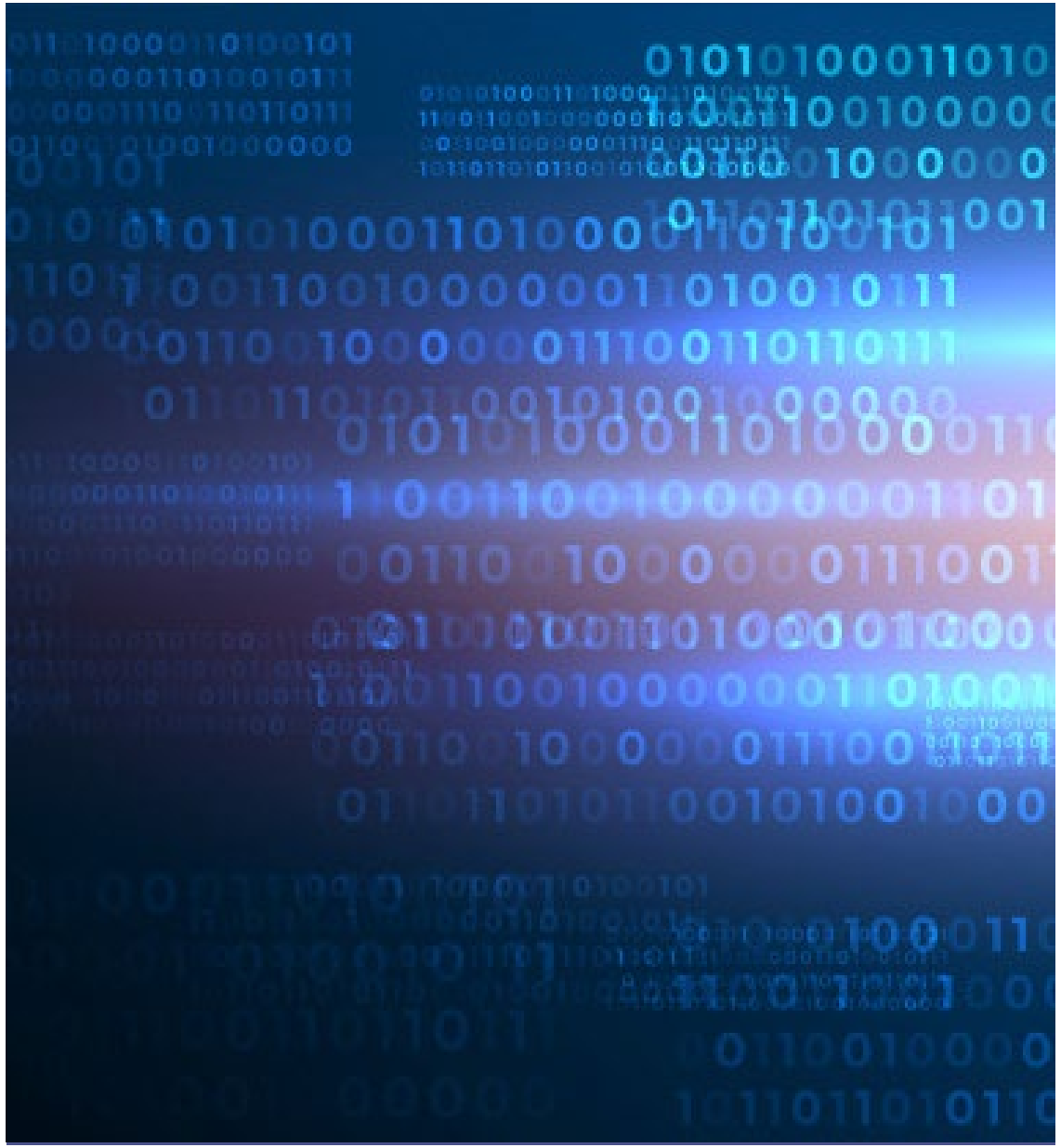
HTML, CSS & JS

REFRESHER



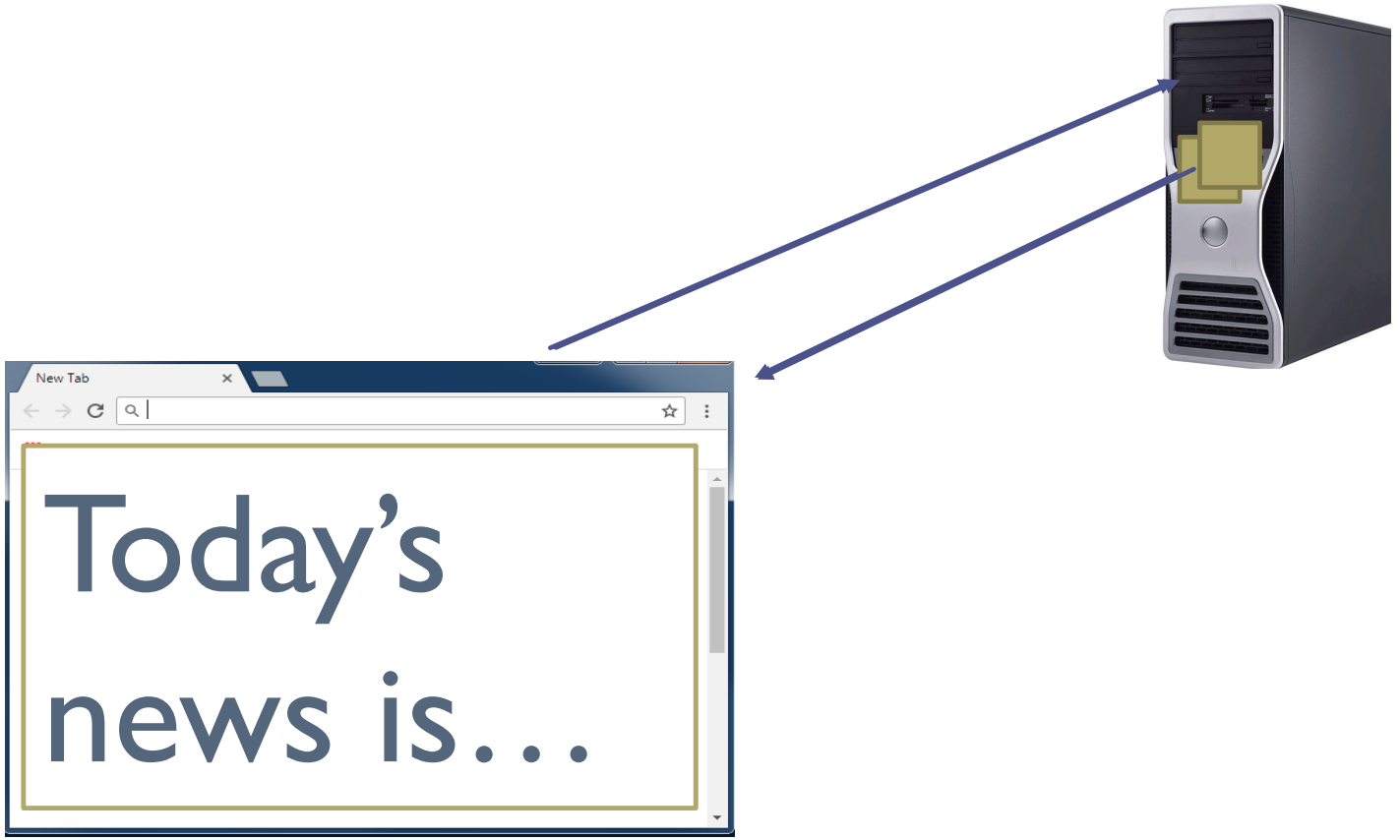
AGENDA

- How does the internet / web pages work?
- HTML basics
- Accessibility issues
- CSS basics
- Responsive web pages
- JS basics
- External Libraries (JQuery, Bootstrap & Font Awesome)

HOW DOES THE INTERNET / WEB PAGES WORK?

1

THE TRADITIONAL APPROACH



THE TRADITIONAL APPROACH

- The browser requests the page from the server (HTML)
- The HTML page contains links to resources (images, css, js)
- The browser requests each of these resources
- The browser parses the HTML + CSS to display the page
- The brower runs any js defined to be run on rendering the page

