Contemporary Web Development
Lesson 1

http://bit.ly/cannotfeelmybones

Hey mlab, Avner here.

- https://avner.js.org
- https://github.com/Avnerus
- avner.peled@aalto.fi

Notable web projects:

- https://github.com/Avnerus/tzina : WebVR Documetary (IDFA Doclab 2016).
- https://github.com/Avnerus/STIR : Mobile Web App (IDFA Doclab 2017) .
- https://github.com/Avnerus/silencecaptive : Wi-Fi Portal (CCA Gallery).
- https://github.com/Raycasters/Marrow : AI Installation (IDFA Doclab 2018).
- https://github.com/Avnerus/softbot : Soft Robotic Avatar Platform (MA Thesis).
- https://github.com/Avnerus/riot-isomorphic : RiotJS Framework powering Feature.fm

Disclaimer

This world is moving so fast that everything might change soon.

Also this the first time I am teaching this course (or any), so this the beta phase.

Evolution of web technology

http://www.evolutionoftheweb.com/

- Static HTML/CSS

  ```
  # telnet www.oocities.org 80
  GET /hollywood/1525/ HTTP/1.1
  Host: www.oocities.org
  ```

- Javascript
- Weird Hybrids (Java, Flash, ActiveX, Silverlight 🤢)
- AJAX
- HTML5 (Canvas, SVG)
- **Node JS**
- **Front-end frameworks**
- **Web Extensions**

The web browser is ***slowly*** replacing the operating system, becoming a standard for content.

