

# INTRODUCTION TO ACCESSIBILITY

Some common terms

Here are some **common terms** that are relevant to accessibility.

The W3C

The **World Wide Web Consortium** or the W3C is an international community that develops the open standards for the Web.

The W3C produces **specifications** on a wide range of web-related topics including HTML, CSS and Accessibility.

W3C technical specifications have **four levels of maturity:**

## **1. Working Draft (WD)**

A document that W3C has published for review by the community, including W3C Members, the public, and other technical organizations.



## **2. Candidate Recommendation (CR)**

A document that satisfies the Working Group's technical requirements, and has already received a comprehensive review.

### **3. Proposed Recommendation (PR)**

A document that has been accepted by the W3C Director as of sufficient quality to become a W3C Recommendation.

## **4. W3C Recommendation (REC)**

A specification or set of guidelines that, after extensive consensus-building, has received the endorsement of W3C Members and the Director.

WAI

Within the W3C, there is a sub-group called the **Web Accessibility Initiative** (WAI) Working Group.

The WAI Working Group has been responsible for developing the **Web Content Accessibility Guidelines** (WCAG).

WCAG

The WCAG guidelines **provide a standard for web content accessibility.**



WCAG 1.0 became a **W3C Recommendation** in May 1999.

<https://www.w3.org/TR/WAI-WEBCONTENT/>

WCAG 2.0 became a **W3C Recommendation** in December 2008.

<http://www.w3.org/TR/WCAG20/>

WCAG 2.1 became a **W3C Recommendation** in June 2018.