URLS

- All pages and resources are accessed via their URL

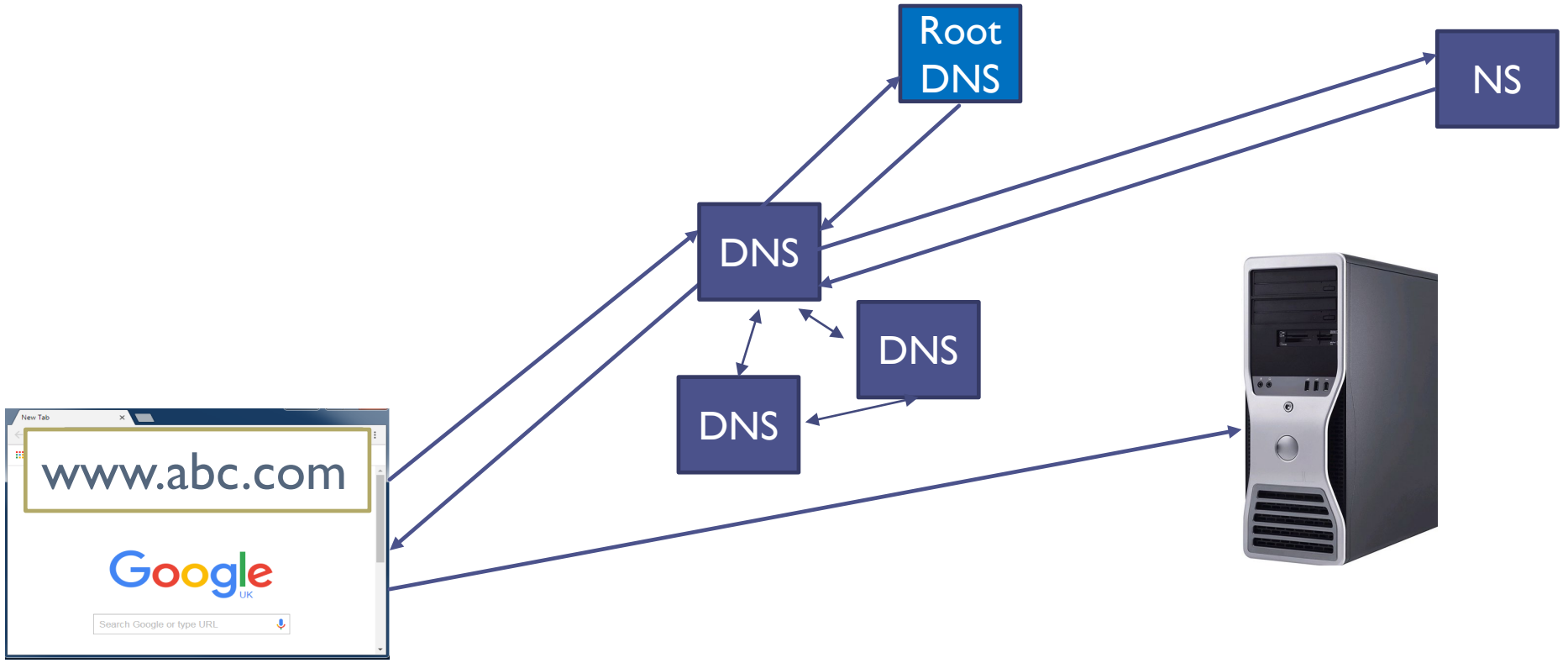
NAMESERVERS + DNS

- Every domain name is listed on 1 or more nameservers
- The nameserver provides a lookup for the IP address associated with each domain + subdomain
- DNS servers cache this data

NAMESERVERS + DNS

- IANA manages the list of top level domains. The list is stored in the root zone database
- There are 13 root DNS servers– these contain the location of the global DNS servers for each top level domain
- The global DNS servers contain the nameserver records for each domain name

NAMESERVERS + DNS



HTML BASICS

2

HTML BASICS

- HTML is a *markup* language similar to XML
- HTML page contains a DOM (Document object model) – defines the page as a hierarchical set of objects
- Each object is defined as an element, specified by tags

```
<p>some para</p>

<input type="text" id="name" >
```

- Current version is HTML5

HTML VS XML

- XML is case sensitive
- Every tag must have a closing tag or be self closing in XML
- Attributes in XML must always be key value pairs
- HTML has a limited set of pre-defined tags.

HTML BASICS

```
<input type="text" name="firstname" id="firstname" required>
```

HTML BASICS

```
<!DOCTYPE html>
<html>
      <head>
            …
      </head>
      <body>
         …
      </body>
</html>
```

HTML BASICS

SEE HTML CHEAT SHEET

ACTIVITY - CREATE A BASIC HTML PAGE

- Create a simple HTML page containing a form which will allow us to enter information for a customer (e.g. name and address). Don't worry about styling for now.
- Use a dropdown/select box
- Display a list of instructions using the list tags

- Explore the page in the browser using the dev tools

HTML BASICS – FORMATTING TEXT

- `<b>` Bold text
- `<strong>` Strong-importance text, typically displayed as bold
- `<i>` Italicized text
- `<em>` Emphasized text, typically italicized
- `<sup>` Superscript text
- `<sub>` Subscript text