Chrome OS*

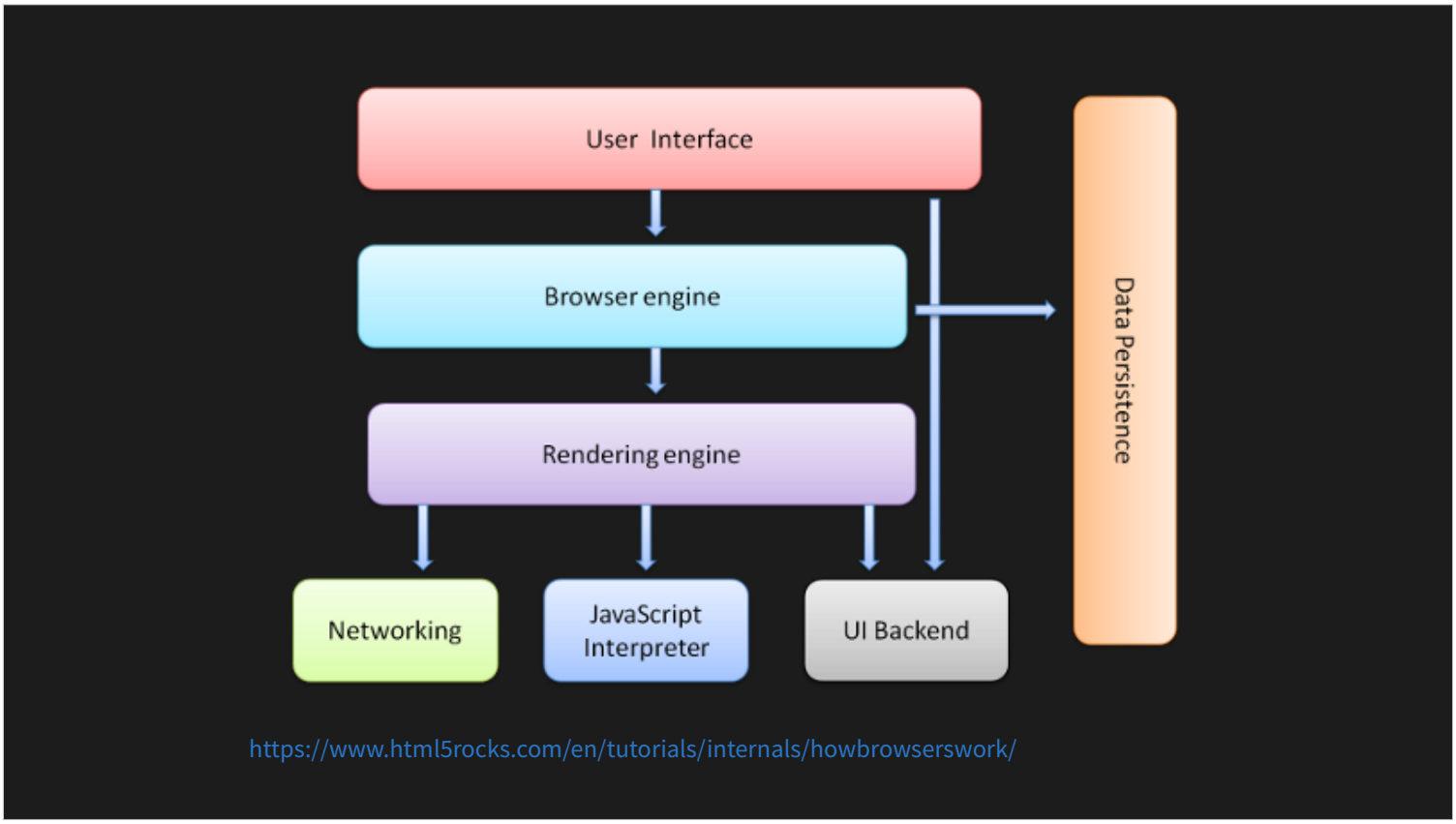
Firefox OS**

* Chrome OS powered notebook now supports Android apps and more.
** Firefox OS has been discontinued and forked by several other projects.

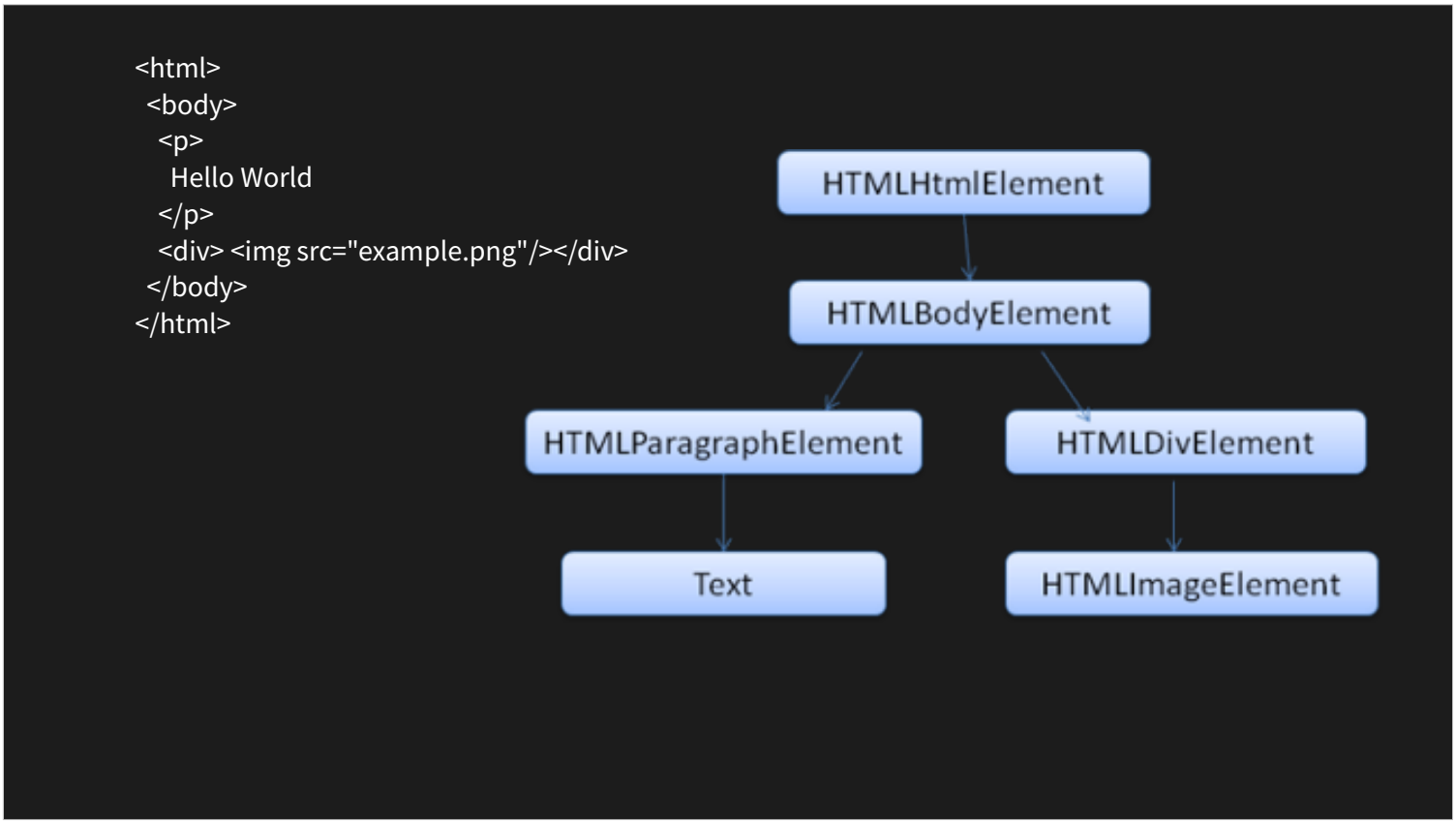
Having the browser as the main operating system solves many issues regarding cross platform compatibility and security. Everything is based on standards and runs in secure environment. Run ,multiple tabs like multiple processes.

