercise 2

Create a responsive grid layout using fluid grids and relative units. The layout should have three equal-width columns and display four boxes in each row. Each box should have a background color, text content, and padding. Implement the following requirements:

- Use CSS Grid to define the grid layout with three equal-width columns.
- Use relative units (percent, em, or rem) to size the boxes and provide responsive behavior.
- Apply appropriate spacing between the boxes.
- Implement a hover effect that changes the background color of the boxes.
- Ensure the layout remains responsive and adapts to different screen sizes.
- Feel free to experiment with different colors, font sizes, and paddings to make your design visually appealing.

Flexible images and media (max-width, object-fit)

- **Max-width Property:** Use the max-width property to ensure that images and media elements do not exceed their container's width, allowing them to scale down proportionally on smaller screens.
- **Object-fit Property:** Apply the object-fit property to control how images and media fill their containers.
- **Responsive Video Embeds:** Use responsive techniques, such as wrapping videos in a container with a fixed aspect ratio and applying CSS rules for max-width, to ensure videos adapt to different screen sizes.
- **Retina Display Optimization:** Serve high-resolution images (2x or 3x) for devices with retina displays using media queries and the srcset attribute, improving image quality and clarity.
- **Image Loading Performance:** Optimize image loading by using modern image formats (e.g., WebP), compressing images, and implementing lazy loading techniques to defer the loading of off-screen images until they are needed.

Exercise 3

Create a web page that displays an image and adjusts its size responsively using the "max-width" property and scales it using the "object-fit" property. The image should initially have a width of 300 pixels and a height of 200 pixels. When the viewport width is smaller than the image width, the image should scale down proportionally. When the viewport width is larger than the image width, the image should maintain its original size. Additionally, add a button that toggles the object-fit property between "contain" and "cover" when clicked.

Requirements:
Use HTML, CSS, and JavaScript to implement the solution.

- The image should have a width of 300 pixels and a height of 200 pixels initially.
- Use the "max-width" property to ensure the image scales down proportionally when the viewport width is smaller than the image width.
- Use the "object-fit" property to control how the image fits within its container.
- Bonus: Add a button that toggles the object-fit property between "contain" and "cover" when clicked.

Hint:
You can use the "addEventListener" method in JavaScript to handle the button click event.
Update the object-fit property of the image element based on the button's state.

CSS media queries and breakpoints

- CSS media queries allow you to apply different styles based on various device characteristics such as screen size, resolution, orientation, and more.
- Media queries use the @media rule to define different styles for different conditions.
- Breakpoints in media queries are specific screen widths at which the layout or styling of a webpage changes.

Commonly used media query breakpoints include:

- Small screens (e.g., smartphones): @media (max-width: 767px)
- Medium screens (e.g., tablets): @media (min-width: 768px) and (max-width: 1023px)
- Large screens (e.g., desktops): @media (min-width: 1024px)
- Media queries can target other device characteristics like aspect ratio, device type, and more.

Exercise 4

Write a responsive webpage layout that adjusts its appearance based on the screen size. The webpage should have the following requirements:

On small screens (max-width: 767px):

- The heading should have a font size of 20 pixels.
- The paragraph text should have a font size of 14 pixels.
- Clicking the "Click Me" button should toggle the background color of the page between light blue (#E3F2FD) and light pink (#F8BBD0).

On medium screens (min-width: 768px) and (max-width: 1023px):

- The heading should have a font size of 24 pixels.
- The paragraph text should have a font size of 16 pixels.
- Clicking the "Click Me" button should toggle the background color of the page between light green (#DCEDC8) and light purple (#D1C4E9).

On large screens (min-width: 1024px):

- The heading should have a font size of 30 pixels.
  ☐The paragraph text should have a font size of 18 pixels.
  ☐Clicking the "Click Me" button should toggle the background color of the page between light yellow (#FFF9C4) and light orange (#FFE0B2).

CSS Flexbox and Grid

CSS Flexbox:

- Provides a flexible layout system for arranging elements in a single dimension (row or column).
- Allows for easy alignment, distribution, and ordering of elements.
- Simplifies the creation of responsive and dynamic layouts

CSS Grid:

- Offers a powerful grid-based layout system for arranging elements in rows and columns.
- Supports both fixed and flexible sizing of grid tracks.
- Enables precise control over grid item placement and alignment

Flexbox vs. Grid:

- Flexbox is best suited for arranging items within a single dimension, such as a row or column.
- Grid is ideal for creating complex grid-based layouts with rows and columns.
- They can be used together to achieve advanced layout requirements.

vijfhart
IT-OPLEIDINGEN

Scalable and readable text across devices

Importance of Scalable Text:

- Ensures that text remains legible and readable across different devices and screen sizes.
- Enhances user experience by accommodating varying viewing conditions.
- Supports accessibility for users with visual impairments.

Techniques for Scalable Text:

- Using relative units like em or rem for font sizes.
- Applying media queries to adjust font sizes based on screen dimensions.
- Implementing fluid typography techniques to maintain optimal text scaling.

Readability Considerations:

- F Choosing appropriate font families and styles for readability.
- Setting appropriate line-height and letter-spacing for improved legibility.
- Ensuring sufficient contrast between text and background colors.

Testing and Optimization:

- FChecking text legibility on different devices and screen resolutions.
- Conducting user testing to assess readability and adjust as needed.
- Optimizing performance by balancing text quality and load times

vijfhart
IT-OPLEIDINGEN

Exercise 5

Create a webpage layout that consists of two sections: a header and a main content area. The header should contain a logo on the left and a navigation menu on the right, aligned horizontally. The main content area should display a grid of four items, each with an image and a caption.