HTML BASICS – ADDITIONAL TAGS

- `<br>` Line break
- `<hr>` Horizontal rule
- `<q>` Short quote, typically displayed in quotes
- `<blockquote>` Block quote, typically indented
- `<address>` Address, typically italicized
- `<bdo>` Specifies direction of text
- `<abbr>` Explains an abbreviation

HTML BASICS – ENTITIES

- Currencies `&pound; &euro;`
- Accented characters `&eacute; &aring;`
- Less than / greater than `< >`
- Math symbols `&sum; &infin;`

HTML5 SEMANTIC TAGS

```
<body>
  <nav>...</nav>
  <article>...</article>
  <aside>...</aside>
  <main>...</main>
  <section>...</section>
</body>
```

HTML5 LABEL

```
<label for="email">Email:</label>
<input type="text"
name="email" id="em"/>
```

HTML INPUTS

The input element can have the following types:

(HTML 4)

- text, password, checkbox, radio, file, hidden, button, reset, submit, image, imagesubmit

(HTML 5)

- color, date, datetime-local, time, month, week, range, search, number, email, url, tel

ACTIVITY - EXPLORING INPUTS

- Add some additional inputs to your form to explore the different types and see how they render in the browser

ACCESSIBILITY

2

WHAT IS ACCESSIBILITY

- the process of making sure that your website is accessible to everyone

P.O.U.R.

- P. - Perceivable
  - Everyone has a similar experience on your website irrespective of ability
- O. - Operable
  - Everyone can interact with your website irrespective of ability
- U. - Understandable
  - Everyone can understand what to do on your website irrespective of ability
- R. - Robust
  - Everyone can use a range of technologies to access your website.

ACTIVITY - CHECK OUT A BAD UI

- tee.mn/frustrated

THINGS TO THINK ABOUT

- Do all the images have alt tags?
- Have you used headings appropriately?
- Do your form fields have labels?
- Have you used colours that may be difficult to differentiate?
- Have you fixed font sizes (and made them very small?)
- Have you made use of ARIA?

ARIA

- Aria = "Accessible Rich Internet Applications"
  - Enables you to add semantics and metadata to HTML content
  - Used in conjunction with screen readers, to make UI accessible

- Aria features available in HTML5:
  - Accessible forms
  - Landmark roles
  - Live regions
  - Audible validation

ARIA WITH FORMS

- You can link an input field to a tag that describes it
  - Use the `aria-describedby` attribute
  - Screen readers will announce the descriptive text on input focus

```
<label for="fromText">From airport:</label>
<input id="fromText" aria-describedby="fromDescriptor">
<span  id="fromDescriptor">Airport you're travelling from</span>
```

- You can tell a screen reader what to say
  - Use the `aria-labelledby` attribute
  - Screen readers speak the specified items

```
<label id="rangeLabel" for="rangeText">Travel within</label>
<input id="rangeText" aria-labelledby="rangeLabel rangeText rangeUnit">
<span  id="rangeUnit">days</span>
```

ARIA WITH CONTENT

- You can designate a tag as a "live" region
  - Tells the screen reader to announce content changes

```
<someElement

  aria-live="off" | "polite" | "assertive"

  aria-atomic="true" | "false"

  aria-relevant="additions" | "removals" | "text" | "all" > …
```

ACTIVITY - ACCESSIBILITY CHALLENGE

- Try to order a pizza from Dominoes only using your keyboard... I'd like a medium pizza with a stuffed crust and extra olives please!
- Check out Amazon – A Promised Land : Barack Obama, hardback edition – how useful are the "from the publisher" images?
- Look at https://w3.org/wai/aria/apg
- Look at Chrome plugin - Silktide - website accessibility simulator

CSS BASICS

4

CSS

- Describes how HTML elements are to be displayed on screen
- The "cascade" is the algorithm that browsers use to determine which rules apply to HTML elements.
- "the property value from the origin with the highest precedence gets applied, even if the selector from a lower precedence origin or layer has greater specificity."