Progressive web apps are standards for running offline and/or cached web content. They also support multiple threads.

But let’s focus on one tab.



The rendering engine parses the DOM tree.



DOM is the object tree model that is represented by the tree of tags.

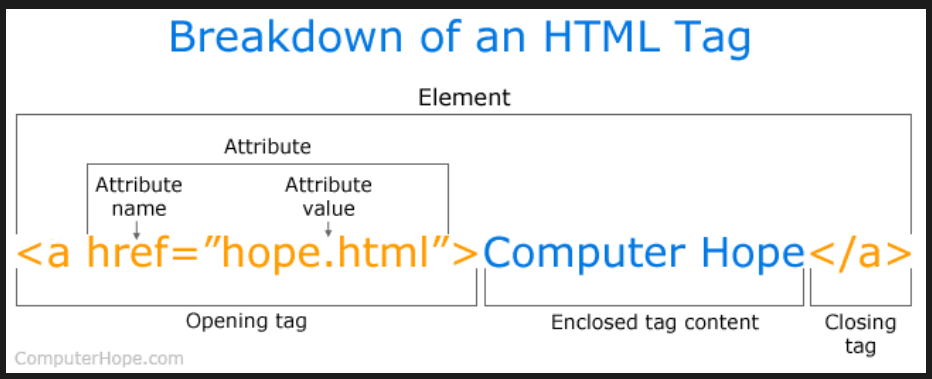
HTML elements with no content are called empty elements.

 is an empty element without a closing tag (the
 tag defines a line break).

Empty elements can be "closed" in the opening tag like this:
.

HTML5 does not require empty elements to be closed. But if you want stricter validation, or if you need to make your document readable by XML parsers, you must close all HTML elements properly.

HTML is almost XML



A more standard HTML is XHTML
An empty tag must still have / (<img/>)

A happy family

```
<father name=”Papa” age=50>
        <refrigerator>
                ice cream
        </refrigerator>
</father>
<mother name=”Mama” age=45>
        <son name=”Johnny” age=8>
                sugar
        </son>
        <daughter
                name=”Neene”
                age=7
                 oneatsugar=”changeMood(‘happy’)
        />
</mother>
```

A happy family

```
<father name=”Papa” age=50>
        <refrigerator>
                ice cream
        </refrigerator>
</father>
<mother name=”Mama” age=45>
        <son name=”Johnny” age=8>
                sugar
        </son>
        <daughter
                name=”Neene”
                age=7
                oneatsugar=”changeMood(‘happy’)
        />
</mother>
```

Don't actually do this in web development. Structure and behavior should be separated. *"Unobtrusive Javascript"*

Code style guide

```
<johnny
        age=8
        shirt-color=”pink”
        pants-color=”pink”
        favorite-food=”sugar”
/>
```

Let's think about Reddit.