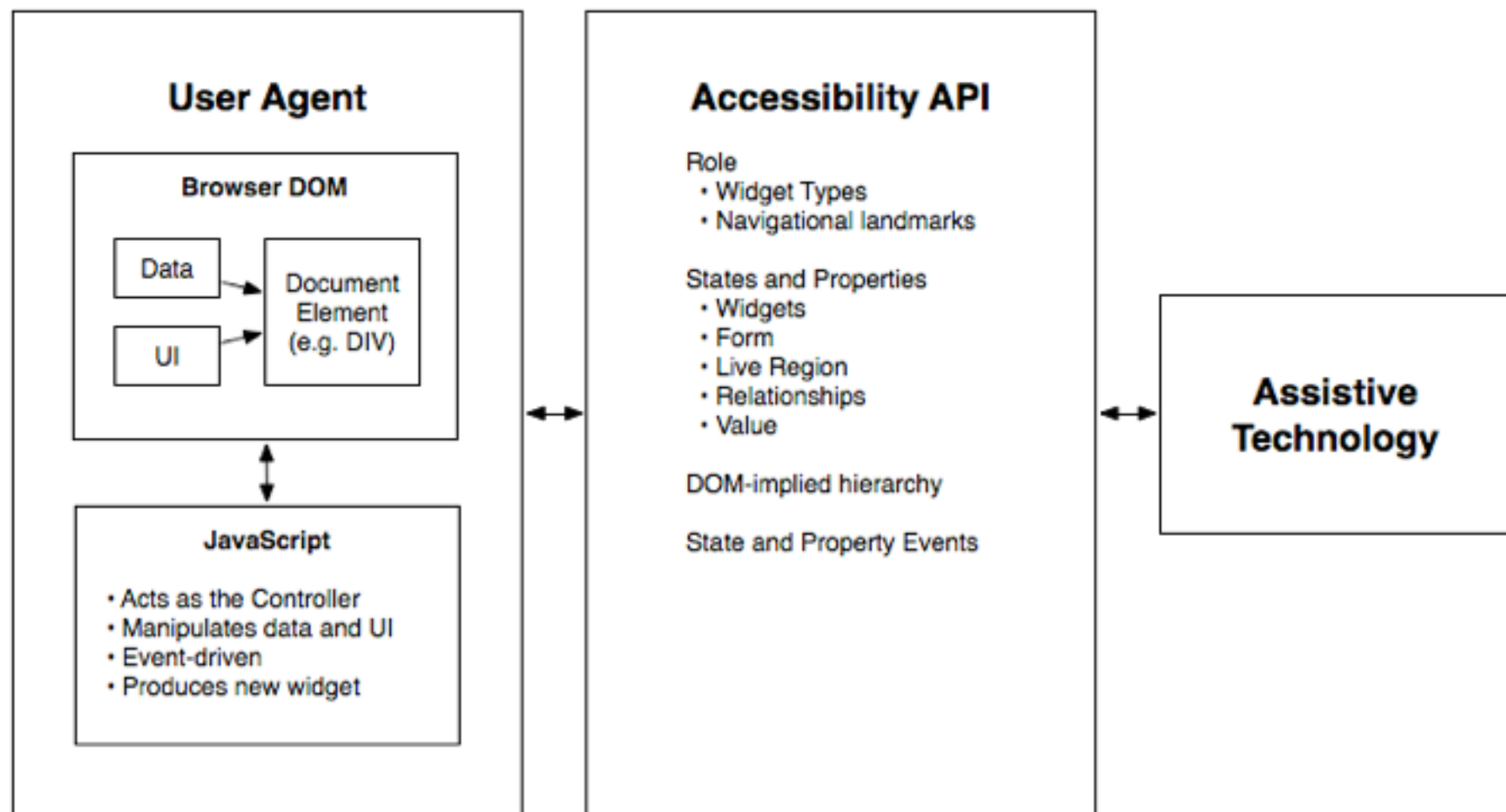
<https://www.w3.org/TR/WCAG21/>

# Accessibility API

Accessibility application programming interfaces (APIs) are used to **communicate semantic information about the user interface** to Assistive Technologies.

“Accessibility APIs constitute a **contract between applications and assistive technologies**, to enable them to access the appropriate semantics needed to produce a usable alternative to interactive applications.”

For example, the Accessibility API **helps screen reading software** determine whether a particular UI widget is a menu, button, text field, list box, etc.





Accessibility APIs **expose information about each object** within the application such as:

1. The object's **role** (e.g. a menu, a button, an input, an image).

2. A **name that identifies the object within the interface** (e.g. a visible label or a name that has been encoded directly in the object).

3. The object's **state** (e.g. selected, unselected, checked, unchecked).

More than one API?

In OS X Safari and Chrome support  
**NSAccessibility.**

In iOS Safari and Chrome support  
**UIAccessibility.**

Some browsers support **one or more of the available accessibility APIs** for the platform they're running on.



In Windows, Firefox and Chrome support **MSAA/IAccessible** and **IAccessible2**.

And Internet Explorer supports **MSAA/**  
**IAccessible** and **UIAExpress**.

This is why you should always test against **more than one** Browser/ Assistive Technology combination.

## Windows

IE: JAWS & NVDA

FireFox: JAWS & NVDA

Chrome: JAWS & NVDA

## OSX

Safari: VoiceOver

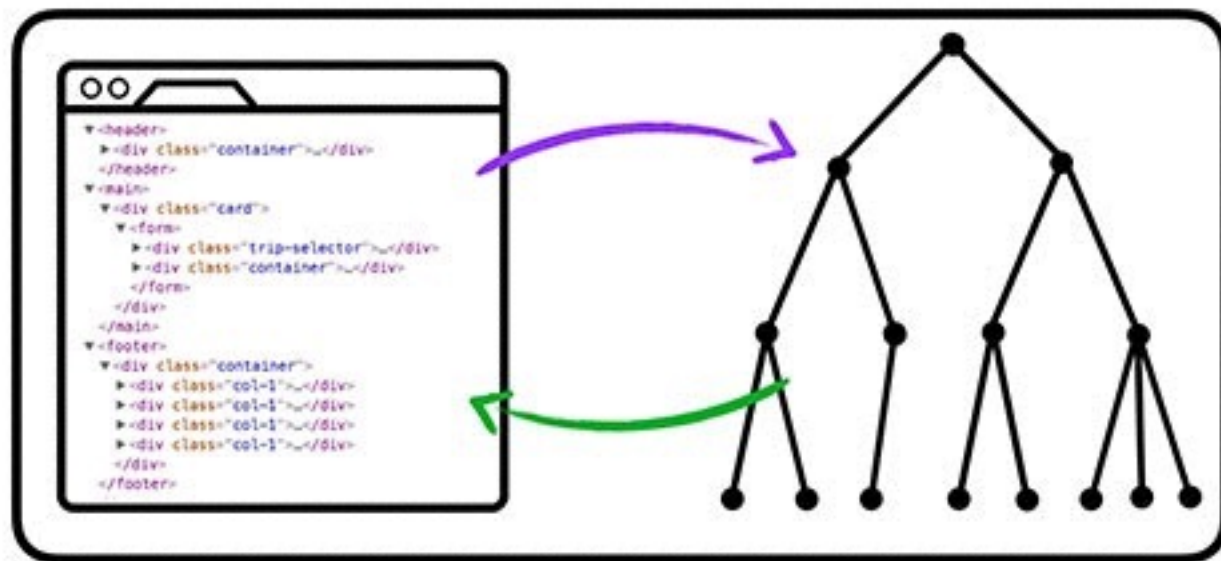
FF: VoiceOver

Chrome: VoiceOver

# Accessibility Tree

Browsers take the DOM tree **and modify it**, to turn it into a form that is useful for assistive technologies.

This modified tree, is referred to as **the accessibility tree** - a subset of of the DOM tree.



DOM

accessibility  
tree



The accessibility tree contains **only** **“Accessible objects”**. These are nodes that have states, properties or events.