INLINE STYLES

```
<h1 style="color: red">
  Here's an h1 element
</h1>

<p style="color: blue">
  Here's a p element
</p>
```

```
<style>
  h1 {
    color: red;
  }

  p {
    color: blue;
  }
</style>
```

STYLE RULES

- Each style rule has a {} declaration block
- The declaration block contains a series of declarations
- Each declaration is a property:value pair, terminated by ;

```
h1 {
  color: red;
  font-style: italic;
}

p {
  color: blue;
  margin-bottom: 10px;
  border: lightblue solid 5px;
}
```

USING EXTERNAL STYLE SHEETS

- If you want to define common styles for multiple web pages:
  - Define the style rules in a `.css` style sheet file

```
body {
  font-family: consolas, sans-serif;
}

h1 {
  color: red;
  font-style: italic;
}
```

- To link a web page to the CSS style sheet file
  - Use a `<link>` tag

```
<link rel="stylesheet" type="text/css"
      href="MyStylesheet1.css">
```

CSS SELECTORS

- Each style rule specifies a CSS selector
  - The selector defines which parts of the HTML document will be affected by the declarations
- There are several types of CSS selector:
  - Element selectors
  - Class selectors
  - ID selectors

```
/* Applies to all
elements */
* {
  font-family: verdana;
  color: #9c1738;
}
```

```
/* Applies to all <div> elements */
div {
  color: red;
}

/* Applies to all <p> elements */
p {
  color: blue;
}
```

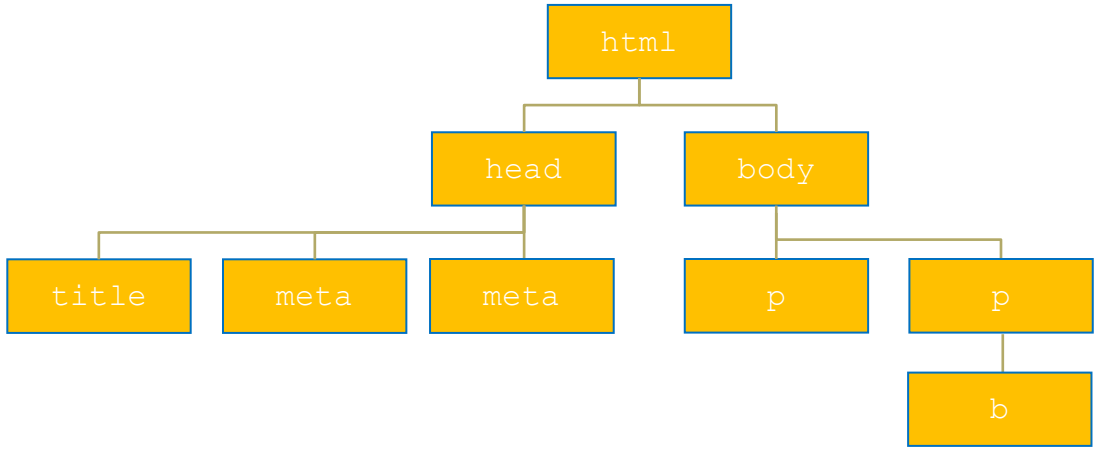
CSS SELECTORS

```
/* Applies to elements that have "optional" class */
.optional {
  border: 1px solid lightblue;
}

/* Applies to elements that have "required" class */
.required {
  border: 1px solid pink;
}
```

```
/* Applies to element with ID "mainContent" */
#mainContent {
  color: blue;
  background-color: #eeeeff;
}

/* Applies to element with ID "additionalContent" */
#additionalContent {
  color: red;
  background-color: #ffeeee;
}
```

CSS WITH THE HTML STRUCTURE

```
<html>
  <head>
    <title>This is my simple page</title>
    <meta name="author" content="John Smith"/>
    <meta name="description" content="My example"/>
  </head>
  <body>
    <p id="intro">Intro to my page</p>
    <p id="main">The <b>main</b> content</p>
  </body>
</html>
```



CSS SELECTORS

```
/*Applies to element with a style of e2, which must be
a descendant of an element with an ID of e1 */
#e1 .e2 {
  declarations…
}
```

```
/*Applies to element with a style of e2, which must be
a direct child of of an element with an ID of e1 */
#e1 > .e2 {
  declarations…
}
```

```
/*Applies to element with an ID of e2 but only if it
is at the same level as an element with an ID of e1
(shared parent / adjacent sibling) */
#e1 + #e2 {
  declarations…
}
```

COMBINING CSS SELECTORS

```
/* For general elements with class "standout" */
.standout {
  color: orange;
  font-size: 18pt;
  font-weight: bold
}

/* For <h1> elements with class "standout" */
h1.standout {
  font-size: 36pt;
}
```

```
/* This rule applies to each of the elements */

h1, h2, h3, h4, h5, h6 { color: orange; }
```