All about <components>

Web components

Polymer

React

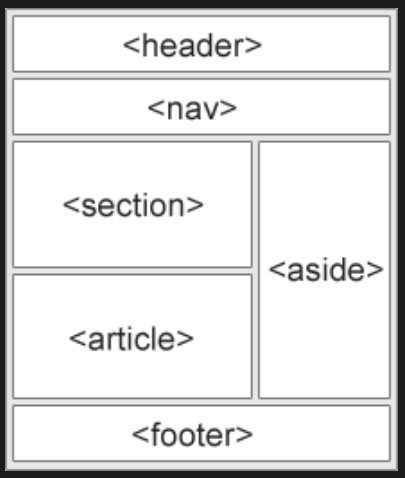
Riot JS

We are moving beyond the strict definition of HTML elements.

The named components are being processed by the front-end engine and rendered into HTML UI elements.

The upcoming web components standard is powered by the “Shadow DOM”

HTML5 Semantic Elements

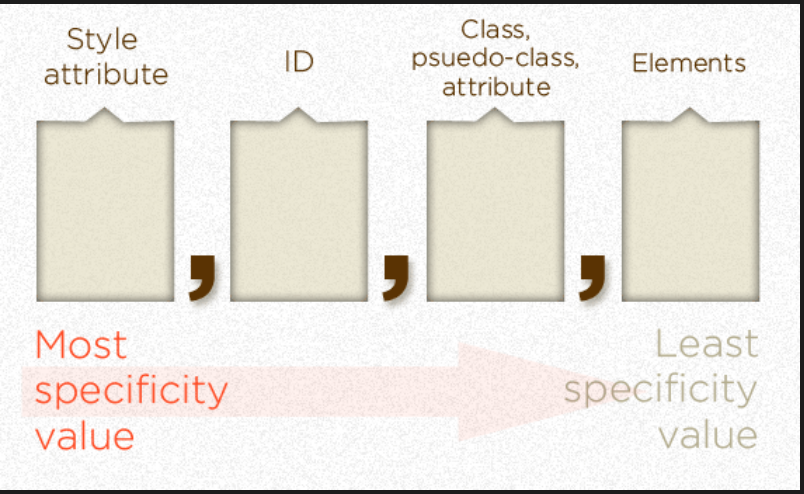


Styling

```
.human {
        empty-cells: show;
}
father {
        overflow: visible;
}
#child1 {
        border: none;
}

.human refrigerator {
        box-shadow: 6px 6px #666
}
```

```
<father name=”Papa” age=50 class=”human”>
        <refrigerator>
                ice cream
        </refrigerator>
</father>
<mother name=”Mama” age=45
class=”human”>
        <son id=”child1” name=”Johnny” age=8>
                sugar
        </son>
        <daughter
                name=”Neene”
                age=7
                 oneatsugar=”changeMood(‘happy’)
        />
</mother>
```

