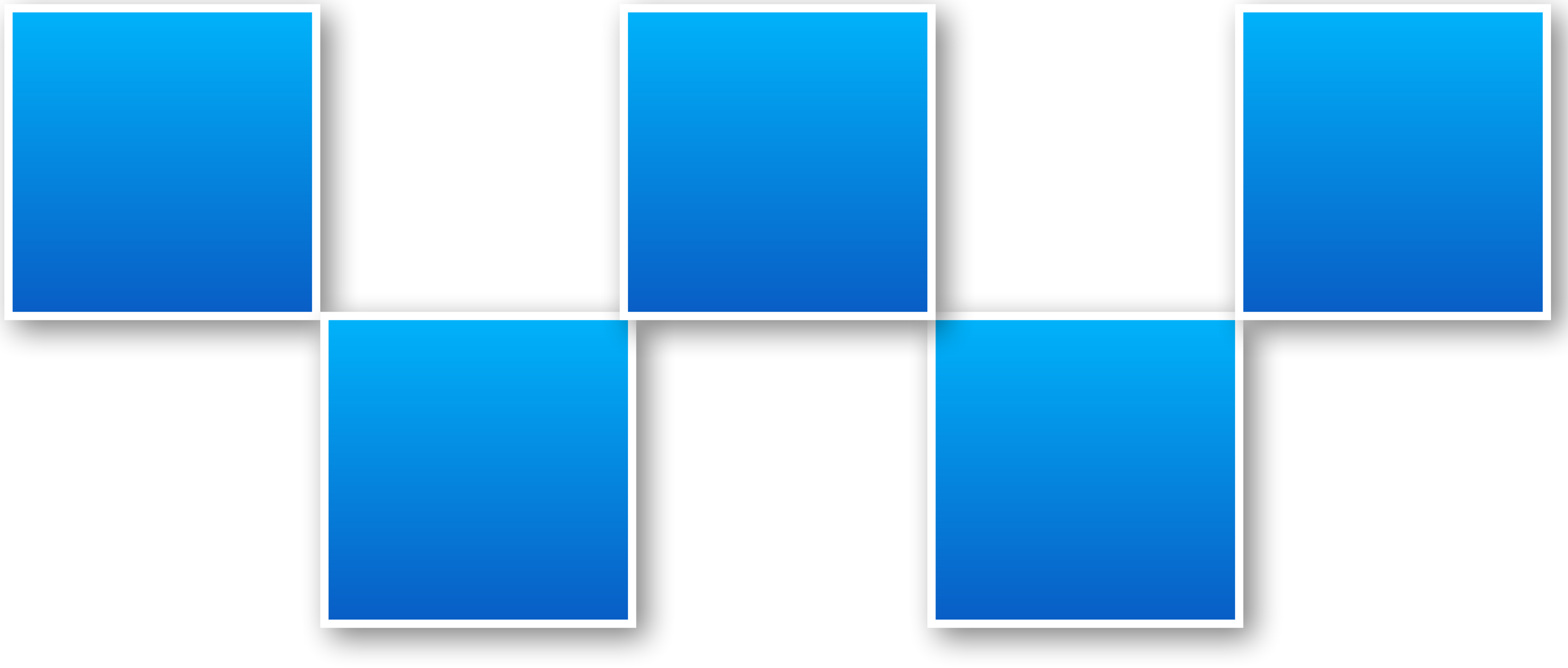

Mobile-first Responsive Web Design

Progressive enhancement

- Progressive enhancement views web design in a series of layers. The first layer is the content. Combine that with semantic markup to create structured content. After you’ve got the basics out of the way, you add a presentation layer using CSS and a behavior layer using JavaScript. You never assume the browser supports those features, but if it does, visitors get a better experience.

Mobile-first Responsive Web Design

- Mobile-first Responsive Web Design (RWD) techniques that start from a mobile template. Despite its simplicity, there is a lot of power that comes from this approach.
- Mobile-first RWD isn’t that different from progressive enhancement. Actually many people call it content-first design instead because content is the first layer of progressive enhancement. Starting from the most basic document not only reaches the most people, it also has beneficial side effects.

Mobile-first Responsive Web Design

Feature phones

- Basic HTML
- Simple layout
- Small images
- Limited CSS and JS

Smartphones

- Add newer HTML5 features if supported
- Simple layout
- Small images, but bigger than feature-phone size
- More CSS and JS

Tablets

- Because there is more room, we can add optional content like sideb
- Multiple column layouts
- Larger images

Desktops

- Add widescreen layouts
- Larger images

Steps of Enhancing the example using Mobile-First approach

- In the Index.html, comment out the <iframe> section that is for google map. That iframe causes a lot of files to download. Notice that even though you set it to display:none, some web browser still download it.

- Note the "visit" is not so essential on this page, it should be moved after the "oncolor" main section, especially on the one-column mobile version.

    - After that we need to fix the float content issue in the CSS file, because we change the order of "visit" and "oncolor". Basically, we need to swap the float left and float right.

- In the CSS file, we also want to move the mobile media query block above the desktop media query to be consistent with our mobile-first approach in the index.html.

Using conditional comments

- Be aware that the older version of Internet Explorer does not support media query. Fortunately, Microsoft provides conditional comments to overcome it.
- Create a blank text file called layout.css and copy the desktop rules into it. After you copy them, remove the rules and the surrounding media query from styles.css.
- Notice that the header image is so big. Even though we set it to display:none, it is still get downloaded. Move the header from the styles.css to the layout.css. So it is only downloaded when the desktop screen is detected.

Using Mobile-first Responsive Web Design

- Make the HTML as simple as possible and swap the order of the CSS so that the mobile version is first.
- Fix CSS background images so that only one file gets downloaded per image. Make sure display:none is being used appropriately.
- Supply different source files for <img> tags at different screen resolutions. Make sure the right size image is downloaded.
- Use JavaScript to add Google Maps to the page when the browser can support it and the document is wide enough to accommodate it.

Add the map using JavaScript

- We'll add the map back if the browser window is wide enough. As we know that if we hide and show the map using CSS, the resources for the map will still get downloaded. So we're going to need to use JavaScript to add the map when appropriate.
- First, we need to add a link to the map. To do that, we'll need a <div> that our JavaScript can reference. The <div> will contain a <p> tag with a link to the map.
- Add the JavaScript to the page. Because the map is a nice-to-have feature and not essential, We're going to add our JavaScript as the very last thing on the page before the closing </body> tag. That way makes a page load faster. The browser will parse all of the HTML and CSS before it gets to the JavaScript. Our visitors will not be waiting for the map code to load.

Summary

- Adding media queries to an existing desktop site may make it look good on mobile, but doesn't mean that it is mobile optimized.
- Because most mobile browsers don't support plug-ins, there are fewer tools to assist mobile web developers.
- Mobile-first Responsive Web Design helps optimize web pages by making sure that smaller resources are downloaded by default.
- Mobile-first RWD is another form of progressive enhancement that uses screen size to determine how to enhance web pages.
- Designing for mobile first forces you to focus on what really matters, thus helping you remove badly designed, unnecessarily complicated, or unwanted code from pages.
- Internet Explorer 8 and below do not support media queries. Conditional comments are a workaround.
- JavaScript can supplement media queries by testing for screen size and adding content when appropriate.
- Instead of designing breakpoints based on the typical screen resolutions, let the content dictate the resolutions at which you need to modify the layout.