CSS PSEUDO CLASSES

- CSS supports several pseudo-classes for hyperlinks

Pseudo-class	Description
`:link`	Selects all unvisited links
`:visited`	Selects all visited links
`:hover`	Selects links on mouse-over
`:active`	Selects the active link

- The `:focus` pseudo-class selects the element that currently has input focus

CSS – INJECTING CONTENT

- The `::before` and `::after` pseudo-elements can be used to insert content before/after the content of an element

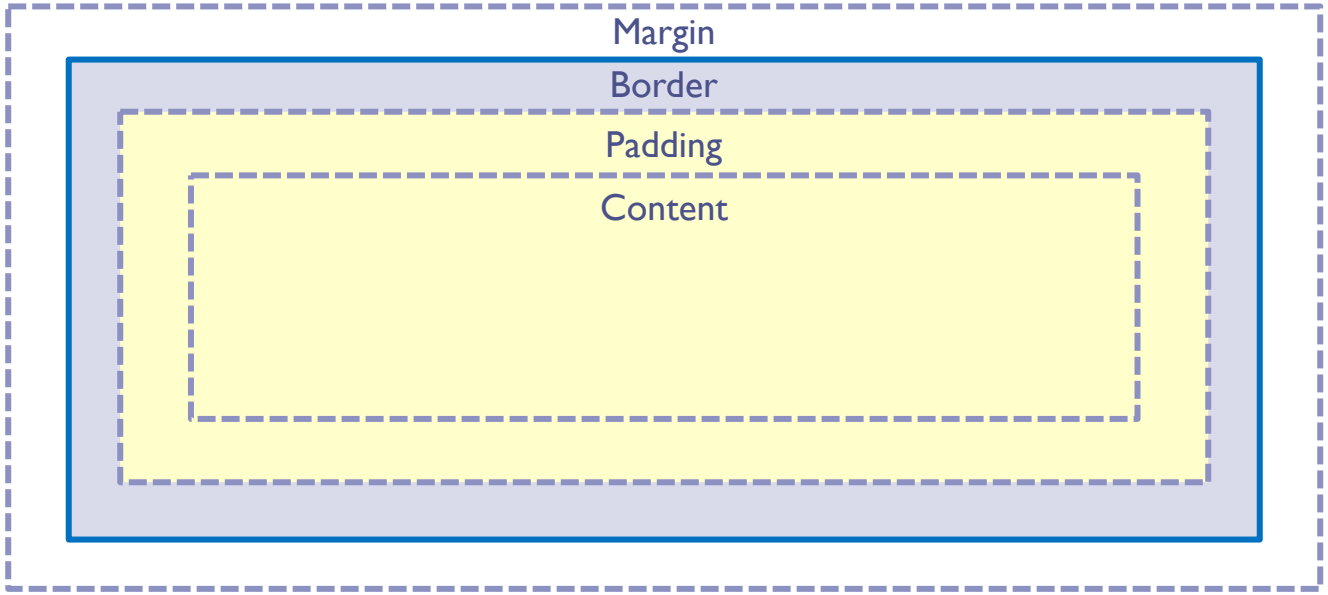
```
aSelector::before {
  content: someContent
}
aSelector::after {
  content: someContent
}
```

STYLING – BACKGROUND + TEXT

- background: #ff00dd
- color: rgb(100,150,166)
- text-decoration: none
- text-decoration: underline
- text-transform: uppercase
- text-align: center
- font: arial
- font-weight: bold
- font-size: 12px
- font-size: large
- font-size: 1.2 em
- font-style: italic

STYLING –TABLES AND DIVS

- `border: 1px solid #000`
- `Border-collapse: collapse`
- `width: 400px`
- `width: 60%`
- `max-width: 1000px`
- `height: 200px`
- `margin: 10px; margin-top:5px;`
- `padding: 5px 10px 5px 10px;`



STYLING – BLOCK, INLINE & FLOATING

- Block elements – take full width, with line-break before and after
  - `<h1> <p> <div> <li>`
- Inline elements – take up as much width as needed. (can't use margin/padding)
  - `<span> <a>`
- This can be changed with: `display: inline-block` (allows you to set a height and width)
- Block elements can be made to appear next to other elements with
- `float: left / float:right`
- Clear floats with `clear: both`

USING WEB FONTS

- All browsers include a few basic fonts – additional fonts can be added using stylesheets

DEBUGGING / CASCADING

- origin (author, user, user-agent)
- specificity (inline, ID, classes, elements)
- order (last takes precedence) nothing overrides inline style except !important
- inheritance