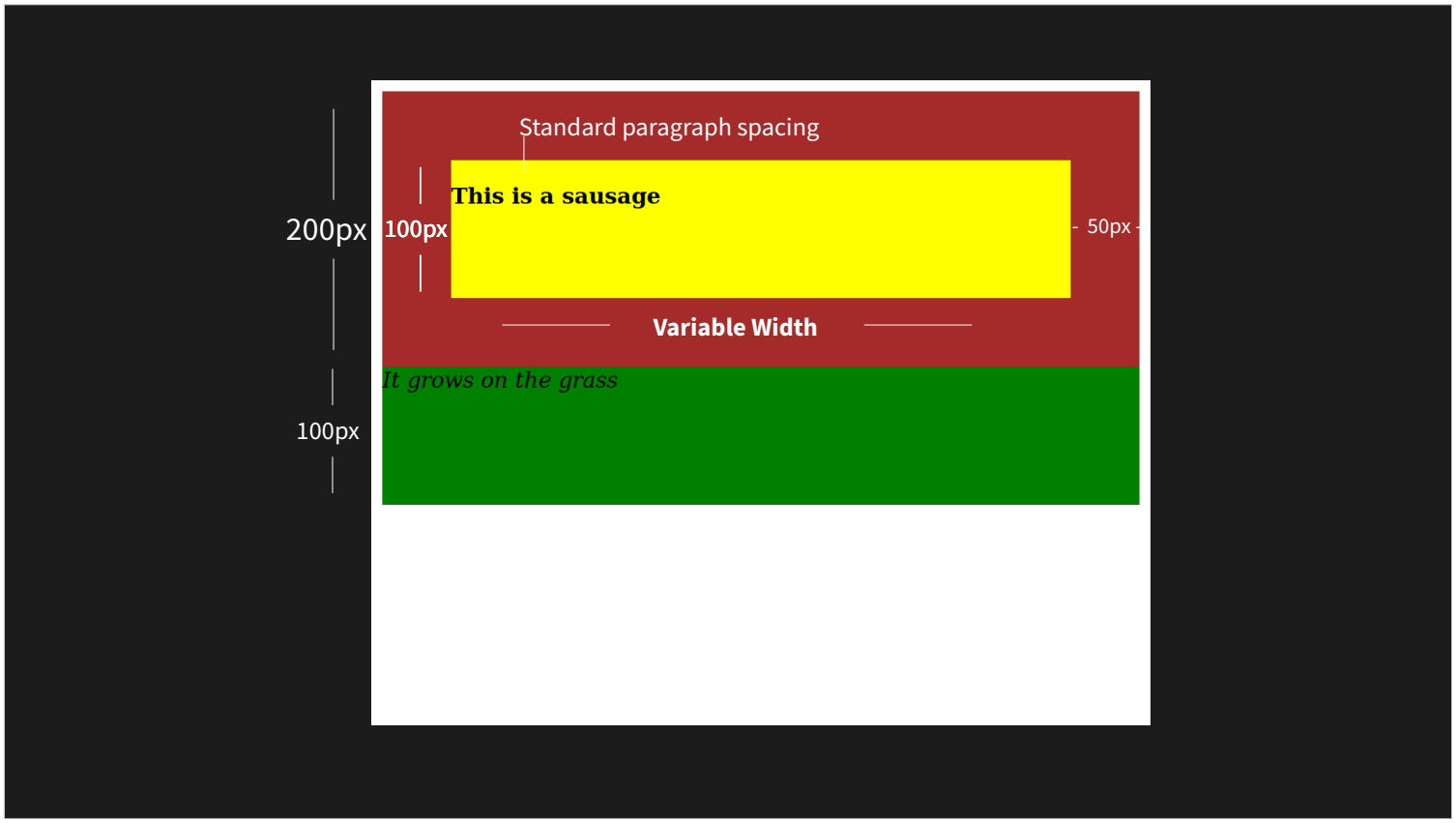
CSS Specificity - Example



https://css-tricks.com/specifics-on-css-specificity/

Contemporary Style:
SASS - Example

Also showing attribute selectors and inherited vs non inherited properties.



Can also be done using postion:relative and top:

Why not just margin? They collapse

https://www.w3.org/TR/CSS21/box.html#collapsing-margins

"Margins collapse between adjacent elements. In simple terms, this means that for adjacent vertical block-level elements in the normal document flow, only the margin of the element with the largest margin value will be honored, while the margin of the element with the smaller margin value will be collapsed to zero"

Three methods for centering elements.