All other DOM nodes (that do not have states, properties or events) **are not presented in the accessibility tree.**

For example, **a section within the DOM tree** could be:



The Accessibility tree would **only**  
**present the following:**

```
<form action="#">
```

```
  <label for="name">Name</label>
```

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

```
  <button type="submit">Submit</button>
```

```
</form>
```

Each browser could potentially present  
a **slightly different accessibility tree**.

# Widgets



Within the various WAI ARIA specifications, there are multiple references to **“widgets”**.

A widget is a component that **enables a user to perform a function** or access a service such as a dropdown menu, a modal or a tooltip.

# Exercise 01a:

## Using keyboard-only

Before using any screen reader, it is important to understand how to navigate websites and applications **using the keyboard only.**

We'll start by opening a  
**demonstration page to practice on.**

Open **exercise01-keyboard-only/  
exercise1.html**

Moving forwards and  
backwards

The **TAB** keystroke moves focus to the **next focusable element** on the page.



The **SHIFT TAB** keystroke moves focus to the **previous focusable element** on the page.

Select menus

The **DOWN ARROW** keystroke will move focus to the **next option in dropdown menu**.

The **UP ARROW** keystroke will move focus to the **previous option in dropdown menu.**

The **ENTER** and **SPACEBAR** keystrokes will **select the option** that is currently in focus.

The **ESC** keystroke will **close the dropdown menu.**

Radio buttons

As long as all radio buttons have a matching name value, **they will act as a radio button group.**



This means you can **only select one radio button** from within the group at a time.

The **TAB** will move focus **into and out of a radio button group.**

The **SPACEBAR** keystroke will **select the current radio button.**

The **DOWN ARROW** keystroke will **move focus to next radio button and select it.**

The **UP ARROW** keystroke will **move to focus to the previous radio button and select it.**

When a radio button has been selected from within a radio group, **it is impossible to uncheck radio buttons from within this group**. The selection can be changed, but not unselected.

# Checkboxes

Unlike radio buttons within a group, checkboxes are **always treated as individual form controls.**



The **TAB** will move focus **into and out of each checkbox.**

The **SPACEBAR** keystroke will **select and unselect the current checkbox.**

# Exercise 01b:

## VoiceOver

Open **exercise01-keyboard-only/  
exercise1.html**

VO keys

VoiceOver uses “**VO**” **keys for control**.  
The default VO keys are the **CONTROL** +  
**OPTION** keystrokes.

The + symbol indicates that these keys are **used together**.

These two keys can be **changed in VoiceOver settings** as needed.



Starting and stopping

The **COMMAND** + **F5** keystrokes will **start VoiceOver**.

Alternatively, **VoiceOver** can be  
**started manually via:**

System Preferences > Accessibility > VoiceOver > Enable  
VoiceOver

The **COMMAND** + **F5** keystrokes will also  
**quit VoiceOver.**

Alternatively, VoiceOver can be **quit manually** by clicking the “X” icon in the top left corner of the VoiceOver panel.

Reading

The VO + A keystrokes will trigger VoiceOver to **start reading**.

The **CONTROL** keystroke will trigger VoiceOver to **stop reading**.



The VO + RIGHT ARROW keystrokes will  
**read the next item.**

The VO + LEFT ARROW keystrokes will  
**read the previous item.**

The **VO** + **B** keystrokes will **read from top of the page to the current location.**

# Navigating

The VO + COMMAND + L keystrokes will  
**take you to the next link.**

The **VO** + **COMMAND** + **H** keystrokes will  
**take you to the next heading.**

The **VO** + **COMMAND** + **J** keystrokes will  
**take you to the next form control.**

The VO + COMMAND + X keystrokes will  
**take you to the next list.**



The VO + COMMAND + T keystrokes will  
**take you to the next table.**

The VO + COMMAND + U will **display the Rotar.**

# Exercise 01c: NVDA

Open **exercise01-keyboard-only/  
exercise1.html**

NVDA key

The **NVDA key** is set to the **INSERT** key by default, but it can be changed to the **Caps lock** key when installing NVDA for the first time.

If you want to **change your NVDA key preferences**, press **CTRL + NVDA + K**.

Starting and stopping



The NVDA application **needs to be manually opened** in order to begin reading.

The **INSERT** + **Q** keystrokes will **quit**  
**NVDA.**

Reading

The **INSERT + DOWN ARROW** keystrokes will trigger NVDA to **start reading continuously from this point on.**

The **CONTROL** keystroke will trigger NVDA to **stop reading**.

The DOWN ARROW keystrokes will **read the next item.**

The UP ARROW keystrokes will **read the previous item.**

# Navigating



The **K** keystrokes will **take you to the next link.**

The L keystrokes will **take you to the next list.**

The **H** keystrokes will **take you to the next heading.**

The **T** keystrokes will **take you to the next table.**