RESPONSIVE WEB PAGES

5

RESPONSIVE SITE DESIGN

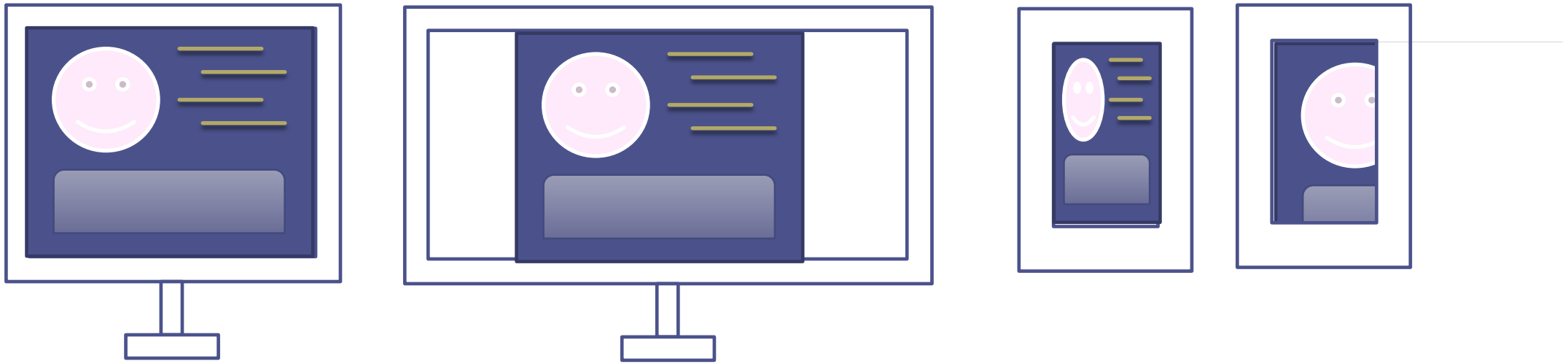
- Websites need to be usable on different devices (mobile vs desktop)
- Sometimes you may wish to provide a separate website for smaller devices

```
if(screen.width <= 600)
 window.location.replace("https://mobile.mysite.com");
```

- Most of the time we can adjust layouts using 3 strategies:
  - Resizing
  - Scaling
  - Floating

THE VIEWPORT

- Originally web pages were designed to be a fixed size, and we designed for (e.g.) 800px wide. This worked ok for wider screens – the page was just centered on the screen
- But for smaller devices, you'd see either a squashed page, or only part of it, with a horizontal scrollbar



THE VIEWPORT

- Including the viewport meta tag in the head section tells the browser whether to scale the page
- This is the standard setting – use this when you are adjusting the styling or layout to take into account the screen size

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- This version would stop users being able to zoom in

```
<meta name="viewport" content="width=device-width,
    initial-scale=1.0, maximum-scale=1.0, user-scalable=0">
```

USING CSS TO RESIZE FOR SCREEN SIZES

Example: set an image's width to:

- 100% if the screen width < 768px,
- 50% if the screen width is between 768 and 1100px,
- 768px exactly if the screen width is > 1100px

```
.pic1 { width: 768px}
@media only screen and (max-width: 768px)
{
  .pic1 { width: 100% }
}
@media only screen and (min-width: 769px) and (max-width: 1100px)
{
  .pic1 {width: 50%}
}
```

USING CSS TO RESIZE FOR SCREEN SIZES

Suggested breakpoints:

- 576px
- 768px
- 992px
- 1200px
- 1400px

ACTIVITY - STYLE A BASIC HTML PAGE

- Create a css file and link to it within the HTML page with the customer form you created earlier
- Style the form as follows:
  - it has a fixed width of 900px, centered on the screen, which reduces to 600px on devices with a screen size of less than 1024px, or the 95% of the width of the screen for devices with a screen size of less than 640px;
  - Use a Google font of your choice
  - Ensure the fields line up
  - The button should change colour when the mouse hovers over it

JAVASCRIPT BASICS

6

JAVASCRIPT VERSIONS

- Most JavaScript applications have traditionally been written using "classic" JavaScript - ECMAScript 5 (ES5)
- The ECMAScript standard has evolved a lot since then
- ES2015, aka ES6 - major changes, e.g. lambdas etc.
- ES2016, aka ES7 - async/await etc.
- ES2017, aka ES8 - more async/await features etc.
- ES2018, aka ES9 - variadic functions etc.
- ES2019, aka ES10 - array changes etc.