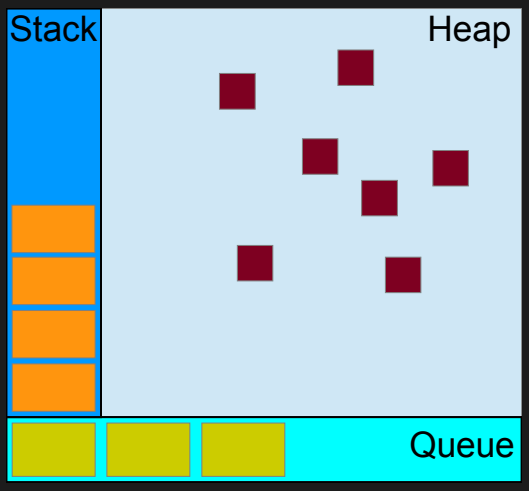
Even more here http://thenewcode.com/723/Seven-Ways-of-Centering-With-CSS

Responsive Design with Media Queries - Example

```
/* Large desktop */
@media (min-width: 1200px) { /* CSS Classes */ }

/* Portrait tablet to landscape and desktop */
@media (min-width: 768px) and (max-width: 979px) { /* CSS Classes */ }

/* Landscape phone to portrait tablet */
@media (max-width: 767px) { /* CSS Classes */ }

/* Landscape phones and down */
@media (max-width: 480px) { /* CSS Classes */ }
```

Javascript

The runtime model



More info

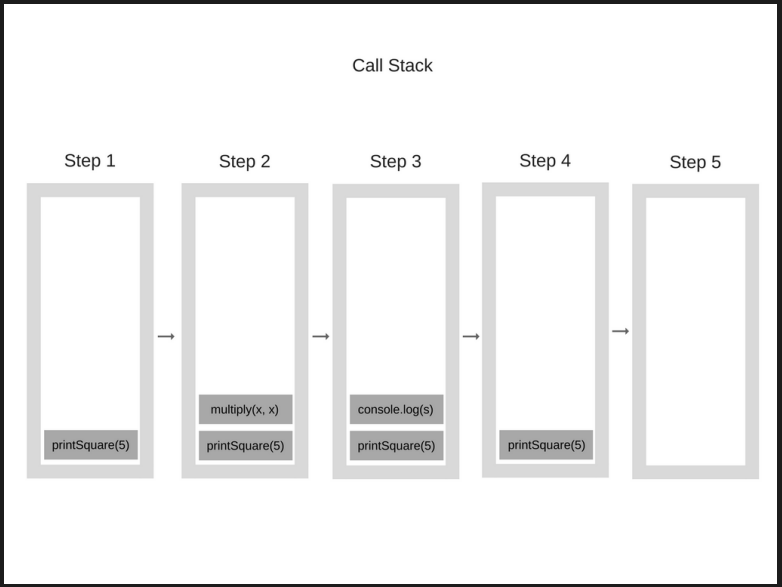
Stack – The call stack of normal javascript functions, it remebers the state of each function executing.

Heap – All javascript variables are objects that are passed by reference and are allocated on the memory heap. They are ‘reference-counted’, so when an object has 0 references, it will be ‘garbage collected’.

Queue – The Event Message Queue of user and asynchronous events.