On small screens (max-width: 767px):

- Use CSS Flexbox to align the logo and navigation menu horizontally in the header.
- Utilize CSS Grid to arrange the four items in a 2x2 grid layout in the main content area.
- Apply appropriate spacing and styling to achieve a visually appealing design.
- Make the layout responsive, ensuring that it adapts gracefully to different screen sizes.
- Implement a JavaScript functionality where clicking on an item in the grid will display a modal with additional details about the item

Unit types, font scaling and line-height

- **Unit types for font sizes:** Pixels (px), em, rem, and percentages (%).
- **Font scaling techniques:** Using relative units (em, rem) for font sizes, applying media queries for responsive font scaling.
- **Importance of line-height:** Enhances readability and visual spacing between lines of text.
- **Adjusting line-height:** Consider font size, line length, and content to determine an appropriate line-height value.
- **Unitless line-height:** Using a unitless value maintains a consistent line-height regardless of font size.
- **Responsive typography:** Implementing fluid typography techniques for seamless font scaling across different devices.

Exercise 6

Create a webpage that showcases a responsive text section with the following: requirements

- The base font size for the webpage should be 16 pixels (px).
- Implement three different sections with headings and paragraphs, each using a different unit type for font sizes: em, rem, and percentage (%).
- Apply appropriate line-height values to ensure readability and visual spacing between lines of text.
- Use media queries to adjust font sizes and line-heights for different screen sizes: smaller screens (max-width: 767px), medium screens (min-width: 768px) and (max-width: 1023px), and larger screens (min-width: 1024px).
- Test the webpage on different screen sizes to ensure the responsive behavior and font scaling.

Your goal is to create a visually appealing and readable text section that scales appropriately across different devices.

Accessibility and web typography best practices

- **Importance of accessibility:** Ensuring that web content is accessible to users with disabilities, including visual impairments, hearing impairments, and motor disabilities.
- **Contrast ratio and color accessibility:** Following WCAG guidelines for text-color contrast ratios to ensure readability for all users, especially those with visual impairments.
- **Proper use of heading tags:** Using heading tags (h1, h2, h3, etc.) in a hierarchical manner to provide a clear structure and aid screen readers in understanding the content.
- **Semantic HTML:** Using semantic HTML elements (such as <nav>, <article>, <section>, etc.) to provide meaningful structure and enhance accessibility.
- **Alternative text for images:** Including descriptive alt text for images to provide context and ensure accessibility for visually impaired users who rely on screen readers.
- **Keyboard accessibility:** Designing web pages that can be navigated and interacted with using only a keyboard, without relying on mouse or touch input.

vijfhart
IT-OPLEIDINGEN

Responsive images and media

- **Image optimization:** Optimize images for different screen sizes and resolutions.
- **Use srcset attribute:** Provide multiple image sources for different resolutions or sizes.
- **Utilize the picture element**: Specify different image sources for different scenarios.
- **Responsive media queries:** Adjust the size and layout of media elements based on viewport size.
- **Implement lazy loading:** Defer loading of non-visible images or media.
- **High-resolution displays:** Provide high-resolution images for devices with high pixel densities.

vijfhart
IT-OPLEIDINGEN

Exercise 7

Create a responsive webpage that includes an image and a video element. Implement the following requirements:

- The image should change based on the screen size. Use "image-large.jpg" for screen widths larger than 1200 pixels, "image-medium.jpg" for screen widths between 768 pixels and 1200 pixels, and "image-small.jpg" for screen widths smaller than 768 pixels.
- The video should be displayed with controls and should support both MP4 and WebM formats. Use "video.mp4" for the MP4 source and "video.webm" for the WebM source.
- Ensure that the image and video maintain their aspect ratios and adjust their sizes responsively.
- Implement lazy loading for the image. The image should only load when it enters the viewport.
- Apply the provided CSS styles to ensure the image is displayed with a maximum width of 100% and the video has a responsive layout.

Write the HTML, CSS, and JavaScript code to achieve the above requirements.

Responsive images and media

- **User experience:** Fast-loading websites improve user experience, leading to higher satisfaction and engagement.
- **Conversion rates:** Slow loading times can decrease conversions, emphasizing the need for performance optimization.
- **Search engine ranking:** Loading time affects search engine rankings, impacting visibility and organic traffic.
- **Mobile responsiveness:** Mobile users expect fast-loading websites, making performance optimization crucial.
- **Reduced bounce rates:** Performance optimization reduces bounce rates and encourages further engagement.
- **Competitive advantage:** A fast-loading website sets businesses apart and builds trust with users.

Tools to measure and improve performance

- **Performance measurement tools:** Use tools like Google PageSpeed Insights, GTmetrix, or WebPageTest to assess website performance and obtain actionable recommendations.
- **Code profiling tools:** Utilize browser-based developer tools (e.g., Chrome DevTools, Firefox Developer Tools) to analyze code performance and identify areas for improvement.
- **Caching mechanisms:** Implement browser caching, server-side caching, and CDN caching to store static resources and reduce server requests.
- **Minification and compression:** Minify HTML, CSS, and JavaScript files and compress them using tools like Gzip or Brotli to reduce file size.
- **Image optimization:** Optimize images by resizing, compressing, and using modern formats like WebP.
- **Continuous optimization:** Regularly monitor performance, optimize code and assets, and stay updated with latest techniques and best practices.

vijfhart IT-OPLEIDINGEN

RESPONSIVE **WEB DESIGN**