JAVA VS JAVASCRIPT

Java and Javascript both use c style syntax

Java is OO, Javascript is prototype functional

Java is compiled, Javascript is interpreted

Java runs in JVM, Javascript runs in Browser / nodeJS

Why is Javascript called Javascript? Marketing!

DATA TYPES

Javascript has the following data types:

- number
- boolean
- string
- symbol
- object
- null
- undefined

DECLARING VARIABLES

- Variables do not have a fixed type
- var x = 6 (old – don't use this!)
- let x = 6 - mutable
- const x = 6 – immutable

- Arrays are defined with [ ] and can contain mixed variable types
- let sizes = ["small", "large", "extra large"]

CODING RULES

- Javascript is case sensitive
- Whitespaces (spaces / tabs / newlines) can be used freely
- Indent as you wish (or not)
- Semi-colons are optional (but required if not using a new line)
- Comments are like Java – either // or /* … */

TESTING EQUALITY

- Use === or !=== for strict equality (data and object type must match)
- Use == or !== for “loose” equality (data must match but object type doesn’t have to)

Generally

- Always use === or !== unless you are comparing to null

OPERATORS

- = assignment
- + / - * plus, divide, minus, multiply
- += plus and assign (and /= -= etc)
- ++ increment
  - y = x++ is the same as y = x, x = x+ 1
  - y = ++x is the same as x = x + 1, y = x
- && and
- || or

STRINGS

- Can be defined using “ or ‘ or `
- Use backticks for interpolation `my name is ${name}`
- + concatenation
- Change a string to a number by preceeding it with a +

OBJECTS / JSON

- Objects are a set of key,value pairs
- The values could be key:value pairs, or a list of values

```
var x = {  name:'matt',
           age:26,
           foods : ['pizza', 'pasta'],
           preferences: {[
               {colour: 'blue'},
               {season: 'summer'}
           ]}
         }
```

OBJECTS / JSON

- Replace a value in an object using . notation
- Use the spread operator to override part of an object

CONDITIONS AND LOOPS

- if / if else = works like Java
- switch = works like Java

- for = works like Java for incrementors
  - for (x = 1; x <10; x++)
  - for (String s **of** someStrings)

CREATING FUNCTIONS

```
function sayHello(firstname,surname) {
   alert('Hello ' + firstname + ' ' + surname);
}


const sayHello = (firstname, surname) => {
 alert('Hello ' + firstname + ' ' + surname);
}


const sayHello = (firstname, surname) =>  alert('Hello ' + firstname + ' ' + surname)
```

THE DOM

- We can manipulate the visible page from Javascript code – we treat the HTML elements as object (Document Object Model)
- Find elements using (for example):
- let someElement = document.getElementById('para1');
- let someElements = document.getElementsByClassName('class1');

- Trigger code when HTML events occur using (for example):
- <button onclick="someFunction();">blue</button>

HELPFUL FUNCTIONS

- alert() – creates a popup window
- console.log() – outputs to the console

ACTIVITY - CREATE SOME BASIC JAVASCRIPT

- Using the HTML page with the customer form you created earlier
- Add the following functionality to the form:
  - The instructions should not be visible by default, but should be made visible when the user clicks on the title – use an arrow (the keyboard characters ^ or >) to indicate this.

EXTERNAL LIBRARIES

7

BOOTSTRAP - HTTPS://GETBOOTSTRAP.COM/

- Provides CSS and scripts to make building responsive sites easier
  - Grid system allows for dynamic reflow of elements
  - Container provides default padding, alignments and other features
- Provides default (but customisable) themes
  - Easier way to achieve consistent styling of elements
- Provides components for extra functionality
  - Navigation bars, dropdows, progressbars
- But be careful – different versions have different syntaxes

JQUERY - HTTPS://JQUERY.COM/

- Provides scripts to make interacting with the DOM easier
  - Find and adjust elements with simpler syntax
  - Easier way to make ajax requests

- Bootstrap 3 and 4 use Jquery. Bootstrap 5 doesn't but you can add it separately

FONTAWESOME - HTTPS://FONTAWESOME.COM/

- Provides css and scripts to allow you to make it easier to use small regular icons
  - Large library of standard images
  - Different styles, sizes and ways to embed.

USEFUL TOOLS

8

USEFUL TOOLS

- https://squoosh.app/ - reduce photo sizes (+ convert to webp format)
- https://web.dev/measure/ - page quality measurement
- https://wave.webaim.org/ - accessibility measurement