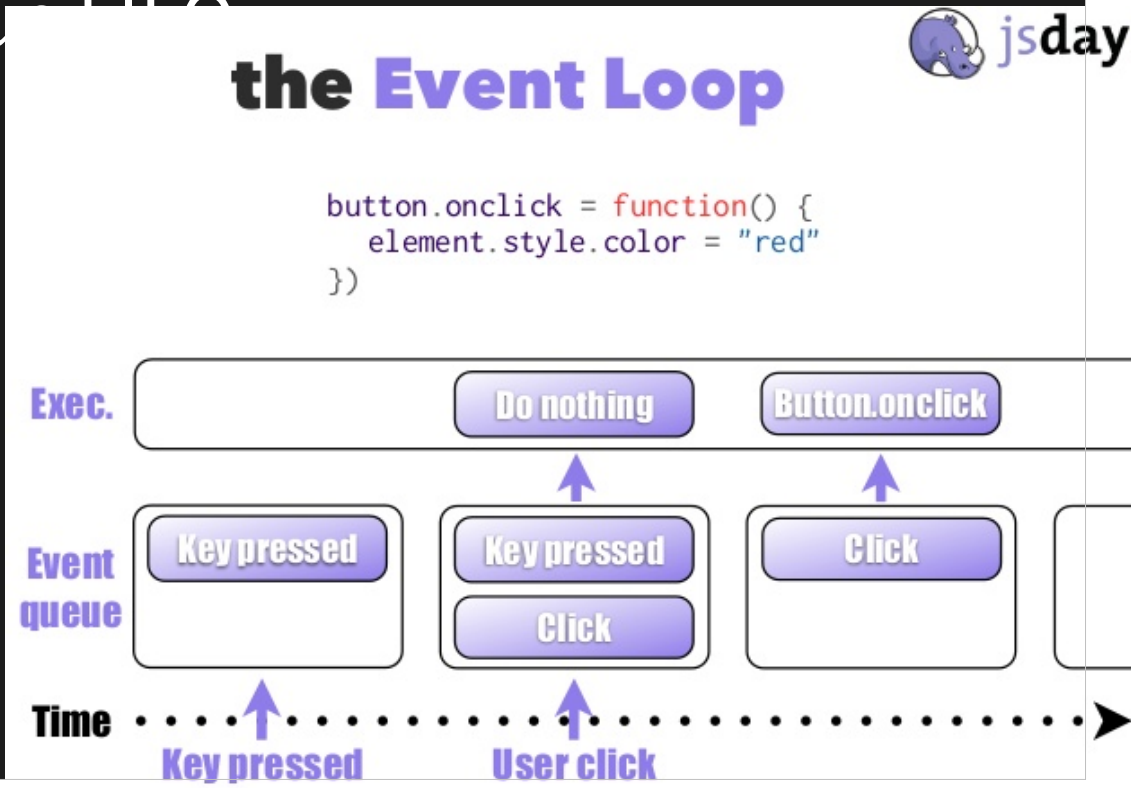
Call Stack LIFO

```
function multiply(x, y) {
  return x * y;
}
function printSquare(x) {
  var s = multiply(x, x);
  console.log(s);
}
printSquare(5)
```



From https://blog.sessionstack.com/how-does-javascript-actually-work-part-1-b0bacc073cf

Event Queue - FIFO



Blocking / Non-blocking
Example

I/O is such as network requests are never blocking,
They get offloaded to the browser's API threads.
But normal functions and code loops are blocking
since javascript is single-threaded,

Good video:
https://www.youtube.com/watch?v=8aGhZQkoFbQ

Listening to Events and Event Bubbling

Example

I/O is such as network requests are never blocking,
But normal functions and code loops are blocking
since javascript is single-threaded,

Warm up exercise – your dream website OR a sample website.

Don’t think about styling (css positioning) details.
Just think about the tree structure, what are the components? What is contained in what? What are the component attributes?

Think about actions and reaction to “events”, either user generated events or any ‘external’ event.

Something starts running on startup?

```
Window.onload = () => {

}
```

Home exercise – Stack/Queue tracing

```
window.onload = function run() {
  let cracker =   document.querySelector("#cracker");
  cracker.addEventListener("click", dipCracker);
}
function dipCracker(event) {
  let cracker = event.target;
  if (cracker.style.backgroundColor != "green") {
     cracker.style.backgroundColor = "green";
    dipCracker(event);
  } else {
    setTimeout(function eat() {
      console.log("Yummy!")
      cracker.style.display = "none";
    },2000);
    console.log("Nom Nom...");
  }
}
```

Pen

```
<html>
        <body>
                <div id=”cracker”>
                </div>
        </body>
</html>

#cracker {
        background-color: brown;
        width: 100px;
        height: 200px;
}
```

Imagine the user opens the page and then clicks on the cracker.

Trace the state of the stack queue and event queue throughout the process.

Also practice:

- CSS Selectors: https://flukeout.github.io/